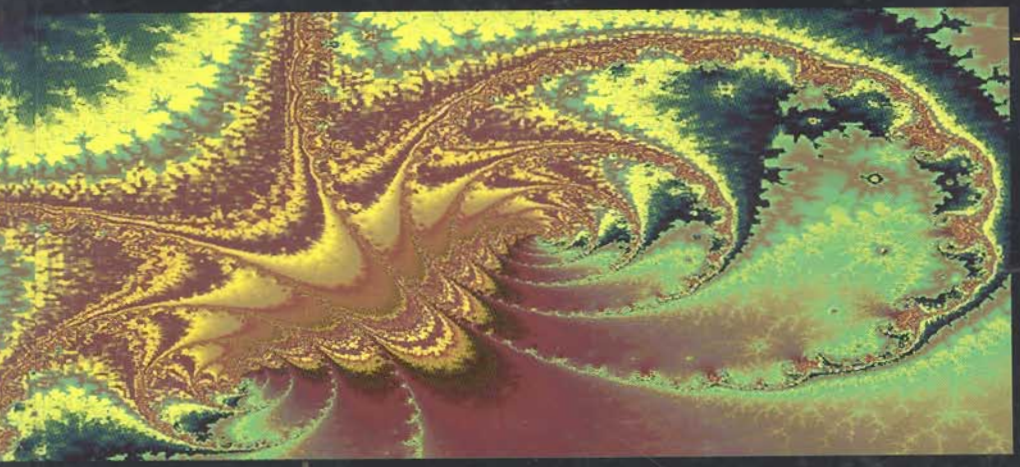
INCLUYE CD-ROM



PROGRAMACIÓN

DESARROLLO WEB CON PHP Y MYSQL

LUKE WELLING
LAURA THOMSON

ANAYA MULTIMEDIA

Índice de contenidos

Introducción **30**

Razones para leer este libro 31
Objetivos que puede alcanzar con este libro 32
Concepto de PHP 33
Novedades de la versión 4.3 de PHP 33
Concepto de MySQL 34
Razones para utilizar PHP y MySQL 34
Alguna de las cualidades de PHP 35
Rendimiento 35
Integración de base de datos 35
Bibliotecas incorporadas 36
Coste 36
Aprendizaje de PHP 36
Portabilidad 36
Código fuente 36
Algunas de las ventajas de MySQL 37
Rendimiento 37
Bajo coste 37
Facilidad de uso 37
Portabilidad 37
Código fuente 37
Cómo está organizado este libro 38
Finalmente 38

Parte I. Uso de PHP 39

1. Curso acelerado de PHP 40

Uso de PHP 42
Aplicación de ejemplo: Bob's Auto Parts 42
El formulario de pedidos 43
Procesamiento del formulario 44
Cómo incrustar PHP en HTML 45
Estilos de etiquetas PHP 46
Instrucciones de PHP 47
Espacios en blanco 47
Comentarios 48
Cómo agregar contenido dinámico 49
Llamada de funciones 49
Función date() 50
Cómo acceder a las variables de formulario 50
Variables de formulario 50
Concatenación de cadenas 53
Literales y variables 54
Identificadores 54
Variables declaradas por el usuario 55
Asignación de valores a variables 55
Tipos de variables 55
Tipos de datos de PHP 56
Control de tipos 56
Conversión de tipos 57
Variables de tipo variable 57
Constantes 58
Ámbito de variables 58
Operadores 59
Operadores aritméticos 60
Operadores de cadena 61
Operadores de asignación 61
Devolución de valores de asignación 61
Combinación de operadores de asignación 62
Incremento y decremento previo y posterior 62
Referencias 63
Operadores de comparación 63
El operador iguales 63
Otros operadores de comparación 64
Operadores lógicos 64
Operadores bit a bit 65
Otros operadores 66
Operador ternario 66

Operador de supresión de error 66
Operador de ejecución 67
Uso de operadores: calcular los totales de los formularios 67
Precedencia y asociatividad: evaluación de expresiones 69
Funciones de variables 70
Cómo probar y establecer tipos de variables 70
Cómo probar el estado de las variables 71
Reinterpretación de variables 72
Estructuras de control 72
Toma de decisiones con estructuras condicionales 72
Las instrucciones if 72
Bloques de código 73
Instrucciones else 73
Instrucciones elseif 74
Instrucción switch 75
Comparación de condiciones diferentes 77
Iteración: repetición de acciones 77
Bucles while 78
Los bucles for y foreach 80
Bucles do...while 81
Cómo salir de una estructura de control o una secuencia de comandos 81
Siguiente paso: guardar el pedido del cliente 82

2. Almacenamiento y recuperación de datos 84

Cómo guardar datos para su lectura posterior 86
Almacenamiento y recuperación de los pedidos de Bob 86
Introducción al procesamiento de archivos 87
Cómo abrir un archivo 88
Modos de archivo 88
Uso de fopen() para abrir un archivo 88
Cómo abrir archivos a través de FTP o HTTP 91
Problemas al abrir el archivo 91
Cómo escribir en un archivo 93
Parámetros de fwrite() 94
Formatos de archivo 94
Cómo cerrar un archivo 95
Lectura desde un archivo 95
Apertura de un archivo para su lectura: fopen() 96
Cómo saber cuándo parar: feof() 97
Cómo leer línea a línea: fgets(), fgetss() y fgetcsv() 97
Lectura de todo el archivo: readfile(), fpassthru(), file() 98
Lectura de un carácter: fget()+ 99
Lectura de una longitud arbitraria de bytes: fread() 99
Otras funciones de archivo útiles 100

Comprobación de la existencia de un archivo: file_exists() 100
Cómo averiguar el tamaño de un archivo: filesize() 100
Eliminación de un archivo: unlink() 100
Navegación dentro de un archivo: rewind(), fseek() y ftell() 101
Bloqueo de archivos 102
La opción más acertada: los sistemas de administración de base de datos 103
Problemas con el uso de archivos planos 103
Cómo resolver estos problemas 104
Lecturas adicionales 105
A continuación 105

3. Uso de matrices 106

¿Qué es una matriz? 108
Matrices indexadas numéricamente 108
Inicialización de matrices indexadas numéricamente 109
Acceso a los contenidos de matrices 109
Uso de bucles para acceder a la matriz 110
Matrices asociativas 111
Inicialización de una matriz asociativa 111
Cómo acceder a elementos de matriz 111
Uso de bucles con matrices asociativas 111
Matrices multidimensionales 113
Cómo ordenar matrices 117
Uso de sort() 117
Uso de asort() y ksort() para ordenar matrices asociativas 117
Inversión del orden 118
Ordenación de matrices multidimensionales 118
Ordenaciones definidas por el usuario 118
Ordenaciones de usuario inversas 120
Reordenación de matrices 120
Uso de shuffle() 121
Uso de array_reverse() 122
Carga de matrices desde archivos 123
Otras manipulaciones de cadenas 125
Navegación dentro de una matriz con each(), current(), reset(), end(), next(), pos(), and prev() 126
Aplicación de cualquier función a cada elemento de una matriz: array_walk() 126
Cómo contar elementos de una matriz: count(), sizeof(), and array_count_values() 128
Conversión de matrices en variables escalares: extract() 129
Lecturas adicionales 130
A continuación 130

4. Manipulación de cadenas y expresiones regulares 132

Aplicación de ejemplo: Smart Form Mail 133
Aplicación de formato a cadenas 136
Limpieza de cadenas: chop(), ltrim() y trim() 136
Aplicación de formato a cadenas para presentaciones 136
Uso de formato HTML: la función nl2br() 136
Aplicación de formato a una cadena para su impresión 137
Cambio de mayúsculas y minúsculas en una cadena 139
Aplicación de formato a las cadenas para su almacenamiento: AddSlashes() y StripSlashes() 140
Cómo combinar y dividir cadenas con funciones de cadena 141
Uso de explode(), implode() y join() 142
Uso de strtok() 142
Uso de substr() 143
Comparación de cadenas 144
Ordenación de cadenas:strcmp(),strcasecmp() y strnatcmp() 144
Comprobación de la longitud de una cadena con strlen() 145
Cómo buscar subcadenas y reemplazarlas con funciones de cadena 145
Búsqueda de cadenas en cadenas: strstr(), strchr(), strrchr(), stristr() 146
Búsqueda de la posición de una subcadena: strpos(), strrpos() 147
Sustitución de subcadenas: str_replace(), substr_replace() 148
Introducción a las expresiones regulares 149
Los fundamentos 150
Conjuntos y clases de caracteres 150
Repetición 152
Subexpresiones 152
Recuento de subexpresiones 152
Anclajes al principio o al final de una cadena 152
Bifurcación 153
Cómo buscar coincidencias de caracteres especiales 153
Resumen de los caracteres especiales 153
Uso de estos elementos en Smart Form 154
Búsqueda de subcadenas con expresiones regulares 155
Sustitución de cadenas con expresiones regulares 156
División de cadenas con expresiones regulares 156
Comparación de funciones de cadenas y funciones de expresiones regulares 157
Lecturas adicionales 157
A continuación 157

5. Reutilización de código y creación de funciones 158

¿Por qué reutilizar código? 160
Costes 160
Legibilidad 160

Uniformidad 160
Uso de require() e include() 161
Uso de require() 161
Extensiones de nombre de archivo y require() 162
Etiquetas de PHP y require() 162
Uso de require() para plantillas de sitios Web 163
Uso de auto_prepend_file y auto_append_file 167
Uso de include() 168
Uso de las funciones de PHP 170
Llamada de funciones 170
Cómo llamar a una función no definida 171
Funciones y el uso de mayúsculas y minúsculas 172
¿Por qué debería crear funciones propias? 173
Estructura básica de una función 173
Designación de funciones 174
Parámetros 175
Ámbito 177
Llamadas por referencia frente a llamadas por valor 179
Final de las funciones 180
Cómo devolver valores desde funciones 182
Bloques de código 182
Recursión 183
Lecturas adicionales 185
A continuación 185

6. PHP orientado a objetos 186

Conceptos orientados a objetos 187
Clases y objetos 188
Polimorfismo 189
Herencia 189
Creación de clases, atributos y operaciones en PHP 190
Estructura de una clase 190
Constructores 191
Instanciación 191
Uso de los atributos de clase 192
Llamadas a operaciones de clase 194
Implementación de la herencia en PHP 195
Reemplazamientos 196
Herencia múltiple 197
Diseño de clases 198
Escritura del código para nuestra clase 199
A continuación 207

Parte II. Uso de MySQL 209

7. Diseño de la base de datos Web 210

Conceptos de base de datos relacionales 212
Tablas 212
Columnas 212
Filas 213
Valores 213
Claves 213
Esquemas 214
Relaciones 214
Cómo diseñar nuestra base de datos Web 215
Piense en los objetos del mundo real que está modelando 215
Cómo evitar el almacenamiento de datos redundantes 216
Uso de valores de columna únicos 217
Selección de claves lógicas 218
Reflexiones sobre las preguntas que desea formular a la base de datos 218
Evite diseños con varios atributos vacíos 219
Resumen de los tipos de tablas 219
Arquitectura de base de datos Web 220
Arquitectura 220
Lecturas adicionales 221
A continuación 221

8. Creación de la base de datos Web 222

Nota sobre el uso del monitor de MySQL 224
Cómo registrarse en MySQL 225
Creación de bases de datos y usuarios 226
Creación de la base de datos 226
Usuarios y privilegios 227
Introducción al sistema de privilegios de MySQL 227
Principio de asignación del privilegio más bajo 227
Cómo configurar usuarios: el comando GRANT 228
Tipos y niveles de privilegio 229
El comando REVOKE 231
Ejemplos de uso de GRANT y REVOKE 232
Cómo configurar un usuario para la Web 233
Cómo cerrar la sesión como administrador 233
Uso de la base de datos correcta 233
Creación de tablas de base de datos 234
¿Cuál es el significado del resto de las palabras clave? 235
Tipos de columna 236
Cómo examinar la base de datos con SHOW y DESCRIBE 238
Identificadores de MySQL 239
Tipos de dato de columna 240
Tipos numéricos 241

Tipos de fecha y hora 242
Tipos de cadena 244
Lecturas adicionales 245
A continuación 246

9. Cómo trabajar con la base de datos de MySQL 248

¿Qué es SQL? 250
Inserción de datos en la base de datos 250
Recuperación de datos de la base de datos 252
Recuperación de datos con criterios específicos 253
Recuperación de datos desde varias tablas 256
Combinaciones sencillas de dos tablas 256
Combinación de varias tablas 257
Búsqueda de filas que no coincidan 258
Uso de otros nombres para designar tablas: los alias 260
Resumen de los tipos de combinación 260
Recuperación de datos con un orden dado 261
Agrupación y agregación de datos 262
Cómo escoger las filas que recuperar 264
Actualización de registros en la base de datos 264
Alteración de tablas tras su creación 265
Eliminación de registros de la base de datos 267
Eliminación de tablas 267
Eliminación de una base de datos entera 267
Lecturas adicionales 268
A continuación 268

10. Acceso a la base de datos de MySQL desde la Web con PHP 270

Funcionamiento de las arquitecturas de base de datos 272
Pasos básicos para consultar una base de datos desde la Web 275
Comprobación y filtrado de datos entrantes 275
Cómo configurar de una conexión 277
Selección de una base de datos 278
Cómo consultar la base de datos 279
Recuperación de resultados de consulta 279
Desconexión de una base de datos 281
Cómo colocar nueva información en la base de datos 281
Otras funciones útiles PHP y MySQL 284
Liberación de recursos 285
Creación y eliminación de bases de datos 285
Otras interfaces de base de datos y PHP 285
Uso de una interfaz de base de datos genérica: PEAR DB 286
Lecturas adicionales 288
A continuación 289

11. MySQL Avanzado 290

Análisis detallado del sistema de privilegios 291
La tabla de usuario 292
Las tablas db y host 294
Las tablas tables_priv y columns_priv 295
Control de acceso: ¿cómo utiliza MySQL las tablas de concesión de privilegios? 296
Actualización de privilegios: ¿cuándo surten efecto los cambios? 297
Cómo proteger la bases de datos MySQL 297
MySQL desde el punto de vista del sistema operativo 297
Contraseñas 298
Privilegios de usuario 299
Aspectos relacionados con la Web 299
Cómo obtener más información sobre bases de datos 300
Cómo obtener información con SHOW 300
Cómo obtener información con DESCRIBE 302
Compresión del funcionamiento de las consultas con EXPLAIN 303
Cómo agilizar consultas con índices 306
Trucos de optimización general 307
Optimización del diseño 307
Permisos 307
Optimización de tablas 307
Uso de índices 308
Uso de valores predeterminados 308
Uso de conexione permanentes 308
Otras sugerencias 308
Tipos diferentes de tabla 308
Carga de datos desde un archivo 309
Cómo realizar una copia de seguridad de la base de datos MySQL 310
Restablecimiento de la base de datos MySQL 310
Lecturas adicionales 311
A continuación 311

Parte III. Comercio electrónico y seguridad 313

12. Creación de un sitio Web de comercio electrónico 314

¿Cuál es nuestro objetivo? 315
Tipos de sitios Web comerciales 316
Medios publicitarios en línea 316
No suministrar información importante 317
Mala presentación 317
No responder a la información generada por el sitio Web 317
No actualizar el sitio 318

No realizar el seguimiento del éxito del sitio 318
Recogida de pedidos de artículos y servicios 319
Preguntas sin respuesta 321
Confianza 321
Facilidad de uso 322
Compatibilidad 323
Suministro de servicios y artículos digitales 323
Cómo añadir valor a los artículos y servicios 324
Recorte de costes 324
Riesgos y amenazas 325
Piratas informáticos 325
Fracaso en la atracción de suficiente negocio 326
Fallos de hardware 326
Fallos de alimentación, comunicación, red y distribución 327
Competencia 327
Errores de software 327
Cambios en las políticas e impuestos gubernamentales 328
Límites de la capacidad del sistema 328
Por qué estrategia optar 328
A continuación 329

13. Aspectos de seguridad relacionados con el comercio electrónico . 330

Importancia de la información 332
Amenazas contra la seguridad 332
Exposición de datos confidenciales 333
Pérdida o destrucción de datos 335
Modificación de los datos 335
Errores en el software 336
Malas especificaciones técnicas 337
Suposiciones erróneas hechas por los desarrolladores 337
Pruebas incompletas 337
Repudio 338
Equilibrio entre usabilidad, rendimiento, coste y seguridad 339
Creación de una política de seguridad 340
Principios de autenticación 340
Uso de la autenticación 341
Fundamentos de la encriptación 342
Encriptación de clave privada 343
Encriptación de clave pública 344
Firmas digitales 344
Certificados digitales 345
Servidores Web seguros 347
Auditorías y registros 348
Cortafuegos 349

Copia de seguridad de los datos 349
Copia de seguridad de los archivos generales 350
Copia de seguridad y restauración de bases de datos MySQL 350
Seguridad física 350
A continuación 351

14. Implementación de la autenticación con PHP y MySQL 352

Identificación de visitantes 353
Implementación del control de acceso 355
Almacenamiento de contraseñas 357
Cifrado de contraseñas 360
Protección de páginas múltiples 361
Autenticación básica 362
Uso de la autenticación básica en PHP 363
Uso de la autenticación básica con los archivos .htaccess de Apache 365
Autenticación básica con IIS 368
Uso de la autenticación mod_auth_mysql 371
Instalación de mod_auth_mysql 371
¿Funcionó? 372
Uso de mod_auth_mysql 372
Creación de su propio sistema de autenticación 373
Lecturas adicionales 374
A continuación 374

15. Implementación de transacciones seguras con PHP y MySQL 376

Cómo suministrar transacciones seguras 377
El equipo del usuario 378
Internet 380
Su sistema 381
Uso SSL 382
Filtrado de la entrada de los usuarios 385
Suministro de un almacenamiento seguro 386
¿Por qué almacenar números de tarjeta de crédito? 387
Uso de encriptación en PHP 388
Instalación de GPG 388
Cómo probar GPG 391
Lecturas adicionales 396
A continuación 396

Parte IV. Técnicas avanzadas de PHP 397

16. Interacción con el sistema de archivos y el servidor 398

Introducción a la carga de archivos 400

HTML para la carga de archivos ... 400
Un inciso sobre seguridad ... 401
Código PHP para procesar la tarea de carga del archivo ... 402
Problemas habituales ... 407
Uso de las funciones de directorio ... 407
Lectura desde directorios ... 408
Cómo obtener información sobre el directorio actual ... 409
Creación y eliminación de directorios ... 409
Interacción con el sistema de archivos ... 410
Cómo obtener información de archivo ... 410
Cómo cambiar las propiedades de archivo ... 413
Creación, eliminación y desplazamiento de archivos ... 413
Uso de funciones de ejecución de programas ... 414
Interacción con el entorno: getenv() y putenv() ... 417
Lecturas adicionales ... 418
A continuación ... 418

17. Uso de funciones de red y de protocolo ... 420

Descripción general de protocolos ... 421
Envío y recepción de correos electrónicos ... 422
Uso de otro sitio Web ... 423
Uso de funciones de búsqueda de red ... 426
Uso de FTP ... 430
Uso de FTP para realizar una copia o reproducir un archivo ... 430
Conexión al servidor FTP remoto ... 433
Inicio de sesión en el servidor FTP ... 433
Comprobación del momento temporal de la actualización de archivos ... 434
Descarga del archivo ... 435
Cierre de la conexión ... 436
Carga de archivos ... 436
Cómo evitar los tiempos de espera ... 437
Uso de otras funciones de FTP ... 437
Comunicaciones de red genéricas con cURL ... 438
Lecturas adicionales ... 440
A continuación ... 441

18. Administración de la fecha y la hora ... 442

Cómo obtener la fecha y la hora en PHP ... 443
Uso de la función date() ... 443
Marcas de tiempo de Unix ... 445
Uso de la función getdate() ... 446
Validación de fechas ... 447
Conversión entre formatos de fecha de PHP y MySQL ... 447

Cálculos de fecha 449
Uso de funciones de calendario 450
Lecturas adicionales 451
A continuación 451

19. Creación de imágenes 452

Configuración de compatibilidad de imágenes en PHP 453
Formatos de imagen 454
JPEG 455
PNG 455
WBMP 455
GIF 455
Creación de imágenes 456
Creación de un lienzo 458
Dibujo o impresión de texto en la imagen 458
Generación del gráfico final 460
Liberación de recursos 461
Uso de imágenes generadas automáticamente en otras páginas 462
Uso de texto y fuentes para crear imágenes 463
Lienzo base 466
Cómo ajustar el texto sobre el botón 466
Colocación del texto 470
Cómo escribir el texto en el botón 470
Para terminar 471
Dibujo de figuras y representación gráfica de datos 471
Otras funciones de imagen 478
Lecturas adicionales 479
A continuación 479

20. Uso del control de sesión en PHP 480

Qué es el control de sesión 481
Funcionalidad de sesión básica 482
Qué es una cookie 482
Configuración de cookies desde HTTP 483
Uso de cookies con sesiones 483
Almacenamiento del Id. de sesión 484
Implementación de sesiones simples 484
Inicio de sesión 484
Registro de variables de sesión 485
Uso de variables de sesión 485
Anulación del registro de variables y eliminación de la sesión 486
Ejemplo de sesión sencilla 487
Configuración del control de sesión 489
Implementación de autenticación con el control de sesión 490

Lecturas adicionales 496
A continuación 497

21. Otras características útiles 498

Uso de comillas mágicas 499
Evaluación de cadenas: eval() 500
Finalización de ejecución: die y exit 501
Serialización 502
Cómo obtener información sobre el entorno PHP 503
Cómo saber qué extensiones se han cargado 503
Identificación del propietario de la secuencia de comandos 504
Cómo saber cuándo se ha modificado una secuencia de comandos 504
Carga dinámica de extensiones 504
Modificación temporal del entorno de ejecución 505
Cómo resaltar fuentes 506
A continuación 506

Parte V. Creación de proyectos PHP y MySQL prácticos 509

22. Uso de PHP y MySQL en grandes proyectos 510

Aplicación de ingeniería de software al desarrollo Web 512
Planificación y ejecución de un proyecto de aplicación Web 513
Reutilización de código 514
Cómo escribir código mantenible 514
Estándares de codificación 515
Convenciones de nomenclatura 515
Comentarios de código 516
Sangrados 517
División del código 518
Uso de una estructura de directorios estándar 518
Cómo documentar y compartir funciones internas 519
Implementación del control de versión 519
Selección de un entorno de desarrollo 520
Documentación de nuestros proyectos 521
Prototipos 521
Separación de lógica y contenido 522
Optimización de código 523
Uso de optimización sencilla 523
Uso de productos Zend 524
Comprobaciones 524
Lecturas adicionales 526
A continuación 526

23. Depuración 528

Errores de programación 530
Errores sintácticos 530
Errores de ejecución 531
Llamadas a funciones que no existen 533
Lectura o escritura de archivos 533
Interacción con MySQL u otras bases de datos 534
Conexiones a redes de servicios 535
Comprobación de entrada de datos incorrecta 536
Errores lógicos 537
Ayuda de depuración de variables 538
Niveles de informes de errores 540
Modificación de los parámetros de informes de error 541
Cómo desencadenar nuestros propios errores 543
Procesamiento correcto de errores 543
A continuación 546

24. Cómo generar autenticación y personalización de usuarios 548

El problema 550
Componentes de la solución 550
Identificación y personalización de usuarios 550
Almacenamiento de marcadores 551
Recomendación de marcadores 551
Presentación de la solución 551
Implementación de la base de datos 554
Implementación del sitio básico 555
Implementación de la autenticación de usuarios 558
Registro 558
Conexión 563
Desconexión 567
Modificación de contraseñas 568
Restablecimiento de contraseñas olvidadas 570
Implementación del almacenamiento y recuperación de marcadores 575
Cómo añadir marcadores 575
Cómo mostrar marcadores 578
Eliminación de marcadores 579
Implementación de recomendaciones 581
Conclusión y posibles ampliaciones 585
A continuación 585

25. Construcción de un carro de la compra 586

El problema 588
Componentes de la solución 588

Generación de un catálogo en línea ... 588
Cómo realizar el seguimiento de las compras de un usuario mientras compra ... 588
Pagos ... 589
Interfaz de administración ... 589
Presentación de la solución ... 590
Implementación de la base de datos ... 594
Implementación del catálogo en línea ... 596
Enumeración de categorías ... 598
Enumeración de libros en una categoría ... 601
Cómo mostrar detalles sobre un libro ... 602
Implementación del carro de la compra ... 604
Uso de la secuencia de comandos show_cart.php ... 604
Cómo ver el carro ... 607
Cómo añadir artículos al carro ... 610
Cómo guardar el carro actualizado ... 611
Impresión de un resumen en la barra de encabezado ... 612
Cómo salir ... 613
Implementación del pago ... 618
Implementación de una interfaz de administración ... 620
Ampliación del proyecto ... 628
Uso de un sistema existente ... 628
A continuación ... 628

26. Creación de un sistema de administración de contenidos ... 630

El problema ... 631
Requisitos de la solución ... 632
Modificación del contenido ... 632
Introducción de contenido en el sistema ... 632
FTP ... 632
Método de carga de archivos ... 633
Cambios en línea ... 633
Bases de datos frente a almacenamiento en archivos ... 633
Estructura de documentos ... 634
Uso de metadatos ... 635
Formato del resultado ... 635
Manipulación de imágenes ... 637
Diseño y presentación de la solución ... 639
Diseño de la base de datos ... 640
Implementación ... 642
Interfaz de usuario ... 642
Sistema ... 646
Búsquedas ... 654
Pantalla del editor ... 657

Ampliación del proyecto 658
A continuación 659

27. Creación de un servicio de correo electrónico basado en la Web ... 660

El problema 661
Componentes de la solución 662
Presentación de la solución 663
Configuración de la base de datos 665
Arquitectura de secuencias de comandos 667
Conexión y desconexión 673
Configuración de cuentas 676
Creación de una nueva cuenta 678
Modificación de una cuenta existente 680
Eliminación de una cuenta 680
Lectura de correo 681
Selección de una cuenta 681
Cómo ver los contenidos del buzón de correo 684
Lectura de un mensaje de correo 687
Cómo ver encabezados de mensaje 690
Eliminación de correo 691
Envío de correo 692
Envío de un nuevo mensaje 692
Cómo responder o reenviar correo 694
Ampliación del proyecto 695
A continuación 696

28. Creación de un gestor de listas de correo 698

El problema 699
Componentes de la solución 700
Definición de una base de datos de listas y suscriptores 700
Carga de archivos 701
Envío de correo con archivos adjuntos 701
Presentación de la solución 702
Definición de la base de datos 704
Arquitectura de la secuencia de comandos 706
Implementación del inicio de sesión 714
Creación de una nueva cuenta 715
Conexión 717
Implementación de funciones de usuario 720
Cómo ver las listas 720
Cómo ver información de listas 725
Cómo ver archivos de listas 727
Suscripciones y anulación de suscripciones 728
Modificación de la configuración de una cuenta 729

Modificación de contraseñas 730
Desconexión 732
Implementación de funciones administrativas 732
Creación de una nueva lista 733
Carga de un nuevo boletín informativo 735
Procesamiento de la carga de varios archivos 737
Vista previa del boletín informativo 742
Envío del mensaje 743
Ampliación del proyecto 749
A continuación 749

29. Creación de foros Web 750

El problema 752
Componentes de la solución 752
Presentación de la solución 754
Diseño de la base de datos 755
Cómo ver el árbol de artículos 758
Despliegue y repliegue 760
Cómo mostrar los artículos 763
Uso de la clase treenode 764
Cómo ver artículos individuales 770
Cómo añadir nuevos artículos 772
Ampliaciones 778
Uso de un sistema existente 779
A continuación 779

30. Generación de documentos personalizados en PDF 780

El problema 781
Evaluación de formatos de documento 782
Papel 782
ASCII 783
HTML 783
Formatos de procesadores de texto 783
Formato de texto enriquecido 784
PostScript 785
Formato de documento portable 785
Componentes de la solución 786
Sistema de preguntas y respuestas 786
Software de generación de documentos 787
Software para crear una plantilla RTF 787
Software para crear una plantilla PDF 787
Software para crear PDF mediante programación 788
Presentación de la solución 788
Cómo responder a las preguntas 790

Cómo calificar las respuestas 792
Generación de un certificado RTF 794
Generación de un certificado PDF a partir de una plantilla 797
Generación de un documento PDF por medio de PDFlib 801
Una secuencia de comandos Hello World para PDFlib 801
Generación de nuestro certificado con PDFlib 805
Problemas con los encabezados 812
Ampliación del proyecto 813
Lecturas adicionales 813

31. Conexión a servicios Web con XML y SOAP 814

El problema 815
Cómo entender XML 816
Servicios Web 820
SOAP 820
WSDL 821
Componentes de la solución 822
Creación de un carro de la compra 822
Uso de las interfaces de servicios Web de Amazon 822
Análisis de XML 823
Uso de SOAP con PHP 823
Almacenamiento en caché 824
Presentación de la solución 824
Aplicación principal 828
Cómo mostrar los libros de una categoría 834
Obtención de un AmazonResultSet 836
Uso de XML sobre HTTP 844
Uso de SOAP 850
Almacenamiento de datos en caché 852
Creación del carro de la compra 854
Cómo salir a Amazon 857
Instalación del código del proyecto 858
Ampliación del proyecto 859
Lecturas adicionales 859

Parte VI. Apéndices 861

A. Instalación de PHP y MySQL 862

Ejecución de PHP como intérprete CGI o como módulo 864
Instalación de Apache, PHP y MySQL en Unix 864
Instalación binaria 865
Instalación desde código fuente 865
Instalación de MySQL 866
Instalación de PDFlib 868

Instalación de cURL 869
Instalación de PHP 869
Fragmentos de código de httpd.conf 872
¿Funciona la compatibilidad con PHP? 873
¿Funciona SSL? 874
Últimos pasos 875
Instalación de Apache, PHP y MySQL en Windows 875
Instalación de MySQL en Windows 876
Definición de la ruta 878
Eliminación del usuario anónimo 878
Definición de la contraseña raíz 879
Instalación de Apache en Windows 879
Instalación de PHP en Windows 881
Uso de InstallShield 881
Instalación manual 881
Cómo añadir PHP a su configuración de Apache 883
Cómo añadir PHP y MySQL a Microsoft IIS y PWS 883
Pruebas 884
Instalación de PEAR 884
Configuraciones adicionales 885

B. Recursos Web 886

Recursos PHP 887
Recursos específicos de MySQL y SQL 889
Recursos para Apache 889
Desarrollo Web 890

C. Contenido del CD-ROM 892

Contenido del CD-ROM 893
Windows 893
Linux / Unix 894
Léalo antes de abrirlo 894

Índice alfabético 895

Bienvenidos a Desarrollo Web con PHP y MySQL. Estas páginas recogen los conocimientos fruto de nuestras experiencias con PHP y MySQL, dos de las herramientas de desarrollo Web más actuales.

En esta introducción, trataremos los siguientes aspectos:

- Razones para leer este libro
- Objetivos que se pueden alcanzar utilizando este libro
- Qué es PHP y MySQL y por qué son unas herramientas geniales
- Una repaso general a las últimas funciones de PHP 4
- Cómo está organizado este libro

Empecemos.

Razones para leer este libro

En este libro se explica cómo crear sitios Web interactivos, desde los formularios de pedidos más sencillos hasta los sitios de comercio electrónico más complejos y seguros, y lo que es más, con ayuda de tecnologías de código abierto.

Este libro va dirigido a lectores con un conocimiento básico de HMTL y con cierta experiencia de programación con lenguajes actuales, aunque no es necesario

que hayan programado para Internet o utilizado una base de datos relacional. Los lectores con menos experiencia en programación también podrán aprovechar este libro aunque tardarán un poco más en digerir sus conceptos. Hemos intentado tratar los conceptos básicos, aunque se aborden rápidamente. Este libro va dirigido a personas que deseen dominar PHP y MySQL para crear un sitio Web comercial o de gran tamaño. Puede que tenga experiencia con otros lenguajes de desarrollo Web. Si así fuera, este libro le ayudará a ponerse al día rápidamente.

La razón que nos ha animado a escribir este libro es que todos los manuales sobre PHP que conocíamos eran básicamente guías de referencia de funciones. Estos libros resultan útiles pero no sirven para mucho cuando nuestro jefe o un cliente nos piden crear un carro de la compra para un sitio Web. Hemos intentado que los ejemplos resulten útiles. De hecho, gran parte de su código se puede utilizar directamente en un sitio Web o bastaría con pequeñas modificaciones para hacerlo.

Objetivos que puede alcanzar con este libro

La lectura de este libro le permitirá crear sitios Web reales y dinámicos. Si ha desarrollado sitios Web utilizando simple HTML, ya conocerá las limitaciones de este enfoque. El contenido estático de los sitios desarrollados únicamente con HTML es exactamente eso, estático. Estos sitios no varían a menos que se actualicen físicamente. Los usuarios no pueden interactuar con el sitio de forma significativa.

El uso de un lenguaje como PHP y una base de datos como MySQL permite crear sitios Web dinámicos, es decir, susceptibles de personalización y dotados de información en tiempo real.

En este libro nos hemos centrado deliberadamente en aplicaciones del mundo real, incluso en los capítulos introductorios. Comenzamos por analizar un sencillo sistema de pedidos en línea y aprovecharemos para examinar los distintos componentes de PHP y MySQL.

Abordaremos aspectos relacionados con el comercio electrónico y la seguridad mientras desarrollamos un sitio Web real y le mostraremos cómo implementarlos en PHP y MySQL.

En la sección final del libro, comentaremos cómo abordar proyectos del mundo real y le guiaremos a través del diseño, planificación y construcción de los siguientes ocho proyectos:

- Autenticación de usuario y personalización
- Carros de la compra
- Sistemas de administración de contenido
- Correo electrónico basado en la Web
- Administradores de listas de correo
- Foros Web

- Generación de documentos.
- Conexión a servicios Web con XML.

Puede utilizar estos proyectos directamente o modificarlos para ajustarlos a sus necesidades. Se han seleccionado porque creemos que representan las ocho aplicaciones Web más comunes desarrolladas por programadores. Si sus necesidades son distintas, este libro debería servirle de ayuda para lograr sus objetivos.

Concepto de PHP

PHP es un lenguaje de secuencia de comandos de servidor diseñado específicamente para la Web. Dentro de una página Web puede incrustar código PHP que se ejecutará cada vez que se visite una página. El código PHP es interpretado en el servidor Web y genera código HTML y otro contenido que el visitante verá.

PHP fue concebido en 1994 y es fruto del trabajo de un hombre, Rasmus Lerdorf. Ha sido adoptado por otras personas de talento y ha experimentado tres trasformaciones importantes hasta convertirse en el producto actual. En octubre de 2002, era utilizado por más de nueve millones de dominios de todo el mundo y su número crece rápidamente. Si desea conocer el número actual de sitios que utilizan este lenguaje, visite el sitio `http://www.php.net/usage.php`.

PHP es un producto de código abierto, lo que quiere decir que puede acceder a su código. Puede utilizarlo, modificarlo y redistribuirlo sin coste alguno.

Las siglas PHP equivalían inicialmente a Personal Home Page (Página de inicio personal) pero se modificaron de acuerdo con la convención de designación de GNU (del inglés, Gnu´s Not Unix, Gnu no es Unix) y ahora equivale a PHP Hipertext Preprocessor (Preprocesador de hipertexto PHP).

En la actualidad, PHP está en su versión 4. Esta versión incorpora mejoras importantes al lenguaje que comentaremos en la siguiente sección.

La dirección Web de la página de PHP está disponible es http://www.php.net.

La dirección de la página de Zend (la compañía cuyos fundadores diseñaron PHP4) se encuentra en http://www.zend.com.

Novedades de la versión 4.3 de PHP

Si ha utilizado PHP anteriormente, observará una serie de mejoras importantes en la versión 4.3.

- La mayor parte de las funciones de entrada y salida utilizan ahora un enfoque de secuencias unificado, lo que significa que se pueden abrir archivos fácilmente, así como conexiones HTTP, HTTPS y FTP.

- PEAR ya ha salido de su versión beta y dispone de un instalador sencillo.
- La biblioteca gráfica GD se ha incorporado a PHP.
- La compatibilidad con Apache se sigue considerando como experimental, pero mejora constantemente.
- Un ejecutable pensado para su uso con secuencias de línea de comandos que se incorpora de manera predeterminada al instalar PHP.

Concepto de MySQL

MySQL es un sistema para la administración de bases de datos relacional (RDBMS) rápido y sólido. Las bases de datos permiten almacenar, buscar, ordenar y recuperar datos de forma eficiente. El servidor de MySQL controla el acceso a los datos para garantizar el uso simultáneo de varios usuarios, para proporcionar acceso a dichos datos y para asegurarse de que sólo obtienen acceso a ellos los usuarios con autorización. Por lo tanto, MySQL es un servidor multiusuario y de subprocesamiento múltiple. Utiliza SQL (del inglés Structured Query Language, Lenguaje de consulta estructurado), el lenguaje estándar para la consulta de bases de datos utilizado en todo el mundo. MySQL lleva disponible desde 1996 pero su nacimiento se remonta a 1979. Ha obtenido el galardón Choice Award del Linux Journal Readers en varias ocasiones.

MySQL se distribuye bajo una licencia de código abierto en la actualidad, pero también existen licencias comerciales.

Razones para utilizar PHP y MySQL

Al desarrollar un sitio de comercio electrónico, se pueden utilizar una gran cantidad de productos diferentes:

- Hardware para el servidor Web
- Un sistema operativo
- Software de servidor Web
- Un sistema de administración de base de datos
- Un lenguaje de secuencia de comandos o de programación

Algunas de estas opciones dependen de otras. Por ejemplo, no todos los sistemas operativos se ejecutan sobre todo el hardware ni todos los lenguajes de secuencia de comandos se pueden conectar a bases de datos, etc.

En este libro no vamos a prestar mucha atención al hardware, ni a los sistemas operativos ni al software de servidor Web. No será necesario. Una de las ventajas

de PHP es que está disponible para Microsoft Windows, para muchas versiones de Unix y para cualquier servidor Web completamente funcional. MySQL resulta igualmente versátil.

Para demostrarlo, los ejemplos de este libro se han escrito y probado en dos configuraciones muy utilizadas:

- Linux con el servidor Web Apache
- Microsoft Windows 2000 con Microsoft Internet Information Server (IIS)

Sea cual sea el hardware, sistema operativo y servidor Web que elija, le recomendamos que considere seriamente la opción de utilizar PHP y MySQL.

Alguna de las cualidades de PHP

Entre los competidores principales de PHP se puede citar a Perl, Microsoft Active Server Pages (ASP), Java Server Pages (JSP) y Allaire ColdFusion.

En comparación con estos productos, PHP cuenta con muchas ventajas, entre las que se encuentran las siguientes:

- Alto rendimiento
- Interfaces para una gran cantidad de sistemas de base de datos diferentes
- Bibliotecas incorporadas para muchas tareas Web habituales
- Bajo coste
- Facilidad de aprendizaje y uso
- Portabilidad
- Acceso al código abierto

A continuación se comentan en más detalle estas cualidades.

Rendimiento

PHP es muy eficiente. Mediante el uso de un único servidor, puede servir millones de acceso al día. Los indicadores comparativos de rendimiento publicados por Zend Technologies (http://www.zend.com) muestran que PHP supera ampliamente a sus competidores en esta faceta.

Integración de base de datos

PHP dispone de una conexión propia a todos los sistemas de base de datos. Además de MySQL, puede conectarse directamente a las bases de datos de

PostgreSQL, mSQL, Oracle, dbm, filePro, Hyperwave, Informix, InterBase y Sybase, entre otras.

El uso de ODBC (del inglés Open Database Connectivity Standard, Estándar de conectividad abierta de base de datos) permite establecer una conexión a cualquier base de datos que suministre un controlador ODBC. Entre ellas, se incluyen los productos de Microsoft, y muchos otros.

Bibliotecas incorporadas

Como se ha diseñado para su uso en la Web, PHP incorpora una gran cantidad de funciones integradas para realizar útiles tareas relacionadas con la Web. Puede generar imágenes GIF al instante, establecer conexiones a otros servicios de red, enviar correos electrónicos, trabajar con cookies y generar documentos PDF, todo con unas pocas líneas de código.

Coste

PHP es gratuito. Puede descargar la última versión de http://www.php.net cuando lo desee sin coste alguno.

Aprendizaje de PHP

La sintaxis de PHP se basa en otros lenguajes de programación, principalmente en C y Perl. Si ya conoce C o Perl, o un lenguaje de tipo C como C++ o Java, no tardará nada en utilizar PHP de manera productiva.

Portabilidad

PHP está disponible para una gran cantidad de sistemas operativos diferentes. Puede escribir código PHP en todos los sistemas operativos gratuitos del tipo Unix, como Linux y FreeBSD, versiones comerciales de Unix, como Solaris y IRIX o en las diferentes versiones de Microsoft Windows.

Su código funcionará sin necesidad de aplicar ninguna modificación a los diferentes sistemas que ejecute PHP.

Código fuente

Dispone de acceso al código fuente de PHP. A diferencia de los productos comerciales y de código cerrado, si desea modificar algo o agregar un elemento al programa, puede hacerlo con total libertad.

No necesitara esperar a que el fabricante publique parches, ni tendrá que preocuparse porque el fabricante cierre sus puertas o decida abandonar el producto.

Algunas de las ventajas de MySQL

Entre los competidores principales de MySQL, se puede citar a PostgreSQL, Microsoft SQL Server y Oracle.

MySQL cuenta con muchas ventajas, entre las que se encuentran las siguientes:

- Alto rendimiento
- Bajo coste
- Facilidad de configuración y aprendizaje
- Portabilidad
- Accesibilidad a código fuente

A continuación se comentan en más detalle estas cualidades.

Rendimiento

MySQL es muy rápido. Si lo desea, puede consultar la página de indicadores comparativos de sus desarrolladores en el sitio Web mysql.com. Estos indicadores revelan en muchos casos una diferencia de velocidad abismal con respecto a los productos de la competencia.

Bajo coste

MySQL está disponible de manera gratuita, bajo una licencia de código abierto, o por un precio reducido en forma de licencia comercial si resultara necesario para su aplicación.

Facilidad de uso

Las bases de datos más modernas utilizan SQL. Si ha utilizado otros RDBMS, no debería tener problemas para adaptarse a este sistema. MySQL resulta además más sencillo de configurar que otros productos similares.

Portabilidad

MySQL se puede utilizar en una gran cantidad de sistemas Unix diferentes así como bajo Microsoft Windows.

Código fuente

Como en el caso de PHP, puede obtener y modificar el código fuente de MySQL.

Cómo está organizado este libro

Este libro se divide en cinco partes principales.

La parte I, "Uso de PHP", proporciona una visión general de las partes fundamentales del lenguaje PHP. Los ejemplos escogidos son ejemplos del mundo real utilizados para desarrollar un sitio de comercio electrónico y no código sin utilidad práctica. Esta sección se inicia con el capítulo "Curso acelerado". Si ya tiene experiencia en programación con PHP, puede repasar rápidamente esta sección. Si ha trabajado antes con PHP o acaba de llegar al mundo de la programación, conviene que se detenga un poco más sobre ella.

La parte II, "Uso de MySQL" aborda los conceptos y el diseño implicado en el uso de sistemas de bases de datos relacionales como MySQL, el uso de SQL, la conexión de la base de datos MySQL al mundo con PHP y temas avanzados sobre MySQL, como la seguridad y la optimización.

La parte III, "Comercio electrónico y seguridad" trata alguno de los temas generales implicados en el desarrollo de un sitio de comercio electrónico utilizando cualquier lenguaje. El aspecto más importante es la seguridad. Seguidamente, explicaremos cómo utilizar PHP y MySQL para autenticar usuarios y recoger, transmitir y almacenar datos de manera segura.

La parte IV, "Técnicas avanzadas de PHP", trata de manera detallada alguna de las funciones principales que integra PHP. Se han seleccionado aquellos grupos de funciones que tienen una mayor probabilidad de aparición al crear un sitio de comercio electrónico. Aprenderá a interactuar con el servidor, a interactuar con la red, a generar imágenes, a manipular fechas y horas y variables de sesión.

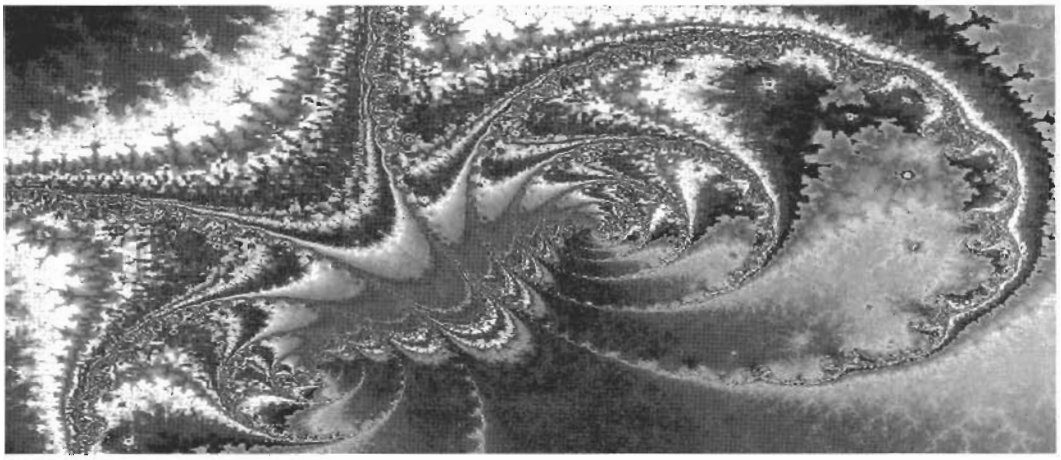
En la parte V, "Construcción de proyectos prácticos con PHP y MySQL", trata temas del mundo real como la gestión de proyectos de gran tamaño y funciones de depuración. Se incluyen proyectos de ejemplo que muestran el potencial y la versatilidad de PHP y MySQL.

Finalmente

Esperamos que disfrute de este libro y que se lo pase tan bien aprendiendo a utilizar PHP y MySQL como lo hicimos nosotros cuando empezamos a utilizar estos productos. Es un placer trabajar con ellos. En poco tiempo, podrá unirse a los miles de desarrolladores que utilizan potentes y robustas herramientas para desarrollar sitios Web dinámicos y en tiempo real.

Parte I

Uso de PHP



1

Curso acelerado de PHP

En este capítulo se realiza un rápido repaso de la sintaxis y de las estructuras de lenguaje de PHP. Si ya programa en PHP, puede utilizar este capítulo para colmar sus lagunas de conocimientos. Si tiene experiencia en programación con C, ASP u otro lenguaje de programación, este capítulo le ayudará a ponerse al día rápidamente.

En este libro aprenderá a utilizar PHP a través de una serie de ejemplos del mundo real, tomados de nuestra experiencia en el desarrollo de sitios de comercio electrónico. Por regla general, los textos sobre programación utilizan ejemplos muy simples para explicar elementos de sintaxis. Nosotros hemos decidido no seguir esta práctica. Nos hemos dado cuenta de que lo que el lector quiere con frecuencia es poder ver un sitio ya creado y operativo, para entender cómo se utiliza el lenguaje, en lugar de otra guía de sintaxis y funciones sin mayor interés que un manual en línea.

Pruebe los ejemplos, introdúzcalos y cárguelos desde el CD-ROM, modifíquelos, divídalos en partes y aprenda a unirlas de nuevo.

En este capítulo comenzaremos por presentar un ejemplo de un formulario de producto en línea para aprender a utilizar las variables, operadores y expresiones en PHP. También analizaremos los tipos de variables y el orden de precedencia de los operadores. Le enseñaremos a acceder a variables de formulario y a manipularlas para calcular los totales y los impuestos correspondientes en un pedido de cliente.

Seguidamente, desarrollaremos un ejemplo de formulario de pedido en línea utilizando nuestra secuencia de comandos de PHP para validar los datos. Examina-

remos el concepto de valores booleanos y presentaremos ejemplos de uso de `if`, `else`, del operador `?:` y de la instrucción `switch`.

Por último, analizaremos el tema de los bucles para lo cual escribiremos código de PHP con el que generar tablas HTML repetitivas. Entre los temas clave que se analizarán están los siguientes:

- Cómo incrustar PHP en HTML
- Cómo agregar contenido dinámico
- Cómo acceder a variables de formulario
- Identificadores
- Variables declaradas por el usuario
- Tipos de variables
- Asignación de valores a variables
- Constantes
- Ámbito de variables
- Operadores y su prioridad
- Expresiones
- Funciones de variables
- Toma de decisiones con `if`, `else` y `switch`

Uso de PHP

Para realizar los ejemplos de ese capítulo, así como los del resto del libro, necesitará disponer de acceso a un servidor Web con PHP instalado. Para sacar el máximo partido a los ejemplos y casos prácticos que se presentarán, debería ejecutarlos e intentar modificarlos. Para ello, necesitará un banco de pruebas sobre el que experimentar.

Si no tiene PHP en su equipo, deberá instalarlo o pedirle a su administrador del sistema que lo haga en su lugar. En el apéndice A encontrará las instrucciones y en el CD-ROM se incluye todo lo necesario para instalar PHP en UNIX y Windows NT.

Aplicación de ejemplo: Bob's Auto Parts

Una de las aplicaciones más comunes de cualquier lenguaje de secuencia de comandos de servidor es el procesamiento de formularios Web. Empezaremos nuestro estudio de PHP implementando un formulario para Bob's Auto Parts, una compañía

de repuestos de coches. El código utilizado para este ejemplo se incluye en el directorio correspondiente del CD.

El formulario de pedidos

Por el momento, el programador de HTML de la empresa de Bob ha creado un formulario de pedido para los repuestos que vende la empresa. En la figura 1.1 se ilustra el aspecto del formulario. Resulta bastante sencillo y es muy parecido a los que se pueden encontrar en Internet. Lo primero que Bob quiere saber es qué pide el cliente, obtener el total del pedido e incluir los impuestos que se deben pagar.

En el listado 1.1 se recoge parte del código HTML asociado a esta página. En este código debe fijarse en dos cosas importantes.

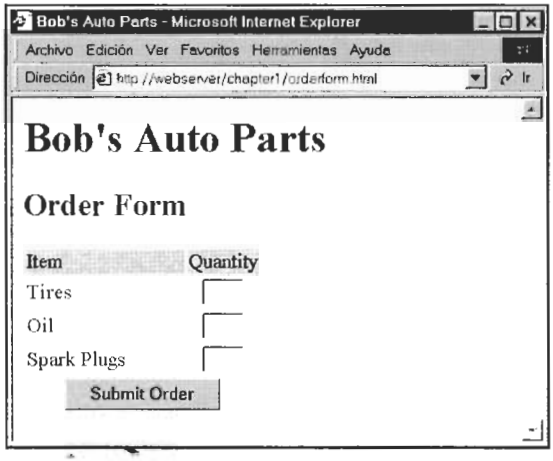


Figura 1.1. El formulario inicial de pedido de Bob sólo registra productos y cantidades

Listado 1.1. orderform.html. Código HTML correspondiente al formulario básico de pedido de Bob

```
<form action="processorder.php" method=post>
 <table  border=0>
 <tr  bgcolor=#cccccc>
  <td  width=150>Item</td>
  <td  width=15>Quantity</td>
 </tr>
 <tr>
  <td>Tires</td>
  <td  align="center"><input  type="text"  name="tireqty"  size="3"
      maxlength="3"></td>
 </tr>
 <tr>
  <td>Oil</td>
```

```
   <td  align="center"><input  type="text"  name="oilqty"  size="3"
maxlength="3"></td>
  </tr>
 <tr>
   <td>Spark Plugs</td>
   <td align="center"><input type="text" name="sparkqty" size="3"
maxlength="3"></td>
 </tr>
 <tr>
   <td colspan="2" align="center"><input type="submit" value="Submit
Order"></td>
 </tr>
 </table>
 </form>
```

En primer lugar, hemos establecido la acción del formulario en el nombre de la secuencia de comandos PHP que procesará el pedido del cliente. (Escribiremos esta secuencia de comandos a continuación.) Por regla general, el valor del atributo `ACTION` es el URL que se cargará cuando el usuario pulse el botón de envío. Los datos indicados por el usuario en el formulario se enviarán a este URL a través del método especificado en el atributo `METHOD`, ya sea `GET` (ajuntado al final del URL) o `POST` (enviado como un paquete diferente).

En segundo lugar, debería fijarse en los nombre s de los formularios `tireqty`, `oilqty` y `sparkqty`. Utilizaremos estos nombres de nuevo en nuestra secuencia de comandos de PHP. Por ello, resulta importante asignar nombres significativos a los campos de formulario que sean sencillos de recordar al escribir la secuencia de comandos de PHP. Algunos editores de PHP generan nombres de campo del tipo `campo23` de manera predeterminada. Estos campos resultan difíciles de recordar. Su vida como programador de PHP resultará mucho más sencilla si estos nombres reflejan los datos que se introducen en el campo.

Es aconsejable adoptar un estándar de codificación para nombres de campos de manera que todos los nombres del sitio utilicen el mismo formato. De esta forma le resultará más sencillo recordar, por ejemplo, si abrevió una palabra en el nombre de un campo o si utilizó guiones bajos para representar espacios.

Procesamiento del formulario

Para procesar el formulario, tendremos que crear la secuencia de comandos mencionada en el atributo `ACTION` de la etiqueta `FORM` llamada `processorder.php`. Abra su editor de texto y cree este archivo. Escriba el siguiente código:

```
 <head>
   <title>Bob's  Auto  Parts  -  Order  Results</title>
 </head>
 <body>
 <h1>Bob's Auto Parts</h1>
 <h2>Order Results</h2>
  </body>
  </html>
```

Como observará, todo lo que hemos escrito hasta ahora es simple código HTML. Ha llegado el momento de agregar algo de código PHP a nuestra secuencia de comandos.

Cómo incrustar PHP en HTML

Agregue las siguientes líneas debajo del encabezado <h2>:

```
<?php
   echo '<p>Order processed.</p>';
?>
```

Guarde el archivo y cárguelo en el navegador. Para ello, rellene el formulario de Bob y haga clic sobre el botón **Submit Order**. El resultado debería parecerse al ilustrado en la figura 1.2.

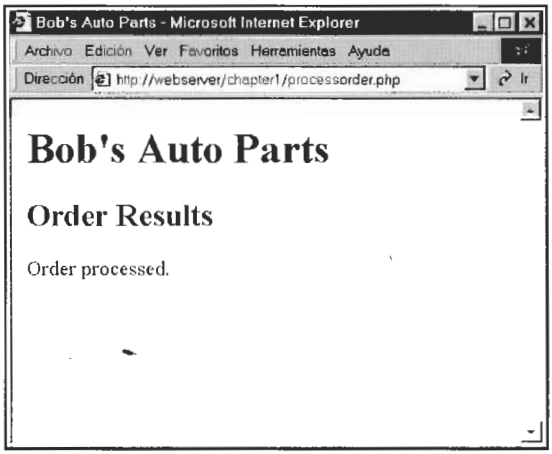


Figura 1.2. El texto pasado a la instrucción echo de PHP se refleja en el navegador

Fíjese en cómo se incrustó el código de PHP que escribimos dentro de un archivo HTML de aspecto normal. Intente visualizar el código fuente en su navegador. Debería mostrarse la siguiente secuencia:

```
<html>
<head>
  <title>Bob's Auto Parts - Order Results</title>
</head>
<body>
<h1>Bob's Auto Parts</h1>
<h2>Order Results</h2>
<p>Order processed.</p></body>
</html>
```

No se ve ninguna secuencia de PHP ya que el intérprete de PHP ha recorrido la secuencia de comandos y la ha sustituido con su resultado. Por lo tanto, se puede generar código HTML limpio y visible a partir de PHP desde cualquier navegador. En otras palabras, el navegador del usuario no necesita entender PHP.

Este hecho ilustra de forma resumida el concepto de secuencias de comandos del lado del servidor. El código PHP se ha interpretado y ejecutado en el servidor Web, lo que difiere de JavaScript y de otras tecnologías de cliente que se interpretan dentro del navegador Web del equipo de un usuario.

El código de archivo se compone de cuatro elementos:

- HTML
- Etiquetas PHP
- Instrucciones PHP
- Espacios en blanco

También podemos agregar

- Comentarios

La mayor parte de las líneas de este ejemplo son sencillo código HTML.

Estilos de etiquetas PHP

En PHP existen cuatro estilos diferentes de etiquetas que podemos utilizar. Los siguientes fragmentos de código son equivalentes:

- Estilo XML

```
<?php  echo  '<p>Order processed.</p>';  ?>
```

  Éste es el tipo de etiqueta que utilizaremos en el libro. Se trata del estilo de etiqueta preferido que usar con PHP 3 y 4. El administrador del servidor no puede desactivarlo lo que garantiza su disponibilidad para todos los servidores. Este estilo de etiqueta se puede utilizar con documentos XML (del inglés Extensible Markup Language, Lenguaje de marcado extensible). Si tiene previsto utilizar XML en su sitio, debería utilizar este estilo de etiqueta.

- Estilo corto

```
<?  echo  '<p>Order  processed.</p>';  ?>
```

  Este estilo de etiqueta es el más simple y sigue el estilo de una instrucción de procesamiento SGML (del inglés Standard Generalized Markup Language, Lenguaje de marcado generalizado estándar). Para utilizar este tipo de etiqueta (la más corta), debe habilitar el parámetro `short_tags` en el archivo de configuración o compilar PHP para que utilice este tipo de etiquetas. En el apéndice A encontrará más información sobre su instalación.

- Estilo SCRIPT

```
<script language='php'> echo '<p>Order processed.</p>'; </script>
```

Este estilo de etiqueta es la más larga y le resultará familiar si conoce JavaScript o VBScript. Puede utilizarla si está usando un editor HTML que dé problemas con otros estilos de etiquetas.

- Estilo ASP

```
<% echo '<p>Order processed.</p>'; %>
```

Este estilo de etiqueta es el que utilizan las Páginas activas de servidor (ASP). Se puede utilizar si ha activado el parámetro `asp_tags`. Puede recurrir a este estilo de etiquetas si está utilizando un editor diseñado para ASP o si ya ha programado en ASP.

Instrucciones de PHP

Para indicarle al intérprete de PHP qué hacer, se introducen instrucciones de PHP entre las etiquetas de cierre y las etiquetas de apertura. En este ejemplo, sólo utilizamos un tipo de instrucción:

```
echo '<p>Order processed.</p>';
```

Como habrá supuesto, la función de la instrucción `echo` es muy sencilla: imprime (o repite) la cadena pasada en el navegador. Si examina la figura 1.2, el resultado devuelto es la cadena "Order processed" en el ventana del navegador.

Como observará, al final de la instrucción `echo` aparece un punto y coma. Este signo se utiliza para separar instrucciones en PHP como si se tratara de un punto al escribir frases en español. Si ha programado antes en C o en Java, estará familiarizado con el uso del punto y coma con dicha función.

Uno de los errores de sintaxis más comunes al programar es olvidarse de introducir los puntos y comas. Sin embargo, resulta igualmente sencillo descubrir el error y corregirlo.

Espacios en blanco

Los caracteres de espaciado como las líneas nuevas (retornos del carro), los espacios y los tabuladores se conocen como espacios en blanco. Como probablemente sepa, los navegadores ignoran los espacios en blanco en HTML. Y otro tanto hace el motor de PHP. Examine los siguientes dos fragmentos de HTML:

```
<h1>Welcome to Bob's Auto Parts!</h1><p>What would you like to order
today?</p>
```

y

```
<h1>Welcome              to Bob's
Auto Parts!</h1>
```

```
<p>What would you like
to order today?</p>
```

Estos dos fragmentos de HTML devuelven el mismo resultado al representarlos en el navegador. Sin embargo, conviene utilizar espacios en blanco en HTML para mejorar la legibilidad del código. Y otro tanto se puede decir de PHP. El uso de espacios en blanco no es obligatorio pero facilita la lectura del código si colocamos cada instrucción en una línea separada. Por ejemplo:

```
echo 'hello ';
echo 'world';
```

y

```
echo 'hello ';echo 'world';
```

son equivalentes, pero la primera versión resulta más sencilla de leer.

Comentarios

Los comentarios son exactamente eso, comentarios. Su función es la de proporcionar indicaciones para aquéllos que leen el código. Los comentarios se pueden utilizar para explicar el objetivo de la secuencia de comandos, quién la escribió, por qué se escribió de esa forma, cuándo se modificó por última vez, etc. Por regla general, todas las secuencias de comandos de PHP incluyen comentarios, salvo las más sencillas.

El intérprete de PHP ignorará cualquier texto de un comentario. En concreto, el analizador de PHP omite los comentarios que equivalen a espacios en blanco.

PHP admite el uso de comentarios del estilo de C, C++ y secuencias de comandos del núcleo.

A continuación se recoge un comentario del estilo de C multilínea que puede aparecer al inicio de una secuencia de comandos de PHP:

```
/* Autor: Bob Smith
   Modificado por última vez: 10 de Abril
   Esta secuencia de comandos procesa pedidos de clientes.
*/
```

Los comentarios multilínea deben empezar y acabar con el símbolo /*. Como ocurre en C, los comentarios en línea no se pueden anidar.

También se pueden utilizar comentarios de una sola línea, bien en estilo de C++:

```
echo '<p>Order processed.</p>'; // Empezar a imprimir el pedido
```

o en estilo de secuencia de comandos del núcleo:

```
echo '<p>Order processed.</p>'; # Empezar a imprimir el pedido
```

En ambos casos, todo lo que venga después del símbolo de comentario (# o //) se considera con un comentario hasta el final de la línea y el final de la etiqueta de PHP, según qué ocurra primero.

Cómo agregar contenido dinámico

Hasta ahora, no hemos utilizado PHP para hacer algo que no pudiésemos conseguir con HTML.

La razón principal para utilizar un leguaje de secuencia de comandos de servidor es suministrar contenido dinámico a los usuarios de un sitio. Se trata de una aplicación importante porque el contenido que cambia en función de las necesidades del usuario o con el tiempo garantiza la vuelta de los visitantes. PHP nos permite realizar esta tarea de manera sencilla.

Vamos a empezar por un ejemplo sencillo. Sustituya el código de PHP del archivo `processorder.php` por el siguiente código:

```
<?php
   echo '<p>Order processed at ';
   echo date('H:i, jS F');
   echo '</p>';
?>
```

En este fragmento, se utiliza la función `date()` integrada en PHP para indicar al cliente la fecha y la ahora a la que procesar el pedido. Esta información será diferente cada vez que se ejecute la secuencia de comandos. En la figura 1.3 se ilustra el resultado de ejecutar la secuencia de comandos en una ocasión:

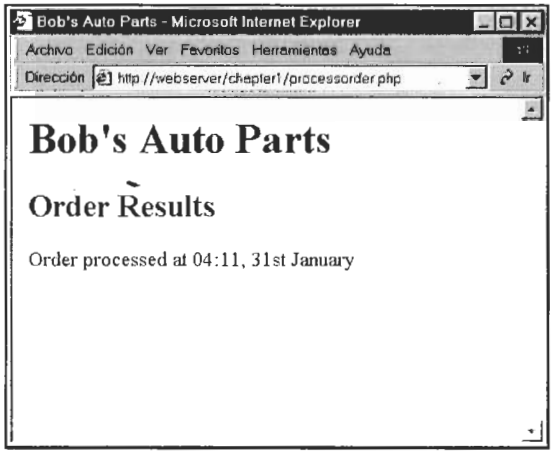


Figura 1.3. La función date() devuelve una cadena de fecha con formato

Llamada de funciones

Examine la llamada a la función `date()`. Se trata de la forma general que adoptan las llamadas de función. PHP incorpora una extensa biblioteca de funciones que puede utilizar para desarrollar aplicaciones Web. La mayor parte de estas funciones pasan y devuelven datos.

Examine la siguiente llamada de función:

```
date('H:i, jS F')
```

Fíjese en que estamos pasando una cadena (datos de texto) a la función dentro de un par de paréntesis. Esta operación se conoce como llamar al argumento o parámetro de la función. Estos argumentos son los datos de entrada utilizados por la función para devolver resultados específicos.

Función date()

La función `date()` espera a que el argumento que se le pase sea una cadena de formato, que represente el estilo que se desea utilizar para devolver los datos. Cada una de las letras de la cadena representa una parte de la fecha y la hora. `H` es la hora en formato 24 horas, `i` son los minutos precedidos de un cero cuando resulte necesario, `j` es el día del mes sin utilizar un cero inicial, `s` representa el sufijo ordinal (en este caso "th" del inglés) y `F` es el nombre completo del mes.

(En un capítulo posterior se incluye una lista completa de los formatos que admite la función `date()`).

Cómo acceder a las variables de formulario

El único objetivo de utilizar el formulario de pedido es recoger el pedido del cliente. Con PHP resulta muy sencillo obtener los detalles de lo que acaba de escribir el cliente, pero el método exacto depende de la versión del lenguaje que esté utilizando y de un parámetro del archivo php.ini.

Variables de formulario

Dentro de una secuencia de comandos de PHP, podemos acceder a cada uno de los campos del formulario como una variable de PHP cuyo nombre se relaciona con el nombre del campo del formulario. En PHP los nombres de variables se reconocen porque comienzan con el signo del dólar ($). Un error habitual es olvidarse de incluir este símbolo.

Puede acceder a los contenidos del `tireqty` de las siguientes formas:

```
$tireqty                        // estilo corto
$_POST['tireqty']               // estilo intermedio
$HTTP_POST_VARS['tireqty']      // estilo largo
```

En este ejemplo, y a lo largo del libro, se utiliza el estilo largo para hacer referencia a las variables de formulario pero puede crear versiones cortas de las variables para facilitar su uso. Se trata de una opción práctica y segura de procesar los datos que funcionará en todos los sistemas sea cual se su versión o configuración.

En su código, puede utilizar otro enfoque, pero para ello debería conocer las diferentes opciones.

En resumen:

- El estilo corto es práctico, pero requiere activar el parámetro de configuración `register_globals`. (El valor predeterminado varía de una versión a otra de PHP.) En este estilo resulta sencillo cometer errores que podrían convertir a su código en inseguro.
- El estilo intermedio resulta bastante práctico, pero hizo su aparición en la versión PHP 4.1.0, por lo que no funcionará en las versiones antiguas.
- El estilo largo es el estilo que incluye más detalles, pero es el único que garantiza el funcionamiento del código en todos los servidores, con independencia de su configuración. Sin embargo, tenga en cuenta que se considera como obsoleto y que es probable que con el tiempo acabe desapareciendo.

Nuestra intención es que el código de ejemplo utilizado en el libro funcione sin necesidad de ninguna modificación en la mayor cantidad de sistemas posibles. Por esta razón, hemos optado por el estilo largo, pero puede escoger otro método.

En el estilo corto, los nombres de las variables utilizadas en la secuencia de comandos son iguales a los nombres de los campos del formulario HTML. No es necesario declarar las variables en la secuencia de comandos o adoptar ninguna medida para crear dichas variables. Se pasan en la secuencia de comandos básicamente de la misma forma que se pasan argumentos a una función. Si ha escogido este estilo, puede utilizar una variable como `$tireqty`. El campo `tirdqty` del formulario crea la variable `$tireqty` en la secuencia de comandos de procesamiento.

El uso del estilo corto exige la activación de un comando llamado `register_globals` en el archivo de configuración `php.ini`. En PHP, desde la versión 4.2.0 en adelante, este parámetro está desactivado de manera predeterminada. Por el contrario, en las versiones anteriores está activado.

Resulta práctico disponer de este tipo de acceso a las variables, pero antes de activar el parámetro `register_globals` conviene conocer las razones que han llevado al equipo de desarrollo de PHP a dejarlo desactivado.

Como hemos dicho, este tipo de acceso a las variables resulta práctico, pero es proclive a cometer errores de programación que podrían comprometer la seguridad de las secuencias de comandos. Si las variables de función se convierten automáticamente en variables globales, no existirá una diferencia clara entre las variables creadas por nosotros y las variables no fiables procedentes directamente del usuario.

Si no asignamos un valor inicial a nuestras variables, los usuarios de las secuencias de comandos pueden pasar variables y valores en forma de variables de formulario que se mezclarán con las nuestras. Si se opta por utilizar el estilo corto para acceder a las variables, es aconsejable asignar un valor inicial a las variables.

El estilo medio implica la recuperación de variables de formulario de una de las matrices `$_POST`, `$_GET` y `$_REQUEST`. Una de estas matrices contendrá los detalles de todas las variables de formulario. La variable utilizada dependerá del método

utilizado para enviar el formulario, POST o GET, respectivamente. Además, todos los datos enviados a través de los métodos POST y GET estarán disponibles a través de `$_REQUEST`.

Si el formulario se envía por el método POST, los datos introducidos en el campo `tireqty` se almacenarán en `$_POST['tireqty']`. Si el formulario se envía utilizando GET, los datos se almacenarán en `$_GET['tireqty']`. En cualquiera de los dos casos, los datos estarán disponibles en `$_REQUEST['tireqty']`.

Estas matrices se incluyen dentro de los conocidos conjuntos superglobales. Volveremos sobre ellos al examinar el ámbito de las variables.

Si está utilizando una versión antigua de PHP, puede que no disponga de acceso a `$_POST` o `$_GET`. En las versiones anteriores a la 4.1.0, esta información se almacenaba en las matrices `$HTTP_POST_VARS` y `$HTTP_GET_VARS`. Este estilo es el que llamaremos estilo largo y se puede utilizar con versiones antiguas y nuevas de PHP, pero ha sido considerado como obsoleto por lo que es probable que no funcione en las versiones futuras. En este estilo no existe un equivalente de `$_REQUEST`.

Si está utilizando el estilo largo, puede acceder a la respuesta del usuario a través de `$HTTP_POST_VARS['tireqty']` o `$HTTP_GET_VARS['tireqty']`.

A lo largo de este libro, hemos intentado indicar dónde no funcionará el código con las versiones antiguas de PHP. Los ejemplos utilizados se han probado con la versión 4.3 de PHP, por lo que es posible que a veces resulten incompatibles con versiones anteriores a la 4.1.0. Por lo tanto, le recomendamos que utilice la versión actual.

Como habrá observado, en el código no se comprueba el contenido de las variables para garantizar que se introducen datos correctos en los campos del formulario. Intente introducir datos erróneos deliberadamente para observar qué ocurre. Cuando termine de leer este capítulo, es probable que desee agregar algún tipo de validación de datos a la secuencia de comandos. Vamos a examinar el ejemplo.

Como los nombres de variable resultan un tanto pesados en el estilo lago y nos vamos a basar en un tipo de variables conocidas como matrices, que no se analizarán de lleno hasta un capítulo posterior, comenzaremos por crear copias que resulten sencillas de utilizar.

Para copiar el valor de una variable en otra, puede utilizar el operador de asignación, que en PHP es el signo igual (=). La siguiente línea de código crea una nueva variable denominada `$tireqty` y copia los contenidos de `$HTTP_POST_VARS['tireqty']` dentro de la nueva variable:

```
$tireqty = $HTTP_POST_VARS['tireqty'];
```

Coloque el siguiente bloque de código al inicio de la secuencia de comandos de procesamiento. El resto de secuencias de comandos de este libro que procesen datos desde un formulario contendrán un bloque similar al inicio. Como no se generará ningún resultado, no importa si se coloca por encima o por debajo de la etiqueta `<html>` ni de otras etiquetas de HTML con las que se inicie la página. En nuestro caso, solemos colocar este bloque al inicio de la secuencia de comandos para asegurarnos de que resulta fácil de encontrar.

```
<?php
   //cree nombres de variables cortos
   $tireqty = $HTTP_POST_VARS['tireqty'];
   $oilqty = $HTTP_POST_VARS['oilqty'];
   $sparkqty = $HTTP_POST_VARS['sparkqty'];
?>
```

El código crea tres nuevas variables `$tireqty`, `$oilqty` y `$sparkqty` y las configura para que contengan los datos enviados a través del método POST desde el formulario.

Para que la secuencia de comandos haga algo visible, agregue las siguientes líneas a la parte final de la secuencia de comandos de PHP:

```
echo '<p>Your order is as follows: </p>';
echo $tireqty.' tires<br />';
echo $oilqty.' bottles of oil<br />';
echo $sparkqty.' spark plugs<br />';
```

Si carga ahora este archivo en el navegador, el resultado de la secuencia de comandos se parecerá al ilustrado en la figura 1.4. Los valores reales se mostrarán en función de lo que escriba en el formulario.

En las siguientes secciones se comentan un par de cosas interesantes sobre este ejemplo.

Concatenación de cadenas

En la secuencia de comandos, utilizamos `echo` para imprimir el valor que escribe el usuario en cada uno de los campos de formulario, seguido por texto explicativo. Si examina de cerca las instrucciones `echo`, verá que el nombre de la variable y que el texto que sigue están separados por un punto (.), como se puede ver a continuación:

```
echo $tireqty.' tires<br />';
```

Éste es el operador de concatenación de cadenas y se utiliza para unir cadenas (secciones de texto). Lo utilizará con frecuencia con `echo` para mostrar información en pantalla. Se suele recurrir a él para no tener que escribir varios comandos `echo`.

Para las variables que no sean matrices, puede encerrarlas entre comillas dobles para su representación en el navegador. (El estudio de las matrices se aborda en un capítulo posterior.) Por ejemplo:

```
echo "$tireqty tires<br />";
```

Esta secuencia es equivalente a la primera. Los dos formatos son válidos y la elección de uno u otro es una cuestión personal. Fíjese en que no es lo mismo utilizar comillas dobles que utilizar comillas simples para encerrar nombres de variables. Si se ejecuta la siguiente línea de código:

```
echo '$tireqty tires<br />';
```

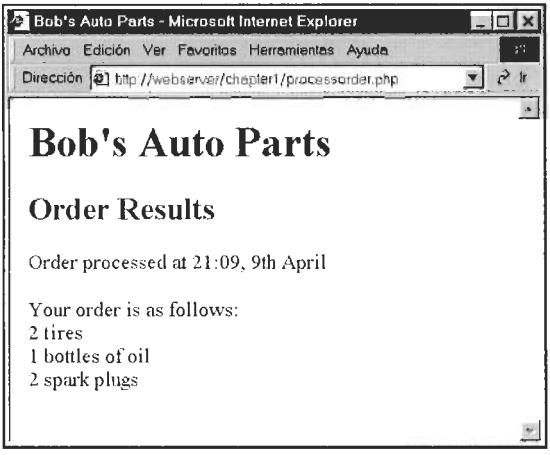


Figura 1.4. Las variables de formulario introducidas por el usuario resultan de fácil acceso en processorder.php

enviará la secuencia "`$tireqty tires<br />`" al navegador. Si se utilizan comillas dobles, el nombre de la variable se sustituirá por su valor. Si se utilizan comillas simples se enviará el nombre de la variable, o cualquier otro texto, sin alterar.

Literales y variables

La variable y la cadena que concatenamos en cada una de las instrucciones `echo` son cosas diferentes. Las variables son un símbolo para sustituir a los datos. Las cadenas son datos en sí mismas. Cuando se utiliza un segmento de datos como tal en un programa como éste, se designa como literal para distinguirlo de las variables. `$tireqty` es una variable, un símbolo que representa los datos escritos por el cliente. Por el contrario, `' tires<br />'` es un literal. Se puede tomar por su valor literal. Bueno, casi. ¿Recuerda el segundo ejemplo anterior? PHP sustituye el nombre de la variable `$tireqty` en la cadena por el valor almacenado en la variable.

Recuerde que existen dos tipos de cadenas en PHP, unas encerradas entre comillas dobles y otras entre comillas simples. PHP intentará evaluar las cadenas con comillas dobles, lo que dará como resultado el comportamiento visto anteriormente. Las cadenas con comillas simples se tratarán como verdaderos literales.

Identificadores

Los identificadores son nombres de variables. (Los nombres de funciones y clases son también identificadores. Los nombres y las clases se examinarán en capítulos posteriores.) Los identificadores siguen algunas reglas sencillas:

- Los identificadores puede tener cualquier longitud y pueden incluir letras, números, guiones bajos y signos de dólar. Sin embargo, conviene poner

cuidado al utilizar los signos de dólar en los identificadores. En una sección posterior explicaremos por qué.

- Los identificadores no pueden comenzar por un número.
- En PHP, los identificadores discriminan entre mayúsculas y minúsculas. `$tireqty` no es lo mismo que `$TireQty`. Un error común en programación consiste en intentar utilizarlos indistintamente. Los nombres de las funciones son una excepción a esta regla (el uso de mayúsculas y minúsculas es indiferente).
- Una variable puede tener el mismo nombre que una función. Este hecho puede dar lugar a confusiones por lo que conviene evitarlos. Así mismo, no se pueden crear dos funciones con el mismo nombre.

Variables declaradas por el usuario

Puede declarar y utilizar sus propias variables además de las variables que se pasan desde el formulario HTML.

Uno de los rasgos de PHP es que no requiere declarar variables antes de utilizarlas. Las variables se crean al asignarles un valor (véase la siguiente sección para conocer los detalles).

Asignación de valores a variables

Para asignar valores a las variables se utiliza el operador de asignación, =, como hicimos al copiar el valor de una variable en otra. En el sitio de Bob, queremos obtener el número total de artículos pedidos y la cantidad total que se debe abonar. Podemos crear dos variables para almacenar estos valores. En primer lugar, inicializamos cada una de estas variables en cero.

Agregue las siguientes líneas al final de la secuencia de comandos de PHP:

```
$totalqty = 0;
$totalamount = 0.00;
```

Cada una de estas líneas crea una variable y le asigna un valor literal. También puede asignar valores de variables a las variables, por ejemplo:

```
$totalqty = 0;
$totalamount = $totalqty;
```

Tipos de variables

Un tipo de variable hace referencia al tipo de datos que se almacenan en ella.

Tipos de datos de PHP

PHP admite los siguientes tipos de datos

- Entero: utilizado para números enteros
- Doble: utilizado para números reales
- Cadena: utilizado para cadenas de caracteres
- Booleano: utilizado para valores verdaderos o falsos
- Matriz: utilizados para almacenar conjuntos de datos del mismo tipo
- Objeto: utilizado para almacenar instancias de clases

PHP 4 agregó tres tipos de variables adicionales: booleanas, NULL y de recurso. Las variables a las que no se les ha asignado un valor, no están definidas o se les ha asignado el valor NULL son de tipo NULL. Algunas funciones incorporadas (como las funciones de base de datos) devuelven variables con tipo de recurso. Es muy poco probable que necesite manipular directamente este tipo de variables.

PHP también admite los tipos `pdfdoc` y `pdfinfo` si se ha instalado con compatibilidad para PDF (del inglés, Portable Document Format, Formato de documento portable). Este aspecto se tratará en un capítulo posterior.

Control de tipos

PHP es un lenguaje con un control de tipos muy débil. En la mayoría de los lenguajes, las variables sólo pueden contener un tipo de datos y dicho tipo de datos debe declararse antes de poder utilizar la variable, con ocurre en C. En PHP, el tipo de variable viene determinado por el valor que se le asigne.

Por ejemplo, al crear `totalqty` y `$totalamount`, deben determinarse sus tipos iniciales, de la siguiente forma:

```
$totalqty = 0;
$totalamount  =  0.00;
```

Como hemos asignado 0, un valor entero, a `$totalqty`, se tratará de una variable de tipo entero. De manera similar, `$totalamount` es de tipo doble.

Pero, por extraño que pueda parecer, podemos agregar una línea a nuestra secuencia de guiones de la siguiente forma:

```
$totalamount = 'Hello';
```

La variable `$totalamount` será entonces de tipo cadena. PHP cambia el tipo de la variable en función del valor almacenado en ella en cualquier momento dado.

Esta capacidad para cambiar los tipos de manera transparente al instante puede resultar extremadamente útil. Recuerde que PHP sabe "automáticamente" qué tipo de datos se está colocando en una variable y devolverá los datos con el mismo tipo tras recuperarlos de la variable.

Conversión de tipos

Puede simular que un tipo de variable o valor sea de un tipo diferente utilizando una conversión de tipo. Este recurso funciona de la misma forma que en C. Basta con colocar el tipo temporal entre corchetes delante de la variable que se desea convertir.

Por ejemplo, podríamos haber declarado las dos variables anteriores utilizando una conversión.

```
$totalqty = 0;
$totalamount  =  (double)$totalqty;
```

La segunda línea indica "toma el valor almacenado en `$totalqty`, interprétalo como doble y almacénalo como `$totalamount`". La variable `$totalamount` será de tipo doble. La variable de conversión no cambia de tipo por lo que `$totalqty` seguirá siendo de tipo entero.

Variables de tipo variable

PHP proporciona otro tipo de variable: la variable de tipo variable. Las variables de tipo variable permiten cambiar el nombre de una variable dinámicamente.

(Como puede observar, PHP proporciona una gran libertad en este sentido: todos los lenguajes permiten cambiar el valor de la variable, pero no hay muchos que permitan cambiar el tipo de variable y menos aún que permitan cambiar su nombre.)

Su funcionamiento consiste en utilizar el valor de una variable como nombre de otra. Por ejemplo, podemos establecer

```
$varname = 'tireqty';
```

A continuación podemos utilizar `$$varname` en lugar de `$tireqty`. Por ejemplo, podemos establecer el valor de `$tireqty`:

```
$$varname = 5;
```

Esto equivale a

```
$tireqty = 5;
```

Puede que parezca un poco confuso, pero no se preocupe porque volveremos sobre uso posteriormente. En lugar de tener que enumerar y utilizar cada variable de formulario por separado, podemos utilizar un bucle y una variable para proce-

sarlas todas automáticamente. En la sección dedicada a los bucles se incluye un ejemplo que ilustra este hecho.

Constantes

Como vimos anteriormente, podemos cambiar el valor almacenado en una variable. También podemos declarar constantes. Una constante almacena un valor como una variable con la diferencia de que se establece una vez y no se puede cambiar en ningún otro punto de la secuencia de comandos.

En nuestra aplicación de ejemplo, podríamos almacenar los precios de los artículos de venta en forma de constantes. Para definir constantes puede utilizar la función `definition`:

```
define('TIREPRICE', 100);
define('OILPRICE', 10);
define('SPARKPRICE', 4);
```

Agregue estas líneas de código a la secuencia de comandos.

Los nombres de las constantes utilizan siempre mayúsculas. Se trata de una convención tomada de C que facilita su distinción de las variables. Esta convención no es obligatoria pero contribuye a que la lectura y el mantenimiento del código resulten más sencillos.

Tenemos tres constantes que se pueden utilizar para calcular el total del pedido del cliente.

Una diferencia importante entre las constantes y las variables es que cuando hacemos referencia a una constante, no lleva antepuesto el símbolo del dólar. Si desea utilizar el valor de una constante, utilice únicamente su nombre. Por ejemplo, para utilizar una de las constantes que acabamos de crear, podemos escribir:

```
echo TIREPRICE;
```

Además de las constantes definidas, PHP establece un gran número de constantes propias. Una forma sencilla de verlas consiste en ejecutar el comando `phpinfo()`:

```
phpinfo();
```

Este comando devuelve una lista de variables y constantes predefinidas de PHP, además de otra información útil. Iremos comentándola a medida que vayamos avanzando.

Ámbito de variables

El término ámbito hace referencia a los lugares dentro de las secuencias de comandos en los que resulta visible una variable dada. A continuación se describen los cuatro tipos de ámbitos de PHP:

- Las variables superglobales incorporadas resultan siempre visibles dentro de una secuencia de comandos.

- Las variables globales declaradas en una secuencia de comandos resultan visibles a lo largo de la secuencia de comandos pero no dentro de las funciones.
- Las variables utilizadas dentro de funciones tienen restringido su ámbito a la función.
- Las variables utilizadas dentro de funciones que se declaran como globales hacen referencia a la variable global del mismo nombre.

A partir de la versión 4.2 de PHP, las matrices `$_GET` y `$_POST` así como otras variables especiales llevan asignadas sus propias reglas de ámbito. Éstas se conocen como superglobables y se pueden ver en todas partes, tanto dentro como fuera de las funciones.

A continuación se recoge la lista completa de variables globales:

- `$GLOBALS`, una matriz con todas las variables globales
- `$_SERVER`, una matriz con las variables de entorno del servidor
- `$_GET`, una matriz con las variables pasadas a la secuencia de comandos a través del método `GET`
- `$_POST`, una matriz con las variables pasadas a la secuencia de comandos a través del método `POST`
- `$_COOKIE`, una matriz con las variables de cookies
- `$_FILES`, una matriz con las variables relacionadas con las cargas de archivos
- `$_ENV`, una matriz de variables de entorno
- `$_REQUEST`, una matriz con todas las variables de entrada de usuario
- `$_SESSION`, una matriz con las variables de sesión

Volveremos sobre cada uno de estos tipos a medida que vayamos avanzando. Analizaremos el ámbito de las variables al estudiar las funciones. Por el momento, todas las variables que utilicemos serán globales de manera predeterminada.

Operadores

Los operadores son símbolos que se pueden utilizar para manipular valores y variables realizando una operación sobre ellos. Tendremos que utilizar algunos para calcular los totales y los impuestos que aplicar al pedido del cliente.

Ya hemos mencionado dos operadores: el operador de asignación, `=`, y `.`, el operador de concatenación. A continuación examinemos la lista completa.

En general, los operadores pueden tomar dos o tres argumentos (aunque la mayoría toman dos). Por ejemplo, el operador de asignación toma dos argumentos:

la ubicación de almacenamiento en la parte izquierda del símbolo = y una expresión en la parte derecha. Estos argumentos se conocen como operandos, es decir, los elementos sobre los que se opera.

Operadores aritméticos

Los operadores aritméticos son bastantes claros: se trata de los operadores matemáticos de uso más común. En la tabla 1.1 se recogen los operadores aritméticos.

Tabla 1.1. Operadores aritméticos de PHP

Operador	Nombre	Ejemplo
+	Suma	$a + $b
-	Resta	$a - $b
*	Multiplicación	$a* $b
/	División	$a / $b
%	Módulo	$a % $b

Con todos estos operadores podemos guardar el resultado de la operación. Por ejemplo:

```
$result = $a + $b;
```

La suma y la resta funcionan como se esperaría. Estas operaciones suman o restan, respectivamente, los valores almacenados en las variables $a y $b.

También puede utilizar el símbolo de resta, -, como operador unitario (es decir, un operador que toma un argumento u operando) para indicar valores negativos; por ejemplo:

```
$a = -1;
```

La multiplicación y la división también funcionan de la forma esperada. Tenga en cuenta que se usa el asterisco como operador de multiplicación en lugar de símbolo habitual de multiplicación y la barra inclinada como símbolo de la división.

El operador de módulo devuelve el resto de la división de la variable `$a` por la variable `$b`. Considere el siguiente fragmento de código:

```
$a = 27;
$b = 10;
$result = $a%$b;
```

El valor almacenado en la variable `$result` es el resto de dividir 27 por 10, es decir, 7.

Tenga en cuenta que los operadores aritméticos se suelen aplicar a las variables enteras o dobles. Si se aplican a cadenas, PHP intentará convertir la cadena en un número. Si contiene una "e" o una "E", la convertirá en una variable doble, de lo contrario la convertirá en un entero. PHP busca números al principio de la cadena y los utiliza como valores (si no hubiera ninguno, el valor de la cadena sería cero).

Operadores de cadena

Ya hemos visto y utilizado el único operador de cadena. Puede utilizar el operador de concatenación de cadenas para agregar dos cadenas, y que genere y almacene un resultado como haríamos si utilizáramos el operador de suma para sumar dos números.

```
$a = "Bob's ";
$b = 'Auto Parts';
$result = $a.$b;
```

La variable `$result` contendrá la cadena "Bob's Auto Parts".

Operadores de asignación

Ya hemos visto el operador de asignación básico =. A este operador se hará siempre referencia como operador de asignación y se lee como "establecer en". Por ejemplo:

```
$totalqty = 0;
```

Esta secuencia se lee como "$totalqty se establece en cero". Ya explicaremos por qué al hablar de los operadores de comparación en una sección posterior.

Devolución de valores de asignación

El uso del operador de asignación devuelve siempre un valor general similar a otros operadores. Si escribe

```
$a + $b
```

el valor de la expresión es el resultado de agregar las variables $a y $b. De manera análoga, puede escribir:

```
$a = 0;
```

El valor de toda la expresión es cero.

Esta secuencia permite realizar operaciones como

```
$b = 6 + ($a = 5);
```

Esta secuencia establece el valor de la variable `$b` en 11. Por regla general, el valor de toda la instrucción de asignación es el valor que se asigna en el operando de la izquierda.

Al calcular el valor de una expresión, se pueden utilizar los paréntesis para asignar prioridad a una subexpresión como hemos hecho en el ejemplo. Su funcionamiento es el mismo que en matemáticas.

Combinación de operadores de asignación

Además de la asignación simple, existe un conjunto de operadores de asignación combinados. Se trata de formas abreviadas de realizar otra operación sobre una variable y de asignarle el resultado. Por ejemplo:

```
$a += 5;
```

Esta secuencia equivale a escribir:

```
$a = $a + 5;
```

Existen operadores de asignación combinados para cada uno de los operadores aritméticos así como para el operador de concatenación de cadenas.

En la tabla 1.2 se recoge un resumen de todos los operadores de asignación combinados y su efecto.

Tabla 1.2. Operadores de asignación combinados de PHP

Operador	Uso	Equivale a
+=	$a += $b	$a = $a + $b
-=	$a -= $b	$a = $a - $b
*=	$a *= $b	$a = $a * $b
/=	$a /= $b	$a = $a / $b
%=	$a %= $b	$a = $a % $b
.=	$a .= $b	$a = $a . $b

Incremento y decremento previo y posterior

Los operadores de incremento (++) y decremento (—) previo y posterior son similares a los operadores += y -=, pero con un par de variaciones.

Los operadores de incremento presentan dos efectos: incrementan un valor y lo asignan. Considere la siguiente secuencia:

```
$a=4;
echo ++$a;
```

La segunda línea utiliza el operador de incremento previo, denominado así porque ++ aparece delante de `$a`. Este operador tiene dos efectos: en primer lugar incrementa `$a` en una unidad y en segundo lugar devuelve el valor incrementado. En este caso, `$a` se incrementa en 5 y se devuelve e imprime dicho valor. El resultado de

toda la expresión es 5. (Fíjese en que el valor que se almacena en realidad en `$a` se ha variado: no estamos devolviendo simplemente `$a + 1`.)

Sin embargo, si colocamos ++ después de `$a`, utilizamos el operador de incremento posterior. Su efecto es diferente al anterior. Considere la siguiente secuencia:

```
$a=4;
echo $a++;
```

En este caso, los efectos se invierten. En primer lugar, se devuelve e imprime el valor de `$a` y en segundo lugar se incrementa. El resultado de toda la expresión es 4. Éste es el valor que se imprimirá. Sin embargo, el valor de `$a` tras ejecutar esta instrucción será 5.

Como probablemente habrá adivinado, el comportamiento es similar para el operador –, con la diferencia de que el valor de `$a` se reduce en lugar de incrementarse.

Referencias

Una novedad de PHP 4 es el operador de referencia, `&` (ampersand), que se utiliza en combinación con el operador de asignación. Por regla general, cuando una variable se asigna a otra, se realiza una copia de la primera y se almacena en memoria. Por ejemplo:

```
$a = 5;
$b = $a;
```

Estas líneas de código realizan una segunda copia del valor almacenado en `$a` y lo guardan en `$b`. Si cambiamos el valor de `$a` en un momento posterior, `$b` no variará:

```
$a = 7; // $b seguirá siendo 5
```

Para evitar que se haga la copia y que se guarde en memoria puede utilizar el operador de referencia, `&`. Por ejemplo:

```
$a = 5;
$b = &$a;
$a = 7; // $a y $b son ahora 7 las dos
```

Operadores de comparación

Los operadores de comparación se utilizan para comparar dos valores. Las expresiones que utilizan estos operadores devuelven el valor lógico `true` o el valor lógico `false` en función del resultado de la comparación.

El operador iguales

El operador de comparación iguales, == (dos signos iguales), permite determinar si dos valores son iguales. Por ejemplo, podemos utilizar la expresión

```
$a == $b
```

para determinar si los valores almacenados en `$a` y en `$b` son iguales. El resultado devuelto por esta expresión será `true` si son iguales o `false` si no lo son.

Resulta sencillo confundir este operador con el operador de asignación (=). Esta confusión no generará un error pero impedirá que se obtengan los valores deseados. En general, los valores distintos a cero se evalúan como `true` y los valores iguales a cero se evalúan como `false`. Suponga que hemos inicializado dos variables de la siguiente forma:

```
$a = 5;
$b = 7;
```

Si prueba `$a = $b`, el resultado será `true`. ¿Por qué? El valor de `$a = $b` es el valor asignado a la parte izquierda de la expresión, que en este caso es 7. Se trata de un valor distinto a cero, por lo que la expresión se evalúa como `true`. Si su intención es probar `$a == $b`, que devuelve `false`, habrá introducido un error de lógica en su código que puede resultar extremadamente difícil de detectar. Compruebe siempre el uso de estos dos operadores y verifique si ha utilizado el que tenía previsto.

Se trata de un error muy sencillo de cometer y es muy probable que caiga en él muchas veces a lo largo de su carrera como programador.

Otros operadores de comparación

PHP admite otros operadores de comparación, que se recogen en la tabla 1.3.

Uno de los más destacados es el nuevo operador de identidad, ===, introducido en PHP, que devuelve `true` sólo si los dos operadores son iguales y del mismo tipo.

Tabla 1.3. Operadores de comparación de PHP

Operador	Nombre	Uso
==	igual	$a == $b
===	idéntico	$a === $b
!=	distinto	$a != $b
<>	distinto	$a <> $b
<	menor que	$a < $b
>	mayor que	$a > $b
<=	menor o igual que	$a <= $b
>=	mayor o igual que	$a >= $b

Operadores lógicos

Los operadores lógicos se utilizan para combinar los resultados de condiciones lógicas. Por ejemplo, puede que nos interese que el valor de una variable, `$a`, se

encuentre entre 0 y 100. Para ello tendríamos que probar las condiciones `$a >= 0` y `$a <= 100`, utilizando el operador AND, como se indica a continuación:

```
$a >= 0 && $a <=100
```

PHP admite el uso de los operadores lógicos AND, OR, XOR (o exclusivo) y NOT. En la tabla 1.4 se resumen los operadores lógicos y su uso.

Tabla 1.4. Operadores lógicos de PHP

Operador	Nombre	Uso	Resultado
!	NOT	!$b	Devuelve `true` si `$b` es `false` y viceversa
&&	AND	$a && $b	Devuelve `true` si `$a` y `$b` son `true`; de lo contrario devuelve `false`
||	OR	$a || $b	Devuelve `true` si `$a` o `$b` o ambas son `true`; de lo contrario devuelve `false`
and	AND	$a and $b	Igual que `&&`, pero con prioridad más baja
or	OR	$a or $b	Igual que `||`, pero con prioridad más baja

Los operadores `and` y `or` tienen una prioridad inferior a la de los operadores `&&` y `||`. En una sección posterior se analizará el tema de la prioridad.

Operadores bit a bit

Los operadores bit a bit permiten tratar un entero como una serie de bits utilizados para representarlos.

Estos operadores no se utilizan demasiado en PHP. En la tabla 1.5 se recoge un resumen de los operadores bit a bit.

Tabla 1.5. Operadores bit a bit de PHP

Operador	Nombre	Uso	Resultado
&	AND bit a bit	$a & $b	Los bits asignados en `$a` y `$b` se establecen en el resultado
|	OR bit a bit	$a | $b	Los bits asignados en `$a` o `$b` se establecen en el resultado
~	NOT bit a bit	~$a	Los bits asignados en `$a` no se establecen en el resultado y viceversa

Operador	Nombre	Uso	Resultado
^	XOR bit a bit	$a ^ $b	Los bits asignados en $a o $b pero no en ambos se establecen en el resultado
<<	desplazamiento a la izquierda	$a << $b	Mueve $a a la izquierda en función de los bits de $b
>>	desplazamiento a la derecha	$a >> $b	Mueve $a a la derecha en función de los bits de $b

Otros operadores

Además de los operadores vistos hasta ahora, existen otros.

El operador coma, `,`, se utiliza para separar argumentos de función y otros elementos de lista. Se suele utilizar de manera ocasional.

Los operadores `new` y `->` se utilizan para instanciar una clase y para acceder a los miembros de clase, respectivamente. Estos operadores se analizarán detalladamente en un capítulo posterior.

Los operadores de elementos de matrices, `[]`, se utilizan para acceder a elementos de matrices. También utilizaremos el operador `=>` en algunos contextos de matrices. Éstos se analizarán en un capítulo posterior.

Existen otros tres operadores que se examinan brevemente a continuación.

Operador ternario

Este operador, `?:`, funciona de la misma forma que en C. Adopta la siguiente forma:

```
condición ? valor si true : valor si false
```

El operador ternario es similar a la versión de la expresión de una instrucción `if-else`, que se analizará más adelante en este capítulo.

Vemos un sencillo ejemplo:

```
($grade > 50 ? 'Aprobado' : 'Suspenso');
```

Esta expresión evalúa la nota del estudiante como 'Aprobado' o 'Suspenso'.

Operador de supresión de error

El operador de supresión de error, `@`, se puede utilizar por delante de cualquier expresión, es decir, de todos aquellos elementos que generen o tengan un valor. Por ejemplo:

```
$a = @(57/0);
```

Sin el operador `@`, esta línea generará un aviso de división por cero (inténtelo). Si se incluye el operador, se suprimirá el error.

Si tiene previsto suprimir advertencias de esta forma, debería escribir código de control de errores que compruebe cuándo tiene lugar una advertencia. Si ha configurado PHP con la función `track_errors` activada, el mensaje de error se almacenará en la variable global `$php_errormsg`.

Operador de ejecución

En realidad el operador de ejecución está formado por un par de operadores: dos acentos graves (`). Este símbolo es diferente a la comilla simple (en el teclado se sitúa en la tecla con el símbolo de diéresis).

PHP intentará ejecutar todo lo que se incluya entre estos símbolos como un comando en la línea de comandos del servidor. El valor de la expresión es el resultado del comando.

Por ejemplo, en sistemas operativos del tipo UNIX, puede utilizar:

```
$out = `ls -la`;
echo  '<pre>'.$out.'</pre>';
```

O el equivalente en un servidor Windows:

```
$out = `dir c:`;
echo '<pre>'.$out.'</pre>';
```

Cualquiera de estas versiones devolverá un listado de directorio y lo almacenará en `$out`. A continuación se puede imprimir en el navegador o manipular de otra forma.

Existen otras formas de ejecutar comandos en el servidor, como veremos en un capítulo posterior.

Uso de operadores: calcular los totales de los formularios

Ahora que ya sabemos cómo utilizar operadores de PHP, podemos calcular los totales y los impuestos en el formulario de pedido de Bob. Para ello agregue el siguiente código al final de la secuencia de comandos de PHP:

```
$totalqty = 0;
$totalqty = $tireqty + $oilqty + $sparkqty;
echo 'Items ordered: '.$totalqty.'<br />';

$totalamount = 0.00;

define('TIREPRICE', 100);
define('OILPRICE', 10);
define('SPARKPRICE', 4);

$totalamount = $tireqty * TIREPRICE
```

```
    + $oilqty * OILPRICE
    + $sparkqty * SPARKPRICE;

echo 'Subtotal: $'.number_format($totalamount,3).'<br />';

$taxrate = 0.10; // local sales tax is 10%
$totalamount = $totalamount * (1 + $taxrate);
echo 'Total including tax: $'.number_format($totalamount,2).'<br />';
```

Si actualiza la página en la ventana del navegador, verá un resultado similar al ilustrado en la figura 1.5.

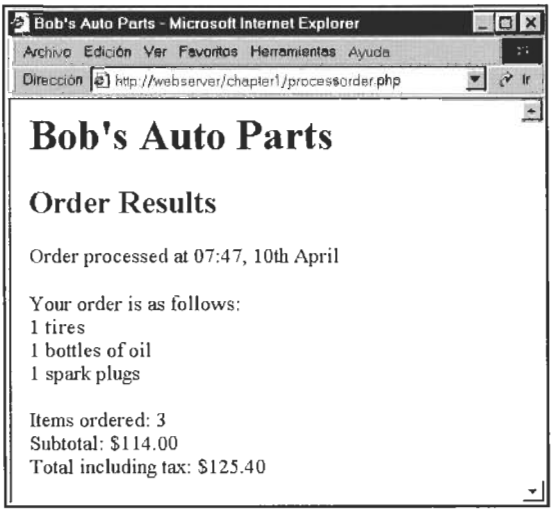


Figura 1.5. Se han calculado los totales del pedido de clientes, se les ha aplicado formato y se ha mostrado

Como puede observar, en esta secuencia de código hemos utilizado varios operadores. El operador de suma (+) y de multiplicación (*) se encargan de calcular las cantidades y el operador de concatenación (.) establece el resultado en el navegador.

También hemos utilizado la función `number_format()` para aplicar formato a los totales como cadenas con dos decimales. Se trata de una función de la biblioteca de matemáticas de PHP.

Si se detiene a examinar los cálculos, es posible que se pregunte por qué se realizaron en ese orden. Por ejemplo, considere esta instrucción:

```
$totalamount = $tireqty * TIREPRICE
    + $oilqty * OILPRICE
    + $sparkqty * SPARKPRICE;
```

La cantidad total parece correcta, pero ¿por qué realizar las multiplicaciones antes que las sumas? La respuesta reside en la precedencia de los operadores, es decir, en el orden en el que se calculan.

Precedencia y asociatividad: evaluación de expresiones

Por regla general, los operadores se evalúan siguiendo una precedencia u orden de prioridad fijado. Los operadores llevan asignada una asociatividad, que es el orden en el que se evalúan los operadores con el mismo orden de prioridad. Suele ser de izquierda a derecha (o izquierda simplemente), de derecha a izquierda (o derecha simplemente) o no resulta relevante. En la tabla 1.6 se recoge la precedencia de operadores y la asociatividad en PHP. En esta tabla, se incluyen los operadores de prioridad más baja en la parte superior y la precedencia va aumentado según se desciende por ella.

Tabla 1.6. Precedencia de operadores en PHP

Asociatividad	Operadores
izquierda	,
izquierda	or
izquierda	xor
izquierda	and
derecha	print
izquierda	= += -= *= /= .= %= &= |= ^= ~= <<= >>=
izquierda	? :
izquierda	||
izquierda	&&
izquierda	|
izquierda	^
izquierda	&
n/a	== != ===
n/a	< <= > >=
izquierda	<< >>
izquierda	+ - .
izquierda	* / %
derecha	-! ~ ++ — (entero) (doble) (cadena) (matriz) (objeto) @
derecha	[ ]
n/a	new
n/a	()

Fíjese en que el operador con mayor prioridad no se ha analizado todavía: los paréntesis. El efecto de los paréntesis consiste en incrementar la prioridad de todos los elementos incluidos en su interior. Esto símbolos permiten alterar las reglas de prioridad cuando resulte necesario.

Recuerde la siguiente sección del último ejemplo:

```
$totalamount = $totalamount * (1 + $taxrate);
```

Si hubiéramos escrito

```
$totalamount = $totalamount * 1 + $taxrate;
```

el operador de multiplicación, que tiene precedencia sobre el operador de suma, se aplicaría en primer lugar, lo que nos daría un resultado incorrecto. El uso de los paréntesis permite forzar el cálculo de la `subexpresión 1 + $taxrate` en primer lugar.

Puede utilizar tantos paréntesis como desee en una expresión. En primer lugar se calcularán los más internos.

Funciones de variables

Antes de abandonar el mundo de las variables y de los operadores, vamos a examinar las funciones de variables de PHP. Se trata de una biblioteca de funciones que permite manipular y probar variables de distintas formas.

Cómo probar y establecer tipos de variables

La mayor parte de estas funciones se utilizan para probar el tipo de una variable.

Las dos más generales son `gettype()` y `settype()`. Éstas tienen los siguientes prototipos de funciones; es decir, lo que esperan los argumentos y lo que devuelven.

```
string gettype(mixed var);
bool settype (mixed var, string type);
```

Para utilizar `gettype()`, la pasamos en una variable. Determinará su tipo y devolverá una cadena que contenga el tipo de nombre o "tipo desconocido" si no es uno de los tipos estándar, es decir, entero, doble, cadena, matriz u objeto.

Para utilizar `settype()`, le pasamos una variable cuyo tipo deseemos modificar y una cadena que contenga un nuevo tipo para dicha variable a partir de la lista anterior. Podemos utilizarla de la siguiente forma:

```
$a = 56;
echo gettype($a).'<br />';
settype($a, 'double');
echo gettype($a).'<br />';
```

Cuando se llama a `gettype()` por primera vez, `$a` es de tipo entero. Tras llamar a `settype()`, el tipo se convierte en doble.

PHP también incorpora funciones para probar tipos. Cada una de estas toma una variable como argumento y devuelve `true` o `false`. Las funciones son:

- is_array()
- is_double(), is_float(), is_real() (Toda la misma función)
- is_long(), is_int(), is_integer() (Toda la misma función)
- is_string()
- is_object()

Cómo probar el estado de las variables

PHP dispone de varias formas de probar el estado de una variable.

La primera de estas formas es `isset()`, que consta del siguiente prototipo:

```
bool isset(mixed var);
```

Esta función toma el nombre de una variable y devuelve `true` si existe y `false` en caso contrario. Puede eliminar una variable utilizando `unset()`. Éste es su prototipo:

```
void  unset(mixed  var);
```

Esta función suprime la variable si se pasa y devuelve `true`.

Por último, tenemos la función `empty()`. Esta función comprueba si existe una variable y si contiene un valor no vacío o distinto a cero y devuelve `true` o `false` según el caso. Su sintaxis es la siguiente:

```
boolean empty(mixed var);
```

Vamos a examinar un ejemplo del uso de estas funciones.

Intente agregar el siguiente código a su secuencia de comandos temporalmente:

```
echo isset($tireqty);
echo isset($nothere);
echo empty($tireqty);
echo empty($nothere);
```

Actualice la página para ver los resultados.

La variable `$tireqty` debería devolver `true` de `isset()` con independencia del valor introducido o de si se introdujo algún valor en el campo del formulario. Que sea `empty()` o no depende del valor introducido.

La variable `$nothere` no existe, por lo que generará `false` desde `isset()` y `true` desde `empty()`.

Estas funciones pueden resultar útiles para asegurarnos de que el usuario rellenó los campos apropiados del formulario.

Reinterpretación de variables

Puede lograr el equivalente de convertir una variable llamando a una función. En este sentido, existen tres funciones útiles:

```
int    intval(mixed   var);
float  doubleval(mixed var);
string strval(mixed var);
```

Cada una de éstas acepta una variable como entrada y devuelve el valor de la variable convertida en el tipo apropiado. En este libro, y en la documentación de php.net, se utiliza el tipo de datos `mixed`.

Este tipo de datos no existe, pero como PHP es tan flexible en materia de procesamiento de tipos, gran parte de las funciones pueden tomar muchos (o todos) los tipos de datos como argumento. Para identificar los argumentos en los que se permiten utilizar muchos tipos se utiliza `mixed`.

Estructuras de control

Las estructuras de control de un lenguaje permiten controlar el flujo de la ejecución de un programa o secuencia de comandos. Las estructuras de control se pueden agrupar en estructuras condicionales (o de bifurcación) y en estructuras de repetición, o bucles.

En las siguientes secciones se examinarán las implementaciones específicas de cada una de ellas en PHP.

Toma de decisiones con estructuras condicionales

Si deseamos responder lógicamente a las entradas de nuestros usuarios, nuestro código debe ser capaz de tomar decisiones. Esta función recae sobre las estructuras condicionales.

Las instrucciones if

Podemos utilizar una instrucción `if` para tomar una decisión. Debemos darle una condición a la instrucción `if` para que la utilice. Si la condición fuera `true`, se ejecutará el siguiente bloque de código. Las condiciones de las instrucciones `if` deben ir incluidas entre paréntesis.

Por ejemplo, si mandamos un pedido en el que no se incluyan neumáticos, latas de aceite ni bujías en el sitio de Bob, probablemente se deberá a que se ha pulsado accidentalmente el botón **Submit** En lugar de indicarnos "Order processed", la página podría devolver un mensaje mucho más útil.

Cuando el visitante realiza un pedido sin ningún artículo, podríamos indicárselo. Para ello podemos utilizar la siguiente instrucción `if`:

```
if( $totalqty == 0 )
echo 'You did not order anything on the previous page!<br />';
```

La condición que estamos utilizando es `$totalqty == 0`. Recuerde que el operador iguales (==) se comporta de manera distinta al operador de asignación (=).

La condición `$totalqty == 0` será `true` por lo que `$totalqty` es igual a cero. Si `$totalqty` no es igual a cero, la condición será `false`. Cuando la condición sea `true`, la instrucción `echo` se ejecutará.

Bloques de código

A menudo necesitaremos ejecutar más de una instrucción dentro de una secuencia condicional como `if`. No es necesario colocar una nueva instrucción `if` para cada una de ellas. En su lugar, podemos agrupar un número de instrucciones en un bloque. Para declarar un bloque, enciérrelo entre llaves:

```
if( $totalqty == 0 )
{
   echo '<font color=red>';
   echo 'You did not order anything on the previous page!<br />';
   echo '</font>';
}
```

Las tres líneas de código encerradas entre llaves forman ahora un bloque de código. Si la condición es `true`, se ejecutarán las tres líneas. Si la condición es `false`, se ignorarán las tres líneas.

Nota

Como se mencionó anteriormente, en PHP no es importante la disposición del código. Sin embargo, conviene sangrarlo por cuestiones de legibilidad. Las sangrías nos permiten distinguir de un vistazo qué líneas se ejecutarán sólo si se cumplen una condición, qué instrucciones se agrupan en bloques y qué instrucciones forman parte de bucles o funciones. En los ejemplos anteriores, se ha sangrado la instrucción que depende de la instrucción `if` y las instrucciones que forman el bloque.

Instrucciones else

Con frecuencia no sólo querrá decidir si desea que se ejecute una acción sino seleccionar una entre un conjunto de ellas.

Las instrucciones `else` permiten establecer la adopción de una acción alternativa cuando la condición de una instrucción `if` resulte `false`. Queremos avisar a los

clientes de Bob si envían una pedido sin ningún artículo. Por otra parte, si realizan un pedido, en lugar de una advertencia, queremos mostrarles lo que han pedido.

Si reorganizamos nuestro código y agregamos una instrucción `else`, podemos mostrar un aviso o un resumen de los artículos solicitados.

```
if( $totalqty == 0 )
{
   echo 'You did not order anything on the previous page!<br />';
}
else
{
   echo $tireqty.' tires<br />';
   echo $oilqty.' bottles of oil<br />';
   echo $sparkqty.' spark plugs<br />';
}
```

Podemos desarrollar procesos lógicos más complejos anidando instrucciones `if`. En el siguiente código, no sólo se mostrará el resumen si la condición `$totalqty == 0` resulta ser cierta, sino que además cada línea del resumen sólo se mostrará si se cumple su propia condición.

```
if( $totalqty == 0)
{
   echo 'You did not order anything on the previous page!<br />';
}
else
{
   if ( $tireqty>0 )
      echo $tireqty.' tires<br />';
   if ( $oilqty>0 )
      echo $oilqty.' bottles of oil<br />';
   if ( $sparkqty>0 )
      echo $sparkqty.' spark plugs<br />';
}
```

Instrucciones elseif

Para muchas de las decisiones que tomamos suele haber más de dos opciones. Podemos crear una secuencia de varias opciones utilizando la instrucción `elseif`. Esta instrucción es una combinación de `else` e `if`. Al suministrar una secuencia de condiciones, el programa puede comprobar cada una de ellas hasta que encuentre una que sea `true`.

Bob ofrece un descuento por grandes pedidos de neumáticos. La oferta se articula de la siguiente forma:

- Menos de 10 neumáticos, sin descuento
- De 10 a 49, 5% de descuento
- De 50 a 99, 10% de descuento
- 100 o más de 100, 15% de descuento

Podemos crear código para calcular el descuento utilizando condiciones e instrucciones `if` y `elseif`. Necesitamos utilizar el operador AND (`&&`) para combinar las dos instrucciones en una.

```
if( $tireqty < 10 )
   $discount = 0;
elseif( $tireqty >= 10 && $tireqty <= 49 )
   $discount = 5;
elseif( $tireqty >= 50 && $tireqty <= 99 )
   $discount = 10;
elseif( $tireqty >= 100 )
   $discount = 15;
```

Fíjese en que da lo mismo escribir `elseif` o `else if`, ya que ambas son correctas.

Si va a escribir un conjunto de instrucciones `if` en cascada, debería se consciente de que sólo se ejecutará uno de los bloques o instrucciones. En este ejemplo no importa porque todas las condiciones son mutuamente excluyentes, sólo una puede ser verdadera a la vez.

Si escribiéramos las condiciones de forma que más de una pueda ser verdadera, sólo se ejecutaría el bloque o instrucción situada a continuación de la primera condición cierta.

Instrucción switch

La instrucción `switch` funciona de una forma similar a la instrucción `if`, pero permite que la condición tome más de dos valores. En una instrucción `if`, la condición puede ser `true` o `false`.

En una instrucción `switch`, la condición puede tomar cualquier número de valores diferentes, siempre y cuando se evalúe en un tipo único (entero, cadena o doble).

Es necesario incluir una instrucción `case` para cada valor al que desee reaccionar y, opcionalmente, una instrucción `case` predeterminada para procesar aquellos valores para los que no se hayan incluido una instrucción `case` específica.

Bob quiere saber qué tipo de publicidad atrae visitantes a su sitio. Para ello podemos agregar una pregunta a nuestro formulario de pedidos.

Inserte el siguiente código HTML dentro del formulario de pedidos y el formulario presentará un aspecto parecido al ilustrado en la figura 1.6.

```
<tr>
  <td>How did you find Bob's</td>
  <td><select name="find">
          <option value = "a">I'm a regular customer
          <option value = "b">TV advertising
          <option value = "c">Phone directory
          <option value = "d">Word of mouth
        </select>
  </td>
</tr>
```

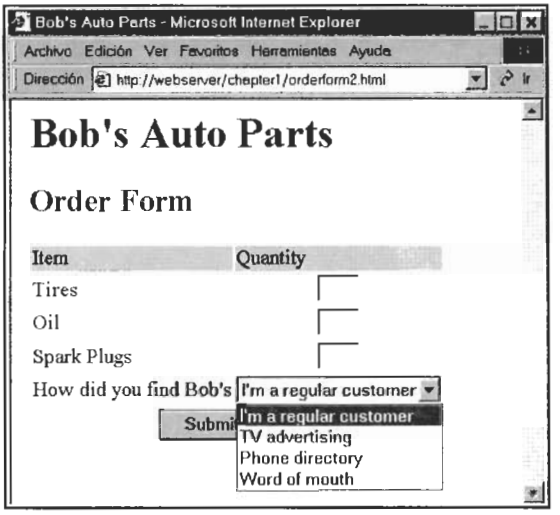


Figura 1.6. El formulario de pedidos pregunta a los visitantes cómo encontraron el sitio Bob's Auto Parts.

Este código de HTML agrega una nueva variable de formulario cuyo valor será "a", "b", "c", o "d". Podríamos procesar esta nueva variable con una serie de instrucciones `if` y `elseif` como la siguiente:

```
if($find == 'a')
   echo '<p>Regular customer.</p>';
elseif($find == 'b')
   echo '<p>Customer referred by TV advert.</p>';
elseif($find == 'c')
   echo '<p>Customer referred by phone directory.</p>';
elseif($find == 'd')
   echo '<p>Customer referred by word of mouth.</p>';
```

Como alternativa podríamos utilizar una instrucción `switch`:

```
switch($find)
{
   case 'a' :
      echo '<p>Regular customer.</p>';
      break;
   case 'b' :
      echo '<p>Customer referred by TV advert.</p>';
      break;
   case 'c' :
      echo '<p>Customer referred by phone directory.</p>';
      break;
   case 'd' :
      echo '<p>Customer referred by word of mouth.</p>';
      break;
   default :
      echo '<p>We do not know how this customer found us.</p>';
      break;
}
```

La instrucción `switch` se comporta de una forma ligeramente diferente a una instrucción `if` o `elseif`. Una instrucción `if` afecta sólo a una instrucción a menos que se utilicen llaves de manera deliberada para crear bloques de instrucciones. Una instrucción `switch` se comporta de forma contraria. Cuando se activa un caso en una instrucción `switch`, PHP ejecutará las instrucciones hasta que alcance una instrucción `break`.

Sin una instrucción `break`, la instrucción `switch` ejecutaría todo el código situado detrás del caso que resultara ser cierto. Al alcanzar la instrucción `break`, se ejecutará la línea de código situada tras la instrucción `switch`.

Comparación de condiciones diferentes

Si no está familiarizado con estas instrucciones, es probable que se esté preguntando cuál es la mejor.

Sin embargo, no podemos dar una respuesta exacta a esta pregunta. No hay nada que pueda hacer con una o varias instrucciones `else`, `elseif` o `switch` que no puede hacerse con un conjunto de instrucciones `if`. Debería intentar utilizar aquella opción que le resulte más sencilla de leer Con el tiempo irá aprendiendo a seleccionar una u otra.

Iteración: repetición de acciones

Una de las cosas para las que los ordenadores han demostrado siempre ser buenos es automatizar tareas repetitivas. Si hay algo que necesite hacer de la misma forma una serie de veces, puede utilizar un bucle para repetir partes de un programa.

Bob quiere una tabla que muestre el coste de envío en función de la distancia a la que se está enviando el paquete.

El coste se puede calcular con una sencilla fórmula.

Queremos que la tabla de costes de envío se parezca a la ilustrada en la figura 1.7.

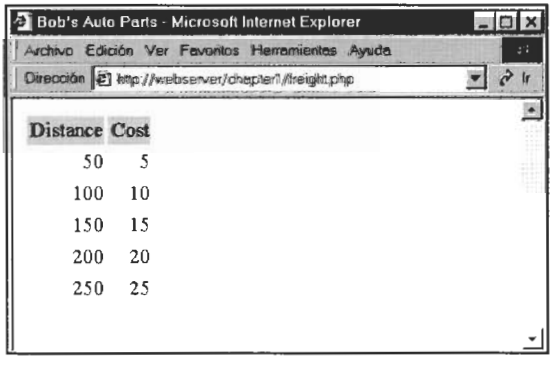


Figura 1.7. Esta tabla muestra el coste de envío en función de la distancia

El listado 1.2 incluye el código HTML utilizado para mostrar esta tabla. Como observará, el código es largo y repetitivo.

Listado 1.2. freight.html: la tabla de costes de envío de Bob

```
<html>
<body>
<table border="0" cellpadding="3">
<tr>
   <td  bgcolor="#CCCCCC"  align="center">Distance</td>
   <td  bgcolor="#CCCCCC"  align="center">Cost</td>
</tr>
<tr>
   <td  align="right">50</td>
   <td align="right">5</td>
</tr>
<tr>
   <td align="right">100</td>
   <td align="right">10</td>
</tr>
<tr>
   <td align="right">150</td>
   <td align="right">15</td>
</tr>
<tr>
   <td align="right">200</td>
   <td align="right">20</td>
</tr>
<tr>
   <td  align="right">250</td>
   <td align="right">25</td>
</tr>
</table>
</body>
</html>
```

Convendría delegar en un ordenador barato e incansable la tarea de escribir este código de HTML en lugar de recurrir a un humano que se aburriría con facilidad (y al que habría que pagar). Las instrucciones de bucle hacen que PHP ejecute una instrucción o un bloque de manera repetida.

Bucles while

El tipo de bucle más sencillo en PHP es el bucle `while`. Como en el caso de las instrucciones `if`, se basan en una condición. La diferencia entre un bucle `while` y una instrucción `if` es que ésta ejecuta el siguiente bloque de código si la condición resulta ser `true`. Un bucle `while` ejecuta el bloque repetidamente mientras la condición sea `true`.

Los bucles `while` se utilizan cuando no se sabe cuántas iteraciones resultarán necesarias para que la condición resulte cierta. Si el número de iteraciones debe ser fijo, considere la posibilidad de utilizar el bucle `for`.

La estructura básica de un bucle `while` es la siguiente:

```
while( condición ) expresión;
```

El siguiente bucle mostrará los números del 1 al 5.

```
$num = 1;
while ($num <= 5 )
{
   echo $num."<br />";
   $num++;
}
```

Al principio de cada operación, se prueba la condición. Si es falsa, el bloque no se ejecuta y el bucle finaliza. A continuación, se ejecutará la instrucción situada por detrás del bucle.

Podemos utilizar un bucle `while` para realizar algo más útil como mostrar la tabla de costes de envío repetitivos de la figura 1.7.

El listado 1.3 utiliza un bucle `while` para generar la tabla de gastos de envío.

Listado 1.3. freight.php. Generación de la tabla de gastos de envío de Bob en PHP

```
<body>
<table border="0" cellpadding="3">
<tr>
   <td bgcolor="#CCCCCC" align="center">Distance</td>
   <td bgcolor="#CCCCCC" align="center">Cost</td>
</tr>
<?
$distance = 50;
while ($distance <= 250 )
{
   echo "<tr>\n  <td align="right">$distance</td>\n";
   echo "  <td align="right">". $distance / 10 ."</td>\n</tr>\n";
   $distance += 50;
}
?>
</table>
</body>
</html>
```

Para que el código HTML generado por nuestra secuencia de comandos resulte legible, debe incluir nuevas líneas y espacios. Como se indicó anteriormente, los navegadores ignoran estos elementos pero resultan importantes para facilitar la lectura de los humanos.

Con frecuencia necesitará examinar el código HMTL si el resultado obtenido no es el esperado.

En el listado 1.3 verá que algunas cadenas incluyen los caracteres `\n`. Si se incluye dentro de una cadena encerrada entre comillas dobles, representa un carácter de nueva línea.

Los bucles for y foreach

La forma en la que utilizamos los bucles `while` anteriormente resulta muy común. En primer lugar establecimos un contador. Antes de cada interacción, probamos el contador en una condición. Al final de cada iteración, modificamos el contador.

Podemos escribir este estilo de bucle de forma más compacta utilizando un bucle `for`.

La estructura básica de los contadores `for` es la siguiente:

```
for( expresión1; condición; expresión2)
   expresión3;
```

- La *expresión1* se ejecuta una vez al principio. En este parámetro se suele establecer el valor inicial de un contador.
- La *condición* se prueba antes de cada iteración. Si la expresión devuelve `false`, la iteración se detendrá. En este parámetro se suele probar el contador con respecto a un límite.
- La *expresión2* se ejecuta al final de cada iteración. En este parámetro se suele ajustar el valor del contador.
- La *expresión3* se ejecuta una vez por iteración. Esta expresión suele ser un bloque de código y contendrá el grueso del código de bucle.

Podemos rescribir el ejemplo del bucle `while` del listado 1.3 como un bucle `for`. El código PHP se convertirá en

```
 <?
for($distance = 50; $distance <= 250; $distance += 50)
{
   echo "<tr>\n    <td align='right'>$distance</td>\n";
   echo "  <td align='right'>". $distance / 10 ."</td>\n</tr>\n";
}
?>
```

Tanto la versión `while` como la versión `for` funcionan de manera idéntica. El bucle `for` resulta un poco más compacto y ahorra dos líneas.

Ambos tipos de bucles son equivalentes. Ninguno de los dos es mejor o peor que el otro. En una situación dada, puede utilizar el que le resulte más intuitivo.

Como comentario al margen, hay que decir que se puede combinar una variable de tipo variable con un bucle `for` para procesar una iteración a través de una serie de campos de formulario repetitivos. Si, por ejemplo, tiene campos de formulario con nombres como `name1`, `nam2`, `name3`, etc. puede procesarlos de la siguiente forma:

```
for (i=1; $i <= $numnames; $i++)
{
   temp= "name$i";
   cho $$temp.'<br />'; // o cualquier procesamiento que desee realizar
}
```

Al crear dinámicamente los nombres de las variables, podemos acceder a cada campo uno por uno.

Además del bucle `for` existe el bucle `foreach`, diseñado específicamente para su uso con matrices. En un capítulo posterior comentaremos cómo utilizarlo.

Bucles do...while

El tipo final de bucle que mencionaremos se comporta de modo ligeramente diferente. La estructura general de una instrucción `do...while` es la siguiente:

```
do
   expresión;
while( condición );
```

Un bucle `do..while` se diferencia de un bucle `while` en que la condición se prueba al final. Por lo tanto en un bucle `do...while`, la instrucción o el bloque incluido en el bucle se ejecuta siempre una vez al menos.

Incluso si tomamos un ejemplo en el que la condición será `false` al principio y nunca puede ser `true`, el bucle se ejecutará una vez antes de comprobar la condición y el final.

```
$num = 100;
do
{
   echo $num.'<br />';
}
while ($num < 1 );
```

Cómo salir de una estructura de control o una secuencia de comandos

Si desea detener la ejecución de un fragmento de código, existen tres opciones que dependen del efecto que esté persiguiendo.

Si desea detener la ejecución de un bucle, puede utilizar la instrucción `break` como se comentó anteriormente en la sección sobre `switch`. Si utiliza esta instrucción en un bucle, la ejecución de la secuencia de comandos continuará en la línea situada tras el bucle.

Si desea saltar hasta la siguiente iteración de bucle, puede utilizar la instrucción `continue`.

Si desea terminar de ejecutar la secuencia de comandos PHP entera, puede utilizar la instrucción `exit`. Esta opción resulta útil al realizar tareas de comprobación de errores. Por ejemplo, podemos modificar nuestro ejemplo anterior de la siguiente forma:

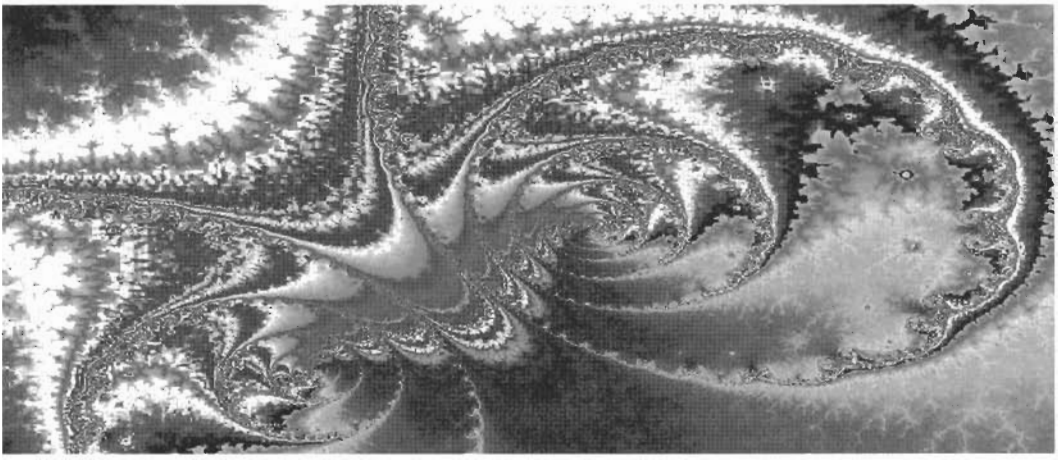
```
if( $totalqty == 0)
```

```
{
   echo 'You did not order anything on the previous page!<br />';
   exit;
}
```

La llamada a `exit` impide que PHP ejecute el resto de la secuencia de comandos.

Siguiente paso: guardar el pedido del cliente

Ahora ya sabe cómo recibir y manipular pedidos de un cliente. En el siguiente capítulo examinaremos cómo almacenar el pedido para poder recuperarlo y servirlo.



2

Almacenamiento y recuperación de datos

Ahora que ya sabemos cómo acceder y manipular datos introducidos en un formulario HTML, podemos pasar a examinar variar formas de almacenar dicha información para su uso posterior. En la mayor parte de los casos, incluido el ejemplo examinado en el capítulo anterior, querrá almacenar los datos y cargarlos en un momento posterior. En nuestro caso, necesitamos escribir pedidos de cliente para almacenarlos y poder servirlos después.

En este capítulo vamos a explicar cómo podemos escribir el pedido del cliente del ejemplo anterior en un archivo y cómo volverlo a leer. También veremos por qué esta solución no es siempre la mejor. Si tenemos un gran número de pedidos, deberíamos utilizar un sistema de administración de base de datos como MySQL.

Entre los temas clave que trataremos en este capítulo se incluyen los siguientes:

- Cómo guardar datos para su uso posterior
- Cómo abrir un archivo
- Cómo crear y escribir en un archivo
- Cómo cerrar un archivo
- Lectura de un archivo
- Bloqueo de un archivo
- Eliminación de archivos
- Otras funciones útiles con archivos

- Una opción mejor: los sistemas de administración de bases de datos
- Lecturas adicionales

Cómo guardar datos para su lectura posterior

Básicamente existen dos formas de almacenar datos: en archivos planos o en una base de datos. Los archivos planos pueden tener múltiples formatos pero, en general, cuando hacemos referencia a un archivo plano, nos estamos refiriendo a un archivo simple de texto. En este ejemplo, vamos a escribir los pedidos de los clientes en un archivo de texto, uno en cada línea.

Este método es muy sencillo, pero resulta a la vez muy restrictivo, como veremos en una sección posterior. Si está tratando con información de un volumen significativo es probable que prefiera utilizar una base de datos. Sin embargo, los archivos planos se utilizan en determinadas ocasiones y hay situaciones en las que necesitará saber cómo hacerlo.

Las operaciones de lectura y escritura de archivos en PHP se realizan prácticamente de la misma forma que en C. Si ha programado alguna vez en C o ha desarrollado secuencias de comandos de núcleo para sistemas UNIX, la mecánica le resultará bastante familiar.

Almacenamiento y recuperación de los pedidos de Bob

En este capítulo, utilizaremos una versión modificada del formulario de pedido visto en el último capítulo. Comenzaremos por este formulario y por el código de PHP que escribimos para procesar los datos del pedido.

Nota

Las secuencias de comandos de HTML y de PHP que se utilizan en este capítulo se pueden encontrar en la carpeta correspondiente del CD-ROM.

Hemos modificado el formulario para incluir un método rápido para obtener la dirección de envío del cliente. En la figura 2.1 se ilustra su aspecto.

El campo del formulario en el que se recoge la dirección de envío del cliente se denomina `address`. Este campo nos da una variable a la que podemos acceder como `$address` al procesar el formulario en PHP, si hemos activado el parámetro `register_globals` o como `$_POST['address']` o `$_GET['address']` si el parámetro `register_globals` no está activado (consulte el capítulo 1 si desea obtener más detalles).

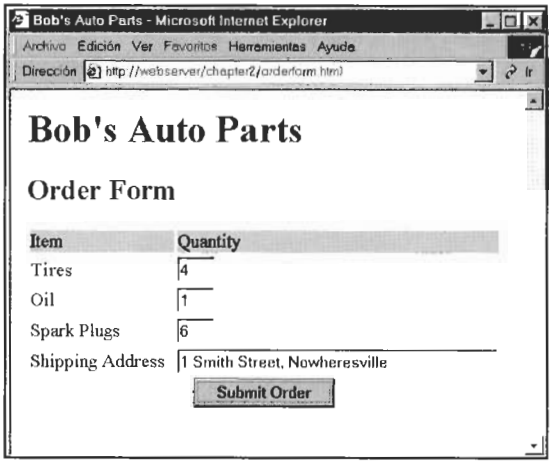


Figura 2.1. Esta versión del pedido obtiene la dirección de envío del cliente

Escribiremos cada pedido que entre en el mismo archivo. Seguidamente, construiremos una interfaz Web para permitir que la plantilla de Bob pueda ver los pedidos que entran.

Introducción al procesamiento de archivos

La operación de escribir datos en un archivo incluye los siguientes pasos:

1. Abrir el archivo. Si el archivo no existiese, tendríamos que crearlo.
2. Escribir los datos en el archivo.
3. Cerrar el archivo.

La operación de lectura de los datos de un archivo también incluye de tres pasos:

1. Abrir el archivo. Si no se puede abrir (por ejemplo, si no existiese), tendríamos que reorganizar el proceso y salir con elegancia.
2. Leer los datos del archivo.
3. Cerrar el archivo.

Cuando queremos leer los datos de un archivo, podemos determinar qué cantidad de elementos del archivo leer cada vez. En una sección posterior examinaremos dichas opciones.

En primer lugar vamos a comenzar por abrir un archivo.

Cómo abrir un archivo

Para abrir un archivo en PHP, se utiliza la función `fopen()`. Al abrir un archivo, tenemos que especificar para qué tenemos pensado utilizarlo. Es lo que se conoce como los modos de archivo.

Modos de archivo

El sistema operativo del servidor necesita saber qué queremos hacer con el archivo que vamos a abrir. Necesitamos saber si otra secuencia de comandos puede abrir el archivo que ya tenemos abierto y determinar si el titular de la secuencia de comandos dispone de permiso para utilizarlo de esa forma. Básicamente, los modos de archivo ofrecen al sistema operativo un mecanismo para determinar la forma de procesar las peticiones de acceso procedentes de otras personas o secuencias de comandos, y un método para comprobar si dispone de acceso y permiso para utilizar el archivo.

Al abrir un archivo dispone de tres opciones:

1. Puede abrir un archivo para leerlo para escribir en él o para ambas acciones.
2. Si está escribiendo en un archivo, puede sobrescribir los contenidos existentes o adjuntar nuevos datos al final del archivo.
3. Si está intentando escribir en un archivo sobre un sistema operativo que diferencie entre archivos binarios y archivos de texto, puede que desee especificar esta circunstancia.

La función `fopen()` admite combinar estas tres opciones.

Uso de fopen() para abrir un archivo

Supongamos que desea escribir el pedido de un cliente en el archivo de pedidos de Bob. Puede abrir el archivo para escribir el pedido con siguiente secuencia:

```
$fp = fopen("$DOCUMENT_ROOT/../orders/orders.txt", "w");
```

Al llamar a `fopen`, se esperan dos o tres parámetros. Por regla general se utilizarán dos, como se muestra en la línea de código.

El primero debería ser el archivo que queremos abrir. Puede especificar una ruta hasta este archivo como hemos hecho en el código anterior; nuestro archivo `orders.txt` se encuentra dentro del directorio de pedidos. Hemos utilizado la variable incorporada `$HTTP_SERVER_VARS['DOCUMENT_ROOT']` pero, como ocurre con los incómodos nombres completos de las variables de formulario, le hemos asignado un nombre más corto.

Esta variable apunta a la base del árbol de documentos de su servidor Web. Hemos utilizado "`..`" para indicar "el directorio superior del directorio raíz del documento".

Este directorio se encuentra fuera del árbol de directorios, por razones de seguridad. No queremos que este archivo resulte accesible desde la Web salvo a través de una interfaz que proporcionaremos nosotros. Esta ruta se conoce como ruta relativa ya que describe una posición en el sistema de archivos en relación al directorio del documento.

Como ocurre con los nombres cortos que asignamos a las variables de formulario, si no activamos el parámetro `register_globals`, necesitamos incluir la siguiente línea al principio de la secuencia de comandos:

```
$DOCUMENT_ROOT = $HTTP_SERVER_VARS['DOCUMENT_ROOT'];
```

para copiar los contenidos de la variable de estilo largo a la variable de estilo corto.

Al igual que existen varias formas de acceder a los datos de formulario, existen varias formas de acceder a las variables de servidor predefinidas. En función de la configuración del servidor puede acceder a la raíz del documento a través de:

- $DOCUMENT_ROOT
- $_SERVER['DOCUMENT_ROOT']
- $HTTP_SERVER_VARS['DOCUMENT_ROOT']

Como en el caso de los datos de formulario, el primer estilo, que hemos llamado estilo corto, está disponible automáticamente si ha activado el parámetro `register_globals`. El segundo estilo (el estilo intermedio) no se puede desactivar, pero sólo está disponible en PHP 4.1 y versiones posteriores. El estilo largo está disponible para todos los sistemas, pero se considera obsoleto por lo que es posible que desaparezca en un futuro.

También se especifica una ruta absoluta al archivo. Ésta es la ruta desde el directorio raíz (/ en los sistemas UNIX y, por regla general, `c:\` en Windows). En nuestro servidor UNIX, sería `/home/book/orders`. El problema de utilizar este método es que si alojamos el sitio en el servidor de un tercero, la ruta absoluta podría variar. A nosotros nos tocó aprenderlo de la forma difícil ya que tuvimos que modificar las rutas absolutas de una gran cantidad de secuencias de comandos cuando los administradores del sistema decidieron cambiar la estructura de directorios sin previo aviso.

Si no se especifica la ruta, el archivo se creará o se buscará en el mismo directorio que la propia secuencia de comandos. Ésta será diferente si está ejecutando PHP a través de algún tipo de contenedor CGI y ello dependerá de la configuración del servidor.

En un entorno UNIX, las barras de los directorios son las barras estándar (/). Si está utilizando la plataforma Windows, puede usar estas barras o las barras invertidas (\). Si utiliza estas últimas, debe marcarlas como caracteres de escape para que `fopen` puede entenderlas correctamente. Para marcar un carácter como carácter especial, basta con agregar otra barra invertida delante, como se muestra a continuación:

```
$fp = fopen("$DOCUMENT_ROOT\\..\\orders\\orders.txt", 'w');
```

Las barras invertidas no se suelen utilizar para indicar rutas en PHP porque el código sólo funcionaría en Windows. El uso de la barra estándar permite mover código entre equipos Windows y UNIX sin variaciones.

El segundo parámetro de `fopen()` es el modo de archivo, que debería ser una cadena. Ésta especifica qué hacer con el archivo. En este caso vamos a pasar 'w' a `fopen()`, que significa abrir el archivo para escritura. En la tabla 2.1 se describen los distintos modos de archivo.

Tabla 2.1. Resumen de los modos de archivo de fopen

Modo	Significado
r	Modo de lectura: abre el archivo para la lectura, empezando por la parte inicial del archivo.
r+	Modo de lectura: abre el archivo para lectura y escritura, empezando por la parte inicial del archivo.
w	Modo de escritura: abre el archivo para escritura, empezando por la parte inicial del archivo. Si el archivo ya existiese, elimina sus contenidos. Si no existiese, intenta crearlo.
w+	Modo de escritura: abre el archivo para escritura y lectura. Si el archivo ya existiese, elimina sus contenidos. Si no existiese, intenta crearlo.
a	Modo de adjunción: abre el archivo para adjuntar (escribir) únicamente, empezando por la parte final de los contenidos existentes. Si no existiese, intenta crearlo.
a+	Modo de adjunción: abre el archivo para adjuntar (escribir) y leer, empezando por la parte final de los contenidos existentes. Si no existiese, intenta crearlo.
b	Modo binario: se utiliza en combinación con uno de los otros modos. Puede utilizarlo si su sistema de archivos distingue entre archivos binarios y archivos de texto. Los sistemas Windows hacen esta diferencia al contrario que los sistemas UNIX.

El modo de archivo para nuestro ejemplo depende de cómo se utilice el sistema. En nuestro caso, hemos utilizado 'w', que sólo permite almacenar un pedido en el archivo. Cada vez que se tome un nuevo pedido, se sobrescribirá el anterior. Esta opción no parece muy adecuada por lo que haremos mejor en seleccionar el modo de adjunción.

```
$fp = fopen("$DOCUMENT_ROOT/../orders/orders.txt", 'a');
```

La función `fopen()` consta de un tercer parámetro opcional. Puede utilizarlo si desea buscar el parámetro `include_path` (establecido en la configuración de PHP; a este respecto consulte el apéndice A) de un archivo. Si desea activar esta opción,

asigne 1 al parámetro. De esta forma no tendrá que suministrar un nombre o una ruta de directorio:

```
$fp = fopen('orders.txt', 'a', 1);
```

Si `fopen()` se abre satisfactoriamente, se devolverá un puntero al archivo que debería guardarse en una variable, en este caso `$fp`. Esta variable se utiliza para acceder al archivo cuando se desee leer o escribir en él.

Cómo abrir archivos a través de FTP o HTTP

Además de abrir archivos locales para su lectura o escritura, puede abrir archivos a través de FTP y HTTP utilizando `fopen()`.

Si el nombre del archivo utilizado comienza por `ftp://`, se abrirá una conexión FTP en modo pasivo al servidor especificado y se devolverá un puntero al inicio del archivo.

Si el nombre del archivo utilizado comienza por `http://`, se abrirá una conexión HTTP al servidor especificado y se devolverá un puntero a la respuesta. Al utilizar el modo HTTP, debe especificar las barras situadas al final de los nombres de directorios, como se muestra a continuación:

```
http://www.server.com/
```

no

```
http://www.server.com
```

Cuando se especifica la última forma de dirección (sin la barra), el servidor Web suele utilizar una redirección HTTP para enviarle a la primera dirección (con barra). Pruebe en su navegador.

En las versiones de PHP anteriores a la 4.0.5, la función `fopen()` no admitía redirecciones HTTP, por lo que debe especificar los URL a los que hacen referencia los directorios con una barra final.

A partir de la versión 4.3.0 también puede abrir archivos sobre SSL siempre y cuando se haya compilado o habilitado la compatibilidad para OpenSSL y utilice la secuencia `https://` delante del nombre del archivo.

Recuerde que los nombres de dominio de su URL no discriminan entre mayúsculas y minúsculas pero que la ruta y el nombre del archivo puede que sí lo hagan.

Problemas al abrir el archivo

Un error que suele producirse de manera habitual es intentar abrir un archivo para el que no se disponga de permiso de lectura o escritura. PHP generará un aviso similar al que se ilustra en la figura 2.2.

Si recibe este error, debe asegurarse de que el usuario utilizado para ejecutar la secuencia de comandos dispone de permiso para acceder al archivo que está inten-

tando utilizar. En función de cómo se configure el servidor, la secuencia de comandos puede que se ejecute como el usuario de servidor Web o como el propietario del directorio en el que se incluye la secuencia de comandos.

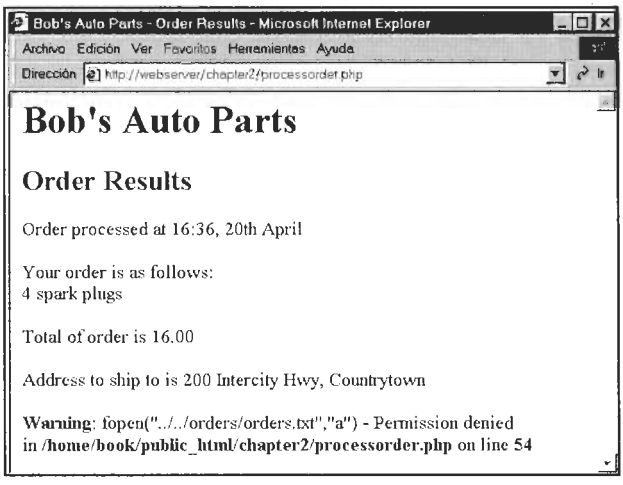


Figura 2.2. PHP le avisa específicamente cuando no puede abrir un archivo

En la mayor parte de los sistemas, la secuencia de comandos se ejecuta como el usuario de servidor Web. Si su secuencia de comandos estuviera en un sistema UNIX dentro del directorio `~/public_html/chapter2/`, se crearía un directorio sobre el que todo el mundo podría escribir y almacenar el pedido mediante la siguiente secuencia:

```
mkdir  ~/orders
chmod  777  ~/orders
```

Tenga en cuenta que los directorios y los archivos sobre los que puede escribir todo el mundo son peligrosos. No debería utilizar directorios susceptibles de escritura que resulten accesibles desde la Web. Por esta razón, nuestro directorio `orders` se encuentra dos subdirectorios más atrás, por encima del directorio `public_html`. El tema de la seguridad se comentará en más profundidad en un capítulo posterior.

Uno de los elementos que suele dar errores al abrir un archivo es el uso de permisos incorrectos, aunque no es el único problema. Si el archivo no se puede abrir, necesitará saberlo para no intentar leer o escribir datos en él.

Si la llamada a `fopen()` falla, la función devolverá `false`. Puede tratar el error de forma que resulte más significativo sustituyendo el mensaje de PHP por otro.

```
@  $fp  =  fopen("$DOCUMENT_ROOT/../orders/orders.txt",  'a',  1);

if  (!$fp)
{
   echo  '<p><strong> Your  order  could  not  be  processed  at  this
```

```
time.'
            .'Please try again later.</strong></p></body></html>';
    exit;
}
```

El símbolo @ situado delante de la llamada a `fopen()` indica a PHP que suprima todos los errores producidos por la llamada de función. Por regla general, conviene saber cuándo ocurre algún error, pero en este caso vamos a tratar el problema en otro lugar.

Esta línea también se puede escribir de la siguiente forma:

```
$fp = @fopen("$DOCUMENT_ROOT/../orders/orders.txt", "a", 1);
```

Sin embargo, con esta secuencia resulta menos obvio que se está utilizando el operador de supresión de errores.

En un capítulo posterior se tratará el tema de la generación de informes de error de manera más detallada.

La instrucción `if` comprueba la variable `$fp` para determinar si se ha devuelto un puntero válido de archivo desde la llamada `fopen`. En caso negativo, imprime un mensaje de error y concluye la ejecución de la secuencia de comandos. Como la página termina aquí, se ha utilizado etiqueta HTML de cierre para que código HTML resulte correcto.

En la figura 2.3 se ilustra el resultado de este enfoque.

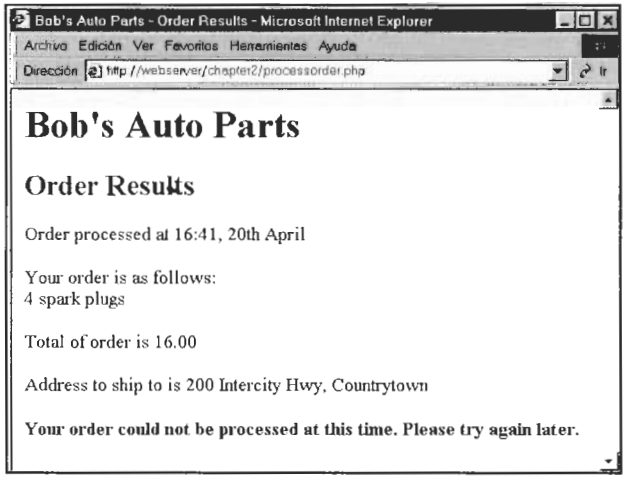


Figura 2.3. El uso de propios mensajes de error propios en lugar de los de PHP puede mejorar la comprensión de los errores

Cómo escribir en un archivo

La operación de escribir en un archivo en PHP resulta relativamente sencilla. Puede utilizar la función `fwrite()` (escribir archivo) o la función `fputs()` (cadena

de archivo); `fputs()` es un alias de `fwrite()`. En el siguiente ejemplo llamamos a la función `fwrite()`:

```
fwrite($fp, $outputstring);
```

Esta línea indica a PHP que escriba la cadena almacenada en `$outputstring` en el archivo al que apunta `$fp`. Antes de analizar los contenidos de `$outputstring`, vamos a comentar la función `fwrite()`.

Parámetros de fwrite()

La función `fwrite()` toma tres parámetros aunque el tercero es opcional. Su sintaxis es la siguiente:

```
int fwrite ( int fp, string cadena [, int longitud])
```

El tercer parámetro, *longitud*, es el número máximo de bytes que escribir. Si se incluye este parámetro, `fwrite()` escribirá la cadena en el archivo al que apunte el parámetro *fp* hasta que alcance el final de la cadena o haya escrito la longitud de bytes, dependiendo de qué venga primero.

Formatos de archivo

Al crear un archivo de datos como el utilizado en el ejemplo, la decisión sobre el formato en el que almacenar los datos será nuestra. (Sin embargo, si tiene previsto utilizar el archivo de datos en otra aplicación, puede que necesite seguir las reglas de dicha aplicación.)

Vamos a crear una cadena que represente un registro en nuestro archivo de datos. Para ello, podemos utilizar el siguiente código:

```
$outputstring = $date."\t".$tireqty." tires \t".$oilqty." oil\t"
        .$sparkqty." spark plugs\t\$".$total
        ."\t". $address."\n";
```

En nuestro sencillo ejemplo, estamos almacenando cada registro de pedido en una línea separada del archivo. Hemos optado por escribir un registro por línea porque de esta forma obtenemos un separador de registros sencillo en forma del carácter de nueva línea. Como las nuevas líneas son invisibles, las representamos con la secuencia de control "\n".

Escribiremos los campos de datos en el mismo orden y utilizaremos el carácter de tabulación para separar los campos. De nuevo, como el carácter de tabulación es invisible, se representa mediante la secuencia de control "\t". Puede seleccionar cualquier delimitador lógico que resulte sencillo de distinguir.

El separador o delimitador no debe ser un carácter que se utilice como entrada ya que debería procesarse para eliminar o utilizar caracteres de escape en todas las instancias del limitador. En un capítulo posterior se analizará cómo procesar las entradas del usuario. Por el momento asumiremos que no se van a introducir tabuladores en un formulario de pedido. Aunque resulta difícil, no es imposible que un usuario introduzca un tabulador o un carácter de nueva línea en un campo de entrada HTML de una sola línea.

El uso de un separador especial de campo nos permitirá dividir los datos en variables distintas de manera más sencilla al leer los datos, como veremos en capítulos posteriores.

Por el momento, trataremos cada pedido como una única cadena.

Tras procesar varios pedidos, los contenidos del archivo se parecerán al ejemplo ilustrado en el listado 2.1.

Listado 2.1. Ejemplo de lo que podría contener el archivo de pedidos

```
15:42, 20th April   4 tires  1 oil 6 spark plugs   $434.00 22 Short St, Smalltown
15:43, 20th April   1 tires  0 oil 0 spark plugs   $100.00 33 Main Rd, Newtown
15:43, 20th April   0 tires  1 oil 4 spark plugs   $26.00  127 Acacia St, Springfield
```

Cómo cerrar un archivo

Cuando se termina de utilizar un archivo, es necesario cerrarlo. Debería hacerlo utilizando la función `fclose()` de la siguiente forma:

```
fclose($fp);
```

Esta función devolverá `true` si el archivo se cerró satisfactoriamente o `false` en caso contrario.

Como la probabilidad de que surja un problema con esta función es muy inferior a la operación de apertura, no procederemos a probarla.

Lectura desde un archivo

Llegados a este punto, los clientes de Bob pueden remitir sus pedidos a través de la Web, pero si los empleados de Bob quieren examinar los pedidos, tendrán que abrir los archivos manualmente.

Vamos a crear una interfaz Web para permitir que los empleados de Bob puedan leer los archivos de manera sencilla. El código correspondiente a esta interfaz se incluye en el listado 2.2.

Listado 2.2. vieworders.php. Interfaz para leer el archivo de pedidos

```
<?php
  //cree un nombre de variable corto
  $DOCUMENT_ROOT = $HTTP_SERVER_VARS['DOCUMENT_ROOT'];
?>
<html>
<head>
  <title>Bob's Auto Parts - Customer Orders</title>
</head>
<body>
```

```
<h1>Bob's Auto Parts</h1>
<h2>Customer Orders</h2>
<?php
@ $fp = fopen("$DOCUMENT_ROOT/../orders/orders.txt", 'r');

    if (!$fp)
    {
        echo '<p><strong>No orders pending.'
             .'Please try again later.</strong></p>';
        exit;
    }

    while (!feof($fp))
    {
        $order= fgets($fp, 999);
        echo $order.'<br />';
    }

    fclose($fp);
?>
  </body>
  </html>
```

Esta secuencia de comandos sigue el orden expuesto anteriormente: abre el archivo, lee el archivo y cierra el archivo. En la figura 2.4 se ilustra el resultado de esta secuencia de comandos utilizando el archivo de datos del listado 2.1.

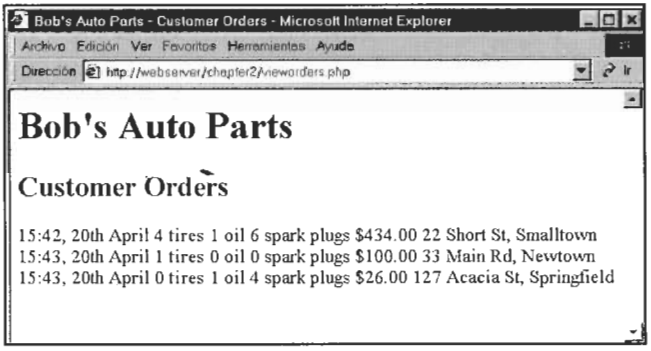


Figura 2.4. La secuencia de comandos vieworders.php muestra todos los pedidos actuales del archivo orders.txt en la ventana del navegador

Vamos a examinar las funciones de esta secuencia de comandos en detalle.

Apertura de un archivo para su lectura: fopen()

De nuevo, abrimos el archivo utilizando `fopen()`. En este caso abrimos el archivo para su lectura únicamente, por lo que utilizamos el modo de archivo `'r'`:

```
$fp = fopen("$DOCUMENT_ROOT/../orders/orders.txt", 'r');
```

Cómo saber cuándo parar: feof()

En este ejemplo, utilizamos el bucle `while` para leer el archivo hasta alcanzar el final del archivo. El bucle `while` busca el final del archivo utilizando la función `feof()`:

```
while (!feof($fp))
```

La función `feof()` toma un puntero de archivo como su único parámetro. Devuelve `true` si el puntero de archivo se encuentra al final del archivo.

En este caso (y en general al leer desde un archivo), leemos desde el archivo hasta que alcanzamos `EOF`.

Cómo leer línea a línea: fgets(), fgetss() y fgetcsv()

En nuestro ejemplo, utilizamos la función `fgets()` para leer desde un archivo:

```
$order= fgets($fp, 999);
```

Esta función se utiliza para leer, línea a línea, un archivo. En este caso, leerá hasta que encuentre un nuevo carácter de línea (`\n`), un `EOF` o haya leído 998 bytes del archivo. La longitud de lectura es la longitud menos un byte.

Se pueden utilizar muchas funciones diferentes para leer archivos. La función `fgets()` resulta útil para tratar con archivos que contengan texto plano con el que queramos trabajar en grupos.

Una variación interesante de esta función es la función `fgetss()`, cuya sintaxis es la siguiente:

```
string fgetss(int fp, int longitud, string [etiquetas_permitidas]);
```

Esta función es muy parecida a la función `fgets()` con la salvedad de que elimina todas las etiquetas PHP y HTML que encuentra. Si desea dejar alguna etiqueta en particular, puede incluirlas en la cadena *etiquetas_permitidas*. Se utiliza `fgetss()` por seguridad al leer un archivo escrito por otra persona o que contenga entradas de usuario. La inclusión de código HTML sin restricciones en el archivo puede estropear el formato atentamente organizado y la inclusión de código PHP podría ofrecer rienda suelta a un usuario malintencionado sobre su servidor.

La función `fgetcsv()` es otra variación sobre `fgets()`. Su sintaxis es la siguiente:

```
array fgetcsv ( int fp, int longitud [, string delimitador [, string
contenedor]])
```

Se utiliza para dividir las líneas del archivo si se ha utilizado un carácter de delimitación, como el carácter de tabulación sugerido antes o una coma como se suele hacer en las hojas de cálculo y otras aplicaciones. Si desea reconstruir las variables del pedido de manera individual en lugar de en forma de línea de texto, `fgetcsv()`

nos permite hacerlo con facilidad. Se invoca de la misma forma que `fgets()`, pero se le pasa el delimitador utilizado para separar campos. Por ejemplo:

```
$order = fgetcsv($fp, 100, "\t");
```

recupera una línea desde el archivo y la divide al encontrar un tabulador (`\t`). Los resultados se devuelven en una matriz (`$order` en el ejemplo de código). En un capítulo posterior examinaremos las matrices.

El parámetro de longitud debería ser mayor que la longitud en caracteres de la línea más larga que incluya el archivo que esté intentando leer.

El parámetro *contenedor* se utiliza para especificar por qué está encerrado cada campo en una línea. Si no se especifica, se tomarán las comillas dobles (") como elemento predeterminado. Este parámetro se agregó en PHP 4.3.0.

Lectura de todo el archivo: readfile(), fpassthru(), file()

En lugar de leer un archivo línea a línea, podemos leer todo el archivo de una vez. Esta operación se puede realizar de cuatro maneras.

La primera utiliza la función `readfile()`. Podemos sustituir la secuencia de comandos entera que escribimos anteriormente por una línea:

```
readfile("$DOCUMENT_ROOT/../orders/orders.txt");
```

Una llamada a la función `readfile()` abre el archivo, imprime el contenido de manera estándar (en el navegador) y cierra el archivo. Su sintaxis es la siguiente:

```
int readfile(string nombre_de_archivo, int [usar_include_path]);
```

El segundo parámetro opcional especifica si PHP debería buscar el archivo en `include_path` y funciona de la misma forma que `fopen()`. La función devuelve el número total de bytes leídos desde el archivo.

En segundo lugar, puede utilizar `fpassthru()`. Necesitará abrir el archivo utilizando `fopen()` primero. A continuación puede utilizar el puntero del archivo como argumento hasta `fpassthru()`, que eliminará los contenidos del archivo desde la posición del puntero hasta la salida estándar. Cierra el archivo cuando termina.

Podemos sustituir la secuencia de comandos anterior por `fpassthru()` de la siguiente forma:

```
$fp = fopen("$DOCUMENT_ROOT/../orders/orders.txt", 'r');
fpassthru($fp);
```

La función `fpassthru()` devuelve `true` si la lectura resulta satisfactoria y `false` en caso contrario.

La tercera opción para leer el archivo entero consiste en utilizar la función `file()`. Esta función es idéntica a `readfile()` con la salvedad de que en lugar de imprimir el archivo en la aplicación estándar, lo convierte en una matriz. Examinaremos esta

opción de manera más detallada en un capítulo posterior. Simplemente como referencia, se invoca utilizando el siguiente código:

```
$filearray = file($fp);
```

Esta línea de código lee el archivo entero en una matriz llamada `$filearray`. Cada línea del archivo se almacena en un elemento separado de una matriz. Esta función no es segura para datos binarios.

Por último, desde la versión PHP 4.3.0 puede utilizar la función `file_get_contents()`. Esta función es idéntica a `readfile()` con la salvedad de que devuelve el contenido de un archivo en forma de cadena en lugar de dirigir la salida al navegador. La ventaja de esta nueva función es que ofrece seguridad binaria, a diferencia de la función file().

Lectura de un carácter: fget()+

Otra opción para procesar un archivo consiste en leerlo carácter a carácter. Para ello, se puede utilizar la función `fgetc()`. Esta función toma un puntero de archivo como único parámetro y devuelve el siguiente carácter de un archivo. Podemos sustituir el bucle `while` en nuestra secuencia de comandos original por una que utilice `fgetc()`:

```
while  (!feof($fp))
{
   $char = fgetc($fp);
   if (!feof($fp))
      echo ($char=="\n" ? '<br />': $char);
}
```

Este código lee un solo carácter del archivo a la vez utilizando `fgetc()` y lo almacena en la variable `$char` hasta que se alcanza el final del archivo. A continuación, sustituimos los caracteres de final de línea del texto, `/n`, por saltos de línea, `<br />`. El único objetivo es limpiar el formato. Como los navegadores no representan una línea nueva en HTML como tal sin este código, todo el archivo se imprimirá en una línea (pruébelo). Utilizaremos el operador ternario para realizar esta operación de manera satisfactoria.

Un pequeño efecto secundario producto del uso de `fgetc()` en lugar de `fgets()` es que devolverá el carácter `EOF` mientras que `fgets()` no lo hará. Es necesario probar `feof()` de nuevo tras la lectura del carácter para evitar que se imprima `EOF` en el navegador.

No resulta normal leer un archivo carácter a carácter a menos que queramos procesar cada uno de ellos por alguna razón concreta.

Lectura de una longitud arbitraria de bytes: fread()

Por último, podemos utilizar la función `fread()` para leer un número arbitrario de bytes de un archivo.

Su sintaxis es la siguiente:

```
string fread(int fp, int longitud);
```

Funciona de la siguiente forma: lee la longitud de bytes que se especifique o hasta llegar al final del archivo, según qué sea lo primero.

Otras funciones de archivo útiles

Existen otra serie de funciones de archivo que resultan útiles de vez en cuando.

Comprobación de la existencia de un archivo: file_exists()

Si desea comprobar si un archivo existe sin abrirlo, puede utilizar la función `file_exists()` como se indica a continuación:

```
if (file_exists("$DOCUMENT_ROOT/../orders/orders.txt"))
      echo 'There are orders waiting to be processed.';
else
      echo 'There are currently no orders.';
```

Cómo averiguar el tamaño de un archivo: filesize()

La función `filesize()` permite comprobar el tamaño de un archivo. Esta función devuelve el tamaño del un archivo en bytes:

```
echo filesize("$DOCUMENT_ROOT/../orders/orders.txt");
```

Se puede utilizar en combinación con `fread()` para leer todo un archivo (o parte de un archivo). Podemos sustituir toda nuestra secuencia de comandos por

```
$fp = fopen("$DOCUMENT_ROOT/../orders/orders.txt", 'r');
echo fread( $fp, filesize("$DOCUMENT_ROOT/../orders/orders.txt" ));
fclose( $fp );
```

Eliminación de un archivo: unlink()

Si desea eliminar un archivo tras procesar los pedidos, puede hacerlo con ayuda de la función `unlink()`. (No existe una función llamada `delete`.) Por ejemplo:

```
unlink("$DOCUMENT_ROOT/../orders/orders.txt");
```

Esta función devuelve `false` si el archivo no se puede eliminar, lo que suele ocurrir si los permisos asociados al archivo no resultan suficientes o si no existe el archivo.

Navegación dentro de un archivo: rewind(), fseek() y ftell()

Puede manipular y descubrir la ubicación del puntero del archivo dentro de un archivo utilizando `rewind()`, `fseek()` y `ftell()`.

La función `rewind()` restablece el puntero del archivo al comienzo del archivo. La función `ftell()` indica la distancia a la que se encuetra el puntero dentro del archivo en bytes. Por ejemplo, podemos agregar las siguientes líneas a la parte final de nuestra secuencia de comandos (antes del comando `fclose()`).

```
echo 'Final position of the file pointer is '.(ftell($fp));
echo '<br />';
rewind($fp);
echo 'After rewind, the position is '.(ftell($fp));
echo '<br />';
```

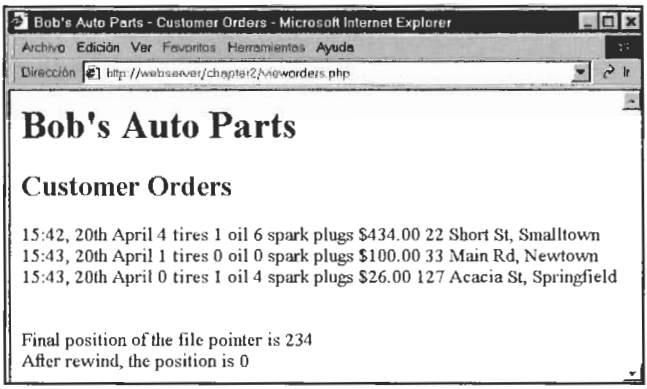


Figura 2.5. Tras leer los pedidos, el puntero apunta al final del archivo, una distancia de 234 bytes. La llamada de retroceso lo establece a la posición 0, es decir al principio del archivo

La función `fseek()` se puede utilizar para establecer el puntero del archivo en otro punto dentro del archivo. Su sintaxis es la siguiente:

```
int fseek ( int fp, int desplazamiento [, int punto de partida])
```

Una llamada a `fseek()` establece el puntero `fp` en un punto desde los que desplaza los bytes indicados. El parámetro *punto de partida* se agregó en PHP 4.0.0. Su valor predeterminado es `SEEK_SET` que es en principio del archivo. Los otros valores posibles son `SEEK_CUR` (la ubicación actual del puntero del archivo) y `SEEK_END` (el final del archivo).

La función `rewind` equivale a llamar a la función `fseek` con un desplazamiento 0. Por ejemplo, podemos utilizar `fseek()` para buscar el registro intermedio de un archivo o para realizar una búsqueda binaria. A menudo, si alcanza el nivel de complejidad de un archivo de datos en el que necesite realizar este tipo de tareas, le resultará mucho más sencillo utilizar una base de datos.

Bloqueo de archivos

Imagine una situación en la que dos clientes intenten realizar un pedido de un producto a la vez. (Algo que no resulta extraño, en especial si el volumen de tráfico del sitio Web comienza a tener cierta importancia.) Piense en qué ocurriría si un cliente llama a la función `fopen()` y comienza a escribir, y entonces otro cliente llama a la misma función y también escribe. ¿Qué contendrá al final el archivo? ¿Será uno de los pedidos o el otro? O algo menos útil, como dos pedidos entremezclados. La respuesta depende del sistema operativo, pero a menudo resulta imposible de saber.

Para evitar problemas como éste, se puede utilizar el bloqueo de archivos. Para implementar este recurso en PHP se utiliza la función `flock()`. Esta función debería invocarse tras abrir un archivo, pero antes de que se lea o escriba ningún dato.

Su sintaxis es la siguiente:

```
bool flock(int fp, int operación [, int &bloquerá])
```

Es necesario pasarle un puntero al archivo abierto y un número que represente el tipo de bloqueo deseado. Devuelve `true` si el bloqueo se logra satisfactoriamente y `false` si no es así.

En la tabla 2.2 se recoge un resumen con los valores posibles del parámetro *operación*. Estos valores cambiaron en la versión 4.0.1 de PHP. En la tabla se recopilan ambos conjuntos de valores.

Tabla 2.2. Valores de operación de la función flock()

Valor de operación	Significado
LOCK_SH (antes 1)	Bloqueo de lectura. El archivo se puede compartir con otros lectores.
LOCK_EX (antes 2)	Bloqueo de escritura. Es exclusivo. El archivo no se puede compartir.
LOCK_UN (antes 3)	Liberar bloqueo existente.
LOCK_NB (antes 4)	Si se agrega el número 4 a la operación, se impide que el bloqueo al intentar obtener uno.

Si va a utilizar `flock()`, tendrá que agregarlo a todas las secuencias de comandos que utilicen el archivo ya que de lo contrario carecerá de valor.

`flock()` no funciona con el sistema de archivos NFS ni con otros sistemas de archivos de red. Tampoco se puede utilizar con sistemas de archivos antiguos que no admitan bloqueos como FAT. En algunos sistemas operativos se implementa en el nivel de proceso y no funcionará correctamente si utiliza un API de servidor multiproceso.

Para utilizarlo en el ejemplo, es necesario modificar `processorder.php` de la siguiente forma:

```
$fp = fopen("$DOCUMENT_ROOT/../orders/orders.txt", 'a');
flock($fp, LOCK_EX); // bloquee el archivo para su escritura
fwrite($fp, $outputstring);
flock($fp, LOCK_UN); // desactive el bloqueo de escritura
fclose($fp);
```

También debería agregar bloqueos a `vieworders.php`:

```
$fp = fopen("$DOCUMENT_ROOT /../orders/orders.txt", 'r');
flock($fp, LOCK_SH); // bloquee el archivo para lectura
// lee desde el archivo
flock($fp, LOCK_UN); // desactive el bloqueo de lectura
fclose($fp);
```

Nuestro código resulta ahora más sólido pero no es perfecto. ¿Qué ocurrirá si dos secuencias de comandos intentan obtener un bloqueo a la vez? Que se creará una carrera en la que los procesos competirán por conseguir bloqueos sin saber quién lo logrará primero, lo que puede ser fuente de más problemas. Lo mejor es un utilizar un DBMS.

La opción más acertada: los sistemas de administración de base de datos

Hasta el momento todos los ejemplos vistos utilizan archivos planos.

En la siguiente sección examinaremos cómo utilizar MySQL, un sistema de administración de bases de datos relacional.

Es posible que se esté preguntando por qué molestarse en recurrir a este sistema.

Problemas con el uso de archivos planos

El uso de archivos planos plantea una serie de problemas:

- Cuando un archivo supera un determinado tamaño, el trabajo se ralentiza.
- La búsqueda de un determinado registro o de un grupo de ellos resulta difícil dentro de un archivo plano. Si los registros están ordenados, puede utilizar algún tipo de búsqueda binaria en combinación con un registro de ancho fijo para buscar sobre un campo clave. Si desea buscar patrones de información (por ejemplo, para extraer todos los clientes que viven en una determinada ciudad), tendrá que leer cada registro y comprobarlo de manera individual.

- Los accesos simultáneos pueden resultar difíciles de solucionar. Ya hemos visto cómo bloquear archivos, pero este mecanismo puede dar lugar a una carrera entre procesos. También pueden dar lugar a cuellos de botella. En determinados niveles de tráfico, puede ocurrir que un grupo grande de usuarios tenga que esperar a que se desbloquee el archivo para poder realizar un pedido. Si la espera es muy larga, se irán a comprar a otra parte.
- El procesamiento de archivos que hemos visto hasta el momento es de tipo secuencial, es decir, partimos del principio del archivo y lo leemos progresivamente hasta llegar al final. Si queremos insertar o eliminar registros de la parte central del archivo (acceso aleatorio), la operación resultará complicada ya que será necesario leer todo el archivo en memoria, realizar los cambios y volver a escribir el archivo completo. Si el archivo de datos es grande, la carga será significativa.
- Aparte de las funciones que ofrecen de los permisos de archivos, no existe una forma sencilla de aplicar diferentes niveles de acceso a datos.

Cómo resolver estos problemas

Los sistemas de administración de bases de datos dan una respuesta a todos estos problemas:

- Los RDBMS proporcionan acceso más rápido a los datos que los archivos planos. MySQL, el sistema de base de datos que utilizaremos en este libro, es uno de los más rápidos.
- Resulta sencillo consultar los RDBMS para extraer conjuntos de datos que se correspondan con determinados criterios.
- Los RDBMS incorporan mecanismos integrados para resolver el problema de los accesos simultáneos para que los programadores no tengan que preocuparse.
- Los RDBMS proporcionan acceso aleatorio a los datos.
- Los RDBMS incorporan sistemas de privilegios. MySQL dispone de funciones especiales en este sentido.

Probablemente, la razón principal para utilizar un RDBMS es que implementan todas las funciones (o al menos la mayor parte de ellas) que se desean para un sistema de almacenamiento.

Puede escribir su propia biblioteca de funciones de PHP, pero ¿por qué tomarse esa molestia?

En la segunda parte de este libro, se analizará el funcionamiento general de las bases de datos y cómo configurar MySQL para crear sitios Web dotados de bases de datos.

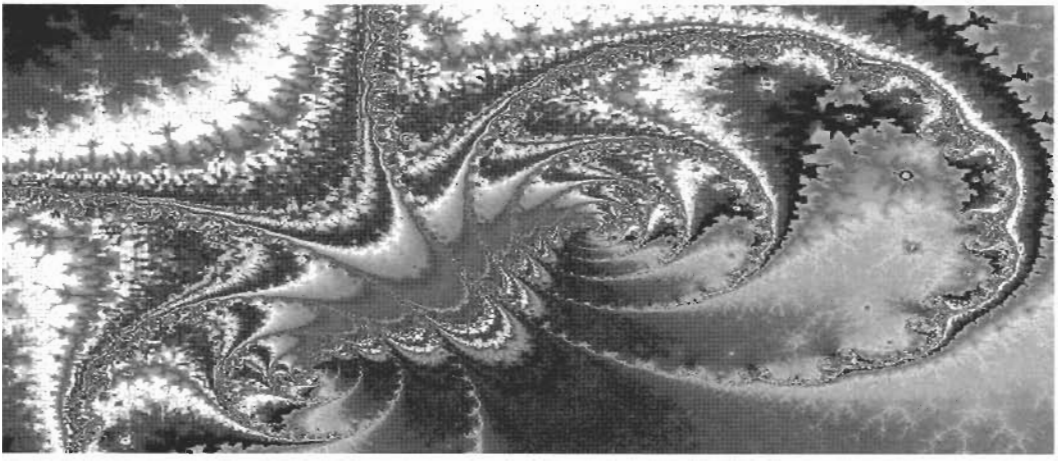
Lecturas adicionales

En un capítulo posterior veremos cómo interactuar con los sistemas de archivos. En concreto, se indicará cómo cambiar permisos, derechos y nombres de archivos, cómo trabajar con directorios y cómo interactuar con el entorno del sistema de archivos.

También puede consultar la sección dedicada al sistema de archivos del manual en línea de PHP en `http://www.php.net/filesystem`.

A continuación

En el siguiente capítulo, analizaremos las matrices y cómo utilizarlas para procesar datos en secuencias de comandos PHP.



3
Uso de matrices

En este capítulo se explica cómo utilizar una importante estructura de programación: las matrices. Las variables que hemos visto en los capítulos anteriores sólo almacenan un valor. Una matriz es una variable capaz de almacenar un conjunto o secuencia de valores. Una matriz puede constar de una gran cantidad de elementos. Cada elemento puede albergar un único valor, como texto o números, u otra matriz. Las matrices que contienen otras matrices se conocen como matrices multidimensio-nales.

PHP admite matrices indexadas numéricamente y matrices asociativas. Es probable que esté familiarizado con las matrices indexadas numéricamente si ha utilizado algún lenguaje de programación, pero si no ha programado nunca en PHP o en Perl es muy probable que no conozca las matrices asociativas. Estas matrices permiten utilizar valores más útiles como índice. Las matrices asociativas permiten asociar a cada elemento palabras u otro tipo de información con significado en lugar de índices numéricos. En este capítulo retomaremos el ejemplo de la aplicación Bob's Auto Parts y utilizaremos matrices para facilitar el trabajo con información repetitiva como los pedidos de los clientes. Así mismo, escribiremos código más breve y claro para realizar parte de las tareas desarrolladas en el capítulo anterior. Nos centraremos en los siguientes aspectos:

- Matrices indexadas numéricamente
- Matrices asociativas
- Matrices multidimensionales
- Ordenación de matrices

¿Qué es una matriz?

En un capítulo anterior vimos las variables escalares. Una variable escalar es una ubicación con nombre en la que se almacena una variable; de manera similar, un matriz es una ubicación con nombre para almacenar un conjunto de valores, que, por tanto, permite agrupar variables escalares.

Utilizaremos la lista de productos de Bob para crear la matriz de nuestro ejemplo.

En la figura 3.1 se puede ver una lista de tres productos almacenados en formato de matriz y una variable, denominada `$products`, que contiene tres valores. (En un instante explicaremos cómo crear un variable como ésta).

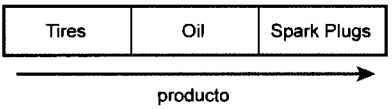


Figura 3.1. Los productos de Bob se pueden almacenar en una matriz

Tras colocar la información en una matriz, podemos trabajar con ella de varias formas útiles. Si utilizamos la estructura de bucle vista en un capítulo anterior, podemos ahorrar trabajo aplicando las mismas acciones a cada valor de la matriz. Podemos trasladar todo el conjunto de información de un sitio a otro como si se tratara de una unidad.

De esta forma, bastará con una sola línea de código para pasar todos los valores a una función. Por ejemplo, podríamos ordenar todos los productos alfabéticamente pasándoselos a la función `sort()` de PHP.

Los valores almacenados en una matriz se denominan elementos de matriz. Cada elemento de matriz lleva asociado un índice (también denominado clave) que se utiliza para acceder al elemento.

Las matrices de la mayor parte de los lenguajes de programación constan de índices numéricos que suelen comenzar en cero o en uno. PHP admite este tipo de matrices.

PHP también admite matrices asociativas con las que estarán familiarizados los programadores de Perl. Estas matrices pueden utilizar prácticamente cualquier valor como índice de matriz, sobre todo cadenas.

Comenzaremos por examinar las matrices indexadas numéricamente.

Matrices indexadas numéricamente

La mayor parte de los lenguajes de programación permite el uso de las matrices indexadas numéricamente. En PHP, los índices comienzan en cero de manera predeterminada pero se puede variar este parámetro.

Inicialización de matrices indexadas numéricamente

Para crear la matriz ilustrada en la figura 3.1, utilice la siguiente línea de código de PHP:

```
$products = array( 'Tires', 'Oil', 'Spark Plugs' );
```

Esta línea de código crea una matriz llamada `products` que contiene los tres valores dados 'Tires', 'Oil' y 'Spark Plugs'. Como en el caso de la instrucción `echo`, `array()` es más una unidad estructural del lenguaje que una función.

Puede que no necesite inicializar manualmente los elementos de una matriz como en el siguiente ejemplo. Todo depende de los contenidos de la matriz.

Si tiene los datos que necesita en otra matriz, puede copiarlos utilizando el operador =.

Si quiere almacenar una secuencia ascendente de números en una matriz, puede utilizar la función `range()` para crear la matriz automáticamente. La siguiente línea de código creará una matriz denominada `numbers` cuyos elementos son los 10 primeros números.

```
$numbers = range(1,10);
```

Si tiene la información almacenada en un archivo del disco duro, puede cargar los contenidos directamente del archivo, como veremos en un sección posterior.

Si los datos destinados a la matriz se almacenan en una base de datos, puede cargar sus contenidos directamente desde la base de datos, como veremos en un capítulo posterior.

También puede utilizar varias funciones para extraer parte de una matriz o para volver a ordenarla. En una sección posterior veremos parte de estas funciones.

Acceso a los conteñidos de matrices

Para acceder a los contenidos de una matriz, utilice su nombre. Si la variable es una matriz, utilice el nombre de variable o el índice para acceder a los contenidos. La clave o el índice indican a qué valores almacenados accedemos. El índice se coloca entre corchetes tras el nombre.

Escriba `$products[0]`, `$products[1]` y `$products[2]` para utilizar los contenidos de la matriz de productos.

El primer elemento de la matriz es el elemento cero. Se trata del mismo sistema de numeración utilizado en C, C++, Java y otros lenguajes, pero puede que al principio le cueste acostumbrarse a él.

Como ocurre con otras variables, para cambiar los contenidos de los elementos de matriz se utiliza el operador =.

La siguiente línea sustituirá el primer elemento de la matriz, `'Tires'`, por `'Fuses'`.

```
$products[0] = 'Fuses';
```

La línea que se incluye a continuación podría utilizarse para agregar un nuevo elemento (`'Fuses'`) al final de la matriz, lo que nos da un total de cuatro elementos:

```
$products[3]  =  'Fuses';
```

Para mostrar los contenidos, podríamos escribir:

```
echo  "$products[0]  $products[1]  $products[2]  $products[3]";
```

Tenga en cuenta que aunque el análisis de cadenas que realiza PHP es bastante inteligente, puede que resulte un poco confuso. Si tiene problemas porque las matrices u otras variables no se interpretan correctamente al encerrarlas entre comillas dobles, puede colocarlas fuera de las comillas. La instrucción `echo` anterior funcionará correctamente, pero en gran parte de los ejemplos posteriores, de mayor complejidad, no se utilizan cadenas con comillas.

Como ocurre con otras variables de PHP, no es necesario inicializar las matrices o crearlas por adelantado ya que se crean automáticamente la primera vez que se utilizan.

El siguiente código creará la misma matriz `$products`:

```
$products[0] = 'Tires';
$products[1] = 'Oil';
$products[2] = 'Spark Plugs';
```

Si la matriz `$products` no existiera todavía, la primera línea crearía una nueva matriz con un único elemento y las siguientes líneas agregarían valores a la matriz.

Uso de bucles para acceder a la matriz

Como la matriz se indexa a partir de una secuencia de números, podemos utilizar un bucle `for` para mostrar los contenidos de manera más sencilla.

```
for ( $i = 0; $i<3; $i++ )
   echo "$products[$i] ";
```

Este bucle produce resultados similares al código anterior, pero exige menos trabajo ya que no es necesario escribir código para trabajar con cada elemento de una matriz. Una de las características destacadas de las matrices indexadas es la posibilidad de utilizar un sencillo bucle para acceder a cada uno de sus elementos. En el caso de matrices asociativas no resulta tan sencillo utilizar bucles para recorrerlas, pero permiten usar índices con valores significativos.

También podemos recurrir el bucle `foreach` que está especialmente diseñado para su uso con matrices. En este ejemplo podemos utilizarlo de la siguiente forma:

```
foreach ($products as $current)
   echo $current.' ';
```

Este código almacena elemento a elemento en la variable `$current` y los imprime.

Matrices asociativas

En la matriz de productos, permitimos que PHP asigne a cada elemento el índice predeterminado. Esto significa que el primer elemento que agregamos se convierte en 0, el segundo en 1 y así sucesivamente. PHP admite el uso de matrices asociativas. En una matriz asociativa, podemos asociar cualquier clave o índice que deseemos a un valor.

Inicialización de una matriz asociativa

El siguiente código crea una matriz asociativa utilizando los nombres de productos y los precios como valores.

```
$prices = array( 'Tires'=>100, 'Oil'=>10, 'Spark Plugs'=>4 );
```

Cómo acceder a elementos de matriz

De nuevo, utilizamos el nombre de la variable y una clave para acceder a los contenidos. De esta forma podemos acceder a la información almacenada en la matriz de precios como `$prices[ 'Tires' ]`, `$prices[ 'Oil' ]` y `$prices[ 'Spark Plugs' ]`.

Como en el caso de las matrices indexadas, las matrices asociativas se pueden crear e inicializar elemento a elemento.

El siguiente fragmento de código crea la misma matriz `$prices`. En lugar de crear una matriz con tres elementos, esta versión crea una matriz con un único elemento y, a continuación, crea dos más.

```
$prices = array( 'Tires'=>100 );
$prices['Oil'] = 10;
$prices['Spark Plugs'] = 4;
```

Aquí también existe una pequeña diferencia, aunque el código sea equivalente. En esta versión, no creamos una matriz de manera explícita sino que se crea automáticamente al agregarle el primer elemento.

```
$prices['Tires']  =  100;
$prices['Oil'] = 10;
$prices['Spark Plugs'] = 4;
```

Uso de bucles con matrices asociativas

Como los índices de esta matriz asociativa no están formados por números, no podemos incluir un simple contador dentro de bucle `for` para que funcione con la matriz. Podemos utilizar el bucle `foreach` o las instrucciones `list()` y `each()`. El bucle `foreach` presenta una estructura ligeramente diferente cuando se utiliza

con matrices asociativas. Podemos usarlo tal y como hicimos en el ejemplo anterior o incorporar las claves:

```
foreach ($prices as $key => $value)
   echo $key.'=>'.$value.'<br />';
```

El siguiente código devuelve los contenidos de la matriz `$prices` utilizando la instrucción `each()`:

```
while( $element = each( $prices ) )
{
   echo $element[ 'key' ];
   echo ' - ';
   echo $element[ 'value' ];
   echo '<br />';
}
```

El resultado de este fragmento de código se muestra en la figura 3.2.

En un capítulo anterior, vimos los bucles `while` y la instrucción `echo`. El código anterior utiliza la función `each()`, que no se ha visto todavía. Como estamos llamando a `each()` dentro de un bucle `while`, los elementos de la matriz se devolverán uno a uno y se detendrá al alcanzar el final de la matriz.

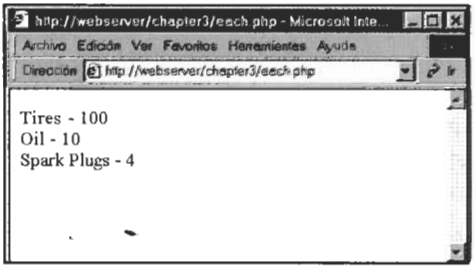


Figura 3.2. Se puede utilizar una instrucción each() para recorrer matrices

En este código, la variable `$element` es una matriz. Al llamar a `each()`, nos devuelve una matriz con cuatro valores y los cuatro índices hasta las ubicaciones de matriz. Las ubicaciones `key` y `0` contienen la clave del elemento actual y las ubicaciones `value` y `1` contienen el valor del elemento actual. Aunque da igual cuál se escoja, en nuestro ejemplo hemos optado por utilizar las ubicaciones con nombre en lugar de las ubicaciones con número.

Existe una forma más habitual y elegante de realizar esta tarea. Podemos utilizar la función `list()` para dividir una matriz en un número de valores y separar los dos valores que nos proporciona la función `each()` de la siguiente forma:

```
$list( $product, $price ) = each( $prices );
```

Esta línea utiliza `each()` para tomar el elemento actual desde `$prices`, lo devuelve en forma de una matriz y convierte al siguiente elemento en el actual.

También utiliza la función `list()` para convertir los elementos `0` y `1` de la matriz devueltos por `each()` en dos nuevas variables llamadas `$product` y `$price`.

Podemos utilizar un bucle para recorrer la matriz `$prices` entera y mostrar los resultados utilizando esta breve secuencia de comandos.

```
while ( list( $product, $price ) = each( $prices ) )
   echo "$product - $price<br />";
```

El resultado de este código es el mismo que el generado por la secuencia de comandos anterior, pero resulta más sencilla de leer porque la instrucción `list()` permite asignar nombres a las variables. Una cosa en la que fijarse al utilizar `each()` es que la matriz realiza el seguimiento el elemento actual. Si queremos utilizar la matriz dos veces en la misma secuencia de comandos, debemos reestablecer el elemento actual al principio de la matriz utilizando la función `reset()`. Para recorrer la matriz de precios de nuevo con ayuda de un bucle, utilice el siguiente código:

```
reset($prices);
while ( list( $product, $price ) = each( $prices ) )
   echo "$product - $price<br />";
```

Este código restablece el elemento actual al principio de la matriz y nos permite recorrerla de nuevo.

Matrices multidimensionales

Las matrices no tienen por qué estar formadas por una simple lista de claves y valores ya que cada ubicación de la matriz puede contener otra matriz. De esta forma podemos crear una matriz de dos dimensiones. Para hacerse una idea del concepto de las matrices bidimensionales, piense en ellas como una tabla, o cuadrícula, con altura y anchura o con filas y columnas. Si queremos almacenar más de un dato de cada producto de Bob, podemos utilizar una matriz bidimensional.

La figura 3.3 muestra los productos de Bob en una matriz bidimensional en la que cada fila representa un producto y cada columna un atributo de producto.

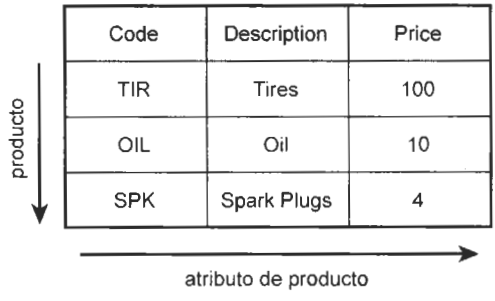


Figura 3.3. Podemos almacenar más información sobre los productos de Bob en una matriz bidimensional

Utilizando PHP podríamos escribir el siguiente código para configurar los datos de la matriz que se muestra en la figura 3.3.

```
$products = array( array( 'TIR', 'Tires', 100 ),
                   array( 'OIL', 'Oil', 10 ),
                   array( 'SPK', 'Spark Plugs', 4 ) );
```

En esta definición puede ver que la matriz de productos contiene ahora tres matrices.

Para acceder a los datos de una matriz unidimensional, recuerde que necesitamos el nombre de la matriz y el índice del elemento.

En una matriz bidimensional la operación es similar con la salvedad de que cada elemento consta de dos índices, una fila y una columna.

La fila superior es la fila 0 y la columna situada más a la izquierda es la columna 0.

Para visualizar los contenidos de esta matriz, podríamos acceder manualmente a cada elemento de la siguiente forma:

```
echo '|'.$products[0][0].'|'.$products[0][1].'|'.$products[0][2].'|<br />';
echo '|'.$products[1][0].'|'.$products[1][1].'|'.$products[1][2].'|<br />';
echo '|'.$products[2][0].'|'.$products[2][1].'|'.$products[2][2].'|<br />';
```

También podemos colocar un bucle `for` dentro de otro bucle `for` para obtener el mismo resultado.

```
for ( $row = 0; $row < 3; $row++ )
{
   for ( $column = 0; $column < 3; $column++ )
   {
      echo '|'.$products[$row][$column];
   }
   echo '|<br />';
}
```

Ambas versiones generan el mismo resultado en el navegador.

```
|TIR|Tires|100|
|OIL|Oil|10|
|SPK|Spark Plugs|4|
```

La única diferencia entre los dos ejemplos es que el código será más breve si utiliza la segunda versión con una matriz de gran tamaño.

Si lo prefiere puede crear nombres de columna en lugar de números, como se ilustra en la figura 3.3.

Para ello, puede utilizar matrices asociativas. Para almacenar el mismo conjunto de productos, con columnas con nombre como las de la figura 3.3, se utiliza el siguiente código:

```
$products = array( array( Code => 'TIR',
                          Description => 'Tires',
                          Price => 100
                        ),
```

```
        array( Code => 'OIL',
               Description => 'Oil',
               Price => 10
             ),
        array( Code => 'SPK',
               Description => 'Spark Plugs',
               Price =>4
             )
);
```

Resulta más sencillo trabajar con esta matriz si sólo se desea recuperar un valor y es más sencillo recordar que la descripción almacenada en la columna `Description` que recordar que está almacenada en la columna 1. El uso de matrices asociativas evita tener que memorizar que se ha almacenado un artículo en [x][y]. Es fácil encontrar los datos utilizando la referencia a una ubicación utilizando nombres de fila y columna significativos.

Sin embargo, perderemos la posibilidad de utilizar un sencillo bucle `for` para recorrer cada columna. Una forma de escribir código para mostrar esta matriz es la siguiente:

```
for ( $row = 0; $row < 3; $row++ )
{
   echo '|'.$products[$row]['Code'].'|'.$products[$row]['Description'].
        '|'.$products[$row]['Price'].'|<br />';
```

El uso de un bucle `for` permite recorrer la matriz externa `$products`, numéricamente indexada. Cada fila de la matriz `$products` es un matriz asociativa. Podemos utilizar las funciones `each()` y `list()` en un bucle `while` para recorrer matrices asociativas. Por lo tanto, necesitamos utilizar un bucle `while` dentro de un bucle `for`:

```
for ( $row = 0; $row < 3; $row++ )
{
   while ( list( $key, $value ) = each( $products[ $row ] ) )
   {
      echo "|$value";
   }
   echo '|<br />';
}
```

No tenemos por qué limitarnos a dos dimensiones. De la misma forma que los elementos de las matrices pueden contener otras matrices, esas nuevas matrices pueden contener a su vez otras.

Una matriz tridimensional tiene altura, anchura y fondo. Si el símil de las matrices bidimensionales como tablas con filas y columnas le ayudó a imaginárselas, piense en las matrices tridimensionales como un conjunto apilado de tablas. Para hacer referencia a cada elemento, se utilizará la capa, la fila y la columna.

Si Bob dividiera sus productos en categorías, podríamos utilizar una matriz tridimensional para almacenarlos. La figura 3.4 muestra los productos de Bob en una matriz tridimensional.

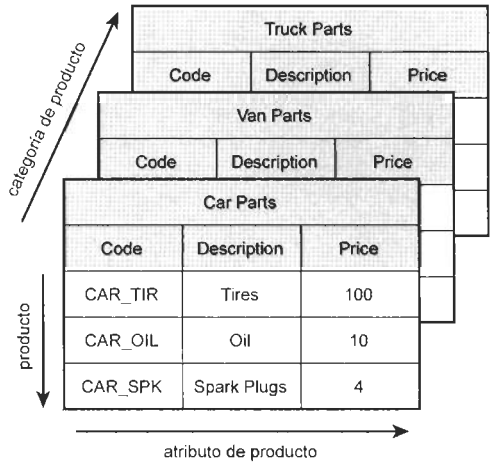


Figura 3.4. Esta matriz tridimensional nos permite dividir los productos en categorías

Si examina el código que define esta matriz, observará que una matriz tridimensional es una matriz que contiene matrices de matrices.

```
$categories = array( array (  array( 'CAR_TIR', 'Tires', 100 ),
                              array( 'CAR_OIL', 'Oil', 10 ),
                              array( 'CAR_SPK', 'Spark Plugs', 4 )
                           ),
                     array (  array( 'VAN_TIR', 'Tires', 120 ),
                              array( 'VAN_OIL', 'Oil', 12 ),
                              array( 'VAN_SPK', 'Spark Plugs', 5 )
                           ),
                     array (  array( 'TRK_TIR', 'Tires', 150 ),
                              array( 'TRK_OIL', 'Oil', 15 ),
                              array( 'TRK_SPK', 'Spark Plugs', 6 )
                           )
                   );
```

Como esta matriz sólo consta de índices numéricos, podemos utilizar bucles `for` anidados para mostrar sus contenidos.

```
for ( $layer = 0; $layer < 3; $layer++ )
{
   echo "Layer $layer<br />";
   for ( $row = 0; $row < 3; $row++ )
   {
      for ( $column = 0; $column < 3; $column++ )
      {
         echo '|'.$categories[$layer][$row][$column];
      }
      echo '|<br />';
   }
}
```

Debido a la forma en la que se crean las matrices multidimensionales, podríamos crear matrices de cuatro, cinco o seis dimensiones. El lenguaje no tiene límites a este

respecto, pero resulta difícil visualizar estructuras con más de tres dimensiones. La mayor parte de los problemas del mundo real se corresponden lógicamente con estructuras de tres o menos dimensiones.

Cómo ordenar matrices

A menudo resulta útil ordenar datos relacionados que estén almacenados en una matriz. Es sencillo tomar un matriz unidimensional y ordenarla.

Uso de sort()

El siguiente código da como resultado una matriz que se ordena en orden alfabético ascendente:

```
$products = array( 'Tires', 'Oil', 'Spark Plugs' );
sort($products);
```

Los elementos de nuestra matriz quedarán ordenados de la siguiente forma: `Oil`, `Spark Plugs`, `Tires`.

También podemos ordenar los valores numéricamente. Si tenemos una matriz con los precios de los productos de Bob, podemos ordenarla en sentido ascendente como se muestra a continuación:

```
$prices = array( 100, 10, 4 );
sort($prices);
```

Los precios se ordenarán ahora de la siguiente forma: 4, 10, 100.

Tenga en cuenta que la función de ordenación discrimina entre mayúsculas y minúsculas, y que las letras mayúsculas van delante de las minúsculas. Por lo tanto, la "A" va antes que "Z", pero la "Z" va antes que la "a".

Uso de asort() y ksort() para ordenar matrices asociativas

Si quisiéramos utilizar una matriz asociativa para almacenar artículos y sus precios, necesitaríamos utilizar diferentes tipos de funciones de ordenación para mantener agrupadas las claves y su valores al ordenarlos.

El siguiente código crea una matriz asociativa en la que se incluyen tres productos y sus precios asociados, y ordena la matriz en orden ascendente por los precios.

```
$prices = array( 'Tires'=>100, 'Oil'=>10, 'Spark Plugs'=>4 );
asort($prices);
```

La función `asort()` ordena la matriz en función del valor de cada elemento. En la matriz, los valores son los precios y las claves son las descripciones textuales. Si

en lugar de ordenar la matriz por el precio, queremos hacerlo por la descripción, se utilizará la función `ksort()`, que ordena la matriz por la clave y no por su valor. Como resultado se ordenarán las claves de la matriz alfabéticamente: Oil, Spark Plugs, Tires.

```
$prices = array( 'Tires'=>100, 'Oil'=>10, 'Spark Plugs'=>4 );
ksort($prices);
```

Inversión del orden

Hemos visto las funciones `sort()`, `asort()` y `ksort()`. Estas tres funciones ordenan las matrices en orden ascendente. Pero cada una de ellas tiene una función correspondiente para ordenar las matrices en orden descendente. Estas funciones inversas son `rsort()`, `arsort()` y `krsort()`.

Las funciones de ordenación inversa se utilizan de la misma forma que las funciones de ordenación normales. La función `rsort()` ordena un matriz unidimensional indexada numéricamente en orden descendente. La función `arsort()` ordena una matriz asociativa unidimensional en sentido descendente utilizando el valor de cada elemento. La función `krsort()` ordena una matriz asociativa unidimensional en orden descendente utilizando la clave de cada elemento.

Ordenación de matrices multidimensionales

La ordenación de matrices con más de una dimensión o aplicando un orden distinto al alfabético o numérico resulta más complicado. PHP sabe cómo comparar dos números o dos cadenas de texto, pero en una matriz multidimensional cada elemento de una matriz es otra matriz. PHP no sabe cómo comparar dos matrices, por lo que es necesario crear un método para compararlas. En la mayor parte de las ocasiones, el orden de las palabras o números resulta bastante obvio pero en el caso de objetos complicados, se convierte en una operación más problemática.

Ordenaciones definidas por el usuario

A continuación se repite la definición de una matriz bidimensional utilizada anteriormente. Esta matriz almacena los tres productos que comercializa Bob con un código, una descripción y un precio.

```
$products = array( array( 'TIR', 'Tires', 100 ),
                   array( 'OIL', 'Oil', 10 ),
                   array( 'SPK', 'Spark Plugs', 4 ) );
```

Si ordenamos esta matriz, ¿qué orden se aplicará finalmente a los valores? Como sabemos qué contenidos representan, existen dos posibilidades al menos. Podemos

ordenar los productos en orden alfabético utilizando su descripción o hacerlo numéricamente por su precio. Ambas opciones son posibles, pero necesitamos utilizar la función `usort()` y decirle a PHP cómo comparar los artículos. Para ello tenemos que escribir nuestra propia función de comparación.

El siguiente código ordena esta matriz en orden alfabético utilizando la segunda columna de la matriz: la descripción.

```
function  compare($x,  $y)
{
   if ( $x[1] == $y[1] )
      return 0;
   else if ( $x[1] < $y[1] )
      return -1;
   else
      return 1;
}

usort($products, 'compare');
```

Hasta el momento, hemos utilizado varias funciones incorporadas en PHP. Para ordenar esta matriz, hemos definido una función propia. En un capítulo posterior se analizará en detalle la creación de funciones.

Las funciones se definen utilizando la palabra clave `function`. Es necesario asignar un nombre a la función. Los nombres deberían ser descriptivos. Muchas funciones toman parámetros o argumentos. Nuestra función `compare()` toma dos, uno llamado `x` y otra llamado `y`. El objetivo de esta función es tomar dos valores y determinar su orden.

En este ejemplo, los parámetros `x` e `y` serán dos matrices dentro de la matriz principal y cada una representará un producto. Para acceder a la descripción de la matriz `x`, escribimos `$x[1]` porque se trata del segundo elemento de estas matrices y la numeración comienza en cero. Utilizamos `$x[1]` y `$y[1]` para comparar las descripciones de las matrices pasadas en la función.

Al final de la función, se puede devolver una respuesta al código de llamada. Para devolver un valor, utilizamos la palabra clave `return`. Por ejemplo, la línea `return 1;` devuelve el valor `1` al código que invocó la función.

Para utilizar la función `compare()` con `usort()`, debemos comparar `x` e `y`. La función debe devolver `0` si `x` es igual a `y`, un número negativo si es menor y un número positivo si es mayor. La función devolverá `0`, `1`, o `-1`, según los valores de `x` e `y`.

La línea final del código llama a la función incorporada de PHP `usort()` dentro de la matriz que queremos ordenar (`$products`) y el nombre de la función de comparación (`compare()`).

Si quisiéramos ordenar la matriz de otra forma, bastaría con escribir una función de comparación diferente. Para ordenar la matriz por el precio, debemos examinar la tercera columna de la matriz y crear la siguiente función de comparación

```
function  compare($x,  $y)
{
```

```
if ( $x[2] == $y[2] )
   return 0;
else if ( $x[2] < $y[2] )
   return -1;
else
   return 1;
}
```

Al llamar a `usort($products, compare)`, la matriz se ordenará en orden ascendente por el precio. La "u" de `usort()` equivale a "usuario" porque requiere una función de comparación definida por el usuario. Las versiones `uasort()` y `uksort()` de `asort` y `ksort` también requieren una función definida por el usuario.

Al igual que `asort()`, la función `uasort()` debería utilizarse al ordenar una matriz asociativa por su valor. Utilice `asort` si los valores son números o texto. Defina una función de comparación y utilice `uasort()` si los valores son objetos más complicados que las matrices.

Al igual que `ksort()`, `uksort()` debería utilizarse para ordenar una matriz asociativa por su clave. Utilice `ksort()` si sus claves son números o texto. Defina una función de comparación y utilice `uksort()` si sus claves son objetos más complicados que matrices.

Ordenaciones de usuario inversas

`sort()`, `asort()` y `ksort()` tienen funciones inversas que se establecen colocando una "r" delante del nombre de la función. Las ordenaciones definidas por el usuario no disponen de variantes inversas, pero se puede ordenar una matriz multidimensional en orden inverso. Para ello, bastará con escribir una función de comparación que devuelva los valores al revés. Para ordenar una matriz en orden inverso, la función tendrá que devolver 1 si x es menor que y y -1 si x es mayor que y. Por ejemplo:

```
function reverseCompare($x, $y)
{
if ( $x[2] == $y[2] )
   return 0;
else if ( $x[2] < $y[2] )
   return 1;
else
   return -1;
}
```

La función `usort($products, reverseCompare)` dará como resultado una matriz en orden descendente por el precio.

Reordenación de matrices

En algunas aplicaciones puede que necesitemos manipular el orden de las matrices de varias formas. La función `shuffle()` reordena de manera aleatoria los

elementos de la matriz. La función `array_reverse()` devuelve una copia de la matriz con todos los elementos en orden inverso.

Uso de shuffle()

Bob quiere destacar un número reducido de sus productos en la página principal del sitio. El número de productos que comercializa es grande, pero quiere seleccionar tres productos de manera aleatoria para que aparezcan en la página principal. Su intención es que los visitantes que vuelvan al sitio vean algo diferente en cada visita y no se aburran. Resultaría muy sencillo conseguir este objetivo si todos los productos se incluyeran en una matriz. El listado 3.1 muestra tres imágenes seleccionadas de manera aleatoria. Para ello, se aplica un orden aleatorio a los elementos de la matriz y se muestra el primero de los tres artículos.

Listado 3.1. bobs_front_page.php. Uso de PHP para generar una página principal dinámica en el sitio Bob's Auto Parts

```
<?php
   $pictures  =  array('tire.jpg',  'oil.jpg',  'spark_plug.jpg',
                       'door.jpg', 'steering_wheel.jpg',
                       'thermostat.jpg', 'wiper_blade.jpg',
                       'gasket.jpg', 'brake_pad.jpg');

   srand ((float)microtime()*1000000);
   shuffle($pictures);
?>
<html>
<head>
   <title>Bob's Auto Parts</title>
</head>
<body>
   <center>
      <h1>Bob's Auto Parts</h1>
      <table width = 100%>
         <tr>
<?php
   for ( $i = 0; $i < 3; $i++ )
   {
      echo '<td align="center"><img src="';
      echo $pictures[$i];
      echo '"width="100" height="100"></td>';
   }
?>
         </tr>
      </table>
   </center>
</body>
</html>
```

Como el código selecciona imágenes aleatorias, genera una página diferente prácticamente cada vez que se carga, como se muestra en la figura 3.5.

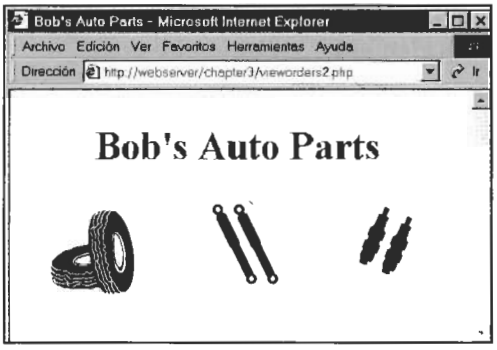


Figura 3.5. La función shuffle() nos permite mostrar tres productos seleccionados de manera aleatoria

Para que las funciones de números aleatorios realicen su trabajo, es necesario suministrar el generador de números primero, llamando a la función `srand()`, como puede apreciar en el listado 3.1.

La función `shuffle()` no se distingue por su pasado. En las versiones más antiguas de PHP, no mezclaba muy bien los resultados aleatoriamente. En la versión 4.2.x para Windows no funcionaba en absoluto (devolvía el mismo resultado inicial). Nuestro consejo es probarla en el servidor para comprobar sus resultados.

Uso de array_reverse()

La función `array_reverse()` toma una matriz y crea una nueva invirtiendo el orden de los elementos. Por ejemplo, existen varias formas de crear una matriz que incluya una cuenta atrás de diez a uno.

Si utilizamos `range()` únicamente se creará una secuencia ascendente. Por ello, debemos utilizar `rsort()` para ordenar los números en orden descendente. Opcionalmente, podemos crear la matriz elemento a elemento escribiendo un bucle `for`:

```
$numbers = array();
for($i=10; $i>0; $i--)
   array_push( $numbers, $i );
```

Los bucles `for()` pueden tener orden descendente como éste. Establecemos un valor alto como valor inicial y al final de cada bucle utilizamos – para reducir el contador a uno.

Creamos una matriz vacía y utilizamos `array_push()` para que cada elemento agregue uno nuevo al final de una matriz. La función contraria de `array_push()` es `array_pop()`. Esta función elimina y devuelve un elemento desde el final de una matriz.

También podemos utilizar la función `array_reverse()` para invertir la matriz creada por `range()`.

```
$numbers = range(1,10);
$numbers  =  array_reverse($numbers);
```

Recuerde que `array_reverse()` devuelve una copia modificada de la matriz. Como no queremos la matriz original, almacenamos la nueva matriz sobre el original.

Carga de matrices desde archivos

En un capítulo anterior, almacenamos los pedidos de los clientes en un archivo. Cada línea del archivo presenta un aspecto parecido a éste:

```
15:42, 20th April 4 tires 1 oil 6 spark plugs $434.00 22 Short St,
Smalltown
```

Para procesar o servir este pedido, podemos volver a cargarlo en una matriz. El listado 3.2 muestra el archivo actual de pedidos.

Listado 3.2. vieworders.php. Uso de PHP para mostrar los pedidos de Bob

```
<?php
//cree un nombre de variable corto
$DOCUMENT_ROOT  =  $HTTP_SERVER_VARS['DOCUMENT_ROOT'];

$orders= file("$DOCUMENT_ROOT/../orders/orders.txt");

$number_of_orders = count($orders);
if ($number_of_orders == 0)
{
   echo '<p><strong>No orders pending.
         Please try again later.</strong></p>';
}
for ($i=0; $i<$number_of_orders; $i++)
{
   echo $orders[$i].'<br />';
}
?>
```

Esta secuencia de comandos genera prácticamente el mismo resultado que el listado 2.2 del capítulo anterior, que se muestra en la figura 2.4. Esta vez vamos a utilizar la función `file()` que carga el archivo entero en una matriz. Cada línea del archivo se convierte en un elemento de una matriz.

Este código también utiliza la función `count()` para comprobar el número de elementos que contiene una matriz.

Adicionalmente, podríamos cargar cada sección del pedido en elementos de matriz separados para procesar las secciones de manera independiente o para aplicarles un formato más llamativo.

Eso es lo que hace el listado 3.3 exactamente.

```
<?php
   //cree un nombre de variable corto
   $DOCUMENT_ROOT = $HTTP_SERVER_VARS['DOCUMENT_ROOT'];
?>
<html>
<head>
   <title>Bob's Auto Parts - Customer Orders</title>
</head>
<body>
<h1>Bob's Auto Parts</h1>
<h2>Customer Orders</h2>
<?php
   //Lea el archivo completo.
   //Cada pedido se convierte en un elemento de la matriz
   $orders= file("$DOCUMENT_ROOT/../orders/orders.txt");
   // cuente el número de pedidos de la matriz
   $number_of_orders  =  count($orders);
   if ($number_of_orders == 0)
   {
      echo '<p><strong>No orders pending.
                Please try again later.</strong></p>';
   }
   echo "<table border=1>\n";
   echo '<tr><th bgcolor="#CCCCFF">Order Date</th>
             <th bgcolor="#CCCCFF">Tires</th>
             <th bgcolor="#CCCCFF">Oil</th>
             <th bgcolor="#CCCCFF">Spark Plugs</th>
             <th bgcolor="#CCCCFF">Total</th>
             <th bgcolor="#CCCCFF">Address</th>
           <tr>';
   for ($i=0; $i<$number_of_orders; $i++)
   {
      //divida cada línea
      $line = explode( "\t", $orders[$i] );
      // Mantenga ordenado únicamente el número de artículos
      $line[1] = intval( $line[1] );
      $line[2] = intval( $line[2] );
      $line[3] = intval( $line[3] );
      // muestre cada pedido
      echo "<tr><td>$line[0]</td>
                  <td align="right">$line[1]</td>
                  <td align="right">$line[2]</td>
                  <td align="right">$line[3]</td>
                  <td align="right">$line[4]</td>
                  <td>$line[5]</td>
                </tr>";
   }
   echo "</table>";
?>
</body>
</html>
```

El código del listado 3.3 carga el archivo completo en una matriz pero a diferencia del ejemplo del listado 3.2, aquí utilizamos la función `explode()` para dividir cada línea para poder procesar y aplicar formato antes de impresión.

En la figura 3.6 se ilustra el resultado de esta secuencia de comandos. La función `explode` tiene la siguiente sintaxis:

```
array explode(string separador, string cadena [, int límite])
```



Figura 3.6. Tras dividir los registros del pedido con la función explode, podemos colocar cada sección del pedido en una celda de tabla diferente para mejorar su aspecto

En el capítulo anterior, utilizamos el carácter de tabulación como delimitador al almacenar estos datos. Por ello, aquí llamamos a:

```
explode( "\t", $orders[$i] )
```

Este código divide las cadenas pasada en partes. Cada carácter de tabulación separa los elementos. Por ejemplo, la cadena

```
"15:42, 20th April\t4 tires\t1 oil\t6 spark plugs\t$434.00\t
22 Short St, Smalltown"
```

se divide en "15:42, 20th April", "4 tires", "1 oil", "6 spark plugs", "$434.00" y "22 Short St, Smalltown". Fíjese en que el parámetro opcional *límite* se puede utilizar para restringir el número máximo de partes devueltas.

En este código no se ha realizado una gran cantidad de operaciones de procesamiento. En lugar de devolver los artículos en cada línea, sólo mostramos el número de cada uno y agregamos una fila de encabezados para mostrar qué representan. Existen varias formas de extraer números de estas cadenas. En este caso, hemos utilizado la función `intval()`. Como se mencionó en un capítulo anterior, `intval()` convierte una cadena en un entero. El proceso de conversión resulta razonablemente inteligente e ignorará partes, como la etiqueta de este ejemplo, que no se pueden convertir en un entero. En el siguiente capítulo abordaremos varias formas de procesar cadenas.

Otras manipulaciones de cadenas

Hasta el momento, sólo hemos analizado aproximadamente la mitad de las funciones disponibles para el procesamiento de matrices. Existen otras muchas que resultarán de utilidad de vez en cuando.

Navegación dentro de una matriz con each(), current(), reset(), end(), next(), pos(), and prev()

Ya mencionamos anteriormente que todas las matrices constan de un puntero interno que apunta al elemento actual de la matriz. Ya se utilizó antes este puntero indirectamente con la función `each()`, pero podemos utilizarlo directamente y manipularlo. Si crea una nueva matriz, el puntero actual se inicializará con el puntero al primer elemento de la matriz. Si llamamos a `current( $nombre_matriz )` se devolverá la primer elemento.

Si llamamos a `next()` o a `each()`, el puntero avanzará un elemento. Si llamamos a `each( $nombre_matriz)`, se devuelve el elemento actual antes de que el puntero avance. La función `next()` se comporta de manera ligeramente diferente: si llamamos a `next( $nombre_matriz )`, el puntero avanzará y devolverá el nuevo elemento actual.

Ya hemos visto que `reset()` devuelve el puntero al primer elemento de la matriz. De manera similar, si llamamos a `end( $nombre_matriz )`, el puntero se enviará al final de la matriz. `reset()` y `end()` devuelven el primer y el último elemento de la matriz, respectivamente. Para recorrer una matriz en orden inverso, podemos utilizar `end()` y `prev()`. La función `prev()` es la opuesta a la función `next()`. Mueve el puntero actual un puesto hacia atrás y devuelve el elemento actual. Por ejemplo, el siguiente código muestra una matriz en orden inverso:

```
$value = end ($array);
while ($value)
{
   echo "$value<br />";
   $value = prev($array);
}
```

Si `$array` se declarara de la siguiente forma:

```
$array = array(1, 2, 3);
```

El resultado se mostraría en el navegador de la siguiente forma:

```
3
2
1
```

El uso de `each()`, `current()`, `reset()`, `end()`, `next()`, `pos()` y `prev()` nos permite escribir nuestro propio código para navegar a través de una matriz en cualquier orden.

Aplicación de cualquier función a cada elemento de una matriz: array_walk()

En ocasiones, puede que necesitemos trabajar o modificar cada elemento de un matriz de la misma forma. Para ello, podemos utilizar la función `array_walk`:

La sintaxis de esta función es la siguiente:

```
int array_walk(array mat, string func, [mixed datos_de_usuario])
```

De manera similar a como hicimos anteriormente al llamar a `usort()`, la función `array_walk()` espera que declaremos una función propia.

Como puede apreciar, `array_walk()` toma tres parámetros. El primero, *mat*, es la matriz que se procesará. El segundo, *func*, es el nombre de una función definida por el usuario que se aplicará a cada elemento de la matriz. El tercer parámetro, *datos_de_usuario*, es opcional. Si lo utiliza, se pasará a través de su función como parámetro. En un instante veremos cómo funciona.

Podríamos crear una función práctica definida por el usuario que muestre cada elemento con opciones de formato diferentes.

El siguiente código visualiza cada elemento en una línea llamando a la función definida por el usuario `myPrint()` con cada elemento de `$array`:

```
function myPrint($value)
{
   echo "$value<br />";
}
array_walk($array, 'myPrint');
```

La función debe tener una firma especial. Para cada elemento de la matriz la función `array_walk` toma la clave y el valor almacenado en la matriz y todo lo que pase como *datos_de_usuario*, y llama a la función de la siguiente forma:

```
Sufunción(valor, clave, datos_de_usuario)
```

En la mayor parte de los casos, la función sólo utilizará los valores de la matriz. En algunos casos, puede que también necesite pasar un parámetro a la función utilizando el parámetro *datos_de_usuario*. En ocasiones puede que esté interesado en la clave de cada elemento así como en su valor. Como en el caso de `MyPrint()`, la función puede optar por ignorar la clave y el parámetro *datos_de_usuario*.

Podemos escribir un ejemplo un poco más complicado, en el que una función modifique los valores de las matrices y necesite un parámetro. Fíjese en que a pesar de no estar interesados en la clave, tenemos que aceptarla para poder aceptar el tercer parámetro.

```
function myMultiply(&$value, $key, $factor)
{
   $value *= $factor;
}
array_walk(&$array, 'myMultiply', 3);
```

En este ejemplo, se define una función, `myMultiply()`, que multiplicará cada elemento de la matriz por un factor suministrado. Necesitamos utilizar el tercer parámetro opcional de `array_walk()` para tomar un parámetro que pasar a nuestra función y utilizarlo como factor por el que multiplicar. Como necesitamos este parámetro, podemos definir la función `myMultiply()` para que tome tres

parámetros: un valor de elemento de matriz (`$value`), una clave de elemento de matriz (`$key`) y otro parámetro (`$factor`). Vamos a optar por ignorar la clave.

Un punto sutil en el que fijarse es la forma de pasar `$value`. El símbolo `&` colocado delante del nombre de la variable en la definición de `myMultiply()` significa que `$value` se pasará por referencia. Este modo de pasar variables permite que la función modifique los contenidos de la matriz. Analizaremos este modo en un capítulo posterior. Por el momento, fíjese simplemente en que al pasar por referencia se utiliza un símbolo `&` delante del nombre de la variable.

Cómo contar elementos de una matriz: count(), sizeof(), and array_count_values()

En un ejemplo anterior utilizamos la función `count()` para contar la cantidad de elementos de una matriz de pedidos. La función `sizeof()` tiene el mismo objetivo. Ambas funciones devuelven el número de elementos de una matriz que se le han pasado. Obtendremos 1 para el número de elementos en una variable escalar normal y 0 si pasamos una matriz vacía o una variable que no se haya establecido.

La función `array_count_values()` es más compleja. Si llama a `array_count_values($array)`, esta función contará el número de veces que tiene lugar cada valor único en la matriz `$array`. (Se determina por la cardinalidad de la matriz). La función devuelve una matriz asociativa que contiene una tabla de frecuencia. Esta matriz contiene los valores exclusivos de `$array` como claves. Cada clave tiene un valor numérico que indica las veces que tiene lugar la clave correspondiente en `$array`.

Por ejemplo, el siguiente código:

```
$array = array(4, 5, 1, 2, 3, 1, 2, 1);
$ac = array_count_values($array);
```

crea una matriz llamada `$ac` que contiene

clave	valor
4	1
5	1
1	3
2	2
3	1

Esto indica que 4, 5 y 3 ocurren una vez en `$arry`, 1 ocurre tres veces y 2 ocurre dos veces.

Conversión de matrices en variables escalares: extract()

Si tenemos una matriz asociativa con una serie de pares de valor y clave, podemos convertirlas en un conjunto de variables escalares utilizando la función `extract()`.

Esta función tiene la siguiente sintaxis:

```
extract(array matriz_var [, int tipo_extracción] [, string prefijo] );
```

El objetivo de `extract()` es tomar una matriz y crear variables escalares con los nombres de las claves de la matriz. Los valores de estas variables se establecen en función de los valores de la matriz.

A continuación se incluye un ejemplo:

```
$array = array( 'key1' => 'value1', 'key2' => 'value2', 'key3' => 'value3');
extract($array);
echo "$key1 $key2 $key3";
```

Este código genera el siguiente resultado:

```
value1 value2 value3
```

La matriz tiene tres elementos con claves: `key1`, `key2` y `key3`. La función `extract()` nos permite crear tres variables escalares, `$key1`, `$key2` y `$key3`. Por el resultado podemos ver que los valores de `$key1`, `$key2` y `$key3` son 'value1', 'value2' y 'value3', respectivamente.

Estos valores proceden de la matriz original.

La función `extract()` consta de dos parámetros opcionales: *tipo_extracción* y *prefijo*. La variable `tipo_extracción` indica a `extract()` cómo resolver las colisiones.

Éstas tienen lugar cuando ya existe una variable con el mismo nombre que una clave. La respuesta predeterminada consiste en sobrescribir la variable existente. En la tabla 3.1 se recogen los valores posibles de este parámetro.

Tabla 3.1. Valores permitidos del parámetro tipos_extracción de la función extract()

Tipo	Significado
`EXTR_OVERWRITE`	Sobrescribe la variable existente cuando tiene lugar una colisión.
`EXTR_SKIP`	Salta un elemento cuando tiene lugar una colisión.
`EXTR_PREFIX_SAME`	Crea una variable denominada `$prefix_key` cuando tiene lugar una colisión. Se debe incluir el parámetro `prefijo`.
`EXTR_PREFIX_ALL`	Coloca delante de todos los nombres de variables el valor del `prefijo`. Se debe incluir el parámetro `prefijo`.

Tipo	Significado
EXTR_IF_EXISTS	Extrae solamente variables que ya existen (es decir, rellena las variables existentes con valores procedentes de la matriz). Esta opción se agregó a la versión 4.2.0 y resulta útil para convertir, por ejemplo, $_REQUEST en un conjunto de variables válidas.
EXTR_PREFIX_IF_EXISTS	Sólo crea una versión con prefijo si ya existiese una versión sin prefijo. Esta opción se agregó en la versión 4.2.0.
EXTR_REFS	Extrae variables como referencia. Esta opción se agregó en la versión 4.3.0.

Las opciones más útiles son las predeterminada, EXTR_OVERWRITE, y EXTR_PREFIX_ALL. Las otras funciones pueden resultar de utilidad si sabemos que se va a producir una colisión y queremos saltarla o utilizar prefijos. A continuación se recoge un sencillo ejemplo utilizando EXTR_PREFIX_ALL. Como puede observar, las variables creadas se denominan prefix_guión bajo_nombre de clave.

```
$array = array( 'key1' => 'value1', 'key2' => 'value2', 'key3' => 'value3');
extract($array, EXTR_PREFIX_ALL, 'myPrefix');
echo "$myPrefix_key1 $myPrefix_key2 $myPrefix_key3";
```

Este código devolverá de nuevo value1 value2 value3.

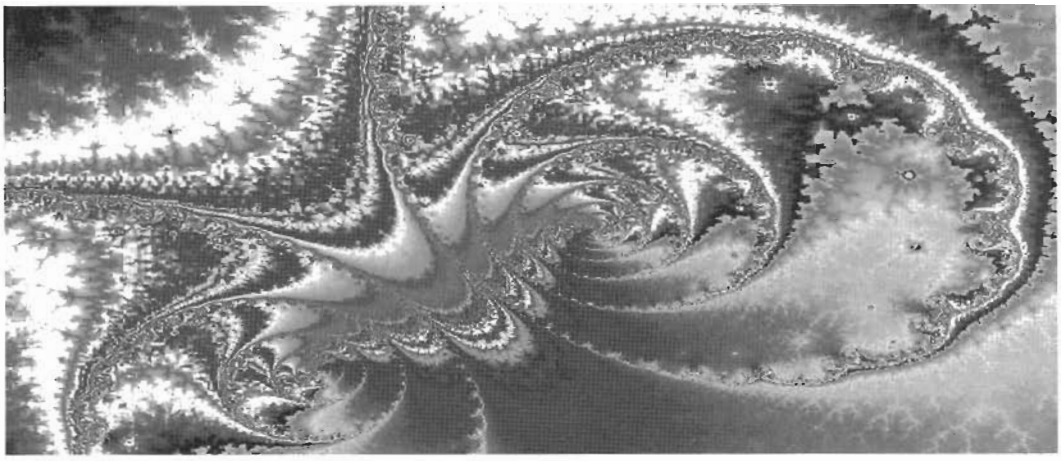
Tenga en cuenta que para que extract() extraiga un elemento, la clave de dicho elemento debe ser un nombre válido de variable, lo que significa que se no se tendrán en cuenta las claves que comiencen por números o que incluyan espacios.

Lecturas adicionales

En este capítulo se trata lo que en nuestra opinión son las funciones de matriz más útiles de PHP, pero no se han estudiado todas. En el manual en línea de PHP, disponible en http://www.php.net/array, encontrará una breve descripción de todas ellas.

A continuación

En el siguiente capítulo se analizarán las funciones de procesamiento. Veremos cómo realizar operaciones de búsqueda, sustitución, división y combinación de cadenas. Así mismo, se examinarán las potentes funciones de expresiones regulares capaces de realizar prácticamente cualquier operación sobre una cadena.



4
Manipulación de cadenas y expresiones regulares

En este capítulo, veremos cómo puede utilizar las funciones de cadena de PHP para aplicar formato y manipular texto. También veremos cómo utilizar funciones de cadena o funciones de expresiones regulares para buscar (y reemplazar) palabras, frases y otros patrones dentro de una cadena.

Estas funciones resultan útiles en muchos contextos. Con frecuencia se suele limpiar o modificar el formato de los datos recibidos de los usuarios para almacenarlos en una base de datos. Las funciones de búsqueda son magníficas para crear motores de búsqueda (entre otras cosas).

En este capítulo, analizaremos los siguientes aspectos:

- Formato de cadenas
- Combinación y división de cadenas
- Comparación de cadenas
- Coincidencia y sustitución de subcadenas con funciones de cadena
- Uso de expresiones regulares

Aplicación de ejemplo: Smart Form Mail

En este capítulo, vamos a examinar las funciones de cadena y de expresiones regulares en el contexto de la aplicación Smart Form Mail. Agregaremos estas se-

cuencias de comandos al sitio Bob's Autor Parts que hemos desarrollado en los últimos capítulos.

En concreto, vamos a crear un sencillo formulario para que los clientes de Bob expresen sus quejas y agradecimientos, como se ilustra en la figura 4.1. Sin embargo, nuestra aplicación incorporará una mejora con respecto a los formularios de este tipo que se incluyen en la Web. En lugar de remitir el formulario por correo electrónico a una dirección genérica como `información@ejemplo.com`, intentaremos que su procesamiento resulte más avanzado. Para ello, se analizará el texto en busca de palabras clave y frases, y se remitirá el correo electrónico al empleado apropiado de la compañía de Bob. Por ejemplo, si el correo contiene la palabra "avertising", el formulario podría dirigirse al departamento de marketing. Si el correo procede de un cliente importante, se dirigirá directamente a Bob.

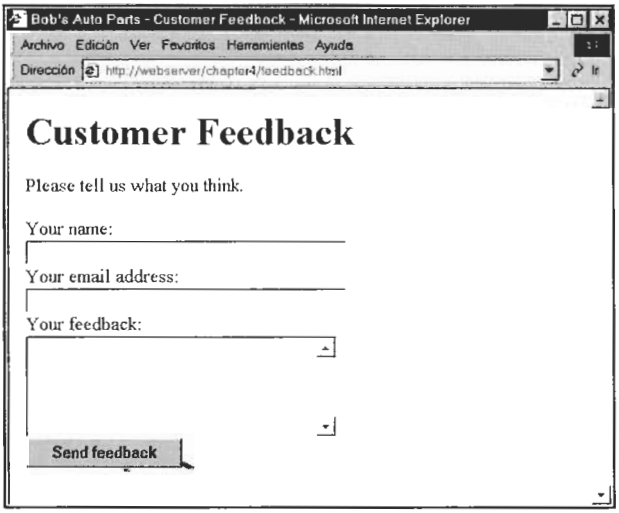


Figura 4.1. El formulario para comentarios de Bob pregunta a los clientes su nombre, dirección de correo electrónico y comentarios

Comenzaremos por la sencilla secuencia de comandos que se muestra en el listado 4.1 e iremos agregando elementos progresivamente.

Listado 4.1. processfeedback.php: secuencia de comandos básica de Email Forms Contents

```
<?php
  //cree nombres de variables
  $name=$HTTP_POST_VARS['name'];
  $email=$HTTP_POST_VARS['email'];
  $feedback=$HTTP_POST_VARS['feedback'];

  $toaddress = 'feedback@example.com';
  $subject = 'Feedback from web site';
```

```
    $mailcontent = 'Customer name: '.$name."\n"
                   .'Customer email: '.$email."\n"
                   ."Customer comments: \n".$feedback."\n";
    $fromaddress = 'From: webserver@example.com';

    mail($toaddress, $subject, $mailcontent, $fromaddress);
?>
<html>
<head>
    <title>Bob's Auto Parts - Feedback Submitted</title>
</head>
<body>
<h1>Feedback submitted</h1>
<p>Your feedback has been sent.</p>
</body>
</html>
```

Por regla general, debería comprobar si los usuarios han rellenado todos los campos obligatorios del formulario con ayuda de `isset()`, por ejemplo. Hemos omitido esta operación en la secuencia de comandos así como en otros ejemplos para no alargar el análisis. En esta secuencia de comandos, hemos concatenado los campos del formulario y hemos utilizado la función `mail()` de PHP para enviar un correo electrónico a `información@ejemplo.com`. A continuación, analizaremos la función `mail()`. La función `mail()` envía correos electrónicos. Su sintaxis es la siguiente:

```
bool mail(string to, string tema, string mensaje,
      string [encabezados_adicionales [, string parámetros_adicionales]]);
```

Los primeros tres parámetros son obligatorios y representan la dirección a la que dirigir el correo electrónico, la línea de asunto y los contenidos del mensaje, respectivamente. El cuarto parámetro se puede utilizar para enviar otros encabezados válidos. Los encabezados válidos de correo electrónicos se describen en el documento RFC822, que está disponible en línea si desea obtener más detalles. (Los RFC o Solicitudes de comentarios son la fuente de una gran cantidad de estándares de Internet, como se comentará en un capítulo posterior.) Aquí hemos utilizado el cuarto parámetro para agregar una dirección `"From:"` al correo electrónico. También puede utilizarlo para agregar campos `"Reply-To:"` y `"Cc:"`, entre otros. Si desea incluir más de un encabezado, sólo será necesario separarlos mediante caracteres de nueva línea (`\n`) en la cadena, de la siguiente forma:

```
$additional_headers="From: webserver@example.com\n"
           .'Reply-To: bob@example.com';
```

El quinto parámetro se puede utilizar para pasar otro argumento al programa que tenga configurado para enviar correo.

Para poder utilizar la función `email()`, configure su instalación de PHP para que apunte al programa utilizado para enviar correos electrónicos. Si la secuencia de comandos no funciona correctamente, consulte el apéndice A.

A lo largo de este capítulo, optimizaremos esta secuencia de comandos básica utilizando funciones de procesamiento de cadenas y expresiones regulares de PHP.

Aplicación de formato a cadenas

A menudo es necesario limpiar las cadenas que envían los usuarios (por regla general procedentes de una interfaz de formulario HTML) para poder utilizarlas.

Limpieza de cadenas: chop(), ltrim() y trim()

El primer paso del proceso de limpieza consiste en quitar todos los espacios en blanco no necesarios de la cadena. Aunque no resulta obligatorio, puede resultar útil si se va a almacenar la cadena en un archivo o base de datos, o si vamos a compararla con otras cadenas.

PHP incorpora tres funciones que resultan de utilidad en este sentido. Utilizaremos la función `trim()` para limpiar los datos de entrada de la siguiente forma:

```
$name=trim($name);
$email=trim($email);
$feedback=trim($feedback);
```

La función `trim()` elimina los espacios en blanco desde al principio al final de una cadena, y devuelve la cadena resultante. De manera predeterminada, limpia los caracteres de nueva línea y retorno del carro (`\n` y `\r`), tabuladores horizontales y verticales (`\t` y `\v`), caracteres de final de cadena (`\0`) y espacios. También permite pasar un segundo parámetro para incorporar otros caracteres que eliminar a la lista predeterminada. Existen otras funciones `ltrim()` o `chop()` que puede utilizar. Ambas son similares a `trim()`: toman la cadena en cuestión como parámetro y devuelven la cadena con formato. La diferencia entre estas tres es que `trim()` elimina los espacios en blanco desde el principio al final de la cadena, `ltrim()` elimina los espacios en blanco desde el principio (o desde la izquierda) únicamente y `chop()` elimina los espacios en blanco desde el final (o desde la derecha) únicamente.

Aplicación de formato a cadenas para presentaciones

PHP consta de un conjunto de funciones que permiten volver a aplicar formato a una cadena de diferentes formas.

Uso de formato HTML: la función nl2br()

La función `nl2br()` toma una cadena como parámetro y sustituye todas las líneas nuevas con la etiqueta XHTML `<br />` (o la etiqueta `<br>` en las versiones anteriores a la 4.0.5). Esta función resulta útil para imprimir una cadena de gran tamaño en el navegador. Por ejemplo, hemos utilizado esta función para aplicar formato a la información recibida del cliente y devolverla:

```
<p>Your feedback (shown below) has been sent.</p>
<p><? echo nl2br($mailcontent); ?> </p>
```

Recuerde que HTML no tiene en cuenta los espacios en blanco sin procesar, por lo que si no filtramos este resultado a través de `nl2br()`, aparecerán en una sola línea (a excepción de las líneas nuevas forzadas por la ventana del navegador), como se ilustra en la figura 4.2.

Aplicación de formato a una cadena para su impresión

Hasta el momento, hemos utilizado la instrucción `echo` para mostrar cadenas en el navegador. PHP también incluye la instrucción `print()`, que realiza la misma función que `echo`, pero devuelve un valor (`true` o `false`, para denotar el resultado de la operación).

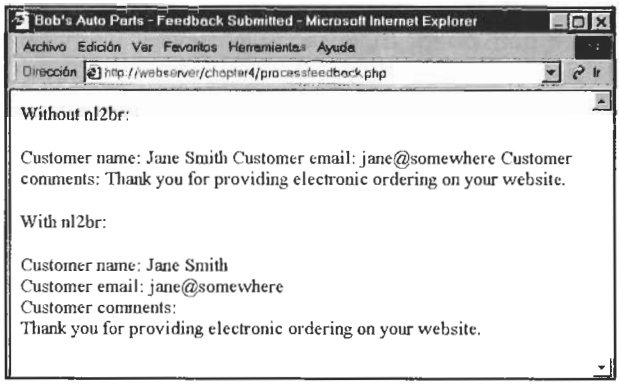


Figura 4.2. El uso de la función n12br() mejora la representación de cadenas largas dentro de HTML

Estas dos técnicas imprimen una cadena sin más. Puede aplicar formato más sofisticado utilizando las funciones `printf()` y `sprintf()`.

Éstas funcionan básicamente de la misma forma, con la excepción de que la primera imprime una cadena con formato en el navegador y la segunda devuelve una cadena con formato.

Si ha programado previamente en C, observará que estas funciones son iguales que sus versiones de C. Si no lo ha hecho, lleva cierto tiempo acostumbrarse pero resultan útiles y potentes.

Las sintaxis de estas funciones es la siguiente:

```
string sprintf (string formato [, mixed args...])
void printf (string formato [, mixed args...])
```

El primer parámetro pasado a ambas funciones es una cadena de formato que describe la forma básica del resultado con código de formato en lugar de variables. Los demás parámetros son variables que se sustituirán en la cadena de formato.

Por ejemplo, con `echo`, utilizamos las variables deseadas para imprimir en línea. Por ejemplo:

```
echo "Total amount of order is $total.";
```

Para obtener el mismo resultado con `printf()`, se utilizará:

```
printf ("Total amount of order is %s.", $total);
```

La secuencia `%s` utilizada en la cadena de formato se denomina especificación de conversión. Su objetivo es "ser reemplazada por una cadena". En este caso, será sustituida con `$total`, interpretada como una cadena. Si el valor almacenado en `$total` fuera 12.4, ambas instrucciones imprimirían 12.4.

La ventaja de `printf()` es que podemos utilizar una especificación de conversión más útil para especificar que `$total` es un número de coma flotante y que debería constar de dos decimales tras el punto, como se indica a continuación:

```
printf ("Total amount of order is %.2f", $total);
```

La cadena de formato puede incluir varias especificaciones de conversión. Si tenemos n especificaciones de conversión, tendrá n argumentos tras la cadena de formato. Cada especificación de conversión será reemplazada por un argumento con nuevo formato en el orden en el que se liste. Por ejemplo:

```
printf ("Total amount of order is %.2f (with shipping %.2f) ",
      $total, $total_shipping);
```

Las especificaciones de conversión utilizarán la variable `$total` pero en la segunda se utilizará la variable `$total_shipping`.

Cada especificación de conversión sigue el mismo formato, a saber:

```
%['carácter_de_relleno][-][anchura][.precisión]tipo
```

Todas las especificaciones de conversión comienzan por un símbolo `%`. Si desea imprimir un símbolo `%`, necesitará utilizar `%%`.

El *carácter de relleno* es opcional. Se utilizará para rellenar los espacios de la variable con la anchura especificada. Por ejemplo, podemos agregar ceros iniciales a un número en un contador. El símbolo - es opcional. Especifica que los datos del campo se justificarán a la izquierda, en lugar de a la derecha (opción predeterminada). El especificador *anchura* indica a `printf()` el espacio (en caracteres) que se debe dejar para que la variable se sustituya. El especificador *precisión* debería comenzar por una coma decimal. Debería contener el número de espacios deseados tras la coma decimal. La parte final de la especificación es un código de tipo. En la tabla 4.1 se recoge un resumen.

Tabla 4.1. Códigos de tipo de especificación de conversión

Tipo	Significado
b	Se interpreta como un entero y se imprime como un número binario.
c	Se interpreta como un entero y se imprime como un carácter.

Tipo	Significado
d	Se interpreta como un entero y se imprime como un número decimal.
f	Se interpreta como un doble y se imprime como un número con coma flotante.
o	Se interpreta como un entero y se imprime como un número octal.
s	Se interpreta como una cadena y se imprime como una cadena.
x	Se interpreta como un entero y se imprime como un número decimal con minúsculas para los dígitos a-f.
X	Se interpreta como un entero y se imprime como un número hexadecimal con mayúsculas para los dígitos A-F.

Desde la versión 4.0.6 se puede utilizar la numeración de argumentos, lo que significa que no es necesario utilizar el mismo orden que las especificaciones de conversión. Por ejemplo:

```
printf ("Total amount of order is %2\$.2f (with shipping %1\$.2f) ",
     $total_shipping, $total);
```

Basta con agregar la posición del argumento directamente tras el símbolo %, seguido por el carácter de escape $. En este caso, 2\$ significa "sustituir con el segundo argumento de la lista". Este método también se puede utilizar para repetir argumentos.

Cambio de mayúsculas y minúsculas en una cadena

Podemos modificar el uso de mayúsculas y minúsculas en una cadena. Este recurso no resulta especialmente útil en nuestra aplicación, pero examinaremos algunos ejemplos. Si comenzamos por la cadena del tema, `$subject`, que vamos a utilizar en nuestro correo electrónico, podemos cambiar el formato de mayúsculas y minúsculas con ayuda de varias funciones.

El efecto de estas funciones se resume en la tabla 4.2. La primera columna muestra el nombre de la función, la segunda describe su efecto, la tercera muestra cómo se aplicará a la cadena `$subject` y la última indica qué valor se devolverá de la función.

Tabla 4.2. Funciones para la aplicación de mayúsculas y minúsculas a cadenas y sus efectos

Función	Descripción	Uso	Valor
		`$subject`	Información procedente del sitio Web

Función	Descripción	Uso	Valor
`strtoupper()`	Convierte la cadena en mayúsculas	`strtoupper ($subject)`	INFORMACIÓN PROCEDENTE DEL SITIO WEB
`strtolower()`	Convierte la cadena en minúsculas	`strtolower ($subject)`	Información procedente del sitio Web
`ucfirst()`	Pone en mayúsculas el primer carácter de la cadena si es un carácter alfabético	`ucfirst ($subject)`	información procedente del sitio Web
`ucwords()`	Pone en mayúsculas la primera letra de cada palabra de la cadena que comience por un carácter alfabético	`ucwords ($subject)`	Información Procedente Del Sitio Web

Aplicación de formato a las cadenas para su almacenamiento: AddSlashes() y StripSlashes()

Además de utilizar funciones de cadena para modificar el formato de las cadenas visualmente, podemos utilizar parte de estas funciones para variar el formato de cadenas para su almacenamiento en una base de datos. Aunque el tema de la escritura de cadenas en bases de datos no se examinará hasta la segunda parte de este libro, seguidamente analizaremos cómo aplicar formato a cadenas para su almacenamiento en bases de datos.

Determinados caracteres son perfectamente válidos como parte de una cadena pero pueden originar problemas, en especial al insertar datos en una base de datos ya que ésta podría interpretarlos como caracteres de control.

Los caracteres problemáticos son las comillas (simples o dobles), las barras invertidas y el carácter NUL.

Debemos buscar una forma de marcar estos caracteres para indicar a las bases de datos, como MySQL, que se trata de caracteres especiales y no de una secuencia de control. Para marcar estos caracteres como caracteres especiales, se agrega una barra invertida delante de ellos. Por ejemplo, las comillas dobles (") se convertirán en \" (barra invertida seguida de las comillas dobles) y la barra invertida (\) se convierte en \\ (doble barra invertida). (Esta regla se aplica universalmente a los caracteres especiales, de manera que si tiene \\ en una cadena, deberá sustituirla por \\\\.)

PHP incorpora dos funciones específicamente diseñadas para marcar caracteres especiales. Antes de escribir cadenas en una base de datos, debería modificar su formato con la función `AddSlashes()`, por ejemplo:

```
$feedback = AddSlashes($feedback);
```

Como muchas otras funciones de cadena, `AddSlashes()` toma una cadena como parámetro y devuelve la cadena con formato nuevo.

Al utilizar `AddSlashes()`, la cadena se almacenará en la base de datos con las barras. Al recuperar la cadena, deberá acordarse de quitar dichas barras. Para ello puede utilizar la función `StripSlashes()`:

```
$feedback = StripSlashes($feedback);
```

La figura 4.3 muestra los efectos de utilizar estas funciones en la cadena.

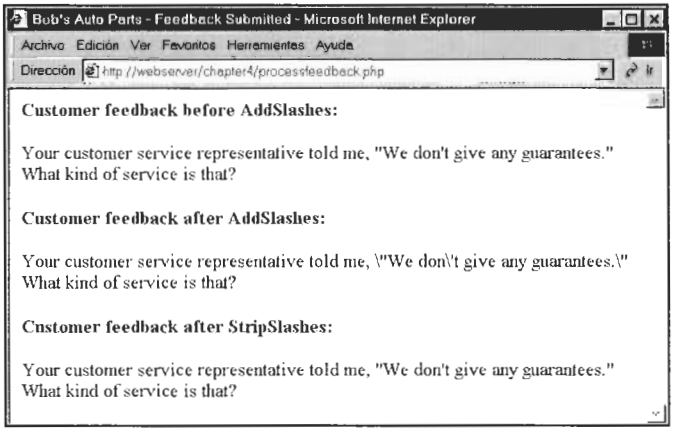


Figura 4.3. Tras llamar a la función AddSlashes(), se colocarán barras delante de todas las comillas. La función StripSlashes() quitará las barras

También puede configurar PHP para agregar y quitar barras automáticamente. Esta función se conoce como comillas mágicas y comentará en mayor detalle en un capítulo posterior.

Cómo combinar y dividir cadenas con funciones de cadena

A menudo, necesitamos examinar partes de una cadena por separado. Por ejemplo, puede que necesitemos examinar las palabras de una frase (para comprobar su ortografía, por ejemplo) o dividir un nombre o una dirección de correo electrónico en sus componentes. PHP incorpora varias funciones de cadena (y una función de expresión regular) que nos permite realizar esta tarea.

En nuestro ejemplo, Bob quiere recibir todos los comentarios procedentes de `bigcustomer.com` directamente, por lo que dividiremos la dirección de correo electrónica escrita por el cliente en partes para determinar si se trata de clientes importantes de Bob.

Uso de explode(), implode() y join()

La primera función que podríamos utilizar para este objetivo es `explode()` cuya sintaxis es la siguiente:

```
array explode(string separador, string entrada [, int límite]);
```

Esta función toma una entrada de cadena y la divide en partes en una cadena de separador especificada. Las partes se reúnen en una matriz. Puede limitar el número de partes con el parámetro opcional de limitación, agregado en PHP 4.0.1.

Para obtener un nombre de dominio de la dirección de correo electrónico del cliente en su secuencia de comandos, podemos utilizar el siguiente código:

```
$email_array = explode('@', $email);
```

Esta llamada a la función `explode()` divide la dirección de correo electrónico del cliente en dos partes: el nombre de usuario, que se puede almacenar en `$email_array[0]` y el nombre de dominio en `$email_array[1]`. Ahora ya podemos probar el nombre de dominio para determinar el origen del cliente y enviar sus comentarios a la persona apropiada.

```
if ($email_array[1]=='bigcustomer.com')
   $toaddress = 'bob@example.com';
else
   $toaddress = 'feedback@example.com';
```

Si el dominio estuviera en mayúsculas, la operación no funcionaría. Podemos evitar este problema convirtiendo el dominio a todo mayúsculas o todo minúsculas y realizar la comprobación seguidamente:

```
$email_array[1] = strtoupper ($email_array[1]);
```

Puede invertir los efectos de `explode()` utilizando las funciones `implode()` o `join()`, que son idénticas. Por ejemplo:

```
$new_email = implode('@', $email_array);
```

Este código toma los elementos de la matriz `$email_array` y los combina con la cadena pasada en el primer parámetro. La llamada a la función es muy similar a `explode()`, pero el efecto es el contrario.

Uso de strtok()

A diferencia de `explode()`, que divide una cadena en pequeñas partes de una sola vez, `strtok()` va extrayendo partes (llamadas símbolos) de la cadena una a una. Esta función es una alternativa útil a la función `explode()` para procesar una cadena palabra a palabra. La sintaxis de la propiedad `strtok()` es la siguiente:

```
string strtok(string entrada, string separador);
```

El separador puede ser un carácter o una cadena de caracteres pero la cadena de entrada se dividirá en cada uno de los caracteres de la cadena del separador en lugar de en toda la cadena (como ocurría con `explode`). La tarea de llamar a la función `strtok()` no resulta tan sencilla como pudiera parecer por su sintaxis.

Para obtener el primer símbolo de una cadena, se invoca `strtok()` con la cadena divida en las partes deseadas y un separador. Para obtener las siguientes partes de la cadena, se pasa un único parámetro: el separador. La función mantiene su propio puntero interno dentro de la cadena. Si desea restablecer el puntero, puede pasarle la cadena de nuevo.

La función `strtok()` se suele utilizar de la siguiente forma:

```
$token = strtok($feedback, ' ');
echo $token.'<br />';
while ($token!='')
{
   $token = strtok(' ');
   echo $token.'<br />';
};
```

Como de costumbre, conviene comprobar que el cliente ha escrito algún comentario en el formulario, utilizando la función `empty()`. En nuestro caso se ha omitido para no alargar el ejemplo.

Este código imprime cada símbolo del comentario del cliente en una línea distinta y recorre en bucle el comentario hasta que no quedan más símbolos. En las versiones de PHP anteriores a la 4.1.0, `strtok()` no funcionaba de forma exactamente igual a como lo hace en C. Si una cadena de destino contiene dos instancias de un separador seguidas (en este caso dos espacios), `strtok()` devuelve una cadena vacía. Este resultado no se puede diferenciar de la cadena vacía que se devuelve al llegar al final de la cadena de destino. Así mismo, si uno de los símbolos es 0, se devolverá la cadena vacía. Esto convierte a la función `strtok()` de PHP en algo menos útil que la de C. La nueva versión funciona correctamente ya que salta las cadenas vacías.

Uso de substr()

La función `substr()` permite acceder a una subcadena situada entre el punto inicial y el punto final de una cadena. En nuestro ejemplo no resulta apropiado pero puede ser útil si intenta acceder a partes de cadenas con formato fijo.

La función `substr()` tiene la siguiente sintaxis:

```
string substr(string cadena, int inicio[, int longitud] );
```

Esta función devuelve una subcadena copiada de *cadena*.

Si se llama a esta función con un número positivo desde el parámetro *inicio* (únicamente), se obtendrá la cadena desde la posición *inicio* hasta el final de la cadena. Por ejemplo:

```
substr($test, 1);
```

devuelve `our customer service is excellent`. Tenga en cuenta que la posición inicial es 0, como ocurre con las matrices.

Si llama a la función `substr()` con un valor *inicio* negativo (únicamente), obtendrá la cadena desde su final menos los caracteres *inicio* hasta el final de la cadena. Por ejemplo:

```
substr($test, -9);
```

En este caso se devuelve `excellent`.

El parámetro de longitud se puede utilizar para especificar un número de caracteres que devolver (si es positivo) o el carácter final de la secuencia de retorno (si es negativo). Por ejemplo:

```
substr($test, 0, 4);
```

devuelve los primeros cuatro caracteres de la cadena, en concreto, `Your`. El siguiente código:

```
echo substr($test, 4, -13);
```

devuelve los caracteres situados entre el cuarto y el carácter decimotercero, es decir, `customer service`.

Comparación de cadenas

Hasta el momento sólo hemos utilizado el operador == para comparar si dos cadenas son iguales. Pero PHP permite realizar comparaciones un poco más avanzadas. Las hemos dividido en dos categorías: coincidencias parciales y otras. En primer lugar estudiaremos las demás y seguidamente pasaremos a estudiar las coincidencias parciales, para lo cual avanzaremos en el desarrollo del ejemplo Smart Form.

Ordenación de cadenas:strcmp(),strcasecmp() y strnatcmp()

Estas funciones se pueden utilizar para ordenar cadenas y, por tanto, para ordenar datos.

La sintaxis de la función `strcmp()` es la siguiente:

```
int strcmp(string str1, string str2);
```

La función espera recibir dos cadenas, que comparará. Si son iguales, devolverá 0. Si la cadena `str1` viene tras (o es mayor que) `str2` en orden lexicográfico, `strcmp()` devolverá un número mayor que cero. Si `str1` es menor que `str2`, `strcmp()` devolverá un número menor que cero. Esta función discrimina entre mayúsculas y minúsculas.

La función `strcasecmp()` es idéntica con la diferencia de que no discrimina entre mayúsculas y minúsculas.

La función `strcasecmp()` y su pareja `strnatcasecmp()`, que no discrimina entre mayúsculas y minúsculas, se agregaron en PHP 4. Estas funciones comparan cadenas en función de un "orden natural", que se parece más a la forma en la que lo hace un humano. Por ejemplo, `strcmp()` ordenará la cadena `"2"` como mayor que la cadena `"12"` porque lexicográficamente resulta mayor.

`strcasecmp()` hace exactamente lo contrario. Si desea saber más sobre el orden natural, diríjase a `http://www.naturalordersort.org/`.

Comprobación de la longitud de una cadena con strlen()

Podemos comprobar la longitud de una cadena con la función `strlen()`. Si se pasa en una cadena, devolverá su longitud. Por ejemplo, `strlen('hello')` devuelve `5`.

Esta función se puede utilizar para validar datos de entrada. Considere la dirección de correo electrónico de nuestro formulario, almacenada en `$email`. Una forma elemental de validar una dirección de correo electrónico almacenada en `$email` consiste en comprobar su longitud. Si partimos de unos cálculos aproximados, la longitud mínima de una dirección de correo electrónico es de seis caracteres, por ejemplo `a@a.to` (suponiendo que el código del país no tiene dominios de segundo nivel, y que el nombre del servidor y la dirección sólo tienen una letra respectivamente). Por lo tanto, si la dirección no tiene esta longitud mínima, se generará un error:

```
if (strlen($email) < 6)
{
   echo 'That email address is not valid';
   exit;  // finalice la ejecución de la secuencia de comandos de PHP
}
```

Obviamente, se trata de una forma muy simple de validar la información. En la siguiente sección examinaremos formas mejores.

Cómo buscar subcadenas y reemplazarlas con funciones de cadena

Resulta habitual comprobar si una subcadena dada está presente en una cadena de mayor tamaño. Esta coincidencia parcial suele ser más útil que comprobar la igualdad entre cadenas.

En nuestro ejemplo Smart Form, queremos buscar determinadas frases clave dentro de los comentarios de los clientes y enviar el correo al departamento adecuado. Si

queremos enviar correos electrónicos relacionados con las tiendas de Bob al encargado del departamentos de ventas, tendremos que saber si aparece la palabra "shop" (o alguna variación) en el mensaje.

Podríamos utilizar funciones ya vistas, como `explode()` o `strtok()` para recuperar palabras individuales del mensaje y compararlas utilizando el operador `==` o `strcmp()`.

Sin embargo, podemos hacer lo mismo con una única llamada de función a una de las funciones de comparación de cadenas o de expresiones regulares. Éstas se utilizan para buscar un patrón dentro de una cadena. Examinaremos cada conjunto de funciones por separado.

Búsqueda de cadenas en cadenas: strstr(), strchr(), strrchr(), stristr()

Para buscar una cadena dentro de una cadena, podemos utilizar cualquiera de las funciones `strstr()`, `strchr()`, `strrchr()` o `stristr()`.

La función `strstr()` es la más genérica y se puede utilizar para buscar una cadena o caracter dentro de una cadena de mayor tamaño. Tenga en cuenta que en PHP, la función `strchr()` es exactamente igual que la función `strstr()`, aunque su nombre implique que se utiliza para buscar un carácter en una cadena, de manera similar a la versión de C de esta función. En PHP, se pueden utilizar las dos funciones para buscar una cadena dentro de otra cadena, incluyendo la posiblidad de buscar una cadena que contenga un solo carácter.

La sintaxis de `strstr()` es la siguiente:

```
string strstr(string pajar, string aguja);
```

En la función se pasa el parámetro *pajar* sobre el que buscar y un parámetro *aguja* que encontrar. Si se encuentra una coincidencia exacta de *aguja*, la función devolverá el *pajar* desde la *aguja* en adelante o, de lo contrario, devolverá `false`. Si la *aguja* tiene lugar en más de una ocasión, la cadena devuelta comenzará desde la primera ocurrencia de *aguja*.

Por ejemplo, en la aplicación Smart Form, podemos decidir dónde enviar el correo electrónico de la siguiente forma:

```
$toaddress = 'feedback@example.com';  // el valor predeterminado

// Cambie $toaddress si se cumple el criterio
if (strstr($feedback, 'shop'))
   $toaddress = 'retail@example.com';
else if (strstr($feedback, 'delivery'))
   $toaddress = 'fulfilment@example.com';
else if (strstr($feedback, 'bill'))
      $toaddress  =  'accounts@example.com';
```

Este código busca determinadas palabras claves en el comentario del usuario y envía el correo a la persona adecuada. Por ejemplo, si el cliente escribe "I still haven't

received delivery of my last order" (todavía no he recibido mi último pedido), se detectará la cadena "delivery" (entrega) y el comentario se enviará a `fulfilment@example.com`.

Existen dos variantes de `strstr()`. La primera es `stristr()`, que es prácticamente idéntica con la salvedad de que no distingue entre mayúsculas y minúsculas. Esta función resulta útil en nuestra aplicación ya que el cliente podría escribir `'delivery'`, `'Delivery'` o `'DELIVERY'`.

La segunda variante es `strrchr()`, que es, de nuevo, prácticamente idéntica, pero devuelve *pajar* desde la última ocurrencia de la *aguja* en adelante.

Búsqueda de la posición de una subcadena: strpos(), strrpos()

Las funciones `strpos()` y `strrpos()` funcionan de la misma forma que `strstr()`, con la diferencia de que en lugar de devolver una subcadena, devuelven la posición numérica de una *aguja* en un *pajar*.

La función `strpos()` tiene la siguiente sintaxis:

```
int strpos(string pajar, string aguja, int [desplazamiento] );
```

El valor entero devuelto representa la posición de la primera ocurrencia de *aguja* dentro de *pajar*. El primer carácter es la posición 0 como de costumbre.

Por ejemplo, el siguiente código imprimirá el valor `4` en el navegador:

```
$test = 'Hello world';
echo strpos($test, 'o');
```

En este caso, sólo hemos pasado un carácter como *aguja*, pero puede ser una cadena de cualquier longitud.

El parámetro opcional de desplazamiento se utiliza para especificar un punto dentro de *pajar* a partir del cual iniciar la búsqueda. Por ejemplo:

```
echo strpos($test, 'o', 5);
```

Este código imprimirá `7` en el navegador porque PHP ha comenzado a buscar el carácter `o` en la posición 5 y, por lo tanto, no verá el situado en la posición 4.

La función `strrpos()` es prácticamente idéntica, pero no devuelve la posición de la última ocurrencia de *aguja* en *pajar*. A diferencia de *strpos()*, sólo funciona con un único carácter *aguja*. Por lo tanto, si la pasamos en una cadena como *aguja*, sólo utilizará el primer carácter de la cadena para establecer la coincidencia. En cualquiera de estos casos, si *aguja* no está en la cadena, `strpos()` o `strrpos()` devolverá `false`, lo cual puede plantear un problema porque `false` en un lenguaje con control débil de tipos como PHP equivale a 0, es decir, al primer carácter de una cadena.

Para evitar este problema podemos utilizar el operador `===` y probar los valores devueltos:

```
$result = strpos($test, 'H');
```

```
if ($result === false)
   echo 'Not found'
else
   echo 'Found at position $';
```

Tenga en cuenta que sólo funcionará en PHP 4 ya que en las versiones anteriores puede probar si es `false` examinando el valor devuelto para determinar si se trata de una cadena (es decir, `false`).

Sustitución de subcadenas: str_replace(), substr_replace()

La función de buscar y reemplazar cadenas puede resultar extremadamente útil con cadenas. Se puede utilizar para personalizar documentos generados por PHP (por ejemplo, sustituyendo `<<nombre>>` con el nombre de una persona y `<<dirección>>` con su dirección). También puede utilizarla para censurar el uso de determinados términos, por ejemplo en un foro o incluso en la aplicación Smart Form.

De nuevo, puede utilizar funciones de cadena o funciones de expresión regular con este fin.

La función de cadena que más se suele utilizar para realizar sustituciones es `str_replace()`. A continuación se recoge su sintaxis:

```
mixed str_replace(mixed aguja, mixed nueva_aguja, mixed pajar);
```

Esta función sustituirá todas las instancias de *aguja* en *pajar* por *nueva_aguja* y devolverá la nueva versión de *pajar*.

> **Nota**
>
> Desde PHP 4.0.5 puede pasar todos los parámetros como matrices y la función se comportará de manera bastante inteligente. Puede pasar una matriz de palabras para su sustitución, una matriz de palabras con la que sustituirlas (respectivamente) y una matriz de cadenas a las que aplicar estas reglas. La función devolverá una matriz de cadenas revisadas.

En el formulario Smart Form podrían aparecer palabras malsonantes incluidas por los usuarios al manifestar sus quejas. Como programadores, podemos evitar que los distintos departamentos de Bob reciban los insultos:

```
$feedback = str_replace($offcolor, '%!@*', $feedback);
```

La función `substr_replace()` se utiliza para buscar y reemplazar una determinada subcadena de una cadena en función de su posición. Su sintaxis es la siguiente:

```
string substr_replace(string cadena, string sustitución, int inicio,
```

```
[,longitud] );
```

Esta función sustituirá parte de la cadena *cadena* con la cadena *sustitución*. La parte que se sustituirá depende de los valores de los parámetros *inicio* y *longitud*.

El valor *inicio* representa el desplazamiento dentro de la cadena en el que comenzará la sustitución. Si el valor fuera 0 o positivo, el desplazamiento se establecería con respecto al principio de la cadena; si fuera negativo, el desplazamiento se establecería con respecto al final de la cadena. Por ejemplo, esta línea de código sustituirá el último carácter de `$test` con "X":

```
$test = substr_replace($test, 'X', -1);
```

El valor *longitud* es opcional y representa el punto en el que PHP dejará de realizar sustituciones. Si no se indica este parámetro, la cadena será sustituida desde el *inicio* hasta el final de la cadena.

Si el valor del parámetro *longitud* es cero, la cadena de sustitución se insertará dentro de la cadena sin sobrescribir la cadena existente.

Una *longitud* positiva representa el número de caracteres que se sustituirán con la nueva cadena.

Una *longitud* negativa representa el punto en el que le gustaría detener la sustitución de caracteres, contados desde el final de la cadena.

Introducción a las expresiones regulares

PHP admite dos estilos de sintaxis de expresión regular: POSIX y Perl. El estilo POSIX de expresión regular se compila en PHP de manera predeterminada, pero se puede utilizar el estilo Perl compilando en biblioteca PCRE (Expresión regular y compatible con Perl).

En nuestro caso, utilizaremos el estilo POSIX que es más sencillo, pero si es programador de Perl o desea saber más sobre la biblioteca PCRE, consulte el manual en línea dirigiéndose a `http://php.net`.

Nota

Las expresiones regulares de POSIX resultan más sencillas de aprender y se ejecutan con más rapidez, pero no son seguras para datos binarios.

Por lo tanto, las coincidencias de patrón realizadas hasta ahora utilizaban funciones de cadena. Nos hemos limitado a las coincidencias exactas o a la coincidencia exacta de subcadenas. Si desea establecer coincidencias más complejas, debería utilizar expresiones regulares.

El uso de las expresiones regulares resulta difícil de comprender al principio pero pueden llegar a ser extremadamente útiles.

Los fundamentos

Una expresión regular es una forma de describir un patrón en un texto. La coincidencia exacta (o literal) que hemos realizado hasta el momento es una forma de expresión regular. Por ejemplo, en una sección anterior buscamos las expresiones regulares `"shop"` y `"delivery"`.

En PHP, el establecimiento de coincidencias con expresiones regulares se parece más a utilizar la función `strstr()` que a realizar una comparación de igualdad, porque se hace coincidir una cadena con alguna parte de otra cadena (puede ser cualquier parte de la cadena a menos que se especifique otra cosa.) Por ejemplo, la cadena `"shop"` coincide con la expresión regular `"shop"`. Pero también con las expresiones regulares `"h"`, `"ho"` etc.

Podemos utilizar caracteres especiales para indicar un metasignificado además de hacer coincidir caracteres de forma exacta.

Por ejemplo, mediante el uso de caracteres especiales se puede indicar que ocurra un patrón al inicio o al final de una cadena, que parte de un patrón se pueda repetir o que los caracteres de un patrón sean de un tipo dado. También podemos buscar coincidencias literales de caracteres especiales. Examinaremos todas estas posibilidades.

Conjuntos y clases de caracteres

El uso de conjuntos de caracteres aumenta el potencial de las expresiones regulares con respecto a las expresiones de coincidencia exacta. Los conjuntos de caracteres se pueden utilizar para hacer coincidir cualquier carácter de un tipo dado (en realidad son un tipo de comodín).

En primer lugar, puede utilizar el carácter como comodín para sustituir cualquier otro carácter individual a excepción del carácter de nueva línea (`\n`). Por ejemplo, la expresión regular

```
.at
```

coincide con las cadenas `'cat'`, `'sat'` y `'mat'`, entre otras.

Este tipo de comodín de coincidencia se suele utilizar para búsquedas de nombres de archivo en sistemas operativos.

Sin embargo, las expresiones regulares permiten especificar el tipo de carácter que se desea buscar y, de hecho, se puede especificar un conjunto al que pertenezca un carácter. En el ejemplo anterior, las expresiones regulares coinciden con `'cat'` y `'mat'` pero también con `'#at'`. Si desea limitar el rango de coincidencias a un carácter entre la a y la z, puede hacerlo de la siguiente forma:

```
[a-z]
```

Todo aquello que quede encerrado entre los corchetes (`[]`) se considerará una clase de carácter (un conjunto de caracteres a los que debe pertenecer un carácter

coincidente). Tenga en cuenta que la expresión incluida entre los corchetes coincide con un único carácter.

Puede incluir un conjunto de caracteres; por ejemplo:

```
[aeiou]
```

significa cualquier vocal.

También puede describir un rango, como hicimos anteriormente utilizando un guión o un conjunto de rangos.

```
[a-zA-Z]
```

Este conjunto de rangos equivale a cualquier carácter alfabético ya sea mayúsculas o minúsculas.

También puede utilizar conjuntos para excluir un carácter del conjunto. Por ejemplo:

```
[^a-z]
```

equivale a cualquier carácter que no esté entre la a y la z. El símbolo de acento circunflejo significa *no* cuando se coloca dentro de corchetes.

Si se utiliza fuera de los corchetes tiene otro significado, que examinaremos en breve.

Además de enumerar conjuntos y rangos, se pueden utilizar varias clases de carácter predefinido en una expresión regular, que se recogen en la tabla 4.3.

Tabla 4.3. Clases de caracteres para su uso en expresiones regulares de estilo de POSIX.

Clase	Coincidencia
`[[:alnum:]]`	Caracteres alfanuméricos
`[[:alpha:]]`	Caracteres alfabéticos
`[[:lower:]]`	Letras en minúsculas
`[[:upper:]]`	Letras en mayúsculas
`[[:digit:]]`	Dígitos decimales
`[[:xdigit:]]`	Dígitos hexadecimales
`[[:punct:]]`	Puntuación
`[[:blank:]]`	Tabuladores y espacios
`[[:space:]]`	Espacios en blanco
`[[:cntrl:]]`	Caracteres de control
`[[:print:]]`	Todos los caracteres imprimibles
`[[:graph:]]`	Todos los caracteres imprimibles a excepción del espacio

Repetición

Con frecuencia necesitamos especificar que se van a dar varias ocurrencias de una cadena o clase de carácter concreta. Para ello, podemos utilizar dos caracteres especiales dentro de una expresión regular. El símbolo * significa que el patrón se puede repetir cero o más veces y el símbolo + indica que el patrón se puede repetir una o varias veces. El símbolo puede aparecer directamente tras la parte de la expresión a la que se aplique. Por ejemplo:

```
[[:alnum:]]+
```

significa "al menos un carácter alfanumérico".

Subexpresiones

Suele resultar útil poder dividir una expresión en subexpresiones para representar, por ejemplo, "al menos una de estas cadenas seguidas exactamente por una de esas otras". Para ello, se pueden utilizar los paréntesis, exactamente de la misma forma que haría en una expresión aritmética. Por ejemplo:

```
(very )*large
```

equivale a `'large'`, `'very large'`, `'very very large'`, etc.

Recuento de subexpresiones

Podemos especificar cuántas veces se repite un elemento utilizando expresiones numéricas entre llaves (`{}`). Podemos mostrar el número exacto de repeticiones (`{3}` significa exactamente 3 repeticiones), un rango de repeticiones (`{2, 4}` significa de 2 a 4 repeticiones) o un rango sin cerrar de repeticiones (`{2,}` significa al menos dos repeticiones).

Por ejemplo,

```
(very ){1, 3}
```

equivale a `'very'`, `'very very '` y `'very very very '`.

Anclajes al principio o al final de una cadena

Podemos especificar si una subexpresión dada debería aparecer al final, al principio o en ambos lugares. Esta opción resulta bastante útil cuando queremos estar seguros de que en la cadena sólo aparece el término de búsqueda y nada más.

Se utiliza el acento circunflejo (^) al principio de una expresión regular para indicar que debe aparecer al principio de una cadena buscada. El símbolo $ se utiliza al final de una expresión regular para indicar que debe aparecer al final.

Por ejemplo, la siguiente expresión busca coincidencias con `bob` al principio de una cadena:

```
^bob
```

La siguiente expresión busca coincidencias con `com` al final de una cadena:

```
com$
```

Por último, la siguiente expresión busca coincidencias con cualquier carácter único de la a la z, en su propia cadena:

```
^[a-z]$
```

Bifurcación

Puede representar una opción en una expresión regular utilizado una barra vertical. Por ejemplo, si deseamos buscar coincidencias con `com`, `edu` o `net`, podemos utilizar la siguiente expresión:

```
(com)|(edu)|(net)
```

Cómo buscar coincidencias de caracteres especiales

Si desea buscar coincidencias con los caracteres especiales mencionados como `.`, `(` o `$`, debe anteponerles una barra invertida (`\`). Si desea representar una barra invertida, debe sustituirla por dos barras inversas, `\\`.

Resumen de los caracteres especiales

En las tablas 4.4 y 4.5 se recogen un resumen de todos los caracteres especiales. La tabla 4.4 muestra el significado de los caracteres especiales fuera de corchetes y la tabla 4.5 muestra su significado cuando se utilizan dentro ellos.

Tabla 4.4. Resumen de caracteres especiales utilizados en expresiones regulares POSIX fuera de corchetes

Carácter	Significado
\	Carácter de escape
^	Coincidencia al principio de la cadena
$	Coincidencia al final de la cadena
.	Coincidencia de cualquier carácter a excepción del carácter de nueva línea (\n)

Carácter	Significado
|	Inicio de una opción alternativa (como OR)
(	Inicio de un subpatrón
)	Final de un subpatrón
*	Repetir 0 o más veces
+	Repetir 1 o más veces
{	Inicio del cuantificador min/máx
{	Inicio del cuantificador min/máx

Tabla 4.5. Resumen de caracteres especiales utilizados en expresiones regulares POSIX dentro de corchetes

Carácter	Significado
\	Carácter de escape
^	NO, solamente si se utiliza en una posición inicial
-	Usado para especificar rangos de carácter

Uso de estos elementos en Smart Form

En la aplicación Smart Form, las expresiones regulares se pueden utilizar con dos fines. El primero consiste en detectar términos concretos en los comentarios enviados por los clientes.

Podemos utilizar expresiones regulares para ahorrarnos trabajo. Si utilizamos funciones de cadena, tendremos que realizar tres búsquedas si queremos encontrar `'shop'`, `'customer service'` o `'retail'` cuando bastaría con una expresión regular para realizar la misma operación:

```
shop|customer service|retail
```

El segundo uso consiste en validar las direcciones de correo electrónico del cliente en nuestra aplicación codificando el formato estandarizado de una dirección de correo electrónico en una expresión regular. El formato incluye varios caracteres alfanuméricos o de puntuación, seguidos por el símbolo `@`, por una cadena alfanumérica y guiones, seguidos por un punto, por más caracteres alfanuméricos y guiones, y posiblemente por más puntos, hasta llegar al final de la cadena, que se codifica de la siguiente forma:

```
^[a-zA-Z0-9_\-\.]+@[a-zA-Z0-9\-]+\.[a-zA-Z0-9\-\.]+$
```

La subexpresión `^[a-zA-Z0-9_\-\.]+` significa "la cadena debe comenzar con una letra, un número, un guión bajo, un guión medio o un punto o una combinación de estos caracteres".

El símbolo `@` equivale al literal `@`.

La subexpresión `[a-zA-Z0-9\-]+` equivale a la primera parte del nombre del host incluyendo caracteres alfanuméricos y guiones. Como puede observar se ha quitado la barra invertida delante del guión dado que se trata de un carácter especial cuando va entre corchetes.

La combinación `\.` equivale al literal `.`.

La subexpresión `[a-zA-Z0-9\-\.]+$` equivale al resto del nombre del dominio, incluyendo letras, números, guiones y más puntos si fueran necesarios, hasta el final de la cadena.

Si analiza esta expresión, descubrirá que es posible crear direcciones no válidas equivalentes a la expresión regular. Resulta prácticamente imposible capturar todos los casos, pero hemos logrado mejorar bastante la situación. Puede volver a definir esta expresión de muchas formas. Por ejemplo, puede enumerar todos los TDL válidos. Ahora bien, es necesario tener cuidado a la hora de restringir elementos ya que una función de validación que rechace un 1% de los datos válidos resulta mucho más molesta que una que permita un 10% de datos no válidos.

Búsqueda de subcadenas con expresiones regulares

La búsqueda de subcadenas es la aplicación principal de las expresiones regulares que acabamos de desarrollar. Las dos funciones disponibles en PHP para buscar coincidencias con expresiones regulares son `ereg()` y `eregi()`.

```
int ereg(string patrón, string búsqueda, array [coincidencias]);
```

Esta función busca la cadena *búsqueda* para encontrar coincidencias con la expresión regular *patrón*. Si se encuentran coincidencias para las subexpresiones de *patrón*, se almacenarán en la matriz coincidencias, un subexpresión por cada elemento de matriz.

La función `eregi()` es idéntica con la salvedad de que no discrimina entre mayúsculas y minúsculas.

Podemos adaptar el ejemplo Smart Form para utilizar expresiones regulares de la siguiente forma:

```
if  (!eregi('^[a-zA-Z0-9_\-\.]+@[a-zA-Z0-9\-]+\.[a-zA-Z0-9\-\.]+$',
$email))
{
   echo 'That is not a valid email address. Please return to the'
        .' previous page and try again.';
   exit;
}
```

```
$toaddress = 'feedback@example.com';// el valor predeterminado
if (eregi('shop|customer service|retail', $feedback))
   $toaddress = 'retail@example.com';
else if (eregi('deliver.*|fulfil.*', $feedback))
   $toaddress = 'fulfilment@example.com';
else if (eregi('bill|account', $feedback))
   $toaddress = 'accounts@example.com';

if (eregi('bigcustomer\.com', $email))
     $toaddress = 'bob@example.com';
```

Sustitución de cadenas con expresiones regulares

También podemos utilizar expresiones regulares para buscar y reemplazar subcadenas de la misma forma que utilizamos `str_replace()`.

Las dos funciones disponibles para realizar esta operación son `ereg_replace()` y `eregi_replace()`. La función `ereg_replace()` tiene la siguiente sintaxis:

```
string ereg_replace(string patrón, string sustitución, string
búsqueda);
```

Esta función busca la expresión regular *patrón* en la cadena *búsqueda* y la reemplaza con la cadena *sustitución*.

La función `eregi_replace()` es idéntica con la diferencia de que no discrimina entre mayúsculas y minúsculas.

División de cadenas con expresiones regulares

Otra función de expresión regular útil es `split()`, cuya sintaxis es la siguiente:

```
array split(string patrón, string búsqueda, int [máx]);
```

Esta función divide la cadena *búsqueda* en subcadenas en la expresión regular *patrón* y devuelve subcadenas en una matriz. El valor entero *máx* limita el número de elementos que puede incluir la matriz.

Esta función resulta útil para dividir nombres de dominio o fechas. Por ejemplo,

```
$domain = 'yallara.cs.rmit.edu.au';
$arr = split ('\.', $domain);
while (list($key, $value) = each ($arr))
   echo '<br />'.$value;
```

Esta función divide el nombre del host en cinco componentes e imprime cada uno en una línea distinta.

Comparación de funciones de cadenas y funciones de expresiones regulares

En general, la ejecución de funciones de expresiones regulares resulta menos eficaz que las funciones de cadena con funcionalidad similar. Si su aplicación es sencilla, utilice expresiones de cadena.

Lecturas adicionales

PHP consta de una gran cantidad de funciones de cadena. En este capítulo hemos visto las más sencillas, pero si tiene una necesidad concreta (como la traducción de caracteres al cirílico), consulte el manual en línea de PHP para averiguar si dispone de ella.

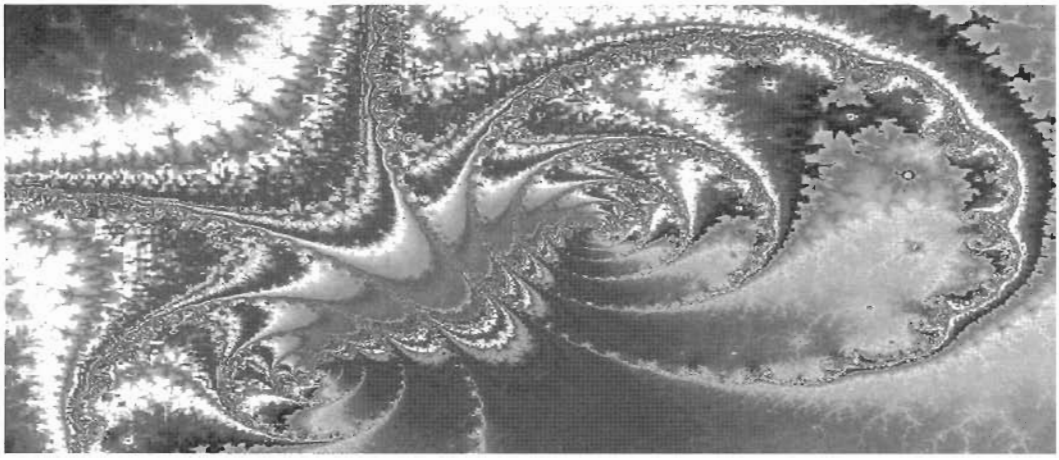
La cantidad de material disponible sobre expresiones regulares es ingente. Puede comenzar por la página `man` de la función `regexp` si está utilizando UNIX. También encontrará artículos muy buenos en `devshed.com` y `phpbuilder.com`.

En el sitio Web de Zend, se incluye una función de validación para correos electrónicos más compleja y potente que la desarrollada en este capítulo. Se denomina `MailVal()` y está disponible en `http://www.zend.com/codex.php?id=88&single=1`.

La correcta comprensión de las expresiones regulares lleva cierto tiempo. Cuanto más ejemplos examine y ejecute, más seguro se sentirá utilizándolas.

A continuación

En el siguiente capítulo, veremos varias formas de utilizar PHP para ahorrar tiempo y esfuerzos de programación, y evitar la redundancia al utilizar código preexistente.



5

Reutilización de código y creación de funciones

En este capítulo explicaremos cómo la reutilización del código permite desarrollar código más coherente, fiable y sencillo de mantener con menos esfuerzo. Mostraremos técnicas para dividir el código en módulos y reutilizarlo. Para ello, comenzaremos por explicar el sencillo uso de las instrucciones `require()` e `include()` que permiten utilizar el mismo código en varias páginas. Explicaremos por qué resultan preferibles a las inclusiones en el lado del servidor. En el ejemplo que presentaremos se recurre a archivos de inclusión para lograr un aspecto visual y operativo uniforme en todo el sitio. Explicaremos cómo escribir funciones propias y llamarlas utilizando funciones de generación de páginas y formularios como ejemplos. En este capítulo abordaremos los siguientes aspectos:

- Reutilización del código
- Utilización de `require()` e `include()`
- Introducción a las funciones
- Definición de funciones
- Parámetros
- Recuperación de valores
- Llamadas por referencias frente a llamadas por valor
- Ámbito
- Recursión

¿Por qué reutilizar código?

Uno de los objetivos de los ingenieros informáticos consiste en reutilizar código para no tener que escribirlo de nuevo. No porque se trate de un grupo especialmente vago sino porque la reutilización de código contribuye a reducir los costes, a aumentar la fiabilidad del código y a mejorar la uniformidad de los resultados. Lo ideal sería que los nuevos proyectos se crearan combinando componentes de código ya existentes, con un mínimo de desarrollo desde cero.

Costes

Durante la vida útil de un fragmento de código, se invierte más tiempo en modificarlo, probarlo y documentarlo que en crearlo. Si está escribiendo código comercial, debería intentar limitar el número de líneas que se utilizan dentro de la organización. Una forma práctica de lograrlo consiste en volver a utilizar código ya creado en lugar de escribir versiones ligeramente diferentes del mismo código para realizar una nueva tarea. Menos código significa menos costes. Si ya existe software que satisface sus necesidades para un proyecto, adquiéralo. El coste de comprar software existente es siempre inferior al coste de desarrollar un producto equivalente. Ponga mucha atención si el software existente no cubre exactamente sus necesidades ya que su modificación podría resultar más difícil que su creación desde cero.

Legibilidad

Si un módulo de código se utiliza en otra parte de la organización, es muy probable que se haya probado exhaustivamente.

Aunque se trata de unas pocas líneas, si se vuelve a escribir, es probable que pasemos algún elemento por alto incorporado por su autor original o que se agregó al código tras descubrir un fallo durante la fase de prueba. El código existente suele ser más fiable que el código recién creado.

Uniformidad

Las interfaces externas de nuestro sistema, incluidas las interfaces de usuario y las interfaces para los sistemas externos, deben ser uniformes. Para escribir código nuevo que resulte uniforme con respecto a la forma en la que funcionan el resto de las partes del sistema se necesita una voluntad y un esfuerzo deliberado. Si reutilizamos código ejecutado en otra parte del sistema, la uniformidad quedará garantizada automáticamente.

Sobre todas estas ventajas, la reutilización del código significa menos trabajo, siempre y cuando dicho código sea modular y esté bien escrito. Al trabajar, intente reconocer secciones de su código a las que pueda llamar en el futuro.

Uso de require() e include()

PHP incorpora dos instrucciones muy sencillas pero de gran utilidad para permitir la reutilización de cualquier tipo de código. Mediante el uso de las instrucciones `requiere()` o `include()` puede cargar un archivo en su secuencia de comandos de PHP. El archivo puede contener todos aquellos elementos que se incluirían por regla general en una secuencia de comandos, incluidas instrucciones de PHP, etiquetas HTML, funciones de PHP o clases de PHP.

Estas instrucciones funcionan de manera similar a las inclusiones del lado del servidor que ofrecen muchos servidores Web y a las instrucciones `#include` de C o C++.

Uso de require()

El siguiente código se almacena en un archivo llamado `reusable.php`:

```
<?php
   echo 'Here is a very simple PHP statement.<br />';
?>
```

El siguiente código se almacena en un archivo llamado `main.php`:

```
<?php
   echo 'This is the main file.<br />';
   require( 'reusable.php' );
   echo 'The script will end now.<br />';
?>
```

Si carga `resuable.php`, es probable que no le sorprenda ver aparecer la secuencia "Here is a very simple PHP statement." en el navegador. Si carga el archivo `main.php` ocurrirá algo un tanto más interesante. El resultado de esta secuencia de comandos se ilustra en la figura 5.1.

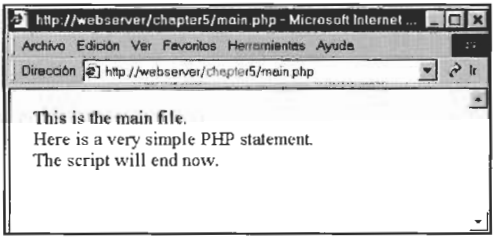


Figura 5.1. El resultado de main.php muestra el resultado de la instrucción require()

Se necesita un archivo para utilizar la instrucción `require()`. En el ejemplo anterior, utilizamos el archivo `reusable.php`. Al ejecutar nuestra secuencia de comandos, la instrucción `require()`

```
require( 'reusable.php' );
```

se sustituye por los contenidos del archivo solicitado y, a continuación, se ejecuta la secuencia de comandos. Esto implica que al cargar `main.php`, se ejecuta como si la secuencia de comandos estuviera escrita de la siguiente forma:

```
<?php
   echo 'This is the main file.<br />';
   echo 'Here is a very simple PHP statement.<br />';
   echo 'The script will end now.<br />';
?>
```

Al utilizar `require()` es necesario tener en cuenta las diferentes formas en las que se procesan las extensiones de nombre de archivo y las etiquetas de PHP.

Extensiones de nombre de archivo y require()

PHP no examina la extensión del nombre de archivo en el archivo requerido. Esto significa que puede asignar el nombre que desee a su archivo siempre y cuando no tenga previsto llamarlo directamente. Al utilizar `require()` para cargar el archivo, se convertirá de manera efectiva en parte de un archivo de PHP y se ejecutará como tal.

Por regla general, las instrucciones de PHP no se procesarán si estuvieran en un archivo llamado, por ejemplo, `page.html`. Sólo se llama a PHP para analizar archivos al definir extensiones como `.php`. Sin embargo, si carga la página `page.html` a través de una instrucción `require()`, se procesarán todas las instrucciones de PHP incluidas en su interior. Por lo tanto, puede utilizar cualquier extensión que desee para archivos de inclusión, pero conviene utilizar una convención lógica, como `.inc`. Recuerde que si almacena los archivos que terminan en `.inc` o alguna otra extensión no estándar en el árbol de documentos Web y los usuarios los cargan directamente en el navegador, podrán ver el código en forma de texto sin procesar, incluidas las contraseñas. Por lo tanto, es importante almacenar los archivos de inclusión fuera del árbol de documentos o utilizar extensiones estándar.

Etiquetas de PHP y require()

En nuestro ejemplo, el archivo reutilizable (`reusable.php`) se escribe de la siguiente forma:

```
<?php
   echo 'Here is a very simple PHP statement.<br />';
?>
```

El código de PHP se coloca dentro del archivo en etiquetas de PHP. Necesitará realizar esta operación si desea que el código PHP incluido en el archivo requerido sea tratado como código de PHP. Si no abre una etiqueta de PHP, su código será tratado como texto o código HTML y no se ejecutará.

Uso de require() para plantillas de sitios Web

Si las páginas Web de su compañía presentan un aspecto visual y operativo uniforme, puede utilizar PHP para agregar la plantilla y los elementos estándar utilizando `require()`.

Por ejemplo, las páginas del sitio Web de la compañía TLA Consulting presentan la apariencia que se muestra en la figura 5.2. Cuando se necesita crear una nueva página, el desarrollador puede abrir una existente, recortar el texto del archivo e introducir el nuevo texto, y guardar el archivo con otro nombre.

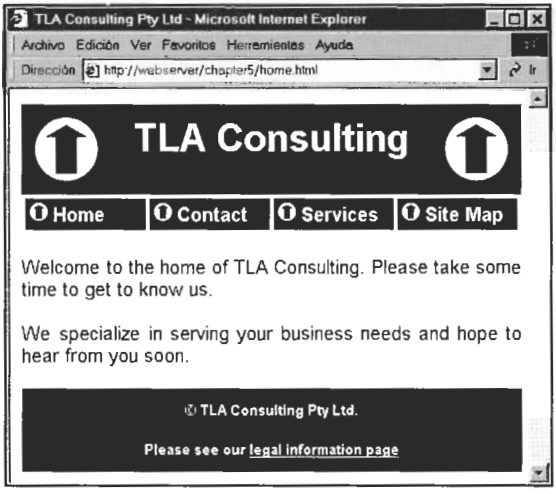


Figura 5.2. El sitio Web de TLA Consulting presenta un aspecto visual y operativo estándar en todas sus páginas

Considere la siguiente situación: imagine un sitio Web que lleva mucho tiempo activo y consta de decenas, centenas o miles de páginas con un estilo común. Entonces se toma la decisión de cambiar parte de su aspecto. Puede tratase de un cambio pequeño, como agregar una dirección de correo electrónico al pie de página de todas las páginas o incluir una nueva entrada en el menú de navegación. Seguro que no le apetece aplicar este pequeño cambio a todas las páginas.

La posibilidad de reutilizar secciones de código HTML comunes a todas las páginas es una opción mucho más práctica que cortar y pegar los cambios en decenas, centenas o miles páginas. En el listado 5.1 se recoge el código fuente de la página de inicio (`home.html`) ilustrada en la figura 5.2.

Listado 5.1. home.html. Código HTML de la página de inicio de TLA Consulting

```
<html>
<head>
```

```
    <title>TLA Consulting Pty Ltd</title>
    <style>
       h1  {color:white;  font-size:24pt;  text-align:center;
            font-family:arial,sans-serif}
       .menu {color:white; font-size:12pt; text-align:center;
              font-family:arial,sans-serif; font-weight:bold}
       td {background:black}
       p {color:black; font-size:12pt; text-align:justify;
          font-family:arial,sans-serif}
       p.foot {color:white; font-size:9pt; text-align:center;
               font-family:arial,sans-serif; font-weight:bold}
       a:link,a:visited,a:active {color:white}
    </style>
 </head>
<body>

    <!- page header ->
    <table width="100%" cellpadding="12" cellspacing="0" border="0">
    <tr bgcolor="black">
       <td  align="left"><img  src="logo.gif"></td>
       <td>
             <h1>TLA Consulting</h1>
       </td>
       <td align="right"><img src="logo.gif"></td>
    </tr>
    </table>

    <!- menu ->
    <table width="100%" bgcolor="white" cellpadding="4" cellspacing="4">
    <tr >
       <td width="25%">
          <img src="s-logo.gif"> <span class="menu">Home</span></td>
       <td width="25%">
          <img src="s-logo.gif"> <span class="menu">Contact</span></td>
       <td width="25%">
          <img src="s-logo.gif"> <span class="menu">Services</span></td>
       <td width="25%">
          <img src="s-logo.gif"> <span class="menu">Site Map</span></td>
    </tr>
    </table>

    <!- page content ->
    <p>Welcome to the home of TLA Consulting.
    Please take some time to get to know us.</p>
    <p>We specialize in serving your business needs
    and hope to hear from you soon.</p>

    <!- page footer ->
    <table width="100%" bgcolor="black" cellpadding="12" border="0">
    <tr>
       <td>
          <p class="foot">&copy; TLA Consulting Pty Ltd.</p>
          <p class="foot">Please see our <a href="legal.php">legal
information page</a></p>
       </td>
    </tr>
```

```
    </table>
</body>
</html>
```

En el listado 5.1 puede ver que el archivo consta de un número de secciones distintas. El encabezado contiene las definiciones CSS (del inglés, Cascading Style Sheet, Hoja de estilo en cascada) utilizada por esta página. La sección titulada "page header" (encabezado de página) muestra el nombre de la compañía y el logotipo, la sección "menu bar" (barra de menús) crea la barra de navegación de la página y la sección "page content" (contenido de página) integra el texto exclusivo de esta página. A continuación, aparece el pie de página. Podemos dividir este archivo y designar las distintas partes como `header.inc`, `home.php` y `footer.inc`. Las secciones `header.inc` y `footer.inc` contendrán código que se reutilizará en otras páginas.

El archivo `home.php` sustituye a `home.html` y contiene el texto exclusivo de esta página y dos instrucciones `require()` como se muestra en el listado 5.2.

Listado 5.2. home.php. El código PHP que produce la página de inicio de TLA

```
<?php
    require('header.inc');
?>
    <!- page content ->
    <p>Welcome to the home of TLA Consulting.
    Please take some time to get to know us.</p>
    <p>We specialize in serving your business needs
    and hope to hear from you soon.</p>
<?php
    require('footer.inc');
?>
```

La instrucción `require()` de `home.php` carga los archivos `header.inc` y `footer.inc`.

Como se mencionó, el nombre asignado a estos archivos no afecta a la forma en la que se procesan al llamarlos con `require()`. Una convención habitual, aunque del todo opcional, consiste en llamar a archivos

parciales que terminarán incluidos en otros archivos `algo.inc` (aquí `inc` equivale a incluir). También resulta habitual, además de conveniente, colocar los archivos que se incluirán dentro de un directorio al que puedan acceder sus secuencias de comandos pero no permitir que se carguen individualmente a través del servidor Web. De esta forma impediremos su carga, lo que dará lugar a a) que se produzcan errores si la extensión del archivo es `.php` pero sólo contienen una página o secuencia de comandos parcial o b) permitir que se pueda leer el código fuente si ha utilizado otra extensión.

El archivo `header.inc` contiene las definiciones de CSS utilizadas por la página, las tablas que muestran el nombre de la compañía y los menús de navegación como se ilustran en el listado 5.3.

El archivo `footer.inc` contiene la tabla que muestra el pie de página en la parte inferior de cada página. Este archivo se muestra en el listado 5.4.

Listado 5.3. header.inc. El encabezado reutilizable para todas las páginas Web de TLA

```
<html>
 <head>
  <title>TLA Consulting Pty Ltd</title>
  <style>
     h1 {color:white; font-size:24pt; text-align:center;
         font-family:arial,sans-serif}
     .menu {color:white; font-size:12pt; text-align:center;
           font-family:arial,sans-serif; font-weight:bold}
     td {background:black}
     p {color:black; font-size:12pt; text-align:justify;
        font-family:arial,sans-serif}
     p.foot  {color:white;  font-size:9pt;  text-align:center;
              font-family:arial,sans-serif; font-weight:bold}
     a:link,a:visited,a:active {color:white}
  </style>
</head>
<body>

  <!- page header ->
  <table width="100%" cellpadding="12" cellspacing="0" border="0">
  <tr bgcolor="black">
     <td align="left"><img src="logo.gif"></td>
     <td>
          <h1>TLA Consulting</h1>
     </td>
     <td align="right"><img src="logo.gif"></td>
  </tr>
  </table>

  <!- menu ->
  <table width="100%" bgcolor="white" cellpadding="4" cellspacing="4">
  <tr >
     <td width="25%">
        <img src="s-logo.gif"> <span class="menu">Home</span></td>
     <td width="25%">
        <img src="s-logo.gif"> <span class="menu">Contact</span></td>
     <td width="25%">
        <img src="s-logo.gif"> <span class="menu">Services</span></td>
     <td width="25%">
        <img src="s-logo.gif"> <span class="menu">Site Map</span></td>
  </tr>
  </table>
```

Listado 5.4. footer.inc. El pie de página reutilizable para todas las páginas Web de TLA

```
<!- page footer ->
  <table width="100%" bgcolor="black" cellpadding="12" border="0">
  <tr>
     <td>
        <p class="foot">&copy; TLA Consulting Pty Ltd.</p>
        <p class="foot">Please see our
                        <a href="legal.php">legal information page</a></p>
     </td>
```

```
    </tr>
    </table>
  </body>
  </html>
```

Este enfoque facilita la tarea de lograr una apariencia uniforme en un sitio Web y permite crear una página Web con el mismo etilo escribiendo algo así como:

```
<?php require('header.inc'); ?>
Here is the content for this page
<?php require('footer.inc'); ?>
```

Pero incluso tras crear una gran cantidad de páginas utilizando este encabezado y este pie de página, resulta sencillo modificar sus respectivos archivos. Con independencia del tipo de cambio que desee realizar (una pequeña variación o modificar el diseño completo del sitio), sólo necesitará realizar el cambio una vez. No tendrá que variar cada página del sitio porque todas ellas se cargan en los archivos de encabezado y pie de página.

El ejemplo visto utiliza únicamente HTML en el cuerpo, en el encabezado y en el pie de página.

Pero no tiene por qué ser así. Dentro de estos archivos, podríamos utilizar instrucciones de PHP para generar dinámicamente partes de las páginas.

Uso de auto_prepend_file y auto_append_file

Si queremos utilizar la instrucción `require()` para agregar el encabezado y el pie de página de todas las páginas, existe otra forma de hacerlo. Dos de las opciones de configuración del archivo php.inc son `auto_prepend_file` y `auto_append_file`.

Si asignamos estas opciones a los archivos de encabezado y pie de página, estaremos seguros de que se cargarán antes y después de cada página.

En Windows, la configuración presentará este aspecto:

```
auto_prepend_file  =  "c:/inetpub/include/header.inc"
auto_append_file = "c:/inetpub/include/footer.inc"
```

En UNIX, la configuración presentará este aspecto:

```
auto_prepend_file  =  "/home/username/include/header.inc"
auto_append_file = "/home/username/include/footer.inc"
```

Si utiliza estas directivas no necesitará escribir las instrucciones `require()`, pero los encabezados y pies de página dejarán de ser opcionales.

Si está utilizando un servidor Web Apache, puede cambiar varias opciones de configuración como éstas en directorios individuales. Para ello, debe configurar el servidor para permitir que el archivo o archivos de configuración principales sean sustituidos.

Para configurar las funciones de anexión al principio o al final de un directorio, cree un archivo llamado `.htaccess` en el directorio. Este archivo debe contener las siguientes dos líneas:

```
php_value  auto_prepend_file  "/home/username/include/header.inc"
php_value  auto_append_file  "/home/username/include/footer.inc"
```

Fíjese en que la sintaxis resulta ligeramente diferente con respecto a la misma opción en php.ini así como el elemento `php_value` situado al principio de la línea. No hay signo igual. También puede alterar otra serie de parámetros de configuración de esta forma.

La sintaxis ha cambiado desde PHP 3. Si está utilizando una versión antigua de PHP, las líneas del archivo `.htaccess` presentarán este aspecto:

```
php3_auto_prepend_file  /home/username/include/header.inc
php3_auto_append_file  /home/username/include/footer.inc
```

La posibilidad de establecer opciones en el archivo `.htaccess` en lugar de hacerlo en el archivo php.ini o en los archivos de configuración de su servidor Web brindan una gran versatilidad. Puede alterar los parámetros en un equipo compartido que sólo afecte a sus directorios. No es necesario reiniciar el servidor Web ni disponer de acceso de administrador. Una de las desventajas del método `.htaccess` es que los archivos se leen y se analizan cada vez que se solicita un archivo de dicho directorio en lugar de uno al inicio, lo que afecta al rendimiento.

Uso de include()

Las instrucciones `require()` y `include()` son prácticamente idénticas. La única diferencia entre ambas es que cuando fallan, la instrucción `require()` devuelve un error terminal, mientras que la instrucción `include()` devuelve una advertencia.

Existe un cambio en cuanto al funcionamiento de estas instrucciones. En las versiones de PHP anteriores a la 4.0.2, existían diferencias importantes en su funcionamiento. Si está utilizando una versión antigua de PHP, lea las siguientes líneas.

La instrucción `include()` se evalúa cada vez que se ejecuta y no se evalúa si no se ejecuta. La instrucción `require()` se ejecuta la primera vez que se analiza la instrucción, independientemente de si el bloque de código que la contiene se ejecuta.

Este hecho no tiene mucha importancia si el tráfico del servidor es reducido pero implica que el código que incluya instrucciones `require()` dentro de estructuras condicionales resultará ineficaz.

```
if($variable == true)
{
   require('file1.inc');
}
else
{
   require('file2.inc');
}
```

Este código cargará innecesariamente ambos archivos cada vez que se cargue la secuencia de comandos, pero sólo utilizará una en función del valor de `$variable`. Sin embargo, si el código se ha escrito utilizando dos instrucciones `include()`, sólo se cargará uno de los archivos y se utilizará como en la siguiente versión:

```
if($variable == true)
{
   include('file1.inc');
}
else
{
   include('file2.inc');
}
```

A diferencia de los archivos cargados a través de una instrucción `require()`, aquellos cargados con `include()` pueden devolver un valor. Por lo tanto, podemos indicar a otras partes del programa sobre el éxito o el fallo del archivo incluido o devolver una respuesta o resultado.

Si vamos a abrir archivos muchas veces, podemos encargar la tarea a un archivo de inclusión en lugar de volver a escribir las mismas líneas de código en cada ocasión. Este archivo podría llamarse "`openfile.inc`" y presentaría un aspecto parecido al siguiente:

```
<?php
@ $fp = fopen($name, $mode);
   if (!$fp)
   {
      echo '<p><strong> Oh No! I could not open the file.</strong></p>';
      return 0;
   }
   else
   {
      return 1;
   }
?>
```

Este archivo intentará abrir el archivo `$name` utilizando el modo indicado en `$mode`. Si fallara, devolvería un mensaje de error y un 0. Si tuviera éxito, devolvería un 1 y no generaría ningún mensaje. Puede llamar a este archivo en una secuencia de comandos de la siguiente forma:

```
$name = 'file.txt';
$mode = 'r';
$result = include('openfile.php');
if( $result == 1 )
{
   // haga lo que desee hacer con el archivo
   // haga referencia a $fp creado en el archivo de inclusión
}
```

Tenga en cuenta que podemos crear variables en el archivo principal o en el de inclusión u obligatorio, y que la variable existirá en ambos.

Este comportamiento es el mismo para las instrucciones `require()` y `include()`.

No puede utilizar `require()` de la misma forma que se ha mostrado aquí porque no es posible devolver valores desde esta instrucción. La devolución de valores puede resultar útil porque nos permite notificar a otras partes del programa sobre un fallo, realizar procesamientos independientes o devolver una respuesta. Las funciones proporcionan un mecanismo que resulta incluso mejor para dividir el código en módulos autónomos, como veremos en la siguiente sección.

Si se está preguntando por qué debería utilizar `require()`, dadas las ventajas de la instrucción `include()` sobre `require()`, la respuesta que es que resulta ligeramente más rápida. Como se mencionó anteriormente, este comportamiento ha cambiado, por lo que puede evitar el problema actualizando su versión de PHP.

Uso de las funciones de PHP

La mayoría de los lenguajes de programación constan de funciones. Éstas se utilizan para separar el código que realiza una tarea simple y bien definida. Las funciones facilitan la lectura del código y nos permiten volver a utilizarlo cuando necesitamos realizar la misma tarea.

Una función es un módulo de código autónomo que establece una interfaz de llamada, realiza alguna tarea y, opcionalmente, devuelve un resultado.

Ya hemos visto una serie de funciones. En los capítulos anteriores, hemos llamado a varias funciones integradas en PHP. También hemos creado algunas funciones sencillas y se han comentado. En esta sección vamos a analizar en mayor detalle las operaciones de llamar y desarrollar funciones.

Llamada de funciones

La siguiente línea representa la llamada más sencilla a una función:

```
nombre_función();
```

Esta línea llama a una función denominada `nombre_función()` que no requiere parámetros. Esta línea de código ignora cualquier valor que pudiera devolver la función.

Existen varias funciones que se llaman de la misma forma. La función `phpinf()` suele resultar útil para realizar pruebas porque muestra la versión de PHP instalada, información sobre PHP, la configuración del servidor y valores sobre varias variables de PHP y de servidor. Esta función no toma parámetros y por regla general ignora el valor devuelto. Las llamadas a `phpinfo()` presentan este aspecto:

```
phpinfo();
```

La mayor parte de las funciones requieren uno o más parámetros: información que se le da a una función al llamarla y que influye sobre el resultado de la ejecución de la

función. Los parámetros se pasan colocando los datos o el nombre de una variable con los datos incluidos dentro del paréntesis tras su nombre. La llamada a una función con un parámetro presenta este aspecto:

```
nombre_función('parámetro');
```

En ese caso, el parámetro utilizado es una cadena que contiene únicamente la palabra `parámetro`, pero las siguientes llamadas también son correctas según la función:

```
nombre_función(2);
nombre_función(7.993);
nombre_función($variable);
```

En la última línea, `$variable` puede ser cualquier tipo de variable de PHP, incluso una matriz. Un parámetro puede ser cualquier tipo de datos pero las funciones concretas suelen necesitar tipos de datos dados.

Para saber cuántos parámetros tomó una función, qué representa cada uno de ellos y el tipo que necesitan, basta con consultar su sintaxis. Al describir las funciones solemos incluir su sintaxis. A continuación, se recoge la sintaxis de la función `fopen()`:

```
int fopen( string nombre_de_archivo, string modo, [int
usar_ruta_inclusión] );
```

Esta sintaxis nos indica varias cosas y es importante saber interpretarlas. En concreto, el término `int` que aparece delante del nombre de la función indica que devolverá un valor entero. Los parámetros de la función se incluyen dentro de los paréntesis. En el caso de `fopen()` tenemos tres. El nombre del archivo y el modo son cadenas y el parámetro es un entero.

Los corchetes que rodean a la expresión *usar_ruta_inclusión* indican que se trata de un parámetro opcional. Podemos incluir valores o ignorar el parámetro, en cuyo caso se utilizará el predeterminado. Tras examinar la sintaxis de esta función, sabemos que el siguiente fragmento de código es una llamada válida a `fopen()`:

```
$name = 'myfile.txt';
$openmode = 'r';
$fp = fopen($name, $openmode)
```

Este código llama a la función `fopen()`. El valor devuelto se almacenará en la variable `$fp`. Hemos optado por pasarle una variable llamada `$name` que contiene una cadena que representa el archivo que queremos abrir y una variable llamada `$openmode` que contiene una cadena que representa el modo en el que queremos abrir el archivo. En cuanto al tercer parámetro opcional, hemos decidido no utilizarlo.

Cómo llamar a una función no definida

Si intenta llamar a una función que no existe, obtendrá un mensaje de error como el ilustrado en la figura 5.3.

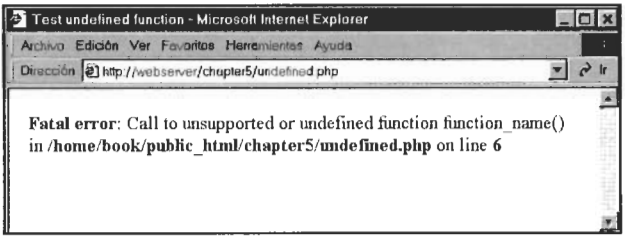


Figura 5.3. Este mensaje de error es el resultado de llamar a una función que no existe

Los mensajes de error que PHP devuelve suelen ser de gran utilidad. Éste nos indica exactamente en qué archivo tuvo lugar el error, en qué línea de la secuencia de comandos y el nombre de la función que intentamos llamar. De esta forma resultará muy sencillo buscar error y corregirlo.

Si se recibe este mensaje de error, hay que comprobar dos cosas:

1. ¿Está escrita correctamente la función?
2. ¿Existe la función en la versión de PHP que está utilizando?

No siempre resulta sencillo recordar cómo se escribe una función. Por ejemplo, algunos nombres de función formados por dos palabras utilizan un guión bajo para separarlas y otras no. La función `stripslashes()` se compone de dos palabras juntas mientras que la función `strip_tags()` utiliza un guión bajo para separar sus dos palabras. Si se escribe incorrectamente el nombre de una función en una llamada se devolverá el error ilustrado en la figura 5.3.

Muchas de las funciones que utilizamos en este libro no existen en PHP 3.0 ya que asumimos que está utilizando la versión PHP 4.0. En cada nueva versión se definen nuevas funciones por lo que las nuevas funcionalidades y el mayor rendimiento justifican la actualización. Si desea saber cuándo se ha agregado una función concreta, puede consular el manual en línea. Si intenta llamar a una función no declarada en la versión que esté ejecutando obtendrá como resultado un error como el que se ilustra en la figura 5.3.

Funciones y el uso de mayúsculas y minúsculas

Recuerde que las llamadas a funciones no discriminan entre mayúsculas y minúsculas por lo que las secuencias `function_name()`, `Function_Name()` o `FUNCTION_NAME()` son igualmente válidas y devuelven el mismo resultado. Puede utilizar el patrón que le resulte más sencillo de leer pero debe intentar mantener cierta uniformidad. En este libro, así como en la mayor parte de la documentación de PHP, se utilizan las minúsculas como convención.

Es importante recordar que los nombres de función se comportan de manera diferente a los nombres de variable. Los nombres de variable discriminan entre mayúsculas y minúsculas, por lo que `$Name` y `$name` serán dos variables diferentes mientras que `Name()` y `name()` identifican la misma función.

En los capítulos anteriores se ha incluido una gran cantidad de ejemplos sobre el uso de las funciones que incorpora PHP. Sin embargo, el verdadero potencial de un lenguaje de programación se mide por su capacidad para crear funciones propias.

¿Por qué debería crear funciones propias?

Las funciones incorporadas de PHP nos permiten interactuar con archivos, utilizar una base de datos, crear gráficos y establecer conexiones a otros servidores. Sin embargo, a lo largo de su vida profesional descubrirá muchas veces que necesita realizar una operación no prevista por los creadores del lenguaje.

Afortunadamente, PHP permite crear funciones para realizar cualquier tarea deseada. Con toda probabilidad, su código combinará funciones existentes con lógica propia desarrollada para realizar una tarea. Si está escribiendo un bloque de código que probablemente desee utilizar de nuevo en varios puntos de una secuencia de comandos o en varias secuencias de comandos, es aconsejable que declare dicho bloque en forma de función. La declaración de una función permite utilizar el código creado de la misma forma que las funciones incorporadas. Bastará con llamar a la función y definir los parámetros necesarios, lo que implica que puede llamar a la función y volver a utilizarla varias veces en la secuencia de comandos.

Estructura básica de una función

La declaración de una función crea o declara una nueva función. La declaración comienza con la palabra clave `function`, incluye el nombre de la función, los parámetros obligatorios y contiene el código que se ejecutará cada vez que se llame a la función. A continuación, se recoge el ejemplo de una función trivial:

```
function  my_function()

{
   echo 'My function was called';
}
```

Esta declaración de función comienza con el término `function`, para que los lectores humanos y el analizador de PHP sepan que lo que viene a continuación es una función definida por el usuario. El nombre de la función es `my_function`. Podemos llamar a nuestra nueva función con la siguiente instrucción:

```
my_function();
```

Como habrá supuesto, la llamada a esta función devolverá la frase "My function was called" en el navegador.

Las funciones integradas están disponibles para su uso en todas las secuencias de comandos de PHP, pero si declara sus propias funciones, sólo estarán disponibles

para la secuencia o secuencias de comandos en las que se declaren. Conviene incluir en un archivo las funciones más utilizadas y recurrir a un instrucción `require()` dentro de las secuencias de comandos para acceder a ellas.

Dentro de una función, las llaves encierran el código que realiza la tarea requerida. En su interior se pueden incluir todos aquellos elementos válidos de PHP como llamadas de función, declaraciones de nuevas variables o funciones, instrucciones `require()` o `include()` y código HTML. Si queremos salir de PHP dentro de una función y escribir HTML, podemos hacerlo de la misma forma que en cualquier otra parte de la secuencia de comandos: mediante una etiqueta de cierre de PHP seguida del código HMTL. A continuación se recoge una modificación válida del ejemplo anterior que genera el mismo resultado:

```
<?php
   function my_function()
   {
?>
My function was called
 <?php
   }
?>
```

Fíjese en que el código de PHP aparece encerrado entre etiquetas de apertura y cierre de PHP. En la mayor parte de los ejemplos de código utilizados en este libro se incluyen estas etiquetas. En este caso se muestran porque resultan necesarias dentro del ejemplo así como por encima y por debajo de él.

Designación de funciones

Lo más importante que recordar a la hora de asignar nombres a las funciones es que deberían ser cortos y descriptivos. Si su función crea un encabezado de página, `encabezadopag()` o `encazado_pag` serían buenas elecciones.

A continuación se indican varias limitaciones:

- No se puede utilizar el mismo nombre asignado a una función existente.
- El nombre de la función sólo puede contener letras, dígitos y guiones bajos.
- El nombre de la función no puede comenzar por un dígito.

Muchos lenguajes permiten volver a utilizar nombres de funciones. Esta opción se conoce como sobrecarga de funciones. Sin embargo, PHP no admite esta posibilidad por lo que sus funciones no pueden tener el mismo nombre que una función incorporada o una función existente definida por el usuario.

Fíjese en que aunque las secuencias de comando de PHP conocen las funciones integradas en el lenguaje, las funciones definidas por el usuario sólo existen en las secuencias de comandos declaradas. Por lo tanto, podemos volver a utilizar el nombre de una función en un archivo diferente aunque esta práctica genera confusión y debería evitarse.

Los siguientes nombres de función son válidos:

```
name()
name2()
name_three()
_namefour()
```

Estos otros no lo son:

```
5name()
name-six()
fopen()
```

(El último sería válido si no existiese una función con el mismo nombre.)

Parámetros

Para poder realizar su trabajo, la mayor parte de las funciones requieren el uso de uno o varios parámetros. Un parámetro permite pasar datos dentro de una función. A continuación, se recoge el ejemplo de una función que requiere un parámetro. Esta función toma una matriz unidimensional y la muestra en una tabla.

```
function create_table($data)

{
   echo '<table border="1">';
   reset($data); // Recuerde que se utiliza para apuntar al inicio
   $value = current($data);
   while ($value)
   {
      echo "<tr><td>$value</td></tr>\n";
      $value = next($data);
   }
   echo '</table>';
}
```

Si llamamos a la función `create_table()` de la siguiente forma:

```
$my_array = array('Line one.','Line two.','Line three.');
create_table($my_array);
```

se generará el resultado ilustrado en la figura 5.4.

Los parámetros nos permiten obtener datos creados fuera de la función (en este caso la matriz `$data`) dentro de la función.

Como ocurre con las funciones incorporadas, las definiciones establecidas por el usuario pueden tener varios parámetros así como parámetros opcionales. Podemos mejorar la función `create_table()` de varias formas, pero una de ellas podría consistir en permitir que el invocador especifique el borde de la tabla u otros atributos. A continuación, se recoge una versión mejorada de la función. Resulta muy similar, pero nos permite establecer la anchura del borde de la tabla así como el espacio y relleno de las celdas.

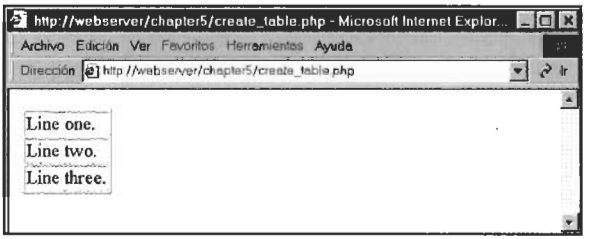


Figura 5.4. Esta tabla HTML es el resultado de llamar a la función create_table()

```
function create_table2( $data, $border =1, $cellpadding = 4, $cellspacing =
4 )
{
   echo "<table border='$border' cellpadding = '$cellpadding'"
        ." cellspacing='$cellspacing'>";
   reset($data);
   $value = current($data);
   while ($value)
   {
       echo "<tr><td>$value</td></tr>\n";
       $value = next($data);
   }
   echo '</table>';
}
```

El primer parámetro `create_table2()` sigue siendo obligatorio. Los otros tres parámetros son opcionales porque se han definido valores predeterminados. Podemos obtener un resultado muy similar al ilustrado en la figura 5.4 con la siguiente llamada a `create_table2()`.

```
create_table2($my_array);
```

Si queremos mostrar los mismos datos con más detalles, podríamos llamar a la función de la siguiente forma:

```
create_table2($my_array, 3, 8, 8);
```

No es necesario indicar los valores opcionales (podemos suministrar algunos e ignorar otros). Los parámetros se asignarán de izquierda a derecha.

Recuerde que no se puede dejar sin definir un parámetro opcional y definir otro situado por detrás. En este ejemplo, si desea pasar un valor para `cellspacing`, también deberá pasar otro `cellpadding`. Ésta suele ser una fuente común de errores de programación. También es la razón por la que los parámetros opcionales se especifican al final en cualquier lista de parámetros.

La siguiente llamada de función:

```
create_table2($my_array,  3);
```

resulta perfectamente válida: establece `$border` en 3 y deja los parámetros `$cellpadding` y `$cellspacing` en sus valores predeterminados.

Ámbito

Es posible que se haya fijado en que cuando necesitamos utilizar variables dentro de un archivo requerido o incluido, basta con declararlo en la secuencia de comandos antes de la instrucción `requiere()` o `include()`, pero al utilizar una función, pasamos dichas variables dentro de la función de manera explícita. Esto se debe en parte a que no existe ningún mecanismo para pasar variables explícitamente a un archivo requerido o incluido y en parte a que el ámbito de las variables se comporta de forma diferente en las funciones.

El ámbito de una variable controla dónde resulta visible y el lugar en el que se puede utilizar. Cada lenguaje de programación adopta reglas diferentes a la hora de establecer el ámbito de las variables. Las de PHP son bastantes simples:

- El ámbito de variables declaradas dentro de una función abarca desde la instrucción en la que se han declarado hasta las llaves de cierre situadas al final de la función. Éste ámbito es el ámbito de función y las variables declaradas así se denominan variables locales.
- El ámbito de las variables declaradas fuera de funciones abarca desde la instrucción en la que se declaran hasta el final del archivo, pero no en el interior de funciones. Este ámbito es el ámbito global y las variables declaradas así se denominan variables globales.
- Las variables superglogales resultan visibles tanto dentro como fuera de las funciones (si desea obtener un listado de estas variables, consulte un capítulo anterior).
- El uso de las instrucciones `require()` e `include()` no afecta al ámbito. Si la instrucción se utiliza dentro de una función, se aplicará el ámbito de la función. Si no se encuentra dentro de una función, se aplica el ámbito de global.
- La palabra clave `global` se puede utilizar manualmente para especificar que una variable definida o utilizada dentro de una función tiene ámbito global.
- Las variables se pueden eliminar manualmente llamando a `unset($nombre_variable)`. Una variable deja de tener ámbito si se elimina.

Los siguientes ejemplos le ayudarán a aclarar estos conceptos.

El fragmento de código que se incluye a continuación no genera ningún resultado. En concreto, se declara una variable llamada `$var` dentro de la función `fn()`. Como la variable se declara dentro de una función, su ámbito es el de una función y sólo existirá desde donde se declara hasta el final de la función. Cuando volvamos a hacer referencia a `$var` fuera de la función, se creará una nueva función llamada `$var`. Esta nueva variable tiene ámbito global y estará visible hasta el final del archivo. Por desgracia, si la única instrucción que utilizamos con la nueva variable `$var` es `echo`, nunca tendrá un valor.

```
function fn()
{
   $var = 'contents';
}
echo  $var;
```

El siguiente ejemplo muestra lo contrario. En él se declara una variable fuera de la función y se intenta utilizar dentro de otra.

```
function  fn()
{
   echo 'inside the function, $var = '.$var.'<br />';
   $var = 'contents2';
   echo 'inside the function, $var = '.$var.'<br />';
}
$var = 'contents 1';
fn();
echo 'outside the function, $var = '.$var.'<br />';
```

El resultado de este código será el siguiente:

```
inside the function, $var =
inside the function, $var = contents 2
outside the function, $var = contents 1
```

Las funciones no se ejecutan hasta que no se llaman, por lo que la primera instrucción es `$var = 'contents 1';`. Ésta crea una variable llamada `$var`, con ámbito global y su contenido es `"contents 1"`. La siguiente instrucción ejecutada es una llamada a la función `fn()`. Las líneas incluidas dentro de la instrucción se ejecutan en orden. La primera línea de la función hace referencia a la variable llamada `$var`.

Al ejecutar esta línea, no se puede ver la variable `$var` creada anteriormente, por lo que crea una nueva con ámbito de función e imprime su valor. Esto crea la primera línea de los resultados.

La siguiente línea de la función establece el contenido de `$var` en `"contents 2"`. Como estamos dentro de la función, esta línea cambia el valor de la variable `$var` local, no de la global. La segunda línea del resultado verifica que este cambio funciona.

Con esto llegamos al final de la función y se ejecuta la secuencia de comandos. La instrucción `echo` demuestra que el valor de la variable global no ha cambiado.

Si queremos que una variable creada dentro de una función tenga ámbito global, podemos utilizar la palabra clave `global` de la siguiente forma:

```
function  fn()
{
   global $var;
   $var = 'contents';
   echo 'inside the function, $var = '.$var.'<br />';
}

fn();
echo 'outside the function, $var = '.$var.'<br />';
```

En el siguiente ejemplo, la variable `$var` se ha definido como global lo que significa que tras llamar a la función, la variable también saldrá fuera de la función. El resultado de esta secuencia de comandos será el siguiente:

```
inside the function, $var = contents
outside the function, $var = contents
```

El ámbito de la variable abarca desde el punto en el que se ejecuta la línea `global $var;`. Podríamos haber declarado la función por encima o debajo de la línea de llamada. (Como observará, existe bastante diferencia entre el ámbito de función y el ámbito de variable.) La ubicación de la declaración de función no tiene relevancia. Lo importante es el lugar en el que se llame a la función y, por lo tanto, desde el que se ejecuta el código incluido en su interior.

También puede utilizar la palabra clave `global` en la parte superior de una secuencia de comandos al utilizar por primera vez una variable para declarar que su ámbito abarca la secuencia de comandos. Éste es probablemente el uso más común de la palabra clave `global`.

Como puede ver por los ejemplos anteriores, resulta perfectamente válido volver a utilizar un nombre de variable para una variable dentro y fuera de una función sin que se produzcan interferencias entre ambas. Sin embargo, no conviene utilizar el mismo nombre porque sin un examen atento del código y del ámbito podría pensarse que se trata de las mismas variables.

Llamadas por referencia frente a llamadas por valor

Si desea escribir una función llamada `increment()` que nos permita incrementar un valor, podríamos vernos tentados a escribir lo siguiente:

```
function increment($value, $amount = 1)
{
   $value = $value +$amount;
}
```

Este código no sirve de nada. El resultado del siguiente código de prueba será `"10"`.

```
$value = 10;
increment ($value);
echo $value;
```

Los contenidos de `$value` no han cambiado.

Esto se debe a las reglas de ámbito. Este código crea una variable llamada `$value` que contiene 10. A continuación llama a la función `increment()`. Se crea la variable `$value` de la función al llamarla. Se le agrega una unidad, por lo que el valor de `$value` pasa a 11 dentro de la función hasta que llega a su fin y volvemos al código

que la llamó. En este código, la variable `$value` es diferente porque tiene ámbito global y no varía.

Una forma de resolver este problema consiste en declarar `$value` en la función como global, pero esto implica que para poder utilizarla, la variable que queremos incrementar deberá llamarse `$value`. Una solución mejor sería utilizar las llamadas por referencia.

La forma normal de llamar a los parámetros de una función es pasándolos por valor. Al pasar un parámetro, se creará una nueva variable que contiene el valor pasado. Se trata de una copia del original. Puede modificar este valor como desee, pero el valor original de la variable fuera de la función seguirá siendo el mismo.

Una solución mejor es utilizar las llamadas por referencia. En este caso, cuando se pasa un parámetro a una función, en lugar de crear una nueva variable, la función recibe una referencia a la variable original. Esta referencia tiene un nombre de variable, que comienza por el símbolo del dólar y se puede utilizar de la misma forma que cualquier otra variable. La diferencia está en que en lugar de tener un valor propio, simplemente hace referencia al original. Cualquier cambio realizado a la referencia sólo afectará al original.

Para indicar que un parámetro debe pasarse por referencia se coloca un símbolo `&` delante del nombre del parámetro en la definición de función. No es necesario ningún cambio en la llamada de función.

El ejemplo de función anterior, `increment()`, se puede modificar para pasar un parámetro por referencia y lograr que funcione correctamente.

```
function increment(&$value, $amount = 1)
{
   $value = $value +$amount;
}
```

Ahora ya disponemos de una función operativa y podemos utilizar el nombre que deseemos para incrementar el valor que nos plazca. Como se indicó anteriormente, resulta confuso utilizar el mismo nombre fuera y dentro de una función, por lo que aplicaremos un nuevo nombre a la variable en la secuencia de comandos principal. El siguiente código de prueba imprimirá 10 antes de llamar a `increment()` y 11 después.

```
$a = 10;
echo $a.'<br />';
increment ($a);
echo $a.'<br />';
```

Final de las funciones

La palabra clave `return` detiene la ejecución de una función. Cuando una función termina porque todas sus instrucciones se han ejecutado o porque se ha utilizado la palabra clave `return`, la ejecución regresa a la llamada de la función.

Si llama a la siguiente función, sólo se ejecutará la primera instrucción `echo`.

```
function test_return()
{
   echo 'This statement will be executed';
   return;
   echo 'This statement will never be executed';
}
```

Obviamente, no se trata de una forma muy útil de utilizar la instrucción `return`. Por regla general, sólo querremos volver desde el centro de una función en respuesta a una condición.

Por ejemplo, una razón habitual para utilizar una instrucción `return` es una condición de error que detenga una función al final. Por ejemplo, si escribimos una función para determinar el número más grande de dos especificados, podemos salir de la función si falta alguno de los números.

```
function larger( $x, $y )
{
   if (!isset($x)||!isset($y))
   {
      echo 'This function requires two numbers';
      return;
   }
   if ($x>=$y)
      echo $x;
   else
      echo $y;
   echo '<br />';
}
```

La función incorporada `isset()` indica si se ha creado una variable y se le ha asignado un valor. En este código, devolvemos un mensaje de error y salimos de la función si uno de los parámetros no lleva definido un valor. Para probarla, utilizamos `!isset()`, que significa "NO isset()", de manera que la instrucción `if` pueda leerse como "si x no está establecida o si y no está establecida". La función regresará si cualquiera de las dos condiciones resulta ser cierta.Si se ejecuta la instrucción `return`, se ignorarán las siguientes líneas de la función. La ejecución del programa regresará al punto en el que se invocó la función. Si se establecen ambos parámetros, la función imprimirá el más grande de los dos. El resultado del siguiente código:

```
$a = 1;
$b = 2.5;
$c = 1.9;
larger($a, $b);
larger($c, $a);
larger($d, $a);
```

será

```
2.5
1.9
This function requires two numbers
```

Cómo devolver valores desde funciones

La salida de una función no es la única razón para utilizar la instrucción `return`. Muchas funciones utilizan instrucciones `return` para comunicarse con el código que las llamó. En lugar de imprimir los resultados de la comparación de la función `larger()`, puede que resulte más útil devolver la respuesta. De esta forma, el código que llame a la función puede decidir si mostrar o utilizarla y cómo. La función incorporada `max()` es equivalente y se comporta de esta forma.

Podemos escribir la función `larger()` así:

```
function larger ($x, $y)
{
   if (!isset($x)||!isset($y))
      return false;
   else if ($x>=$y)
      return $x;
   else
      return $y;
}
```

En este código devolvemos el valor más grande de los pasados. Obviamente, el valor devuelto será diferente en caso de error. Si uno de los números desaparece, devolveremos `false`. El único pero de este enfoque es que los programadores que llamen a esta función deben probar el tipo de devolución con === para asegurarse de que `false` no se confunde con 0.

La función `max()` de PHP no devuelve nada si se establecen ambas variables y si sólo se establece una, será la que devolverá.

El siguiente código:

```
$a = 1; $b = 2.5; $c = 1.9;
echo larger($a, $b)."<br />";
echo larger($c, $a)."<br />";
echo larger($d, $a)."<br />";
```

generará el siguiente resultado porque `$d` no existe y `false` no está visible:

```
2.5
1.9
```

Las funciones que realizan alguna tarea, pero que no necesitan devolver un valor, a menudo devuelven `true` o `false` para indicar si se realizan satisfactoriamente o fallan. Los valores booleanos `true` y `false` se pueden representar con valores 1 y 0, respectivamente, aunque son de tipos diferente.

Bloques de código

Para declarar que un grupo de instrucciones forma un bloque las colocamos entre llaves. Esta estructura no tendrá efectos sobre el funcionamiento del código pero sí

implicaciones específicas como la forma en la que se ejecutan las estructuras de control, como los bucles y las estructuras condicionales.

A continuación se incluyen dos ejemplos que funcionan de forma muy diferente:

Ejemplo sin bloque de código

```
for($i = 0; $i < 3; $i++ )
   echo 'Line 1<br />';
echo 'Line 2<br />';
```

Ejemplo con bloque de código

```
for($i = 0; $i < 3; $i++ )
{
   echo 'Line 1<br />';
   echo 'Line 2<br />';
}
```

En ambos ejemplos, el bucle `for` se itera tres veces. En el primer ejemplo, el bucle `for` sólo ejecuta la línea situada inmediatamente por debajo. El resultado de este ejemplo es el siguiente:

```
Line 1
Line 1
Line 1
Line 2
```

El segundo ejemplo utiliza un bloque de código para agrupar las dos líneas. Esto implica que el bucle `for` ejecuta ambas líneas tres veces.

```
Line 1
Line 2
Line 1
Line 2
Line 1
Line 2
```

Como el código de estos ejemplos está sangrado correctamente, es probable que vea la diferencia enseguida. El sangrado del código tiene el objetivo de ayudar a los lectores a interpretar visualmente las líneas afectadas por el bucle `for`. Sin embargo, fíjese en que los espacios no afectan a la forma en la que PHP procesa el código.

En algunos lenguajes, los bloques de código afectan al ámbito de las variables. Éste no es el caso de PHP.

Recursión

PHP admite las funciones de recursión. Las funciones de recursión se llaman a sí mismas. Estas funcione resultan especialmente útiles para navegar por estructuras de datos dinámicas como listas y árboles vinculados. Sin embargo, son pocas las aplicaciones basadas en Web que requieran una estructura de datos de esta comple-

jidad por lo que el uso de esta función es limitado. La recursión se puede utilizar en lugar de la iteración en muchos casos porque ambas permiten realizar una operación de manera repetida. Las funciones recursivas resultan más lentas y utilizan más memoria que la iteración, razón por la cual debería dar prioridad a estás últimas.

En el listado 5.5 se recoge un breve ejemplo de función recursiva.

Listado 5.5. recursion.php. Resulta sencillo invertir una cadena utilizando la recursión También se muestra la versión iterativa.

```
function  reverse_r($str)
{
   if (strlen($str)>0)
      reverse_r(substr($str, 1));
   echo substr($str, 0, 1);
   return;
}

function reverse_i($str)
{
   for ($i=1; $i<=strlen($str); $i++)
   {
      echo substr($str, -$i, 1);
   }
   return;
}
```

En este listado, hemos implementado dos funciones. Ambas imprimen una cadena al revés. La función `reverse_r()` es recursiva y la función `reverse_i()` es iterativa.

La función `reverse_r()` toma una cadena como parámetro. Al llamarla, se llamará a sí misma y pasará el penúltimo carácter de la cadena. Por ejemplo, si llama a

```
reverse_r('Hello');
```

se llamará a sí misma varias veces con los siguientes parámetros:

```
reverse_r('ello');
reverse_r('llo');
reverse_r('lo');
reverse_r('o');
reverse_r('');
```

Cada llamada que realiza a sí misma crea una nueva copia del código de función en la memoria del servidor pero con un parámetro diferente. Es como si pretendiéramos llamar a una función diferente cada vez. Esto evita que las instancias de la función se confundan. En cada llamada, se prueba la longitud de la cadena pasada. Al llegar al final de la cadena (`strlen()=0`) falla la condición. La instancia más reciente de la función (`reverse_r('')`) seguirá adelante y ejecutará la siguiente línea de código que debe imprimir el primer carácter pasado de la cadena (en este caso, no hay ningún carácter porque la cadena está vacía).

A continuación, esta instancia de la función devuelve el control a la instancia que la llamó, en concreto `reverse_r('o')`.

Esta instrucción imprime el primer carácter en su cadena (`"o"`) y devuelve el control a la instancia que lo llamó.

El proceso continúa (se imprime un carácter y se vuelve a la instancia de la función superior en el orden de llamada) hasta que el control vuelve al programa principal.

Las soluciones recursivas presentan un aspecto muy elegante y matemático. En la mayor parte de los casos, sin embargo, resulta mucho mejor utilizar una solución iterativa. El listado 5.5 también incluye la variante iterativa. Como observará, es igual de larga (aunque no siempre es así) y realiza la misma función.

La diferencia principal es que la función recursiva realiza copias de sí misma en memoria, lo que implica la sobrecarga de varias llamadas de función.

La versión recursiva se puede utilizar cuando el código resulta más breve y elegante que el de la versión iterativa, cosa que en este dominio de aplicaciones no suele ocurrir a menudo.

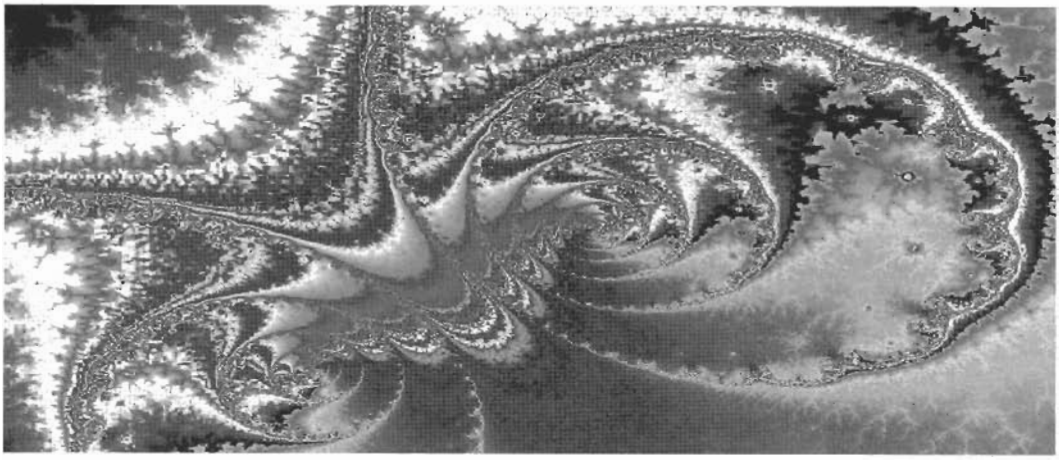
Aunque la recursión parece más elegante, los programadores suelen olvidarse de incluir una condición de finalización, lo cual implica que la función se repetirá hasta que el servidor se quede sin memoria o hasta que se exceda el tiempo de ejecución.

Lecturas adicionales

El uso de las funciones `include()`, `require()`, `function` y `return` también se explica en el manual en línea. Si desea saber más sobre conceptos como el de la recursión, las llamadas por valor o por referencia y el ámbito de las variables que afecta a muchos lenguajes, consulte un libro de texto general sobre informática.

A continuación

Ahora que ya sabemos utilizar los archivos de inclusión, los archivos requeridos y las funciones para lograr que nuestro código resulte más sencillo de mantener y susceptible de utilizar de manera repetida, en el siguiente capítulo se analizará el software orientado a objetos y la compatibilidad que ofrece PHP al respecto. El uso de objetos permite lograr objetivos similares a los conceptos presentados en este capítulo pero con ventajas superiores para proyectos complejos.



6
PHP orientado a objetos

En este capítulo se explican conceptos relacionados con el desarrollo orientado a objetos y se muestra cómo implementarlos en PHP.

Entre los temas clave que trataremos en este capítulo se incluyen los siguientes:

- Conceptos orientados a objetos
- Creación de clases, atributos y operaciones
- Uso de atributos de clase
- Llamadas a operaciones de clase
- Herencia
- Llamadas a métodos de clases
- Diseño de clases
- Desarrollo de código para clases

Conceptos orientados a objetos

Los lenguajes de programación modernos suelen admitir o incluso necesitan adoptar un enfoque orientado a objetos para el desarrollo de software. El desarrollo orientado a objetos (OO) intenta utilizar las clasificaciones, las relaciones y las propiedades de los objetos del sistema para contribuir al desarrollo de programas.

Clases y objetos

En el contexto de software orientado a objetos, un objeto puede ser prácticamente cualquier elemento o concepto (un objeto físico como una mesa o un cliente; o un objeto conceptual que sólo existe en el software, como un campo de entrada de texto o un archivo). Por regla general, estaremos más interesados en objetos conceptuales, incluidos objetos del mundo real que necesitan representarse en software.

El software orientado a objetos se diseña y construye como un conjunto de objetos independientes con atributos y operaciones que interactúan para cubrir nuestras necesidades. Los atributos son propiedades o variables relacionadas con el objeto. Las operaciones son métodos, acciones o funciones que el objeto realiza para modificarse a sí mismo o para obtener algún efecto externo.

La ventaja principal del software orientado a objetos es su capacidad para admitir o fomentar la encapsulación, también conocida como ocultación de datos. Básicamente, el acceso a los datos dentro de un objeto sólo está disponible a través de operaciones con objetos, conocida como la interfaz del objeto.

La funcionalidad de un objeto está vinculada a los datos que utiliza. Podemos alterar los detalles de la implementación del objeto para mejorar el rendimiento, agregar nuevas funciones o resolver errores sin tener que modificar la interfaz, cuyos efectos pueden extenderse a todo el proyecto.

En otras áreas del desarrollo de software, la programación orientada a objetos es la norma y el software orientado a funciones se considera desfasado. Por varias razones, la mayor parte de las secuencias de comandos orientadas a objetos siguen, por desgracia, diseñadas y escritas utilizando una metodología orientada a funciones.

Existen varias razones que explican este enfoque. La mayoría de los proyectos Web son relativamente pequeños y sencillos. En la mayor parte de los casos, los proyectos Web se pueden desarrollar satisfactoriamente sin planificar el enfoque debido a su tamaño. Sin embargo, si intenta aplicar la misma técnica para construir algo de mayor tamaño, la calidad de los resultados se verá afectada, si es que se obtiene algún resultado.

Muchos proyectos evolucionan a partir de un conjunto de páginas vinculadas para convertirse en una aplicación compleja. Estas aplicaciones, ya se presenten a través de cuadros de diálogo y ventanas o páginas HTML generadas dinámicamente, necesitan disponer de una metodología de desarrollo correctamente ideada. La orientación a objetos puede ayudarle a gestionar la complejidad de los proyectos, a incrementar la reusabilidad del código y, por tanto, a reducir los costes de mantenimiento.

En el software orientado a objetos, un objeto es una colección única e identificable de datos y operaciones almacenados que operan sobre dichos datos. Por ejemplo, podemos tener dos objetos que representen botones. Aunque ambos tengan el texto "Aceptar", una anchura de 60 píxeles, una altura de 20 píxeles y otros atributos idénticos, necesitamos disponer de la posibilidad de trabajar con uno o con el otro. En software, existen variables independientes que actúan como identificadores únicos para dichos objetos.

Los objetos se pueden agrupar en clases. Las clases representan un conjunto de objetos que puede variar, pero que deben tener un conjunto de características en común. Una clase contiene objetos con operaciones que se comportan de la misma forma y atributos que representan las mismas cosas, aunque los valores de dichos atributos varíen de objeto a objeto.

Podemos imaginar el término bicicleta como una clase de objetos que describe muchas bicicletas diferentes con una gran cantidad de características o atributos en común; por ejemplo, que tengan dos ruedas, un color y un tamaño, y realicen operaciones, como moverse.

Mi bicicleta es un ejemplo de objeto que encaja en la clase bicicleta Compartirá rasgos comunes a todas las bicicletas, incluida una operación de movimiento que se comporta de la misma forma en la mayoría de las bicicletas (aunque apenas la utilice). Mi bicicleta tiene atributos con valores únicos porque es verde y no todas las bicicletas tienen ese color.

Polimorfismo

Un lenguaje de programación orientado a objetos debe admitir el polimorfismo, lo que significa que las clases pueden tener diferentes comportamientos con respecto a la misma operación. Por ejemplo, si tenemos una clase coche y una clase bicicleta, sus operaciones de desplazamiento serán diferentes. Para los objetos del mundo real, este hecho no plantea un problema ya que es poco probable que se confunda el movimiento de una bicicleta con el de un coche. Sin embargo, un lenguaje de programación no posee el sentido común del mundo real, por lo que el lenguaje debe admitir el polimorfismo para determinar qué operación de movimiento utilizar con un objeto dado.

El polimorfismo es más una característica relacionada con los comportamientos que con los objetos. En PHP, sólo las funciones de miembro de una clase puede ser polimórficas. Una comparación del mundo real es la de los verbos en los lenguajes naturales, que equivalen a las funciones de miembro. Considere las formas en las que una bicicleta se puede utilizar en el mundo real. Se puede limpiar, mover, desmontar, reparar o pintar, entre otras cosas.

Estos verbos describen acciones genéricas porque no se puede saber el tipo de objeto sobre el que se actúa. (Este tipo de abstracción sobre objetos y acciones es una de las características que distinguen a la inteligencia humana.)

Por ejemplo, la operación de mover una bicicleta requiere acciones completamente diferentes a las necesarias para mover un coche, aunque los conceptos sean similares. El verbo mover se puede asociar a un conjunto concreto de acciones una vez que se dé a conocer el objeto sobre el que se actúa.

Herencia

La herencia nos permite crear una relación jerárquica entre clases utilizando subclases. Una subclase hereda atributos y operaciones de su superclase. Por ejem-

plo, un coche y una bicicleta tienen elementos en común. Podemos utilizar una clase vehículo para incluir elementos como un atributo de color y una operación de movimiento común a todos los vehículos y permitir que nuestras clases coche y bicicleta las hereden.

Gracias a la herencia, podemos desarrollar y seguir agregando elementos a clases existentes. A partir de una clase básica se pueden derivar clases más complejas y especializadas a medida que surja la necesidad. Como resultado, nuestro código será más reutilizable, lo cual es una de las ventajas más importantes de un enfoque orientado a objetos.

El uso de la herencia puede ahorrarnos trabajo si las operaciones se escriben una vez en una superclase en lugar de varias veces en subclases diferentes. También puede ayudarlos a modelar de manera más exacta relaciones del mundo real. Si se establece una relación lógica del tipo "es un/una" en una frase entre dos clases, es probable que la herencia resulte adecuada. La frase "un coche es un vehículo" tiene sentido, pero la frase "un vehículo es un coche" no lo tiene porque no todos los vehículos son coches. Por lo tanto, el coche puede derivarse desde la clase vehículo

Creación de clases, atributos y operaciones en PHP

Por el momento, las clases se han estudiado de forma bastante abstracta. Al crear una clase en PHP, se utiliza la palabra clave `class`.

Estructura de una clase

La definición básica de una clase presenta este aspecto:

```
class  nombreclase
{
}
```

Para que la clase resulte útil necesita tener atributos y operaciones. Para crear atributos se declaran variables dentro de una definición de clase utilizando la palabra clave `var`. El siguiente código crea una clase llamada `classname`, con dos atributos `$attribute1` y `$attribute2`.

```
class classname
{
   var $attribute1;
   var  $attribute2;
}
```

Las operaciones se crean declarando funciones dentro de la definición de clase. El siguiente código crea una clase denominada `classname` con dos operaciones que no hacen nada.

La operación `operation1()` no toma parámetros y la operación `operation2()` toma dos parámetros.

```
class  classname
{
   function operation1()
   {
   }
   function operation2($param1, $param2)
   {
   }
}
```

Constructores

La mayor parte de las clases constan de un tipo especial de operación llamado constructor. Un constructor se llama al crear un objeto y suele realizar tareas útiles de inicialización como establecer atributos con valores de inicio lógicos o crear otros objetos necesarios.

Un constructor se declara de la misma forma que otras operaciones, pero tiene el mismo nombre que la clase. Aunque podemos llamar manualmente al constructor, su función principal es hacerlo automáticamente al crear un objeto. El siguiente código declara una clase con un constructor:

```
class classname
{
   function classname($param)
   {
      echo "Constructor called with parameter $param <br />";
   }
}
```

Recuerde que PHP no admite de manera nativa la sobrecarga de funciones, lo que significa que sólo se puede proporcionar una función con un nombre dado, incluido el constructor. (Muchos lenguajes de programación orientados a objetos admiten este recurso.)

Existe una extensión experimental que permite la sobrecarga de métodos get y set, pero no es muy útil.

Instanciación

Tras declarar una clase, tenemos que crear un objeto (un elemento concreto que forme parte de una clase) con el que trabajar. Esta tarea también se conoce como crear una instancia e instanciar una clase.

Para crear un objeto se utiliza la palabra clave `new`. Es necesario especificar a qué clase pertenecerá el objeto y suministrar todos los parámetros que requiera el constructor.

El siguiente código declara una clase llamada `classname` con un constructor y crea tres objetos del tipo `classname`:

```
class  classname
{
   function  classname($param)
   {
      echo "Constructor called with parameter $param <br />";
   }
}

$a = new classname('First');
$b = new classname('Second');
$c = new classname();
```

Como el constructor se llama cada vez que creamos un objeto, este código genera el siguiente resultado:

```
Constructor called with parameter First
Constructor called with parameter Second
Constructor called with parameter
```

Uso de los atributos de clase

Dentro de una clase, disponemos de acceso a una variable especial llamada `$this`. Si un atributo de la clase actual se denominara `$attribute`, se utilizaría la secuencia `$this->attribute` para hacer referencia a él al establecer o acceder a la variable desde una operación dentro de una clase.

El siguiente código muestra cómo establecer o acceder a un atributo dentro de una clase:

```
class classname
{
   var $attribute;
   function operation($param)
   {
      $this->attribute = $param
      echo $this->attribute;
   }
}
```

Algunos lenguajes de programación nos permiten limitar el acceso a atributos declarando dichos datos como privados o protegidos. PHP no admite esta función, por lo que los atributos y operaciones que utilicemos resultarán visibles fuera de la clase (es decir, serán todos públicos).

Podemos realizar la misma tarea como se demostró anteriormente desde fuera de la clase, utilizando sintaxis ligeramente diferente.

```
class  classname
{
```

```
    var  $attribute;
}
$a = new classname();
$a->attribute = 'value';
echo $a->attribute;
```

No es buena idea acceder directamente a los atributos desde fuera de una clase. Una de las ventajas de un enfoque orientado a objetos es que fomenta la encapsulación. Aunque en PHP no se puede forzar la ocultación de datos, con un poco de voluntad se pueden extraer ciertas ventajas.

Si en lugar de acceder a los atributos de una clase directamente, se escriben funciones de descriptores de acceso, todos los accesos se realizarán a través de una única sección de código. Las funciones de descriptores de acceso pueden presentar el siguiente aspecto al principio:

```
class classname
{
    var $attribute;
    function get_attribute()
    {
        return $this->attribute;
    }
    function set_attribute($new_value)
    {
        $this->attribute = $new_value;
    }
}
```

Este código proporciona funciones para acceder al atributo denominado `$attribute`. Tenemos una función llamada `get_attribute()` que devuelve simplemente el valor de `$attribute` y una función llamada `set_attribute()` que asigna un valor nuevo a `$attribute`.

A primera vista, podría parecer que este código no agrega mucho valor. En su forma actual, es probablemente cierto, pero la razón para suministrar las funciones de descripción de acceso es sencilla: sólo tendremos una sección de código que accede a dicho atributo.

Con un solo punto de acceso, podemos implementar elementos de comprobación para asegurarnos de que sólo se almacenan datos pertinentes. Si en un momento posterior necesitamos que el valor de `$attribute` esté entre cero y cien, bastará con agregar unas pocas líneas de código y verificar antes de permitir los cambios. Podemos modificar la función `set_attribute()` para que presente esta aspecto:

```
function set_attribute($new_value)
{
    if( $new_value >= 0 && $newvalue <= 100 )
        $this->attribute  =  $new_value;
}
```

Este cambio resulta sencillo, pero si no utilizamos la función de descriptor de acceso, tendremos que recorrer cada línea de código y modificar cada acceso a `$attribute`, lo cual resulta pesado y proclive a errores.

Con un solo punto de acceso, podemos cambiar libremente la implementación subyacente. Si por alguna razón, optamos por cambiar la forma en la que se almacena `$attribute`, las funciones de descriptor de acceso nos permitirán hacerlo variando el código en un solo punto.

Puede que en lugar de almacenar `$attribute` como variable queremos recuperarla únicamente desde la base de datos cuando resulte necesario, calcular un valor actualizado cada vez que se solicite, inferir un valor de los valores de otros atributos o codificar los datos con un tipo de dato más pequeño. Sea cual sea el cambio que decidamos realizar, bastará con modificar nuestras funciones de descriptor de acceso.

El resto de las secciones de código no se verán afectadas mientras las funciones de descriptor de acceso sigan aceptando o devolviendo los datos que esperan otras partes del programa.

Llamadas a operaciones de clase

Podemos llamar a operaciones de clases de la misma forma que llamamos a los atributos de clase. Si tenemos la siguiente clase:

```
class classname
{
    function operation1()
    {
    }
    function operation2($param1, $param2)
    {
    }
}
```

y creamos un objeto del tipo `classname` llamado `$a` de la siguiente forma:

```
$a = new classname();
```

Seguidamente llamamos a las operaciones de la misma forma que a otras funciones: por su nombre y con todos los parámetros que se necesiten entre paréntesis.

Como estas operaciones pertenecen a un objeto en lugar de a funciones normales, tenemos que especificar a qué objeto pertenecen. El nombre del objeto se utiliza de la misma forma que los atributos de objeto:

```
$a->operation1();
$a->operation2(12, 'test');
```

Si nuestras operaciones devuelven algo, podemos capturar los datos devueltos de la siguiente forma:

```
$x  =  $a->operation1();
$y  =  $a->operation2(12,  'test');
```

Implementación de la herencia en PHP

Si nuestra clase es una subclase de otra, podemos utilizar la palabra clave `extends` para especificarlo.

El siguiente código crea una clase llamada `B` que se deriva de la clase llamada `A` previamente definida.

```
class B extends A
{
   var $attribute2;
   function operation2()
   {
   }
}
```

Si la clase `A` se declara de la siguiente forma:

```
class A
{
   var $attribute1;
   function operation1()
   {
   }
}
```

todos los accesos siguientes a las operaciones y atributos de un objeto de tipo `B` serían válidos:

```
$b = new B();
$b->operation1();
$b->attribute1 = 10;
$b->operation2();
$b->attribute2 = 10;
```

Fíjese en que como la clase `B` extiende la clase `A`, podemos hacer referencia a `operation1()` y `$attribute1`, aunque se declararon en la clase `A`. Como subclase de `A`, `B` tiene la misma funcionalidad y los mismos datos. Además, `B` ha declarado un atributo y una operación propia.

Es importante observar que la herencia sólo funciona en una dirección.

La subclase hereda las funciones de la clase principal o superclase, pero ésta no toma ninguna función de aquélla. Por lo tanto, las dos siguientes líneas de código serían erróneas:

```
$a = new A();
$a->operation1();
$a->attribute1 = 10;
$a->operation2();
$a->attribute2 = 10;
```

La clase `A` no tiene una operación `operation2()` ni un atributo `attribute2`.

Reemplazamientos

Hemos mostrado una subclase que declara nuevos atributos y operaciones. También resulta válido, y a veces útil, volver a declarar los mismos atributos y operaciones. Podemos hacerlo para asignar un atributo de la subclase, un valor diferente para el mismo atributo de su superclase o aplicar a una operación de la subclase funcionalidad diferente con respecto a la misma operación de su superclase. Este recurso se denomina reemplazamiento.

Por ejemplo, si tenemos una clase `A`:

```
class A
{
   var $attribute = 'default value';
   function operation()
   {
      echo 'Something<br />';
      echo "The value of \$attribute is $this->attribute<br />";
   }
}
```

y queremos alterar el valor predeterminado de `$attribute` y suministrar nueva funcionalidad para `operation()`, podemos crear la siguiente clase `B`, que reemplaza `$attribute` y `operation()`:

```
class B extends A
{
   var $attribute = 'different value';
   function operation()
   {
      echo 'Something else<br />';
      echo "The value of \$attribute is $this->attribute<br />";
   }
}
```

La declaración de `B` no afecta a la definición original de `A`. Considere las siguientes dos líneas de código:

```
$a = new A();
$a -> operation();
```

Hemos creado un objeto de tipo `A` y hemos llamado a la función `operation()`. El resultado será:

```
Something
The value of $attribute is default value
```

si la creación de `B` no ha alterado a `A`. Si creamos un objeto de tipo `B`, obtendremos un resultado diferente.

El siguiente código:

```
$b = new B();
$b -> operation();
```

generará

```
Something else
The value of $attribute is different value
```

De la misma forma que el suministro de nuevos atributos y operaciones dentro de una subclase no afecta a la superclase, el reemplazamiento de atributos u operaciones dentro de una subclase no afecta a la superclase.

Una subclase heredará todos los atributos y operaciones de su superclase, a menos que se suministren reemplazamientos. Si suministra una definición de reemplazamiento, ésta tendrá preferencia y reemplazará a la definición original.

A diferencia de otros leguajes orientados a objetos, PHP no permite reemplazar una función y a la vez llamar a la versión definida en la clase superior.

La herencia puede tener varias capas de profundidad. Por ejemplo, podemos declarar una clase `C`, que extienda `B` y que por tanto herede las funciones desde `B` y desde la clase superior `A`. La clase `C` puede a su vez seleccionar atributos y operaciones de sus clases principales para anularlas y reemplazarlas.

Herencia múltiple

Algunos lenguajes orientados a objetos admiten la herencia múltiple pero no así PHP. Esto significa que cada clase sólo puede heredarse desde una clase superior. No existen restricciones en cuanto a la cantidad de clases secundarias que puede compartir una clase principal.

Puede que este concepto no resulte claro al principio. La figura 6.1 muestra tres formas diferentes de derivar las clases `A`, `B` y `C`.

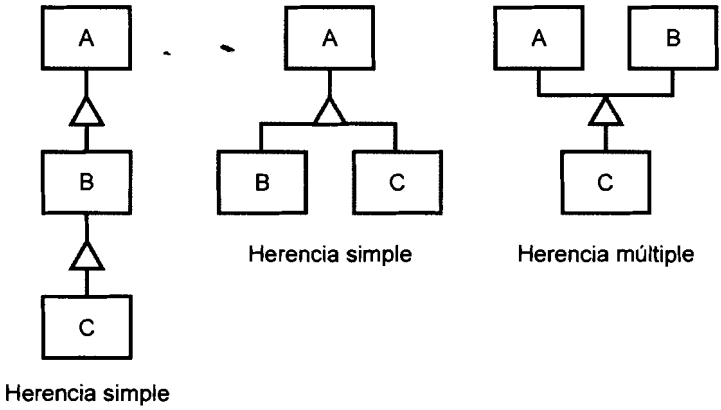


Figura 6.1. PHP no admite la herencia múltiple

La combinación de la izquierda muestra la clase `C` que se hereda de la clase `B`, que a su vez se hereda de la clase `A`. Cada clase tiene un elemento superior al menos, por lo que se trata de una herencia simple perfectamente factible en PHP.

En la combinación central, la clase B y la clase C se heredan de la clase A. Cada clase tiene una clase superior como máximo, por lo que de nuevo estamos ante una herencia permitida en PHP.

La combinación de la derecha muestra la clase C que se deriva de la clase A y de la clase B. En este caso, la clase C consta de dos clases principales lo que la convierte en una clase no válida en PHP.

Diseño de clases

Ahora que ya conocemos algunos de los conceptos en los que se basan los objetos y las clases, y la sintaxis para implementarlos en PHP, ha llegado el momento de examinar cómo diseñar clases útiles.

Muchas clases de nuestro código representan clases o categorías de objetos del mundo real. Las clases que se utilizan en el desarrollo Web pueden incluir páginas, componentes de interfaces de usuario, carros de la compra, resolución de errores, categorías de productos y clientes.

Los objetos de su código también pueden representar instancias específicas de clases mencionadas previamente, por ejemplo, la página principal, un botón dado o el carro de la compra utilizado por Fred Smith en un momento dado. El propio Fred Smith puede representarse mediante un objeto de tipo cliente. Cada artículo comprado por Fred se puede representar como un objeto que pertenezca a una categoría o clase.

En el capítulo anterior, utilizamos sencillos archivos de inclusión para proporcionar a nuestra compañía ficticia, TLA Consulting, un aspecto visual y operativo uniforme en las diferentes páginas del sitio Web. El uso de las clases y la herencia permite crear versiones más avanzadas del mismo sitio.

Queremos disponer de la posibilidad de crear páginas rápidamente para el sitio TLA que se comporten de igual forma y presenten un aspecto semejante. Estas páginas se podrán modificar para su adaptación a diferentes partes del sitio.

Vamos a crear una clase `Page`. El objetivo global de esta clase es limitar la cantidad de código HTML necesario para crear una nueva página. Nos permitirá alterar las partes que cambian de una página a otra además de generar automáticamente los elementos para que no varíen. Las clases ofrecen un marco flexible para crear nuevas páginas y no deberían comprometer nuestra libertad de acción.

Como estamos generando la página desde una secuencia de comandos en lugar de código HTML estático, podemos agregar cualquier elemento que resulte inteligente, incluyendo funcionalidad para:

- Permitirnos alterar elementos de página en un lugar. Si cambiamos la advertencia sobre los derechos de autor o agregamos un botón adicional, sólo necesitaremos realizar el cambio en un punto.
- Disponer de contenido predeterminado para la mayor parte de las páginas, pero disponer de la capacidad de modificar cada elemento donde resulte

necesario y establecer valores personalizados para elementos como el título y las etiquetas meta.

- Reconocer qué página se está visualizando y alterar los elementos de navegación para adaptarlos (no tiene mucho sentido tener un botón vinculado a la página principal dentro de la página principal).
- Permitirnos sustituir elementos estándar para páginas particulares. Por ejemplo, si queremos utilizar diferentes botones de navegación en las secciones del sitio, deberíamos disponer de la posibilidad de sustituir los estándar.

Escritura del código para nuestra clase

Tras decidir el aspecto que queremos que devuelva nuestro código y la aplicación de una serie de funciones, ¿cómo lo implementamos?

En una parte posterior del libro abordaremos el diseño y la gestión de proyectos de gran tamaño. Por el momento, vamos a concentrarnos en las partes específicas relacionadas con la escritura de PHP orientado a objetos.

Nuestra clase necesitará un nombre lógico. Como representa a una página, se denominará `Page`. Para declarar una clase llamada `Page`, escribimos

```
class Page
{
}
```

Esta clase necesita algunos atributos. Estableceremos los elementos que es probable que queramos modificar de una página a otra, como los atributos de nuestra clase. El contenido principal de la página, que será una combinación de etiquetas HMTL y texto, se denominará `$content`. Podemos declarar el contenido con la siguiente línea de código dentro de la función de clase.

```
var  $content;
```

También podemos establecer los atributos para almacenar el título de la página. Es probable que queramos modificar este elemento para indicar con claridad la página que está visualizando nuestro visitante. En lugar de incluir títulos en blanco, suministraremos un título predeterminado con la siguiente declaración:

```
var $title = 'TLA Consulting Pty Ltd';
```

La mayor parte de las páginas Web comerciales incluyen etiquetas meta para ayudar a los motores de búsqueda a indexarlas. Para que resulten útiles, las etiquetas meta deberían cambiar de una página a otra. De nuevo, suministramos un valor predeterminado:

```
var $keywords = 'TLA Consulting, Three Letter Abbreviation,
     some of my best friends are search engines';
```

Los botones de navegación que se muestran en la página original de la figura 5.2 (véase capítulo anterior) deberían mantenerse entre las diferentes páginas para evitar confundir a los usuarios, pero para poder cambiarlos con facilidad los convertiremos en un atributo. Como el número de botones puede ser variable, utilizaremos una matriz y almacenaremos el texto del botón y del URL al que debería apuntar.

```
var $buttons = array( 'Home'     => 'home.php',
                      'Contact'  => 'contact.php',
                      'Services' => 'services.php',
                      'Site Map' => 'map.php'
                    );
```

Para proporcionar alguna funcionalidad, la clase también necesita operaciones. Podemos comenzar por suministrar funciones de descriptores de acceso para establecer y obtener valores de los atributos definidos. Éstos suelen adoptar la siguiente forma:

```
function SetContent($newcontent)
{
   $this->content = $newcontent;
}
```

Como es poco probable que solicitemos alguno de estos valores desde fuera de la clase, hemos optado por utilizar una colección de funciones Get.

La función principal de esta clase es mostrar una página de código HTML, por lo que necesitaremos una función. Hemos llamado a la función `Display()` y presenta de la siguiente forma:

```
function Display()
   {
      echo "<html>\n<head>\n";
      $this -> DisplayTitle();
      $this -> DisplayKeywords();
      $this -> DisplayStyles();
      echo "</head>\n<body>\n";
      $this -> DisplayHeader();
      $this -> DisplayMenu($this->buttons);
      echo $this->content;
      $this -> DisplayFooter();
      echo "</body>\n</html>\n";
   }
```

Esta función incluye varias instrucciones `echo` sencillas para mostrar código HTML, pero se compone fundamentalmente de llamadas a otras funciones de la clase. Como es probable que haya supuesto por los nombres, dichas funciones muestran partes de la página. No es obligatorio dividir las funciones de esta forma. Todas ellas podrían combinarse en una función de mayor tamaño. En nuestro caso, las hemos dividido por varias razones.

Cada función debería realizar una tarea definida. Cuanto más sencilla sea, más fácil resultará escribirla y probarla. Tampoco conviene excederse: si divide su programa en unidades demasiado pequeñas puede resultar difícil de leer.

El uso de la herencia permite reemplazar operaciones. Podemos reemplazar la función `Display()`, pero es poco probable que queramos cambiar la forma en la que se muestra toda la página. Resultará mucho mejor dividir la funcionalidad de visualización en unas pocas tareas independientes y disponer de la posibilidad de reemplazar sólo las partes que deseemos modificar.

Nuestra función `Display` llama a `DisplayTitle()`, `DisplayKeywords()`, `DisplayStyles()`, `DisplayHeader()`, `DisplayMenu()` y `DisplayFooter()`. Por lo tanto, necesitamos definir estas operaciones. Una de las mejoras de PHP 4 sobre PHP 3 es que podemos escribir operaciones o funciones en orden lógico. Para ello, se llama a la operación o a la función antes que al código de la función. En PHP 3 y muchos otros lenguajes, necesitamos escribir la función u operación antes de poder llamarla. La mayor parte de nuestras operaciones son bastante simples y necesitan mostrar HTML y quizás los contenidos de nuestros atributos.

El listado 6.1 muestra la clase completa, que hemos guardado como `page.inc` para incluir o exigir dentro de otros archivos.

Listado 6.1. page.inc. Nuestra clase proporciona una forma fácil y flexible de crear páginas para el sitio TLA

```
<?php
class Page
{
   // atributos de la clase Page
   var $content;
   var $title = 'TLA Consulting Pty Ltd';
   var $keywords = 'TLA Consulting, Three Letter Abbreviation,
        some of my best friends are search engines';
   var $buttons = array( 'Home'  => 'home.php',
                         'Contact'  => 'contact.php',
                         'Services' => 'services.php',
                         'Site Map' => 'map.php'
                        );

   // operaciones de Page

   function SetContent($newcontent)
   {
      $this->content = $newcontent;
   }

   function SetTitle($newtitle)
   {
      $this->title = $newtitle;
   }

   function SetKeywords($newkeywords)
   {
      $this->keywords = $newkeywords;
   }

   function  SetButtons($newbuttons)
```

```
    {
       $this->buttons  =  $newbuttons;
    }

    function Display()
    {
       echo "<html>\n<head>\n";
       $this -> DisplayTitle();
       $this  ->  DisplayKeywords();
       $this -> DisplayStyles();
       echo "</head>\n<body>\n";
       $this -> DisplayHeader();
       $this -> DisplayMenu($this->buttons);
       echo $this->content;
       $this -> DisplayFooter();
       echo "</body>\n</html>\n";
    }

    function DisplayTitle()
    {
       echo '<title> $this->title </title>';
    }

    function DisplayKeywords()
    {
       echo "<META name=\"keywords\" content=\"$this->keywords\">";
    }

    function DisplayStyles()
    {
?>
    <style>
       h1 {color:white; font-size:24pt; text-align:center;
           font-family:arial,sans-serif}
       .menu {color:white; font-size:12pt; text-align:center;
              font-family:arial,sans-serif; font-weight:bold}
       td {background:black}
       p {color:black; font-size:12pt; text-align:justify;
          font-family:arial,sans-serif}
       p.foot {color:white; font-size:9pt; text-align:center;
               font-family:arial,sans-serif; font-weight:bold}
       a:link,a:visited,a:active {color:white}
    </style>
<?php
    }

    function DisplayHeader()
    {
?>
    <table width="100%" cellpadding ="12" cellspacing ="0" border ="0">
    <tr bgcolor ="black">
       <td align ="left"><img src = "logo.gif"></td>
       <td>
             <h1>TLA Consulting Pty Ltd</h1>
       </td>
       <td  align  ="right"><img  src  =  "logo.gif"></td>
```

```
    </tr>
    </table>
<?php
    }

    function DisplayMenu($buttons)
    {
        echo "<table width='100%' bgcolor='white' cellpadding='4'
                    cellspacing='4'\n";
        echo "   <tr>\n";

        //calcular el tamaño del botón
        $width = 100/count($buttons);

        while (list($name, $url) = each($buttons))
        {
            $this -> DisplayButton($width, $name, $url, !$this->
IsURLCurrentPage($url));
        }
        echo "   </tr>\n";
        echo "</table>\n";
    }

    function IsURLCurrentPage($url)
    {
        if(strpos( $GLOBALS['SCRIPT_NAME'], $url )==false)
        {
            return false;
        }
        else
        {
            return true;
        }
    }

    function DisplayButton($width, $name, $url, $active = true)
    {
        if ($active)
        {
            echo "<td width ='$width%'>
                    <a href ='$url'>
                    <img src ='s-logo.gif' alt ='$name' border ='0'></a>
                    <a href ='$url'><span class='menu'>$name</span></a></td>";
        }
        else
        {
            echo "<td width ='$width%'>
                    <img src ='side-logo.gif'>
                    <span class='menu'>$name</span></td>";
        }
    }

    function DisplayFooter()
    {
```

```
?>
      <table width = "100%" bgcolor ="black" cellpadding ="12" border ="0">
      <tr>
         <td>
            <p class="foot">&copy; TLA Consulting Pty Ltd.</p>
            <p class="foot">Please see our
                  <a  href  =""">legal  information  page</a></p>
         </td>
      </tr>
      </table>
<?php
   }
}
?>
```

Al leer esta clase, fíjese en que `DisplayStyles()`, `DisplayHeader()` y `DisplayFooter()` necesitan mostrar un gran bloque de HTML estático sin ningún procesamiento de PHP.

Por lo tanto, sólo hemos utilizado una etiqueta final de PHP (`?>`), hemos escrito nuestro código de HTML y hemos vuelto a introducir PHP con una etiqueta de apertura de PHP (`<?php`) dentro de las funciones.

En esta clase se definen otras dos operaciones. En concreto, la operación `DisplayButton()` muestra un botón de menú.

Si el botón va a apuntar a la página en la que estamos, mostramos un botón inactivo que presenta un aspecto ligeramente diferente sin vinculación. De esta forma, logramos mantener la uniformidad de la página y suministramos a los visitantes una ubicación visual.

La operación `IsURLCurrentPage()` determina si el URL asociado a un botón apunta la página actual. Se pueden utilizar muchas técnicas para descubrirlo. En este caso, hemos utilizado la función de cadena `strpos()` para determinar si el URL dado está contenido en una de las variables establecidas de servidor. La instrucción `strpos() $GLOBALS['SCRIPT_NAME'], $url )` devolverá un número si la cadena de `$url` se encuentra dentro de la variable global o `SCRIPT_NAME`, en caso contrario.

Para utilizar esta clase de página necesitamos incluir `page.inc` en una secuencia de comandos y llamar a `Display()`.

El código del listado 6.2 creará la página de inicio de TLA Consulting y devolverá un resultado muy similar al que generamos previamente en la figura 5.2.

El código del listado 6.2 realiza las siguientes tareas:

1. Utiliza `require` para incluir contenidos de `page.inc`, que contiene la definición de la clase `Page`.
2. Crea una instancia de la clase `Page`, denominada `$homepage`.
3. Llama a la operación `SetContent()` dentro del objeto `$homepage` y pasa texto y etiquetas HTML para que aparezcan en la página.
4. Llama a la operación `Display()` dentro del objeto `$homepage` para obligar a la página a mostrarse en el navegador del visitante.

Listado 6.2. home.php. Esta página de inicio utiliza la clase Page para realizar la mayor parte del trabajo implicado en la generación de la página

```
<?php
   require ('page.inc');

   $homepage = new Page();

   $homepage -> SetContent('<p>Welcome to the home of TLA Consulting.
              Please take some time to get to know us.</p>
              <p>We specialize in serving your business needs
              and hope to hear from you soon.</p>'
              );
   $homepage -> Display();
?>
```

Como puede observar en este listado, se necesita realizar muy poco trabajo para generar nuevas páginas utilizando la clase `Page`. El uso de esta clase de la forma ilustrada implica que todas nuestras páginas deben ser muy similares.

Si queremos que una sección del sitio utilice una variante de la página estándar, bastará con copiar `page.inc` en un nuevo archivo llamado `page2.inc` y realizar algunos cambios. Como resultado, cada vez que actualicemos o modifiquemos partes de la página `page.inc` necesitaremos acordarnos de realizar los mismos cambios en `page2.inc`. Una opción mejor consiste en utilizar la herencia para crear un nueva clase que derive gran parte de su funcionalidad de `Page` y que reemplace las partes que necesiten ser diferentes.

En el sitio de TLA, queremos que la página de servicios incluya una segunda barra de navegación. La secuencia de comandos que se muestra en el listado 6.3 realiza esta tarea mediante la creación de una nueva clase llamada `ServicesPage` que se deriva de `Page`. Suministramos una nueva matriz llamada `$row2buttons` que contiene los botones y vínculos que queremos en la segunda fila. Como queremos que esta clase se comporte prácticamente de la misma manera, sólo anulamos la parte que queremos modificar: la operación `Display()`.

Listado 6.3. services.php. La página de servicios se deriva de la clase Page pero reemplaza Display() para variar el resultado

```
<?php
   require ('page.inc');

   class ServicesPage extends Page
   {
      var $row2buttons = array( 'Re-engineering' => 'reengineering.php',
                                'Standards Compliance' => 'standards.php',
                                'Buzzword Compliance' => 'buzzword.php',
                                'Mission Statements' => 'mission.php'
                              );
      function Display()
      {
         echo  "<html>\n<head>\n";
         $this  ->  DisplayTitle();
```

```
        $this -> DisplayKeywords();
        $this -> DisplayStyles();
        echo  "</head>\n<body>\n";
        $this  ->  DisplayHeader();
        $this  ->  DisplayMenu($this->buttons);
        $this  ->  DisplayMenu($this->row2buttons);
        echo  $this->content;
        $this  ->  DisplayFooter();
        echo  "</body>\n</html>\n";
    }
}

$services  =  new  ServicesPage();
$content  ='<p>At  TLA  Consulting,  we  offer  a  number  of  services.
            Perhaps the productivity of your employees would
            improve if we re-engineered your business.
            Maybe all your business needs is a fresh mission
            statement, or a new batch of buzzwords.</p>';
$services -> SetContent($content);
$services  ->  Display();
?>
```

El sustituto de `Display()` es muy similar con la excepción de que contiene una línea más

```
$this  ->  DisplayMenu($this->row2buttons);
```

para llamar a `DisplayMenu()` una segunda vez y crear una segunda barra de menú. Fuera de la definición de clase, creamos una instancia de la clase `ServicePage`, establecemos los valores para los que no queremos los predeterminados y llamamos a `Display()`.

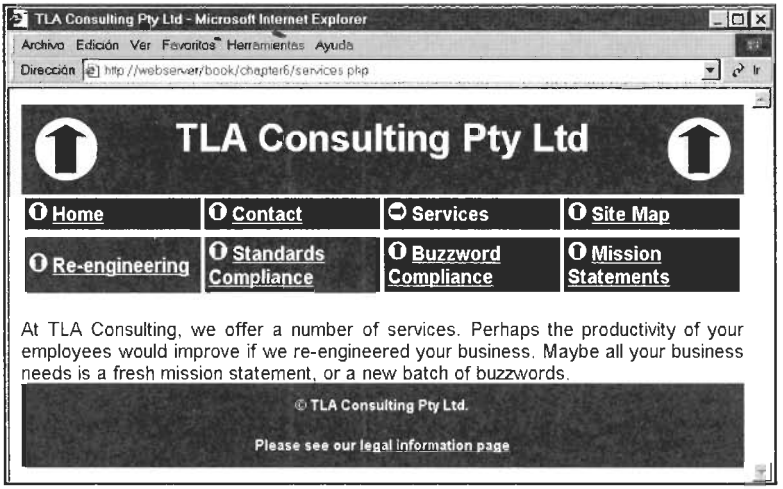


Figura 6.2. La página de servicios se creó utilizando la herencia para volver a utilizar gran parte de nuestra página estándar

Como se ilustra en la figura 6.2, tenemos una nueva variante de la página estándar. El único código nuevo que necesitamos escribir es el que hace referencia a las partes diferentes.

La creación de páginas a través de clases PHP presenta ventajas claras. Con una clase que realice la mayor parte del trabajo por nosotros, necesitaremos realizar menos trabajo para crear una nueva página. Podemos actualizar todas las páginas de una vez actualizando la clase. La herencia nos permite derivar diferentes versiones de la clase desde el original sin perder sus ventajas.

Como ocurre con casi todo en esta vida, estas ventajas tienen un coste.

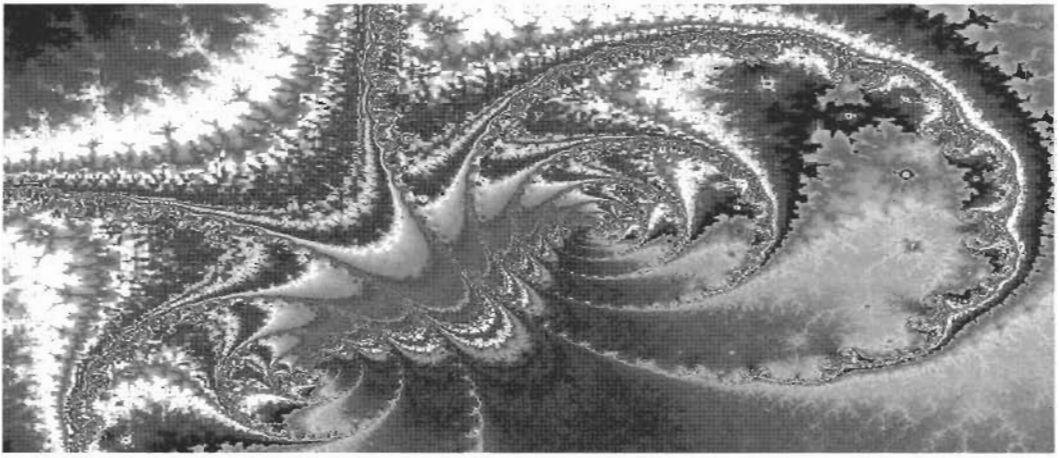
La creación de páginas desde una secuencia de comandos exige un mayor esfuerzo del procesador que la simple carga de una página HTML estándar desde el disco y su envío a un navegador. En un sitio con tráfico este hecho resultará importante por lo que convendría utilizar páginas HTML estáticas o almacenar en caché el resultado de las secuencias de comandos allí donde resulte posible para cargarlo en el servidor.

A continuación

La siguiente sección aborda el estudio de MySQL. Explicaremos cómo crear y completar una base de datos de MySQL, y vincularemos lo aprendido a PHP para poder acceder a nuestra base de datos desde la Web.

Parte II

Uso de MySQL



7

Diseño de la base de datos Web

Ahora que ya se ha familiarizado con los fundamentos de PHP, vamos a ver cómo integrar un base de datos en nuestras secuencias de comandos. En un capítulo anterior comentamos las ventajas de utilizar una base de datos relacional en lugar de un archivo sin procesar:

- Los RDBMS brindan un acceso más rápido a los datos que los archivos sin procesar.
- Los RDBMS permiten realizar consultas fácilmente para extraer conjuntos de datos que encajen con determinados criterios.
- Los RDBMS incorporan mecanismos para tratar con accesos simultáneos para no tener que preocuparnos de esta tarea como programadores.
- Los RDBMS brindan acceso aleatorio a los datos.
- Los RDBMS incorporan un sistema de privilegios.

Para ser más concretos, el uso de una base de datos relacional nos permite responder de forma rápida y sencilla a consultas sobre la procedencia de los clientes, los productos que se están vendiendo mejor o qué tipo de clientes son los que más dinero se dejan.

Esta información puede ayudarle a mejorar el sitio para atraer y fidelizar a los clientes.

En esta sección utilizaremos MySQL como base de datos.

Antes de profundizar en el estudio de las características de MySQL en el siguiente capítulo, examinaremos los siguientes aspectos:

- Conceptos relacionados con las bases de datos relacionales y terminología
- Diseño de bases de datos Web
- Arquitectura de bases de datos Web

Conceptos de base de datos relacionales

Las bases de datos relacionales son, con diferencia, el tipo de base de datos más utilizado.

Estas bases de datos se basan teóricamente en el álgebra relacional. No es necesario comprender la teoría para utilizar una base de datos relacional (aunque vendría bien), pero sí conviene comprender algunos conceptos básicos relacionados con las bases de datos.

Tablas

Las bases de datos relacionales se componen de relaciones, conocidas de manera más común como tablas. Una tabla es exactamente eso, una tabla de datos. Un ejemplo de tabla relacional es una hoja de cálculo electrónica.

Vamos a examinar un ejemplo.

En la figura 7.1, puede ver una tabla de ejemplo. Esta tabla contiene los nombres y las direcciones de los clientes de una librería, Book-O-Rama.

CUSTOMERS

CustomerID	Name	Address	City
1	Julie Smith	25 Oak Street	Airport West
2	Alan Wong	1/47 Haines Avenue	Box Hill
3	Michelle Arthur	357 North Road	Yarraville

Figura 7.1. Los detalles de los clientes de Book-O-Rama se guardan en una tabla

La tabla tiene un nombre (Customers), varias columnas, que hacen referencia a un tipo diferente de datos, y filas que se corresponden con los distintos clientes.

Columnas

Cada columna de la tabla tiene un nombre único y diferentes datos. Cada una de ellas lleva asociado un tipo de datos.

Por ejemplo, en la tabla Customers de la figura 7.1, puede ver que el campo CustomerID es de tipo entero y las otras tres columnas son cadenas. Las columnas a veces se llaman campos o atributos.

Filas

Cada fila de la tabla representa un cliente distinto. Debido al formato tabular, todos tienen los mismos atributos. Las filas también se denominan registros.

Valores

Cada fila se compone de un conjunto de valores individuales que se corresponden con las columnas. Cada valor debe tener el tipo de dato especificado por su columna.

Claves

Necesitamos disponer de una forma de identificar a cada cliente. Los nombres no son una buena opción (si tiene un nombre muy utilizado, entenderá por qué). Tomemos como ejemplo a Julie Smith de la tabla Customers. Si abre una guía de teléfonos de los EEUU, descubrirá una gran cantidad de entradas con ese nombre.

Podemos distinguir a Julie de diferentes formas. Es probable que sea la única Julie Smith que viva en la dirección que se indica. Sin embargo, el uso del nombre y la dirección en conjunto para distinguir a alguien resulta muy pesado y suena un poco a lenguaje legal. Además se necesita utilizar más de una columna de la tabla.

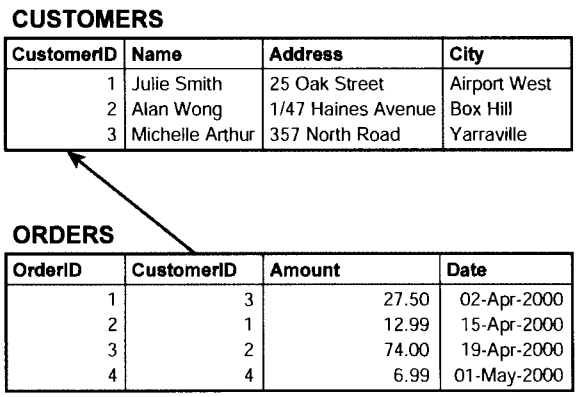
En este ejemplo, lo que hemos hecho, y es probable que se decante por la misma solución en sus aplicaciones, es asignar un número de identificación de cliente exclusivo (CustomerID). Se trata del mismo principio seguido para asignar números de cuenta o número de socios únicos. Esta técnica facilita el almacenamiento de los detalles en una base de datos. Un número de identificación asignado artificialmente garantiza su exclusividad. Existen muy pocos tipos de información real que dispongan de esta propiedad, aunque se utilicen en una combinación.

La columna de identificación de una tabla se denomina clave o clave principal. Las claves pueden constar de varias columnas. Éste sería el caso si optamos por utilizar el nombre y la dirección de un cliente, pero su exclusividad no está garantizada.

Las bases de datos suelen constar de varias tablas y utilizan una clave para asociar una tabla a otra. En la figura 7.2, hemos agregado una segunda tabla a la base de datos. En esta tabla se almacenan los pedidos realizados por los clientes. Cada fila de la tabla Orders representa un único pedido realizado por un único cliente. Sabemos quién es el cliente porque almacenamos su CustomerID. Por ejemplo, si examinamos el pedido con OrderID 2, descubriremos que lo ha realizado el cliente con CustomerID 1. A continuación, si examinamos la tabla Customers, veremos que el cliente con CustomerID 1 es Julie Smith.

El término utilizado en las bases relacionales para designar esta relación es clave secundaria. La columna CustomerIP es la clave principal de la tabla Customer, pero cuando aparece en otra tabla, como la tabla Orders, se conoce como clave secundaria.

Es posible que se esté preguntando por qué hemos optado por utilizar dos tablas distintas y no almacenar la dirección de Julie en la tabla Orders. En la siguiente sección intentaremos dar respuesta a esta cuestión.



CUSTOMERS

CustomerID	Name	Address	City
1	Julie Smith	25 Oak Street	Airport West
2	Alan Wong	1/47 Haines Avenue	Box Hill
3	Michelle Arthur	357 North Road	Yarraville

ORDERS

OrderID	CustomerID	Amount	Date
1	3	27.50	02-Apr-2000
2	1	12.99	15-Apr-2000
3	2	74.00	19-Apr-2000
4	4	6.99	01-May-2000

Figura 7.2. Cada pedido de la tabla Orders hace referencia a un cliente de la tabla Customer

Esquemas

El conjunto completo de diseños de tabla de un base de datos se conoce como esquema de la base de datos. Es como el esqueleto de la base de datos. Un esquema debe mostrar las tablas con sus columnas, los tipos de datos de las columnas e indicar la clave principal de cada tabla y sus claves secundarias. Los esquemas no incluyen datos, pero se puede incluir un ejemplo para explicar su objetivo. Los esquemas pueden adoptar la forma de diagramas, como los que estamos utilizando, diagramas de relaciones de entidad (que no se analizarán en este libro) o pueden tener forma de texto, como

Customers(CustomerID, Name, Address, City)

Orders(OrderID, CustomerID, Amount, Date)

Los términos subrayados con una línea continua en el esquema son las claves principales de la relación. Los términos subrayados con una línea discontinua son las claves secundarias de la relación en la que aparecen.

Relaciones

Las claves secundarias representan una relación entre los datos de dos tablas. Por ejemplo, el vínculo de Orders a Customers representa una relación entre una fila de la tabla Orders y una fila de la tabla Customers.

Existen tres tipos básicos de relaciones en una base de datos relacional. Se clasifican en función del número de elementos que haya a cada lado de la relación. Las relaciones puede ser uno a uno, uno a varios o varios a varios.

Una relación uno a uno significa que sólo hay un elemento a cada lado de la relación. Por ejemplo, si hemos colocado direcciones en una tabla distinta a la de clientes, existiría una relación uno a uno entre ambas. Podríamos tener una clave secundaria desde la tabla de direcciones a la tabla de clientes o al revés (no serían obligatorias ambas).

En una relación uno a varios, una fila de una tabla se vincula a varias filas de otra tabla. En nuestro ejemplo, un cliente podría haber realizado varios pedidos. En estas relaciones, la tabla que contiene varias filas tendrá una clave secundaria hasta la tabla con una sola fila. En este caso, hemos incluido la columna CustomerID dentro de la tabla Order para mostrar la relación.

En las relaciones varios a varios, varias filas de una tabla se asocian a varias filas de otra tabla. Por ejemplo, si tememos dos tablas, `Libros` y `Autores`, puede que un libro haya sido escrito por dos autores y que ambos hayan escrito otros libros, en solitario o en colaboración con otros autores. En este tipo de relaciones se crea una tabla especial para representar las relaciones, con lo que nos tendremos las siguientes: `Libros`, `Autores`, `Libros_Autores`. La tercera tabla contendrá únicamente las claves de las otras tablas como claves secundarias en pares, para mostrar qué autores están relacionados con cada libro.

Cómo diseñar nuestra base de datos Web

La tarea más difícil consiste en determinar cuándo se necesita una tabla y cuál debería ser la clave. Existe una ingente cantidad de material dedicado a estudiar los diagramas de relación de entidad y de normalización de bases de datos, pero quedan fuera del ámbito de este libro. Sin embargo, en la mayor parte de los casos, basta con seguir unos sencillos principios básicos de diseño. Vamos a analizarlos en el contexto de la aplicación Book-O-Rama.

Piense en los objetos del mundo real que está modelando

Al crear una base de datos, lo que hacemos es modelar elementos y relaciones del mundo real, y almacenar la información sobre dichos objetos y relaciones.

Por regla general, cada clase de objeto del mundo real que se modele necesitará su propia tabla.

Piense en ello un momento: queremos almacenar la misma información sobre todos nuestros clientes. Si existe un conjunto de datos con la misma "forma", podremos crear fácilmente una tabla que se corresponda con dichos datos.

En el ejemplo Book-O-Rama, queremos almacenar la información sobre nuestros clientes, los libros que vendemos y los detalles de los pedidos. Todos los clientes tienen un nombre y una dirección. Los pedidos tienen una fecha y una cantidad total, así como los libros pedidos. Los libros constan de ISBN, de un autor, un título y un precio.

Por lo tanto, parece que necesitamos al menos tres tablas en esta base de datos: `Customers`, `Orders` y `Books`. En la figura 7.3 se recoge el esquema inicial.

Por el momento, no podemos determinar qué libros se incluyen en cada pedido a partir del modelo. Lo veremos en un instante.

Cómo evitar el almacenamiento de datos redundantes

En una sección anterior ya nos planteamos por qué no almacenar la dirección de Julie Smith en la tabla de pedidos.

Si Julie realiza varios pedidos a Book-O-Rama, como así esperamos, acabaremos almacenando sus datos varias veces. Es probable que la tabla de pedidos presente un aspecto parecido al ilustrado en la figura 7.4.

CUSTOMERS

CustomerID	Name	Address	City
1	Julie Smith	25 Oak Street	Airport West
2	Alan Wong	1/47 Haines Avenue	Box Hill
3	Michelle Arthur	357 North Road	Yarraville

ORDERS

OrderID	CustomerID	Amount	Date
1	3	27.50	02-Apr-2000
2	1	12.99	15-Apr-2000
3	2	74.00	19-Apr-2000
4	4	6.99	01-May-2000

BOOKS

ISBN	Author	Title	Price
0-672-31687-8	Michael Morgan	Java 2 for Professional Developers	34.99
0-672-31745-1	Thomas Down	Installing Debian GNU/Linux	24.99
0-672-31509-2	Pruitt, et al.	Teach Yourself GIMP in 24 Hours	24.99

Figura 7.3. El esquema inicial consta de las tablas Customers, Orders y Books

ORDERS

OrderID	Amount	Date	CustomerID	Name	Address	City
12	199.50	25-Apr-2000	1	Julie Smith	28 Oak Street	Airport West
13	43.00	29-Apr-2000	1	Julie Smith	28 Oak Street	Airport West
14	15.99	30-Apr-2000	1	Julie Smith	28 Oak Street	Airport West
15	23.75	01-May-2000	1	Julie Smith	28 Oak Street	Airport West

Figura 7.4. El diseño de una base de datos que almacena datos redundantes aumenta el espacio utilizado y puede dar lugar a anomalías en los datos

Esto plantea dos problemas básicos.

El primero supone malgastar espacio innecesariamente. ¿Por qué guardar los detalles de Julie en tres sitios si sólo necesitamos guardarlos en uno?

El segundo problema son las anomalías que pueden surgir al actualizar los datos, es decir, situaciones en las que al cambiar los datos se producen inconsistencias. La integridad de los datos quedará alterada y ya no sabremos qué datos son correctos y

qué datos son incorrectos. Por regla general, esta situación dará lugar a la pérdida de datos.

Es necesario evitar tres tipos de anomalías de actualización: anomalías de modificación, de inserción y de eliminación.

Si Julie se muda de domicilio con pedidos todavía pendientes, tendremos que actualizar su dirección en tres lugares en lugar de en uno, lo que equivale a tener que realizar el trabajo tres veces. Resulta sencillo pasar por alto esta tarea y acabar realizando el cambio en un solo lugar, lo que dará lugar a la incoherencia de los datos de la base de datos (algo muy negativo). Estos problemas se conocen como anomalías de modificación porque tienen lugar cuanto se intenta modificar la base de datos.

En este diseño, necesitamos insertar los detalles de Julie cada vez que recibimos un pedido, lo que significa que deberemos comprobar y asegurarnos de que sus datos son correctos en las filas existentes de la tabla. Si pasáramos por alto la comprobación, podríamos acabar con dos filas de datos desiguales sobre Julie. Por ejemplo, una fila podría indicar que vive en Airport West y otra en Airport. Este tipo de anomalía se conoce como anomalía de inserción porque tiene lugar al introducir los datos.

El tercer tipo de anomalía se conoce como anomalía de eliminación porque tiene lugar (sorpresa, sorpresa) al eliminar filas de la base de datos. Por ejemplo, imagine que tras enviar un pedido, lo eliminamos de la base de datos. Cuando se hayan servido todos los pedidos de Julie, se habrán eliminado todos de la tabla Order. Esto significa que no dispondremos de un registro sobre la dirección de Julie. Por tanto, no podremos enviarles ninguna oferta especial y necesitaremos tomar sus datos de nuevo cuando vuelva a realizar un pedido.

Por regla general, al diseñar las bases de datos se intenta evitar estas anomalías.

Uso de valores de columna únicos

Cada atributo de una fila debe almacenar un solo valor. Por ejemplo, necesitamos saber qué libros componen cada pedido. Existen varias formas de hacerlo.

Podemos agregar una columna a la tabla `Orders` en la que se enumeren todos los libros que hemos pedido, como se ilustra en la figura 7.5.

ORDERS

OrderID	CustomerID	Amount	Date	Books Ordered
1	3	27.50	02-Apr-2000	0-672-31697-8
2	1	12.99	15-Apr-2000	0-672-31745-1, 0-672-31509-2
3	2	74.00	19-Apr-2000	0-672-31697-8
4	3	6.99	01-May-2000	0-672-31745-1, 0-672-31509-2, 0-672-31697-8

Figura 7.5. En este diseño, el atributo Books Ordered de cada fila incluye varios valores

No es una buena idea por varias razones. Lo que en realidad estamos haciendo es anidar una tabla dentro de una columna (una tabla que relaciona pedidos y libros). Al hacerlo, resulta más difícil responder a preguntas como "¿Cuántas copias del libro

Java 2 for Professional Developers quiere pedir?" ya que el sistema no puede contar los campos coincidentes. En su lugar, debe analizar cada valor de atributo y determinar si contiene una correspondencia en su interior.

Como lo que estamos haciendo es crear una tabla dentro de otra, lo que deberíamos hacer es crear una nueva tabla. Esta nueva tabla, se denomina `Order_Items` y se muestra en la figura 7.6.

ORDER_ITEMS

OrderID	ISBN	Quantity
1	0-672-31697-8	1
2	0-672-31745-1	2
2	0-672-31509-2	1
3	0-672-31697-8	1
4	0-672-31745-1	1
4	0-672-31509-2	2
4	0-672-31697-8	1

Figura 7.6. Este diseño facilita la búsqueda de libros concretos pedidos

Esta tabla proporciona un vínculo entre la tabla `Orders` y la tabla `Books`. Es habitual crear este tipo de tablas en una relación varios a varios entre dos objetos: en este caso, un pedido puede constar de varios libros y varias personas pueden pedir un libro.

Selección de claves lógicas

Asegúrese de que las claves seleccionadas son exclusivas. En nuestro caso, hemos creado una clave especial para los clientes (CustomerID) y para los pedidos (OrderID) porque es posible que estos objetos del mundo real no dispongan de manera natural de un identificador con garantías de exclusividad. No necesitamos crear un identificador único para los libros porque ya existe en forma de ISBN. En `Order_Item`, puede agregar una clave adicional si lo desea, pero la combinación de los atributos OrderID (subrayadps???) e ISBN resultará exclusiva siempre y cuando se trate a cada copia de un libro dentro de un pedido como una fila. Por resta razón, la tabla `Order_Item` incluye una columna Quantity.

Reflexiones sobre las preguntas que desea formular a la base de datos

Enlazando con la sección anterior, piense en las preguntas que desea que responda la base de datos. (Recuerde las preguntas que realizamos al principio del capítulo. Por ejemplo, ¿cuáles son los libros que más vende Book-O-Rama?). Asegúrese de que la base de datos contiene todos los datos necesarios y que se han establecido los vínculos adecuados entre las tablas para responder a las preguntas deseadas.

Evite diseños con varios atributos vacíos

Si desea agregar reseñas de libros a la base de datos, dispone al menos dos formas de hacerlo. En la figura 7.7 se recogen ambos enfoques.

La primera consiste en agregar la columna `Review` a la tabla `Books`. En esta solución se incluye un campo nuevo de reseña para cada libro. Si la base de datos consta de una gran cantidad de libros y el encargado de las reseñas no tiene previsto crear una para cada libro, muchas filas quedarán vacías. Es lo que se conoce como valor nulo.

BOOKS

ISBN	Author	Title	Price	Review
0-672-31687-8	Michael Morgan	Java 2 for Professional Developers	34.99	
0-672-31745-1	Thomas Down	Installing Debian GNU/Linux	24.99	
0-672-31509-2	Pruitt, et al.	Teach Yourself GIMP in 24 Hours	24.99	

BOOK_REVIEWS

ISBN	Review

Figura 7.7. Para agregar reseñas, podemos incorporar una nueva columna a la tabla Books o añadir una tabla específica para almacenar esta información

No conviene tener muchos valores nulos en una tabla ya que esta práctica contribuye a desperdiciar espacio de almacenamiento y ocasiona problemas al calcular totales o al aplicar otras funciones a columnas numéricas. Si un usuario ve un valor nulo en una tabla, no sabrá si se debe a que el valor no resulta relevante, a que hay un error en la base de datos o a que no se han introducido los datos todavía.

Por regla general, para evitar los problemas con la inclusión de demasiados valores nulos se recurre a un diseño alternativo. En este caso, podemos utilizar el segundo diseño ilustrado en la figura 7.7. En este caso, sólo se enumeran los libros con una reseña a la tabla `Book_Reviews`, así como sus reseñas.

Fíjese en que este diseño se basa en la idea de tener una persona encargada de la revisión. Como alternativa podríamos permitir que los clientes se encarguen de realizar las reseñas. En este caso, podríamos añadir el campo `CustomerID` a la tabla `Book_Reviews`.

Resumen de los tipos de tablas

Como tendrá la oportunidad de comprobar, las bases de datos suelen componerse de dos tipos de tablas:

- Tablas sencillas que describen un objeto del mundo real. También podrían contener claves a otros objetos simples en los que exista una relación uno a uno o uno a varios. Por ejemplo, nuestro cliente podría contener varios pedidos,

pero cada pedido se asocia a un único cliente. Por lo tanto, asignamos una referencia al cliente en el pedido.

- Tablas de unión que describen una relación varios a varios entre dos objetos reales como la relación entre `Orders` y `Books`. Estas tablas suelen asociarse con algún tipo de transacción del mundo real.

Arquitectura de base de datos Web

Tras comentar la arquitectura interna de nuestra base de datos, vamos a examinar la arquitectura externa de un sistema de base de datos Web y a comentar la metodología para desarrollar un sistema de base de datos Web.

Arquitectura

En la figura 7.8 se ilustra el funcionamiento básico de un servidor Web. Este sistema consta de dos objetos: un navegador Web y un servidor Web. Para unirlos se necesita una forma de comunicación. Un navegador Web realiza una solicitud al servidor. El servidor devuelve una respuesta. Esta arquitectura resulta apropiada para un servidor que devuelva páginas estáticas. La arquitectura que devuelve resultados de un sitio Web basado en una base de datos resulta un tanto más complicada.



Figura 7.8. La relación cliente/servidor entre un navegador Web y un servidor Web requiere comunicación

Las aplicaciones de base de datos Web que desarrollaremos en este libro siguen la estructura general de base de datos Web que se ilustra en la figura 7.9. Gran parte de esta estructura debería resultarle familiar a estas alturas.

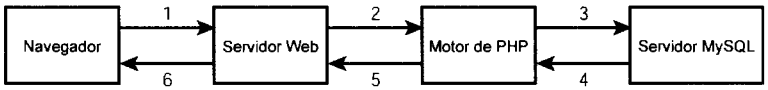


Figura 7.9. La arquitectura básica de base de datos Web se compone de un navegador Web, un servidor Web, un motor de secuencia de comandos y un servidor de base de datos

Una transacción típica de base de datos Web se compone de las fases siguientes, numeradas en función de la figura 7.9. Examinaremos estas fases en el contexto del ejemplo Book-O-Rama.

1. El navegador Web de un usuario envía una petición HTTP a una página Web dada. Por ejemplo, puede tratarse de una búsqueda para extraer todos los libros escritos por Laura Thomson en Book-O-Rama con ayuda de un formulario HTML. La página de resultados de búsqueda se denomina results.php.
2. El servidor Web recibe la petición de results.php, recupera el archivo y lo pasa al motor de PHP para su procesamiento.
3. El motor de PHP comienza a analizar la secuencia de comandos. Dentro de la secuencia de comandos hay un comando que establece la conexión a la base de datos y ejecuta una consulta (realiza la búsqueda de libros). PHP abre una conexión al servidor MySQL y remite la consulta pertinente.
4. El servidor MySQL recibe la consulta de la base de datos y la procesa. A continuación, envía los resultados (una lista de libros) al motor de PHP.
5. El motor de PHP termina de ejecutar la secuencia de comandos, lo que suele implicar la aplicación de formato a los resultados en HTML. Seguidamente, devuelve el código HTML resultante al servidor Web.
6. El servidor Web devuelve el código HTML al navegador donde el usuario puede ver la lista de los libros solicitados.

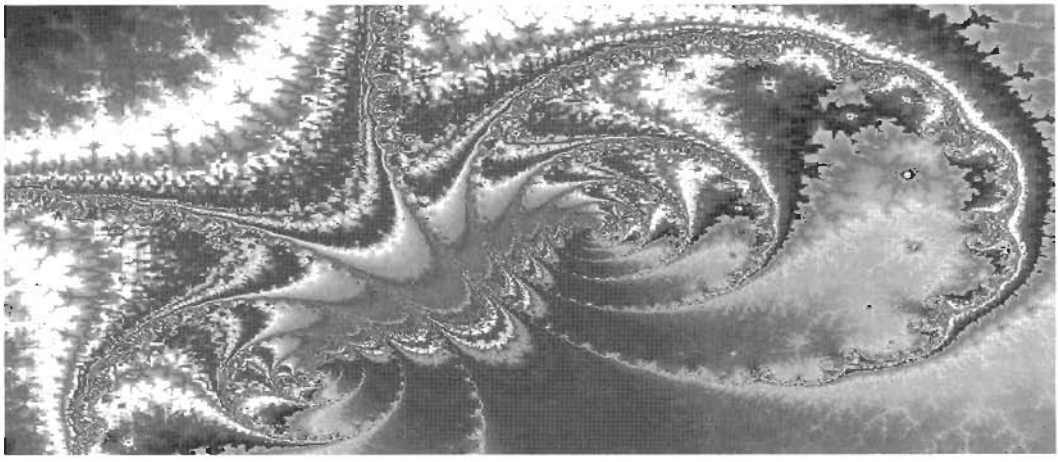
El proceso es básicamente el mismo independientemente del motor de secuencia de comandos o del servidor de base de datos que utilicemos. Por regla general, el software del servidor Web, el motor de PHP y el servidor de la base de datos se ejecutan en el mismo equipo. Sin embargo, también es habitual ejecutar el servidor de la base de datos en un equipo diferente, por razones de seguridad, de una mayor capacidad o de reparto de la carga. Desde el punto de vista de desarrollo, el trabajo será el mismo, pero puede presentar algunas ventajas significativas de rendimiento.

Lecturas adicionales

En este capítulo, hemos analizado alguna de las directrices para el diseño de bases de datos. Si quiere profundizar en la teoría sobre la que descansan las bases de datos relacionales, puede leer algunas de las obras escritas por expertos en el tema como C.J. Date. Eso sí, tenga en cuenta que el material puede resultar bastante teórico y poco relevante a corto plazo para un desarrollador Web comercial. Las bases de datos Web más habituales no suelen alcanzar un nivel tan alto de complejidad.

A continuación

En el siguiente capítulo, comenzaremos a configurar nuestra base de datos MySQL. En primer lugar aprenderá a configurar una base de datos MySQL para la Web y cómo consultarla y, a continuación, cómo consultarla desde PHP.



8
Creación de la base de datos Web

En este capítulo vamos a comentar cómo configurar una base MySQL para su uso en un sitio Web.

En este capítulo abordaremos los siguientes aspectos:

- Creación de una base de datos
- Usuarios y privilegios
- Introducción al sistema de privilegios
- Creación de tablas de base de datos
- Tipos de columna en MySQL

En este capítulo, vamos a seguir adelante con la aplicación de la librería en línea Book-O-Rama comentada en el último capítulo.

A continuación, se recoge el esquema de la aplicación de Book-O-Rama para ayudarle a recordar:

Customers(CustomerID, Name, Address, City)
Orders(OrderID, CustomerID, Amount, Date)
Books(ISBN, Author, Title, Price)
Order_Items(OrderID, ISBN, Quantity)
Book_Reviews(ISBN, Reviews)

Recuerde que las claves principales aparecen subrayadas con una línea continua y las claves secundarias con una línea discontinua.

Para utilizar el material de esta sección, debe disponer de acceso a MySQL, lo cual significa que

- Ha completado la instalación básica de MySQL en su servidor Web. Esto incluye:
  - Instalar los archivos
  - Configurar un usuario para ejecutar MySQL
  - Configurar su ruta
  - Ejecutar `mysql_install_db`, si resultara necesario
  - Establecer la configuración del usuario raíz
  - Eliminar el usuario anónimo y la base de datos de prueba
  - Iniciar el servidor de MySQL
- Si ha realizado todas estas operaciones, puede seguir adelante con la lectura del capítulo. Si no lo ha hecho, en el apéndice A encontrará instrucciones sobre cómo proceder.
- Si tiene problemas durante este capítulo, puede que se deba a la configuración de su sistema MySQL. En este caso, revise esta lista y el apéndice A para asegurarse de que la configuración es correcta.
- Dispone de acceso a MySQL en un equipo de cuya administración no se encarga, como un servicio de alojamiento Web, un equipo en su lugar de trabajo, etc.
  Si este fuera el caso, para poder seguir los ejemplos o crear su propia base de datos, deberá pedir a su administrador que configure un usuario y la base de datos para permitirle trabajar con ella y que le indique el nombre de usuario, la contraseña y el nombre de la base de datos que se le ha asignado.
  Puede saltarse las secciones de este capítulo dedicadas a explicar cómo configurar usuarios y bases de datos o leerlas para explicar mejor lo que necesita su administrador del sistema. Como usuario normal, no podrá ejecutar los comandos para crear usuarios y bases de datos.

Los ejemplos de este capítulo se han elaborado y probado con la versión 3.23.52 de MySQL. Las versiones anteriores disponen de menor funcionalidad. Debería instalar o actualizar su versión a la más reciente, que puede descargar del sitio de MySQL en `http://mysql.com`.

Nota sobre el uso del monitor de MySQL

En este capítulo y en el siguiente, apreciará que en los ejemplos de MySQL cada comando termina en un punto y coma (;). Este símbolo indica a MySQL que ejecute

el comando. Si olvida utilizar el punto y coma, no se aplicará ninguna acción. Se trata de un problema común entre los nuevos usuarios.

También significa que puede utilizar nuevas líneas en mitad de un comando. En el libro, hemos utilizado este recurso para facilitar la lectura de los ejemplos. Apreciará dónde lo hemos hecho porque MySQL incluye un símbolo de continuación. Se trata de una flecha como la incluida en este fragmento:

```
mysql>  grant  select
   ->
```

Este símbolo indica que MySQL espera más entradas. Mientras no se escriba un punto y coma, obtendrá estos caracteres al pulsar **Intro**.

Tenga en cuenta además que las instrucciones de SQL no discriminan entre mayúsculas y minúsculas, pero que las bases de datos y las tablas sí pueden hacerlo (este aspecto se ampliará más adelante).

Cómo registrarse en MySQL

Para registrarse, diríjase hasta la interfaz de línea de comandos de su equipo y escriba lo siguiente:

```
mysql -h nombredehost -u nombredeusuario -p
```

El símbolo de comandos puede presentar un aspecto diferente según el sistema operativo y el núcleo que esté utilizando.

El comando `mysql` invoca al monitor de MySQL. Se trata de un cliente de línea de comandos que nos permite comunicarnos con el servidor MySQL.

El modificador `-h` se utiliza para especificar el host al que establecer la conexión, es decir, el equipo en el que se está ejecutando el servidor MySQL. Si ejecuta este comando en el mismo equipo que el servidor MySQL, no necesitará utilizar el modificador `-h` ni el parámetro `nombredehost`. De lo contrario, deberá sustituir dicho parámetro por el nombre del equipo en el que se esté ejecutando el servidor MySQL.

El modificador `-u` se utiliza para especificar el `nombredeusuario` con el que desea conectarse. Si no se especifica, se utilizará el nombre de usuario predeterminado con el que se inició la sesión en el sistema operativo.

Si ha instalado MySQL en su propio equipo o servidor, deberá registrarse como `root` y crear la base de datos que utilizaremos en esta sección. Si ha realizado una instalación limpia, el usuario raíz será el único usuario del que dispondrá para iniciar la sesión.

Si está utilizando MySQL en un equipo administrado por otra persona, utilice el nombre de usuario que se le haya asignado.

El modificador `-p` indica al servidor que se desea establecer la conexión utilizando una contraseña. Puede evitar su uso, si no se ha establecido una contraseña para el usuario utilizado para iniciar la sesión.

Si se está iniciando la sesión como usuario raíz y no se ha establecido una contraseña para dicho usuario, le recomendamos que consulte el apéndice A de inmediato. Sin una contraseña para el usuario raíz , su sistema es inseguro.

No es necesario que incluya la contraseña en esta línea. El servidor MySQL se la pedirá. De hecho, es mejor no hacerlo. Si introduce la contraseña en la línea de comandos, se mostrará en la pantalla y otros usuarios podrían verla.

Tras introducir el comando anterior, debería obtener un resultado parecido al siguiente:

```
Enter password: ****
```

(Si no funcionase, verifique que el servidor MySQL se está ejecutando y el comando `mysql` aparece en algún lugar de su ruta.)

Debería introducir la contraseña. Si todo va bien, se mostrará una respuesta parecida a la siguiente:

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 5 to server version: 3.23.52-nt
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql>
```

En su equipo, si no obtiene una respuesta similar, asegúrese de haber ejecutado `mysql_install_db`, si resultara necesario, de haber establecido la contraseña del usuario raíz y de haberla escrito correctamente.

Si se trata de otro equipo, asegúrese de haber escrito correctamente la contraseña. Tras ello, debería aparecer el símbolo de comandos de MySQL listo para crear la base de datos.

Si está utilizando su propio equipo, siga las directrices de la siguiente sección.

Si está utilizando el equipo de otra persona, esta fase ya estará completada. Puede saltar a la sección «Uso de la base de datos correcta». Si lo desea, puede leer las secciones intermedias para ampliar sus conocimientos pero no podrá ejecutar los comandos indicados en ellas. (O no debería, al menos).

Creación de bases de datos y usuarios

El sistema de base de datos de MySQL admite una gran cantidad de bases de datos diferentes. Por regla general, se utilizará una base de datos por cada aplicación, En nuestro ejemplo Book-O-Rama, la base de datos se llamará `books`.

Creación de la base de datos

Ésta es la parte más sencilla. Escriba la siguiente secuencia en el símbolo de comandos de MySQL:

```
mysql> create database nombrebd;
```

Sustituya el parámetro *nombrebd* por el nombre de la base de datos deseado. Para comenzar a crear el ejemplo Book-O-Rama, puede crear una base de datos llamada `books`.

Y eso es todo. Debería ver aparecer una respuesta como la siguiente:

```
Query OK, 1 row affected (0.06 sec)
```

Esta secuencia indica que todo ha funcionado correctamente. Si no la obtiene, asegúrese de incluir el punto y coma al final de la línea. Este símbolo indica a MySQL que ha terminado y debería ejecutar el comando.

Usuarios y privilegios

Un sistema MySQL puede tener muchos usuarios. Como norma debería utilizarse el usuario raíz para labores administrativas únicamente, por razones de seguridad. Deberá configurar una cuenta y una contraseña para cada usuario del sistema . No es necesario que sean las mismas utilizadas fuera de MySQL (por ejemplo, los nombres de usuario y contraseñas de UNIX o NT). Los mismos principios se aplican al usuario raíz. Conviene utilizar contraseñas diferentes para el sistema y para MySQL, en especial en el caso de la contraseña del usuario raíz.

No es obligatorio configurar contraseñas para los usuarios, pero sí muy aconsejable.

A la hora de configurar una base de datos Web, conviene establecer al menos un usuario por aplicación Web.

Es probable que se esté preguntado por qué. La respuesta reside en los privilegios.

Introducción al sistema de privilegios de MySQL

Una de las mejores funciones de MySQL es su avanzado sistema de privilegios.

Un privilegio es el derecho a realizar una acción dada sobre un determinado objeto y se asocia a un usuario concreto. El concepto es muy similar a los permisos de archivo.

Al crear un usuario dentro de MySQL, se le conceden un conjunto de privilegios en los que se especifica lo que puede y no puede hacer dentro del sistema.

Principio de asignación del privilegio más bajo

Este principio se puede utilizar para mejorar la seguridad de cualquier sistema informático.

Se trata de un principio elemental pero muy importante que se suele pasar por alto a menudo.

Su contenido es el siguiente:

- Un usuario (o proceso) debería disponer del nivel más bajo de privilegio necesario para realizar la tarea que se le asigne.

Su aplicación debe ir más allá de MySQL. Por ejemplo, para ejecutar consultas desde la Web, un usuario no necesita disponer de todos los privilegios asociados al usuario raíz. Por lo tanto, deberíamos crear otro usuario que sólo disponga de los privilegios necesarios para acceder a la base de datos que acabamos de crear.

Cómo configurar usuarios: el comando GRANT

Los comandos GRANT y REVOKE se utilizan para conceder y retirar los derechos a los usuarios de MySQL en cuatro niveles de privilegio. Estos niveles son:

- Global
- Base de datos
- Tabla
- Columna

En un instante veremos cómo se aplica cada uno de ellos.

El comando GRANT se utiliza para crear usuarios y concederles privilegios. La sintaxis general del comando GRANT es la siguiente:

```
GRANT privilegios [columnas]
ON elemento
TO nombre_usuario [IDENTIFIED BY 'contraseña']
[WITH GRANT OPTION]
```

Las cláusulas incluidas entre corchetes son opcionales. La sintaxis de este comando incluye varios marcadores de posición. El primer lugar, *privilegios*, equivale a una lista de privilegios separados por comas. MySQL consta de un conjunto definido de ellos, que se describen en la siguiente sección.

El marcador de posición *columnas* es opcional. Puede utilizarlo para especificar privilegios a cada columna. Puede utilizar el nombre de una sola columna o una lista de nombres de columna separadas por comas.

El marcador de posición, *elemento* es la base de datos o tabla a la que se aplican los nuevos privilegios.

Puede conceder privilegios a todas las bases de datos especificando `*.*` como elemento. Es lo que se conoce como conceder privilegios globales. También puede hacerlo especificando `*` únicamente si no va a utilizar ninguna base de datos en concreto.

Por regla general, especificará todas las tablas de la base de datos como `nombrebd.*`, una sola tabla de base de datos como `nombrebd.nombretabla` o columnas específicas indicando `nombrebd.nombretabla` y las columnas deseadas en el marcador de posición *columnas*. Si está utilizando una base de datos especí-

fica al remitir este comando y sólo se utiliza `nombretabla`, se interpretará como una tabla de la base de datos actual.

El marcador de posición *nombre_usuario* será el nombre del usuario con el que desea registrarse en MySQL. Recuerde que no tiene por qué ser el mismo que el nombre de registro del sistema. Este parámetro también puede contener un nombre de host. Puede utilizarlo para distinguir, por ejemplo, entre `Laura` (interpretado como laura@hostlocal) y `laura@otrositio.com`. Esta opción resulta bastante útil porque los usuarios pertenecientes a diferentes dominios suelen utilizar los mismos nombres. También permite incrementar la seguridad porque se puede especificar desde dónde se conectan los usuarios e incluso a qué tablas y datos pueden acceder desde una ubicación dada.

El marcador de posición *contraseña* es la contraseña que deseamos utilizar para iniciar la sesión. Se aplican las reglas habituales para la selección de contraseñas. En una sección posterior ampliaremos el tema de la seguridad, pero las contraseñas no deben resultar sencillas de adivinar. Es aconsejable que contengan un combinación de caracteres en mayúsculas y minúsculas y caracteres no alfabéticos.

La opción *WITH GRANT OPTION*, si se especifica, permite que el usuario indicado delegue sus privilegios en otros usuarios.

Los privilegios se almacenan en cuatro tablas del sistema, dentro de la base de datos `mysql`. Estas cuatro tablas se denominan `mysql.user`, `mysql.db`, `mysql.tables_priv` y `mysql.columns_priv` y se relacionan directamente con los cuatro niveles de privilegio mencionados anteriormente. Como alternativa a `GRANT`, puede alterar estas tablas directamente, como se analizará en un capítulo posterior.

Tipos y niveles de privilegio

Existen tres tipos básicos de privilegios en MySQL: los privilegios apropiados para su concesión a los usuarios habituales, los privilegios apropiados para su concesión a los administradores y dos privilegios especiales. Se puede asignar cualquier privilegio a cualquier usuario, pero es aconsejable restringir los privilegios de administrador a los administradores, siguiendo el principio de conceder el nivel más bajo de privilegios.

Debería restringir los privilegios concedidos a los usuarios para las bases de datos y tablas que necesiten utilizar. No debería conceder acceso a la base de datos `mysql` a nadie a excepción de a un administrador. En esta base de datos se almacenan todos los usuarios, contraseñas y otros datos. (En un capítulo posterior analizaremos esta base de datos.)

Los privilegios para el resto de los usuarios deberían establecerse directamente en función de los tipos específicos de comandos SQL y si un usuario puede ejecutarlos. En el siguiente capítulo analizaremos en detalle estos comandos. Por el momento, basta con la descripción conceptual indicada. En la tabla 8.1 se recogen estos privilegios. En la columna **Se aplica a** se enumeran los objetos a los se pueden conceder cada privilegio.

Tabla 8.1. Privilegios para usuarios

Privilegio	Se aplica a	Descripción
SELECT	tablas, columnas	Permite a los usuarios seleccionar filas (registros de tablas).
INSERT	tablas, columnas	Permite a los usuarios insertar nuevas filas en las tablas
UPDATE	tablas, columnas	Permite a los usuarios modificar valores de filas de tablas existentes.
DELETE	tablas	Permite a los usuarios eliminar filas de tablas existentes.
INDEX	tablas	Permite a los usuarios crear y eliminar índices sobre tablas concretas.
ALTER	tablas	Permite a los usuarios alternar la estructura de tablas existentes agregando, por ejemplo, columnas, cambiando el nombre de las columnas o de las tablas y modificando los tipos de datos de las columnas.
CREATE	bases de datos y tablas	Permite a los usuarios eliminar bases de datos o tablas. Si especifica una base de datos o una tabla dada en el comando GRANT, los usuarios sólo podrán crear dicha base de datos o tabla, lo que significa que tendrán que eliminarla primero.
DROP	bases de datos y tablas	Permite a los usuarios eliminar bases de datos o tablas.

La mayor parte de los privilegios para los usuarios normales son relativamente inofensivos en términos de la seguridad del sistema. Se puede utilizar el privilegio ALTER para resolver el sistema de privilegios cambiando el nombre de las tablas, pero los usuarios suelen utilizarlo bastante. La búsqueda de la seguridad consiste en hallar un equilibrio entre la usabilidad y la protección. Debe tomar una decisión con respecto al privilegio ALTER, pero debe saber que se suele conceder a los usuarios a menudo.

Además de los privilegios enumerados en la tabla 8.1, existe el privilegio REFERENCES que no se utiliza y el privilegio GRANT que se concede con el parámetro WITH GRANT OPTION en lugar de en la lista de *privilegios*.

La tabla 8.2 muestra los privilegios apropiados para su concesión a los usuarios administrativos.

Tabla 8.2. Privilegios para administradores

Privilegio	Descripción
`RELOAD`	Permite a un administrador volver a cargar tablas de concesión y eliminación de privilegios, hosts, registros y tablas.
`SHUTDOWN`	Permite a un administrador cerrar el servidor MySQL.
`PROCESS`	Permite a un administrador ver los procesos de servidor y cerrarlos.
`FILE`	Permite la lectura de datos en tablas desde archivos y viceversa.

Estos privilegios se pueden conceder a usuarios que no sean administradores pero debe estar muy atento si está considerando dicha posibilidad. El usuario medio no necesita utilizar los privilegios `RELOAD`, `SHUTDOWN` y `PROCESS`.

El privilegio `FILE` es un tanto diferente. La posibilidad de cargar datos desde archivos puede contribuir a ahorrar mucho tiempo al volver a introducir los datos en la base de datos. Sin embargo, la función de carga de archivos se puede utilizar para cargar cualquier archivo que puede ver el servidor MySQL, incluidas las bases de datos pertenecientes a otros usuarios y, potencialmente, los archivos de contraseñas. Tenga cuidado al conceder este privilegio u ofrezca la posibilidad de realizar la carga de los usuarios.

Existen otros dos privilegios especiales, que se recogen en la tabla 8.3.

Tabla 8.3. Privilegios especiales

Privilegio	Descripción
`ALL`	Concede todos los privilegios enumerados en las tablas 8.1 y 8.2. También se puede escribir `ALL PRIVILEGES` en lugar de `ALL`.
`USAGE`	No concede ningún privilegio. Este privilegio crea un usuario y le permite registrarse, pero no se le concede ningún otro privilegio. Por regla general, la concesión de otros privilegios se suele dejar para un momento posterior.

El comando REVOKE

Se trata del comando opuesto al comando `GRANT`. Se utiliza para retirar privilegios de un usuario.

Su sintaxis es muy similar a la sintaxis de GRANT:

```
REVOKE   privilegios  [(columnas)]
ON elemento
FROM nombre_usuario
```

Si se han concedido privilegios con la cláusula WITH GRANT OPTION, puede revocarlos de la siguiente forma:

```
REVOKE  GRANT  OPTION
ON elemento
FROM nombre_usuario
```

Ejemplos de uso de GRANT y REVOKE

Para configurar un administrador, puede escribir:

```
mysql>  grant all
    ->  on *
    ->  to fred identified by 'mnb123'
    ->  with grant option;
```

Este ejemplo concede todos los privilegios sobre todas las bases de datos a un usuario llamado Fred con la contraseña mnb123 y le permite traspasar dichos privilegios.

No le gustará tener un usuario así en su sistema por lo que conviene revocarlo:

```
mysql>  revoke all
    ->  on *
    ->  from fred;
```

Seguidamente vamos a configurar un usuario normal sin privilegios.

```
mysql>  grant usage
    ->  on books.*
    ->  to sally identified by 'magic123';
```

Hablamos con Sally para determinar qué desea hacer y le concedemos los privilegios adecuados:

```
mysql>  grant select, insert, update, delete, index, alter, create, drop
    ->  on books.*
    ->  to sally;
```

Fíjese en que no necesitamos especificar la contraseña de Sally para realizar esta tarea.

Si decidimos que Sally ha intentado realizar algo no apropiado en la base de datos, podemos reducir sus privilegios:

```
mysql>  revoke alter, create, drop
    ->  on books.*
    ->  from sally;
```

Y posteriormente cuando ya no necesite utilizar la base de datos, podemos revocar todos sus privilegios:

```
mysql>  revoke all
    ->  on books.*
    ->  from sally;
```

Cómo configurar un usuario para la Web

Necesitará configurar un usuario para que sus secuencias de comandos de PHP se conecten a MySQL. Podemos volver a aplicar el principio del privilegio mínimo. ¿Qué se permitirá a las secuencias de comandos?

En la mayor parte de los casos, sólo se necesita seleccionar, insertar, eliminar y actualizar filas de tablas. Podemos configurar estas tareas de la siguiente forma:

```
mysql>  grant select, insert, delete, update
    ->  on books.*
    ->  to bookorama identified by 'bookorama123';
```

Por razones de seguridad, debería seleccionar una contraseña mejor.

Si utilizamos un servicio de alojamiento Web, por regla general obtendremos acceso al resto de los privilegios de tipo usuario en una base de datos creada para nosotros. Se nos asignará un nombre de usuario y una contraseña para el uso de la línea de comandos (desde donde crear tablas y realizar otras tareas por el estilo) y para realizar las conexiones de secuencia de comandos Web (para consultar la base de datos). Este sistema resulta un poco menos seguro. Puede configurar un usuario con este nivel de privilegios de la siguiente forma:

```
mysql>  grant select, insert, update, delete, index, alter, create, drop
    ->  on books.*
    ->  to bookorama identified by 'bookorama123';
```

Proceda a configurar esta segunda versión del usuario ya que es la que utilizaremos en la siguiente sección.

Cómo cerrar la sesión como administrador

Puede cerrar la sesión del monitor de MySQL escribiendo `quit`. Debería volver a registrarse como usuario Web para comprobar si todo funciona correctamente.

Uso de la base de datos correcta

Si ha llegado hasta aquí, debería estar registrado en una cuenta de MySQL de nivel de usuario lista para probar el ejemplo de código, bien porque acaba de configurarla o porque el administrador del servidor Web se ha encargado de ello.

Lo primero que necesita hacer al iniciar una sesión es especificar qué base de datos desea utiliza. Para ello, puede escribir:

```
mysql> use nombrebd;
```

donde *nombrebd* es el nombre de la base de datos.

Como alternativa, puede evitar el comando `use` especificando la base de datos al iniciar la sesión, de la siguiente forma:

```
mysql -D nombrebd -h nombrehost -u nombreusuario -p
```

En este ejemplo, utilizaremos la base de datos de libros:

```
mysql> use books;
```

Al escribir este comando, MySQL debería devolver una respuesta como la siguiente

```
Database changed
```

Si no selecciona una base de datos antes de empezar a trabajar, MySQL devolverá un mensaje de error como el siguiente:

```
ERROR 1046: No Database Selected
```

Creación de tablas de base de datos

El siguiente paso en la configuración de la base de datos consiste en crear las tablas. Para ello puede utilizar el comando `CREATE TABLE` de SQL. Esta instrucción suele presentar esta forma:

```
CREATE TABLE nombretabla (columnas)
```

Debería sustituir el marcador de posición *nombretabla* con el nombre de la tabla que desea crear y el marcador de posición *columnas* con una lista de columnas separadas por comas en la tabla. Cada columna tendrá un nombre seguido por un tipo de dato.

Customers(CustomerID, Name, Address, City)
Orders(OrderID, CustomerID, Amount, Date)
Books(ISBN, Author, Title, Price)
Order_Items(OrderID, ISBN, Quantity)
Book_Reviews(ISBN, Review)

El listado 8.1 muestra el código SQL para crear las tablas, asumiendo que ya hayamos creado la base de datos llamada `books`. Encontrará el código necesario en el archivo correspondiendo a este capítulo del CD-ROM. Puede ejecutar un archivo SQL existente, como el cargado desde el CD-ROM a través de MySQL mediante la siguiente secuencia:

```
> mysql -h host -u bookorama -D books -p < bookorama.sql
```

(Recuerde sustituir el host con el nombre de su host.)

El uso de la redirección de archivos resulta bastante práctico en este caso porque permite editar las secuencias SQL en el editor de texto deseado antes de ejecutarlas.

Listado 8.1. bookorama.sql. Código SQL para crear las tablas de Book-O-Rama

```
create table customers
(  customerid int unsigned not null auto_increment primary key,
   name char(30) not null,
   address char(40) not null,
   city char(20) not null
);

create table orders
(  orderid int unsigned not null auto_increment primary key,
   customerid int unsigned not null,
   amount float(6,2),
   date date not null
);

create table books
(  isbn char(13) not null primary key,
    author char(30),
    title char(60),
    price float(4,2)
);

create table order_items
(  orderid int unsigned not null,
   isbn char(13) not null,
   quantity tinyint unsigned,

   primary key (orderid, isbn)

);
create table book_reviews
(
   isbn char(13) not null primary key,
   review text
);
```

Cada tabla se crea con una instrucción `CREATE TABLE` independiente. Como puede observar hemos creado el esquema con las columnas diseñadas en el último capítulo. Cada columna tiene un tipo de dato listado tras su nombre. Algunas columnas tienen otros especificadores.

¿Cuál es el significado del resto de las palabras clave?

`NOT NULL` significa que todas las filas de la tabla deben tener un valor en este atributo. Si no se especifica, el campo puede dejarse en blanco (`NULL`).

`AUTO_INCREMENT` es una función especial de MySQL que puede utilizar sobre columnas de tipo entero. Significa que si dejamos el campo en blanco al insertar filas en la tabla, MySQL generará automáticamente un valor de identificación exclusivo. Este valor será una unidad mayor que el valor máximo incluido en la columna. Sólo puede tener una columna de este tipo por tabla. Las columnas que especifiquen `AUTO_INCREMENT` deben estar indexadas.

El uso de la palabra clave PRIMARY KEY tras el nombre de una columna determina que la columna se tratará como la clave principal de la tabla. Las entradas de esta columna deben ser exclusivas. MySQL indexará automáticamente esta columna. Fíjese en que en el listado anterior al utilizarlo con customerid en la tabla customer se ha hecho con AUTO_INCREMENT. La indexación automática sobre la clave principal se encarga del índice que necesita AUTO_INCREMENT.

Si especificamos PRIMARY KEY tras el nombre de una columna, sólo se puede utilizar para claves principales formadas por una sola columna. El uso de la cláusula PRIMARY KEY al final de la instrucción order_items es una alternativa. Aquí la hemos utilizado porque la clave principal de esta tabla se compone de dos columnas.

La palabra clave UNSIGNED utilizada tras un tipo entero significa que sólo puede tener como valor cero o un número positivo.

Tipos de columna

Tomemos la primera tabla como ejemplo

```
create table customers
(  customerid int unsigned not null auto_increment primary key,
   name char(30) not null,
   address char(40) not null,
   city char(20) not null
);
```

Al crear una tabla, deberá tomar una decisión con respecto a los tipos de columnas.

En la tabla customers, tenemos cuatro columnas como se especifica en nuestro esquema. La primera, customerid es al clave principal, que hemos especificado directamente. Hemos decidido que sea de tipo entero (tipo int) y que los ID sean unsigned. También hemos aprovechado la función auto_increment para que MySQL pueda encargarse de ello automáticamente y evitarnos una preocupación.

Las otras dos columnas contendrán datos de tipo cadena. En este caso hemos seleccionado char como tipo de dato. Este tipo especifica campos de achura fija a través de corchetes; por ejemplo, name, puede tener hasta 30 caracteres.

Este tipo de datos asignará siempre 30 caracteres de espacio de almacenamiento para el nombre, aunque no se utilicen todos. MySQL rellenará los datos con espacios para asignarles el tamaño exacto.

Como alternativa se puede utilizar el tipo varchar, que sólo utiliza la cantidad de espacio necesario (más un byte). El tipo varchar absorbe menos espacio pero resulta más lento.

Para los clientes reales con nombres y direcciones reales, las anchuras seleccionadas para estas columnas son demasiado estrechas.

Fíjese en que hemos declarado todas las columnas como NOT NULL. Se trata de una pequeña optimización que puede realizar cuando resulte posible.

En un capítulo posterior trataremos el tema de la optimización.

Alguna de las instrucciones `CREATE` presentan variaciones en cuanto a la sintaxis. Examinemos la tabla `orders`:

```
create table orders
(  orderid int unsigned not null auto_increment primary key,
   customerid int unsigned not null,
   amount float(6,2),
   date date not null
);
```

La columna `amount` se especifica como número de coma flotante de tipo `float`. En la mayor parte de los tipos de datos de coma flotante, puede especificar la anchura de visualización y el número de decimales. En este caso, la cantidad del pedido se establecerá en dólares, por lo que podemos asignar un tamaño total razonablemente largo (una anchura de seis espacios) y dos decimales para los céntimos.

La columna `date` lleva asignado el tipo de dato `date`.

En esta tabla concreta, hemos especificado que las columnas puedan definir la cantidad como `NOT NULL`. ¿Por qué? Cuando se introduce un pedido en la base de datos, tendremos que crearlo en `orders`, agregarle los elementos a `order_items` y calcular la cantidad. Puede que no sepamos la cantidad al crear el pedido, por lo que permitimos que sea `NULL`.

La tabla `books` presenta algunas características similares.

```
create table books
(  isbn char(13) not null primary key,
   author char(30),
   title char(60),
   price float(4,2)
);
```

En este caso, no necesitamos generar la clave principal porque los ISBN se generan en otra parte.

Hemos dejado los otros campos como `NULL` porque una librería puede conocer el ISBN de un libro antes de conocer su título, autor o precio.

La tabla `order_items` muestra cómo crear claves principales multicolumna:

```
create table order_items
(  orderid int unsigned not null,
   isbn char(13) not null,
   quantity tinyint unsigned,

   primary key (orderid, isbn)
);
```

Hemos especificado la cantidad de un libro como `TINYINT UNSIGNED`, que equivale a un entero entre 0 y 255.

Como se mencionó anteriormente, las claves principales multicolumna necesitan especificarse con una cláusula de clave principal. Ésta es la que se utiliza aquí.

Por último, examine la tabla `book_reviews`:

```
create  table  book_reviews
(
   isbn char(13) not null primary key,
   review text
);
```

Esta tabla utiliza un nuevo tipo de dato, `text`, que todavía no se ha analizado. Este tipo se utiliza para texto de gran longitud como un artículo. Existen algunas variantes de este tipo que se comentarán en una sección posterior. Para comprender mejor el proceso de creación de tablas vamos a analizar los nombres de las columnas y los identificadores en general, y, a continuación, los tipos de datos que podemos seleccionar para las columnas. Pero en primer lugar, examinaremos la base de datos que hemos creado.

Cómo examinar la base de datos con SHOW y DESCRIBE

Inicie la sesión en el monitor de MySQL y utilice la base de datos `books`. Para ver las tablas de la base de datos, escriba

```
mysql>  show  tables;
```

MySQL mostrará una lista de todas las tablas de la base de datos:

```
+-----------------+
|Tables  in  books  |
+-----------------+
|book_reviews     |
|books            |
|customers        |
|order_items      |
|orders           |
+-----------------+
5 rows in set (0.06 sec)
```

También puede utilizar la instrucción `show` para ver una lista de bases de datos escribiendo

```
mysql>  show  databases;
```

Puede ver más información sobre una tabla concreta, por ejemplo, libros, utilizando `DESCRIBE`:

```
mysql>  describe  books;
```

MySQL mostrará la información suministrada al crear la base de datos:

```
+-------+-----------+--------+--------+-------------+----------+
|Field  |Type       |Null    |Key     |Default      |Extra     |
+-------+-----------+--------+--------+-------------+----------+
|isbn   |char(13)   |        |PRI     |             |          |
```

```
|author |char(30)   |YES    |        |NULL          |          |
|title  |char(60)   |YES    |        |NULL          |          |
|price  |float(4,2) |YES    |        |NULL          |          |
+-------+-----------+-------+--------+--------------+----------+
4 rows in set (0.05 sec)
```

Estos comandos resultan útiles para recordar un tipo de columna o navegar por una base de datos creada por otra persona.

Identificadores de MySQL

MySQL consta de cuatro tipos de identificadores: las bases de datos, las tablas, las columnas (elementos que ya conocemos) y los alias, que examinaremos en el siguiente capítulo.

Las bases de datos de MySQL se asignan a directorios en la estructura de archivos subyacente y las tablas lo hacen a archivos. Este hecho afecta directamente a los nombres que se les puede asignar. También afecta al uso de mayúsculas y minúsculas en dichos nombres. Si su sistema operativo discrimina entre mayúsculas y minúsculas utilizados en los nombres de los directorios o archivos, los nombres seleccionados para designar las bases de datos y las tablas también lo harán (por ejemplo, en UNIX) o, en caso contrario, no lo harán (por ejemplo, en Windows). Los nombres de columnas y los alias no discriminan entre mayúsculas y minúsculas, pero no puede utilizar versiones diferentes de los nombres en la misma instrucción de SQL.

La ubicación del directorio y los archivos que contengan los datos se establece en la configuración. Puede comprarlo en su sistema utilizando la instrucción `mysqladmin` de la siguiente forma:

```
mysqladmin variables
```

Estamos buscando la variable `datadir`.

En la tabla 8.4 se recoge un resumen de los posibles identificadores. La única excepción adicional es que no se puede utilizar ASCII(0), ASCII(255) ni el carácter de comillas en los identificadores (que es muy probable que no utilice).

Tabla 8.4. Identificadores de MySQL

Tipo	Longitud máxima	¿Discrimina entre mayúsculas y minúsculas?	Caracteres permitidos
Base de datos	64	según el sistema operativo	Todos los caracteres permitidos por el sistema operativo para el nombre de un directorio a excepción de los caracteres /, \, y . .

Tipo	Longitud máxima	¿Discrimina entre mayúsculas y minúsculas?	Caracteres permitidos
Tabla	64	según el sistema operativo	Todos los caracteres permitidos por el sistema operativo en un nombre de archivo a excepción de los caracteres / y . .
Columna	64	no	Todos los caracteres.
Alias	255	no	Todos los caracteres

Las reglas son muy abiertas.

Desde la versión MySQL 3.23.6, se pueden utilizar palabras reservadas y caracteres especiales de todos los tipos en identificadores, la única limitación es colocar entre comillas inclinadas los caracteres extraños.

Por ejemplo:

```
create database 'crear base de datos';
```

Las reglas de las versiones de MySQL (anteriores a la 3.23.6) son más restrictivas y no lo permiten.

Obviamente, conviene aplicar sentido común a toda esta libertad. El hecho de poder llamar a una base de datos 'crear base de datos' no significa que tengamos que hacerlo.

Se aplica el mismo principio aplicado en otros tipos de programación: utilice identificadores descriptivos.

Tipos de dato de columna

En MySQL existen tres tipos básicos de columnas: numérico, de fecha y hora, y de cadena. Cada categoría incluye un gran número de tipos. En este capítulo los resumiremos. En un capítulo posterior se analizarán sus puntos fuertes y sus puntos débiles.

Cada uno de los tres tipos dispone de varios tamaños de almacenamiento. Al seleccionar un tipo de columna, el principio general consiste en escoger el tipo más pequeño en el que encajen los datos.

Muchos de los tipos permiten especificar el tamaño máximo de visualización al crear la columna. Esta opción se identifica en las siguientes tablas con una `M`. Si resulta opcional se coloca entre corchetes. El valor máximo que se puede especificar para `M` es 255.

En las siguientes descripciones se utilizan los corchetes para indicar valores opcionales.

Tipos numéricos

Los tipos numéricos son números enteros o de coma flotante. En el caso de los números de coma flotante, puede especificar el número de lugares decimales. En este libro se utiliza `D` para indicarlo. El valor máximo que se puede especificar para `D` es de 30 o `M-2` (es decir, la longitud máxima de visualización menos dos, es decir, un carácter para el punto decimal y uno para la parte entera del número), según cual sea la menor.

Para los tipos enteros también puede especificar si desea que sean `UNSIGNED`, como se muestra en el listado 8.1.

En todos los tipos numéricos, también puede especificar el atributo `ZEROFILL`. Cuando se muestran los valores de una columna `ZEROFILL`, se rellenarán con ceros a la derecha.

Si especifica una columna como `ZEROFILL`, se le asignará automáticamente el atributo `UNSIGNED`.

En la tabla 8.5 se muestran los tipos integrales.

En la columna de rangos se muestran los rangos con firma y sin firma en líneas diferentes.

Tabla 8.5. Tipos de datos integrales

Tipo	Rango	Espacio de almacenamiento (bytes)	Descripción
`TINYINT[(M)]`	127..128 o..255	1	Enteros muy pequeños
`BIT`			Sinónimo de `TINYINT`
`BOOL`			Sinónimo de `TINYINT`
`SMALLINT[(M)]`	-32768..32767 o 0..65535	2	Enteros pequeños
`MEDIUMINT[(M)]`	-8388608.. 8388607 o 0..16777215	3	Enteros de tamaño medio
`INT[(M)]`	$-2^{31}..2^{31}-1$ o 0..16777215	4	Enteros normales
`INTEGER[(M)]`			Sinónimo de INT
`BIGINT[(M)]`	$-2^{63}..2^{63}-1$ o $0..2^{64}-1$	8	Enteros grandes

En la tabla 8.6 se recogen los tipos de coma flotante.

Tabla 8.6. Tipos de datos de coma flotante

Tipo	Rango	Espacio de almacenamiento (bytes)	Descripción
`FLOAT(precisión)`	depende de la precisión	varía	Se puede utilizar para especificar números de coma flotante de precisión única o doble.
`FLOAT[(M,D)]`	±1.175494351E-38 ±3.402823466E+38	4	Número de coma flotante de precisión única. Equivale a FLOAT(4), pero con un ancho de visualización y un número de decimales especificado.
`DOUBLE[(M,D)]`	±1.7976931348623157E +308 ±2.2250738585072014E -308		Equivale a FLOAT(8) pero con un ancho de visualización dado y con un número concreto de decimales.
`DOUBLE`			Sinónimo de `DOUBLE [(M,D)]`.
`PRECISION[(M,D)]`	como arriba		
`REAL[(M,D)]`	como arriba		Sinónimo de `DOUBLE [(M,D)]`.
`DECIMAL[(M[,D])]`	varía	M+2	Número de coma flotante almacenado como `char`. El rango depende de `M`, la anchura de visualización.
`NUMERIC[(M,D)]`	como arriba		Sinónimo de `DECIMAL`.
`DEC[(M,D)]`	como arriba		Sinónimo de `DECIMAL`.

Tipos de fecha y hora

MySQL admite una gran cantidad de tipos de fecha y hora, como se puede observar en la tabla 8.7. Estos tipos permiten recoger datos en formato numérico o de cadena. Conviene destacar que si se utiliza una columna `TIMESTAMP` se establecerá en la fecha y hora correspondiente a la operación más reciente realizada sobre la fila si no se establece manualmente. Este comportamiento resulta útil para el registro de transacciones.

Tabla 8.7. Tipos de datos de fecha y hora

Tipo	Rango	Descripción
`DATE`	1000-01-01 9999-12-31	Una fecha. Se mostrará con formato AAAA-MM-DD.
`TIME`	-838:59:59 838:59:59	Una hora. Se mostrará con formato `HH:MM:SS`. Tenga en cuenta que el rango resulta mucho más amplio de lo que es probable que necesite.
`DATETIME`	1000-01-01 00:00:00 9999-12-31 23:59:59	Una fecha y una hora. Se mostrarán con el formato `YYYY-MM-DD HH:MM:SS`.
`TIMESTAMP[(M)]`	1970-01-01 00:00:00	Una marca de tiempo, que resulta útil para la generación de informes. El formato de visualización depende del valor de M (véase la tabla 8.8 situada a continuación).
	Algún momento en 2037	La parte superior del rango depende del límite de las marcas de tiempo de UNIX.
`YEAR[(2|4)]`	70–69 (1970–2069) 1901–2155	Un año. Puede especificar un formato de 2 a 4 dígitos. Cada uno de éstos tiene un rango diferente, como se muestra.

La tabla 8.8 muestra los diferentes tipos de formatos de visualización de `TIMESTAMP`.

Tabla 8.8. Tipos de formato de visualización de `TIMESTAMP`

Tipo especificado	Formato de visualización
`TIMESTAMP`	AAAAMMDDHHMMSS
`TIMESTAMP(14)`	AAAAMMDDHHMMSS
`TIMESTAMP(12)`	AAMMDDHHMMSS
`TIMESTAMP(10)`	AAMMDDHHMM
`TIMESTAMP(8)`	AAAAMMDD
`TIMESTAMP(6)`	AAMMDD
`TIMESTAMP(4)`	AAMM
`TIMESTAMP(2)`	AA

Tipos de cadena

Los tipos de cadena se clasifican en tres grupos. En primer lugar, están las antiguas cadenas planas, es decir, pequeños fragmentos de texto. Se trata de los tipos CHAR (carácter de longitud fija) y VARCHAR (carácter de longitud variable). Puede especificar la anchura de cada una. Las columnas de tipo CHAR se rellenarán con espacios hasta cubrir la anchura máxima, independientemente del tamaño de los datos. (MySQL elimina los espacios que sobran a la derecha de CHARS al recuperarlos y de VARCHARS al almacenarlos.) En un capítulo posterior examinaremos las ventajas y desventajas en términos de espacio y velocidad de estos dos tipos.

En segundo lugar, están los tipos TEXT y BLOB. Estos tipos disponen de varios tamaños. Equivalen a texto largo y datos binarios, respectivamente. Los BLOB son objetos largos binarios. Pueden incluir todo lo que se necesite, por ejemplo, imágenes o sonidos.

En la práctica, las columnas BLOB y TEXT son iguales con la salvedad de que TEXT discrimina entre mayúsculas y minúsculas, y BLOB no lo hace. Como estos tipos de columna pueden contener grandes cantidades de datos, es necesario tener en cuenta una serie de consideraciones, como se verá en un capítulo posterior.

El tercer grupo consta de dos tipos especiales, SET y ENUM. El tipo SET se utiliza para especificar que los valores de esta columna deben proceder de un conjunto de valores dados. Los valores de columna pueden contener más de un valor del conjunto. Cada conjunto especificado puede contener un máximo de 64 elementos.

ENUM es una enumeración. Es muy similar a SET, con la excepción de que las columnas de este tipo sólo pueden tener uno de los valores especificados o NULL y el máximo número de elementos de la enumeración será de 65.535.

En las tablas 8.9 y 8.10 se recogen los tipos de datos de cadena. La tabla 8.9 muestra los tipos de cadena comunes.

Tabla 8.9. Tipos de cadena habituales

Tipo	Rango	Descripción
[NATIONAL] CHAR(M) [BINARY]	De 0 a 255 caracteres	Cadena de longitud fija M, donde M se establece entre 0 y 255. Se trata del tipo predeterminado en MySQL, pero se incluye ya que forma parte del estándar ANSI SQL. La palabra clave BINARY especifica que los datos deberían tratase como si no discriminaran entre mayúsculas y minúsculas. (De manera predeterminada, discrimina entre mayúsculas y minúsculas.)
CHAR	1	Sinónimo de CHAR(1)
[NATIONAL] VARCHAR(M) [BINARY]	De 1 a 255 caracteres	Igual que el anterior, con la excepción que la longitud es variable.

La tabla 8.10 describe los tipos TEXT y BLOB. La longitud de un campo TEXT en caracteres es el tamaño máximo en bytes de los archivos que podrían almacenarse en dicho campo.

Tabla 8.10. Tipos TEXT y BLOB

Tipo	Longitud máxima (caracteres)	Descripción
TINYBLOB	2^8 -1 (es decir, 255)	Un campo BLOB pequeño
TINYTEXT	2^8 -1 (es decir, 255)	Un campo TEXT pequeño
BLOB	2^{16} -1 (es decir, 65.535)	Un campo BLOB de tamaño normal
TEXT	2^{16} -1 (es decir, 65.535)	Un campo TEXT de tamaño normal
MEDIUMBLOB	2^{24}-1 (es decir, 16.777.215)	Un campo BLOB de tamaño intermedio
MEDIUMTEXT	2^{24}-1 ((es decir, 16.777.215)	Un campo TEXT de tamaño intermedio
LONGBLOB	2^{32}-1 (es decir, 4.294.967.295)	Un campo BLOB de tamaño grande
LONGTEXT	2^{32}-1 (es decir, 4.294.967.295)	Un campo TEXT de tamaño grande

La tabla 8.11 muestra los tipos ENUM y SET.

Tabla 8.11. Tipos SET y ENUM

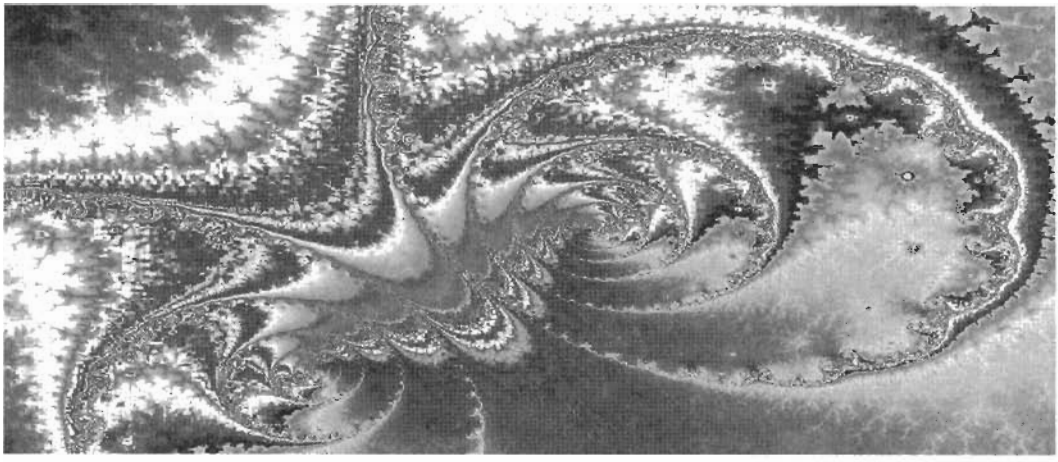
Tipo	Valores máximos del conjunto	Descripción
ENUM('valor1', 'valor2',...)	65.535	Las columnas de este tipo sólo pueden contener uno de los valores listados o NULL.
SET('valor1', 'valor2',...)	64	Las columnas de este tipo pueden contener un conjunto de valores especificados o NULL.

Lecturas adicionales

Si desea obtener más información, puede consultar el manual en línea de MySQL para configurar una base de datos en http://www.mysql.com/.

A continuación

Ahora que ha aprendido a crear usuarios, bases de datos y tablas, puede concentrarse en interactuar con la base de datos. En el siguiente capítulo, veremos cómo introducir datos en tablas, cómo actualizarlos y eliminarlos, y cómo consultar la base de datos.



9

Cómo trabajar con la base de datos de MySQL

En este capítulo vamos a analizar el lenguaje de consulta estructurado (SQL) y su uso para consultar bases de datos. Continuaremos con la base de datos Book-O-Rama. En concreto, veremos cómo insertar, eliminar y actualizar datos y cómo realizar preguntas a la base de datos.

En este capítulo se abordarán los siguientes aspectos:

- Concepto de SQL
- Inserción de datos en la base de datos
- Recuperación de datos de la base de datos
- Combinación de tablas
- Actualización de registros desde tablas
- Modificación de tablas tras su creación
- Eliminación de registros desde la base de datos
- Eliminación de tablas

Comenzaremos por explicar el significado de SQL y por qué resulta útil entender el concepto en el que se basa este lenguaje.

Si no ha creado la base de datos de Book-O-Rama, necesitará hacerlo para poder ejecutar las consultas SQL de este capítulo. En un capítulo anterior encontrará las instrucciones.

¿Qué es SQL?

SQL equivale a lenguaje de consulta estructurado. Se trata del lenguaje estándar para acceder a los sistemas de administración de base de datos (RDBMS). SQL se utiliza para almacenar y consultar datos desde y hasta una base de datos. Se utiliza en sistemas de base de datos como MySQL, Oracle, PostgreSQL, Sybase y Microsoft SQL Server entre otros.

Existe un estándar ANSI de SQL, y los sistemas de bases de datos como MySQL suelen implementarlo. Sin embargo, existen diferencias sutiles entre el SQL estándar y el SQL de MySQL, que en algunos casos está previsto integrar en el estándar y en otros resultan deliberadas. A medida que avancemos, iremos indicando los casos más importantes. En el manual en línea de MySQL se recoge una lista completa de las diferencias entre el SQL de MySQL y el SQL ANSI de cada versión. Esta página se puede encontrar en el siguiente URL así como en otras muchas direcciones:

```
http://www.mysql.com/doc/en/Compatibility.html
```

Puede que haya oído hablar de los lenguajes de definición de datos (DDL), utilizados para definir bases de datos, y de los lenguajes de manipulación de datos (DML), utilizados para consultar bases de datos. SQL trata estos tipos de bases. En un capítulo anterior, vimos la definición de datos (DDL) en SQL, por lo que ya la hemos utilizado. Este lenguaje se utiliza al configurar inicialmente una base de datos. Utilizaremos los aspectos DML de SQL con mucha mayor frecuencia porque nos servirán para almacenar y recuperar datos reales de una base de datos.

Inserción de datos en la base de datos

Para poder trabajar con la base de datos, es necesario almacenar datos en ella. La forma más habitual de llevar a cabo esta operación consiste en utilizar la instrucción `INSERT` de SQL.

Recuerde que los RDBMS contienen tablas, que a su vez contienen filas de datos organizados en columnas. Cada fila de una tabla suele describir un objeto o relación del mundo real y los valores de las columnas para dichas filas almacenan información sobre el mundo real. Podemos utilizar la instrucción `INSERT` para colocar filas de datos dentro de la base de datos.

A continuación, se recoge la sintaxis habitual de una instrucción `INSERT`:

```
INSERT [INTO] tabla [(columna1, columna2, columna3,...)] VALUES
(valor1, valor2, valor3,...);
```

Por ejemplo, para insertar un registro dentro de la tabla `Customers` de Book-O-Rama, puede escribir

```
insert into customers values
   (NULL, "Julie Smith", "25 Oak Street", "Airport West");
```

Como puede observar, hemos sustituido *tabla* por el nombre de la tabla en la que queremos colocar los datos y los *valores* con valores específicos. Los valores de este ejemplo se encierran entre comillas dobles. En MySQL, las cadenas deberían encerrarse siempre entre comillas simples o dobles. (En este libro utilizaremos ambas.) Los números y las fechas no necesitan comillas.

Existen ciertas cosas interesantes en los que fijarse en relación a la instrucción `INSERT`.

Los valores que especificamos se utilizarán para rellenar las columnas de la tabla en orden. Si desea rellenar sólo algunas columnas o si quiere especificarlas en un orden diferente, puede enumerar las deseadas en la parte reservada para las columnas de la instrucción. Por ejemplo:

```
insert into customers (name, city) values
("Melissa Jones", "Nar Nar Goon North");
```

Este enfoque resulta útil si sólo se dispone de datos parciales sobre un registro dado o si algunos campos del registro son opcionales. Puede obtener el mismo efecto con la siguiente sintaxis:

```
insert into customers
set name="Michael Archer",
    address="12 Adderley Avenue",
    city="Leeton";
```

Como puede observar, especificamos un valor NULL para la columna personalizada al agregar a Julie Smith e ignoramos dicha columna al agregar el resto de los clientes. Puede que recuerde que al configurar la base de datos, creamos la columna `customerid` como clave principal de la tabla `Customers` por lo que puede que le resulte extraño. Sin embargo, especificamos el campo como `AUTO_INCREMENT`, lo que significa que si insertamos una fila con un valor NULL o sin ningún valor en este campo, MySQL generará el siguiente número en la secuencia de incremento automática y la insertará sin nuestra intervención, lo cual resulta bastante útil.

También puede insertar varias filas dentro de una tabla de una sola vez. Cada fila debería incluirse en su propio conjunto de corchetes y cada conjunto de corchetes debería separarse por medio de una coma.

Hemos reunido algunos datos de ejemplo para rellenar la base de datos. Se trata de una serie de simples instrucciones `INSERT` que utilizan el enfoque de inserción multifila comentado. La secuencia de comandos que realiza esta tarea podrá encontrarla en el CD que se adjunta al libro, en el archivo correspondiente a esta carpeta. También se recoge en el listado 9.1.

Listado 9.1. book_insert.sql. SQL para rellenar las tablas de Book-O-Rama

```
use books;

insert into customers values
   (NULL, "Julie Smith", "25 Oak Street", "Airport West"),
```

```
    (NULL, "Alan Wong", "1/47 Haines Avenue", "Box Hill"),
    (NULL, "Michelle Arthur", "357 North Road", "Yarraville");

insert into orders values
    (NULL, 3, 69.98, "2000-04-02"),
    (NULL, 1, 49.99, "2000-04-15"),
    (NULL, 2, 74.98, "2000-04-19"),
    (NULL, 3, 24.99, "2000-05-01");

insert into books values
    ("0-672-31697-8", "Michael Morgan", "Java 2 for Professional
Developers", 34.99),
    ("0-672-31745-1", "Thomas Down", "Installing Debian GNU/Linux", 24.99),
    ("0-672-31509-2", "Pruitt, et al.", "Teach Yourself GIMP in 24 Hours",
24.99),
    ("0-672-31769-9", "Thomas Schenk", "Caldera OpenLinux System
Administration Unleashed", 49.99);

insert into order_items values
    (1, "0-672-31697-8", 2),
    (2, "0-672-31769-9", 1),
    (3, "0-672-31769-9", 1),
    (3, "0-672-31509-2", 1),
    (4, "0-672-31745-1", 3);

insert into book_reviews values
    ("0-672-31697-8", "Morgan's book is clearly written and goes well beyond
                       most of the basic Java books out there.");
```

Puede ejecutar esta secuencia de comandos a través de MySQL de la siguiente forma:

```
>mysql -h host -u bookorama -p < book_insert.sql
```

Recuperación de datos de la base de datos

La mula de carga de SQL es la instrucción SELECT. Se utiliza para recuperar datos de una base de datos seleccionando las filas que coinciden con los criterios especificados de una tabla. La instrucción SELECT consta de una gran cantidad de opciones y formas de uso. La forma básica de una instrucción SELECT es la siguiente:

```
SELECT elementos
FROM tablas
[ WHERE condición ]
[ GROUP BY tipo_grupo ]
[ HAVING definición_de_dónde ]
[ ORDER BY tipo_orden ]
[ LIMIT criterios_límite ] ;
```

Comentaremos cada una de estas cláusulas de la instrucción. En primer lugar, vamos a examinar una consulta sin ningún parámetro opcional que selecciona algu-

nos elementos de una tabla dada. Por regla general, se trata de columnas de la tabla. (También pueden ser los resultados de cualquier expresión de MySQL. Analizaremos algunos de los más útiles en una sección posterior.) La siguiente consulta enumera los contenidos de la columna `name` y `city` de la tabla `Customers`:

```
select name, city
from customers;
```

Esta consulta devuelve el siguiente resultado, siempre y cuando haya introducido los datos de ejemplo del listado 9.1 y las dos instrucciones INSERT de ejemplo:

```
+---------+-------+-----------------+
| name            |city               |
+---------+-------+-----------------+
| Julie Smith     |Airport West       |
| Alan Wong       |Box Hill           |
| Michelle Arthur|Yarraville         |
| Melissa Jones   |Nar Nar Goon North|
| Michael Archer |Leeton             |
+---------+-------+-----------------+
```

Como puede observar, tenemos una tabla que contiene los elementos seleccionados (`name` y `city`) de la tabla especificada, `Customers`. Los datos se muestran para todas las filas de la tabla `Customers`.

Puede especificar tantas columnas como desee de una tabla enumerándolas tras la palabra clave `select`. También puede especificar algunos otros elementos. Uno útil es el operador comodín, `*`, que selecciona todas las columnas de la tabla o tablas especificadas. Por ejemplo, para recuperar todas las columnas y todas las filas de la tabla `order_items`, utilizaríamos

```
select *
from order_items;
```

que devolverá el siguiente resultado:

```
+-----+-------+-------------+----------+
| orderid | isbn          | quantity |
+-----+-------+-------------+----------+
|       1 | 0-672-31697-8 |        2 |
|       2 | 0-672-31769-9 |        1 |
|       3 | 0-672-31769-9 |        1 |
|       3 | 0-672-31509-2 |        1 |
|       4 | 0-672-31745-1 |        3 |
+-----+-------+-------------+----------+
```

Recuperación de datos con criterios específicos

Para poder acceder a un subconjunto de filas de una tabla, necesitamos especificar algunos criterios de selección. Para ello se puede utilizar la cláusula WHERE. Por ejemplo,

```
select *
```

```
from orders
where customerid = 3;
```

seleccionará todas las columnas de la tabla de pedidos pero sólo las filas con `customerid` de 3.

La cláusula `WHERE` especifica los criterios utilizados para seleccionar filas concretas.

En este caso, hemos seleccionado las filas con un Id. de cliente de 3. El signo igual se utiliza para probar la igualdad (tenga en cuenta que es diferente con respecto a PHP y que resulta sencillo confundirlos cuando se utilizan juntos).

Además de la igualdad, MySQL admite un conjunto completo de operadores de comparación y de expresiones regulares. En la tabla 9.1 se recogen las más comunes.

Tenga en cuenta que no se trata de una lista completa (si necesita algún operador no enumerado, consulte el manual de MySQL).

Tabla 9.1. Operadores de comparación útiles para las cláusulas WHERE

Operador	Nombre (si resulta de aplicación)	Ejemplo	Descripción
=	igualdad	`customerid = 3`	Comprueba si dos valores son iguales
>	mayor que	`importe > 60.00`	Comprueba si un valor es mayor que otro
<	menor que	`importe < 60.00`	Comprueba si un valor es menor que otro
>=	mayor o igual que	`importe >= 60.00`	Comprueba si un valor es mayor o igual que otro
<=	menor o igual que	`importe <= 60.00`	Comprueba si un valor es menor o igual que otro
!= o <>	diferente	`cantidad != 0`	Comprueba si dos valores no son iguales
`IS NOT NULL`	n/d	`dirección is not null`	Comprueba si el campo contiene un valor
`IS NULL`	n/d	`dirección es null`	Comprueba si el campo no contiene un valor

Operador	Nombre (si resulta de aplicación)	Ejemplo	Descripción
`BETWEEN`	n/d	`importe between`	Comprueba si un valor es mayor o igual que un valor mínimo o menor o igual que un valor máximo
`IN`	n/d	`ciudad in ("Carlton", "Moe")`	Comprueba si un valor se incluye dentro de un conjunto dado
`NOT IN`	n/d	`ciudad in ("Carlton", "Moe")`	Comprueba si un valor no se incluye en un conjunto
`LIKE`	correspondencia de patrón	`nombre like`	Comprueba si un valor coincide con un patrón utilizando simples patrones de correspondencia de SQL
`NOT LIKE`	correspondencia de patrones	`nombre not like ("Fred %")`	Comprueba si un valor no coincide con un patrón
`REGEXP`	expresión regular	`nombre regexp`	Comprueba si un valor coincide con una expresión regular

Las últimas tres entradas de la tabla hacen referencia a `LIKE` y `REGEXP`. Se trata de operadores para la búsqueda de correspondencias en forma de patrón.

`LIKE` utiliza correspondencias de patrón simples de SQL. Los patrones pueden consistir en texto normal más el carácter `%` (porcentaje) para indicar una coincidencia con cualquier número de caracteres y el carácter `_` (guión bajo) para sustituir a un solo carácter en la búsqueda de correspondencias.

La palabra clave `REGEXP` se utiliza para las búsquedas de correspondencias con expresiones regulares. MySQL utiliza expresiones regulares de POSIX. En lugar de `REGEXP`, puede utilizar `RLIKE` ya que son equivalentes. Las expresiones regulares de POSIX también se utilizan en PHP. Puede leer más al respecto en un capítulo anterior.

Puede probar una gran cantidad de criterios de esta forma y combinar los operadores con `AND` y `OR`. Por ejemplo:

```
select *
```

```
from orders
where customerid = 3 or customerid=4;
```

Recuperación de datos desde varias tablas

A menudo, para responder a una pregunta desde la base de datos, es necesario utilizar datos incluidos en más de una tabla. Por ejemplo, si quiere saber los clientes que realizaron pedidos durante el presente mes, necesitará buscar en la tabla `Customers` y en la tabla `Orders`. Si además quiere saber qué artículos pidieron, necesitará consultar la tabla `Order_Items`.

Estos datos se encuentran distribuidos en varias tablas porque hacen referencia a objetos independientes del mundo real. Éste es uno de los principios del buen diseño de bases de datos comentado en un capítulo anterior.

Para reunir esta información en SQL, debe aplicar una operación denominada combinación. Esta operación consiste en unir dos o más tablas para seguir la relación entre los datos. Por ejemplo, si deseamos ver los pedidos realizados por Julie Smith, necesitaremos examinar la tabla `Customers` para buscar el identificador de Julie y, a continuación, mirar en la tabla `Orders` para buscar los pedidos que tengan dicho identificador.

Aunque las combinaciones resultan conceptualmente sencillas, son una de las partes más sutiles y complejas de SQL. En MySQL se implementan varios tipos diferentes de combinación y cada uno de ellos se utiliza para una finalidad diferente.

Combinaciones sencillas de dos tablas

Comenzaremos por examinar algunas instancias de SQL para consultar la información sobre Julie Smith de la que acabamos de hablar:

```
select orders.orderid, orders.amount, orders.date
from customers, orders
where customers.name = 'Julie Smith'
and customers.customerid = orders.customerid;
```

El resultado de esta consulta es

```
+---------+--------+------------+
| orderid | amount | date       |
+---------+--------+------------+
|       2 |  49.99 | 2000-04-15 |
+---------+--------+------------+
```

Tenemos que fijarnos en varios elementos.

En primer lugar, como es necesario obtener información de las dos tablas para contestar a esta consulta, se han listado ambas.

También se ha especificado el tipo de combinación, posiblemente sin ser consciente de ello. La coma situada entre los nombres de las tablas equivale a escribir `INNER JOIN` o `CROSS JOIN`. Éste es un tipo de combinación al que a veces también

se hace referencia como combinación completa o el producto cartesiano de tablas. Significa "toma las tablas indicadas y crea otra con ellas; esta tabla debería constar de una fila para cada combinación posible de filas de cada una de las tablas indicadas, ya tenga sentido o no". En otras palabras, obtenemos otra tabla, que incluye todas las filas de la tabla `Customers` que se correspondan con cada fila de la tabla `Orders`, independientemente de si un cliente realizó un pedido concreto.

En muchas ocasiones el resultado no tiene mucho sentido. Por regla general, lo que queremos es ver las filas coincidentes, es decir, los pedidos realizados por un cliente dado y la información correspondiente del cliente.

Para lograrlo se utiliza una *condición de combinación* en la cláusula `WHERE`. Se trata de un tipo especial de instrucción condicional que explica qué atributos muestran la relación entre dos tablas. En este caso, la condición de combinación es la siguiente:

```
customers.customerid = orders.customerid
```

que indica a MySQL que ponga filas en la tabla de resultados si el `CustomerId` de la tabla `Customers` coincide con el `CustomerId` de la tabla `Orders`.

Al agregar esta condición de combinación en la consulta, hemos creado otro tipo de combinación llamada *combinación de igualdad*.

La notación de punto utilizada deja claro a qué tabla pertenece una columna dada, es decir, `customers.customerid` hace referencia a la columna `customerid` de la tabla `Customers` y `order.customerid` hace referencia a la columna `customerid` de la tabla `Orders`.

Esta notación es obligatoria si el nombre de una columna resulta ambiguo, es decir, si aparece en más de una tabla.

También se puede utilizar para distinguir los nombres de columnas utilizados en diferentes bases de datos. En este ejemplo, se ha utilizado la notación tabla.columna. Puede especificar la base de datos con una notación basededatos.tabla.columna, por ejemplo, para probar una condición como

```
books.orders.customerid = other_db.orders.customerid
```

Sin embargo, puede utilizar la notación de punto para todas las referencias de columna de una consulta. Esta opción es una buena idea, en especial cuando las consultas comienzan a complicarse. Su uso no es obligatorio en MySQL pero las consultas resultarán mucho más legibles y sencillas de mantener. Como puede apreciar, hemos utilizado esta convención en el resto de la consulta anterior, por ejemplo, al utilizar la condición

```
customers.name = 'Julie Smith'
```

El nombre de columna sólo tiene lugar en la tabla de clientes por lo que no necesitaremos especificarlo, aunque resultará más claro.

Combinación de varias tablas

Cuando se combinan más de dos tablas la operación se complica. Como regla general, en las condiciones de combinación se combinan las tablas de dos en dos.

Es como seguir las relaciones entre los datos de una tabla a otra tabla a otra tabla, etc.

Por ejemplo, si queremos saber qué clientes han pedido libros sobre Java (por ejemplo, para poder enviarles información sobre un nuevo libro), necesitaremos rastrear dichas relaciones a través de bastantes tablas.

Tendremos que buscar clientes que hayan realizado al menos un pedido que incluya un libro de Java. Para acceder a la tabla `Orders` desde la tabla `Customers` podemos utilizar la columna `customerid` como hicimos anteriormente. Para acceder a la tabla `Order_Items` desde la tabla `Orders`, podemos utilizar la columna `orderid`. Para acceder a un libro específico de la tabla `Books` desde la tabla `Order_Items`, podemos utilizar el ISBN. Tras establecer estos enlaces, podemos buscar libros que incluyan el término Java en el título y recuperar los nombres de los clientes que hayan comprado alguno de ellos.

Examinemos la consulta que realiza estas operaciones.

```
select  customers.name
from customers, orders, order_items, books
where customers.customerid = orders.customerid
and orders.orderid = order_items.orderid
and  order_items.isbn  =  books.isbn
and  books.title  like  '%Java%';
```

Esta consulta devuelve el siguiente resultado:

```
+--------+--------+
| name            |
+--------+--------+
| Michelle  Arthur |
+--------+--------+
```

Fíjese en que seguimos los datos a través de cuatro tablas diferentes y, para hacerlo con una combinación de igualdad, necesitamos tres condiciones de combinación diferentes. Por regla general, se necesita una condición de combinación para cada par de tablas que se desee combinar, por lo que el total de condiciones de combinación será igual al número de tablas que se desean combinar menos uno. Esta regla puede resultar útil para consultas de depuración que no funcionen. Compruebe las condiciones de combinación y asegúrese de haber seguido la ruta desde lo que sabe a lo que desea saber.

Búsqueda de filas que no coincidan

El segundo tipo principal de consultas que utilizaremos en MySQL son las combinaciones por la izquierda.

En los ejemplos anteriores, sólo se han incluido las filas coincidentes entre las tablas. En ocasiones, se necesita recuperar las filas que no coincidan. Por ejemplo, los clientes que no hayan realizado nunca un pedido o los libros que no se hayan pedido nunca.

La forma más sencilla de responder a este tipo de cuestión en MySQL es utilizar una combinación por la izquierda. En estas combinaciones se comparan filas a partir

de una condición de combinación dada entre dos tablas. Si no existen filas coincidentes en la tabla de la derecha, se agregará una fila al resultado que contenga valores `NULL` en las columnas de la derecha.

Veamos un ejemplo:

```
select customers.customerid, customers.name, orders.orderid
from customers left join orders
on customers.customerid = orders.customerid;
```

La consulta SQL utiliza una combinación por la izquierda para combinar `Customers` con `Orders`. La combinación por la izquierda utiliza una sintaxis un tanto diferente en la condición de combinación. En este caso, dicha combinación va en una cláusula `ON` especial de la instrucción SQL.

El resultado de la consulta es el siguiente:

```
+------+-----+---------------+---------+
| customerid |name             | orderid |
+------+-----+---------------+---------+
|          1 |Julie  Smith     |       2 |
|          2 |Alan  Wong       |       3 |
|          3 |Michelle  Arthur|       1 |
|          3 |Michelle  Arthur|       4 |
|          4 |Melissa  Jones   |    NULL |
|          5 |Michael  Archer  |    NULL |
+------+-----+---------------+---------+
```

Este resultado muestra que no existen Id. de pedido coincidentes para Melissa Jones y Michael Archer porque dichos Id. son `NULL`.

Si sólo deseamos ver los clientes que no hayan realizado ningún pedido, podemos recuperarlos utilizando dichos valores `NULL` en el campo de clave principal de la tabla de la derecha (en este caso `orderid`).

```
select customers.customerid, customers.name
from customers left join orders
using (customerid)
where orders.orderid is null;
```

El resultado será:

```
+------------+---------------+
| customerid | name          |
+------------+---------------+
|          4 | Melissa  Jones |
|          5 | Michael  Archer|
+------------+---------------+
```

Como observará, hemos utilizado una sintaxis diferente para la condición de combinación en este ejemplo. Las combinaciones por la izquierda admiten la sintaxis `ON` utilizada en el primer ejemplo o la sintaxis `USING` del segundo. Tenga en cuenta que la sintaxis `USING` no especifica la tabla de la que procede el atributo de combinación. Por esta razón, las columnas de las tablas deben tener el mismo nombre si queremos utilizar `USING`.

Uso de otros nombres para designar tablas: los alias

A menudo resulta práctico, y a veces necesario, poder utilizar otros nombres para hacer referencia a las tablas. Es lo que se conoce como alias. Los alias se pueden crear al principio de una columna y utilizarlos a lo largo de todas ellas. Suelen resultar prácticos por cuestiones de brevedad. Por ejemplo, examine la consulta que escribimos anteriormente pero escrita con alias:

```
select c.name
from customers as c, orders as o, order_items as oi, books as b
where c.customerid = o.customerid
and o.orderid = oi.orderid
and oi.isbn = b.isbn
and b.title like '%Java%';
```

Al declarar las tablas que vamos a utilizar, agregamos una cláusula `AS` para establecer el alias asociado a dicha tabla. También podemos utilizar alias para columnas. Volveremos sobre este tema al analizar las funciones de agregación.

El uso de alias es obligatorio cuando se trata de combinar una tabla consigo misma. Suena más complicado y extraño de lo que parece. Resulta útil, por ejemplo, si queremos buscar filas de la misma tabla con valores comunes.

Si queremos buscar clientes que vivan en la misma ciudad (por ejemplo, para establecer un sitio de lectura) podemos asignar a la tabla (`Customers`) dos alias diferentes.

```
select c1.name, c2.name, c1.city
from customers as c1, customers as c2
where c1.city = c2.city
and c1.name != c2.name;
```

Lo que estamos haciendo básicamente es simular la existencia de dos tablas `Customers` diferentes, `c1` y `c2`, y realizar la combinación sobre la columna `City`. Como observará también necesitamos la segunda condición, `c1.name != c2.name`, para no recuperar todos los clientes por tener el mismo nombre.

Resumen de los tipos de combinación

En la tabla 9.2 se recogen los diferentes tipos de combinaciones vistos. Existen otros, pero los incluidos en la tabla son los más importantes para nuestros propósitos.

Tabla 9.2. Tipos de combinación en MySQL

Nombre	Descripción
Producto cartesiano	Se combinan todas las filas de todas las tablas. Se utiliza incluyendo una coma entre los nombres de las tablas y sin una cláusula `WHERE`.
Combinación total	Igual a la anterior.

Nombre	Descripción
Combinación cruzada	Igual que las anteriores. También se puede utilizar especificando las palabras clave `CROSS JOIN` entre los nombres de las tablas combinadas.
Combinación interna	Semánticamente, resulta equivalente a la coma. También se puede especificar utilizando las palabras clave `INNER JOIN` Sin una condición `WHERE` resulta equivalente a una combinación total. Por regla general, se especifica una condición `WHERE` para obtener una auténtica combinación interna.
Combinación de igualdad	Utiliza una expresión condicional con un signo = para buscar filas coincidentes entre las diferentes tablas de la combinación. En SQL, equivale a una combinación con una cláusula `WHERE`.
Combinación por la izquierda	Busca filas coincidentes entre tablas y rellena las filas no coincidentes con valores `NULL`. En SQL, se utiliza con las palabras clave `LEFT JOIN`. Se utiliza para buscar valores que falten. También se puede utilizar la combinación `RIGHT JOIN` de manera equivalente.

Recuperación de datos con un orden dado

Si desea mostrar en un orden concreto las filas recuperadas en una consulta, puede utilizar la cláusula `ORDER BY` de la instrucción `SELECT`.

Esta función resulta práctica para presentar los resultados en un formato legible para humanos.

La cláusula `ORDER BY` se utiliza para ordenar las filas de una o varias columnas listadas en la cláusula `SELECT`. Por ejemplo,

```
select name, address
from customers
order by name;
```

Esta consulta devuelve los nombres y las direcciones en orden alfabético por el nombre:

```
+----------------+--------------------+
| name           | address            |
+----------------+--------------------+
| Alan Wong      | 1/47 Haines Avenue |
| Julie Smith    | 25 Oak Street      |
| Melissa Jones  |                    |
| Michael Archer | 12 Adderley Avenue |
| Michelle Arthur| 357 North Road     |
+----------------+--------------------+
```

(Al utilizar el formato de nombre y apellido, los registros se ordenan por el nombre. Si desea ordenarlos por el apellido, necesitará utilizar dos campos diferentes.)

El orden predeterminado es ascendente (de la a a la z o numéricamente hacia arriba). Puede especificar este orden mediante la palabra clave `ASC`:

```
select name, address
from customers
order by name asc;
```

También puede utilizar la palabra clave `DESC` (descendente) para aplicar el orden inverso:

```
select name, address
from customers
order by name desc;
```

Puede ordenar los registros por más de una columna. Además, puede utilizar alias de columna o incluso sus números de posición (por ejemplo, 3 es la tercera columna de la tabla) en lugar de nombres.

Agrupación y agregación de datos

A menudo nos interesa saber cuántas filas contiene un conjunto dado o el valor promedio de una columna (por ejemplo, el promedio en metálico por pedido). MySQL incorpora un conjunto de funciones de agregación que resultan útiles para responder a una consulta.

Estas funciones de agregación se pueden aplicar a una tabla en su conjunto o a grupos de datos dentro de una tabla. Lo normal es utilizar las listadas en la tabla 9.3.

Tabla 9.3. Funciones de agregación de MySQL

Nombre	Descripción
`AVG(columna)`	Promedio de los valores de la columna especificada.
`COUNT(elementos)`	Si especifica una columna, devolverá el número de valores que no sean `NULL` de la columna. Si agrega la palabra `DISTINC` delante del nombre de la columna, obtendrá un contador de los valores diferentes de dicha columna únicamente. Si especifica `COUNT(*)`, obtendrá un recuento de fila independientemente de los valores `NULL`.
`MIN(columna)`	Mínimo de los valores de la columna especificada.
`MAX(columna)`	Máximo de los valores de la columna especificada.
`STD(columna)`	Desviación estándar de los valores de la columna especificada.

Nombre	Descripción
`STDDEV(columna)`	Igual que `STD(columna)`.
`SUM(columna)`	Suma de los valores de la columna especificada.

Vamos a examinar algunos ejemplos, comenzando por el mencionado anteriormente. Podemos calcular la media total de un pedido de la siguiente forma:

```
select  avg(amount)
from orders;
```

El resultado será el siguiente:

```
+---------------+
| avg(amount)   |
+---------------+
|    54.985002  |
+---------------+
```

Para obtener información detallada, podemos utilizar la cláusula `GROUP BY`. Esta cláusula nos permite ver la media de pedidos por grupos (por ejemplo, por el número de cliente). De esta forma sabremos qué clientes realizan los mayores pedidos.

```
select  customerid,  avg(amount)
from orders
group by customerid;
```

Al utilizar una cláusula `GROUP BY` con una función de agregación, se modifica el comportamiento de la función. En lugar de obtener la media de las cantidades de pedidos de toda la tabla, esta consulta recupera la media de las cantidades de pedidos de cada cliente (o, para ser más específico, para cada `customerid`):

```
+------------+---------------+
| customerid | avg(amount)   |
+------------+---------------+
|          1 |    49.990002 |
|          2 |    74.980003 |
|          3 |    47.485002 |
+------------+---------------+
```

Antes utilizar la agrupación y agregación de funciones, tenga en cuenta que en ANSI SQL, si utiliza una función de agregación o la cláusula `GROUP BY`, lo único que puede aparecer en la cláusula `SELECT` son las funciones de agregación y las columnas designadas en la cláusula `GROUP BY`. Así mismo, si quiere utilizar una columna en una cláusula `GROUP BY`, debe listarla en la cláusula `SELECT`.

MySQL brinda una mayor libertad en este sentido ya que admite sintaxis extendida, lo que nos permite dejar elementos fuera de la cláusula `SELECT` si no los deseamos.

Además de agrupar y agregar datos, podemos probar los resultados de una operación de agregación utilizando la cláusula `HAVING`. Ésta se coloca directamente detrás

de la cláusula GROUP BY y es como una cláusula WHERE aplicada únicamente a grupos y agregados.

En nuestro ejemplo anterior, si queremos saber qué clientes tienen pedidos cuyo importe medio supere los 50 dólares, podemos utilizar la siguiente consulta:

```
select customerid, avg(amount)
from orders
group by customerid
having avg(amount) > 50;
```

Fíjese en que la cláusula HAVING se aplica a los grupos. Esta consulta devolverá el siguiente resultado:

```
+------------+-------------+
| customerid | avg(amount) |
+------------+-------------+
|          2 |   74.980003 |
+------------+-------------+
```

Cómo escoger las filas que recuperar

Una cláusula de la instrucción SELECT que puede resultar especialmente útil en aplicaciones Web es la cláusula LIMIT. Esta cláusula se utiliza para especificar qué filas deberían devolverse desde el resultado. Toma dos parámetros: el número de fila desde la que empezar y el número de filas que devolver.

La siguiente consulta ilustra el uso de LIMIT:

```
select name
from customers
limit 2, 3;
```

Esta consulta se puede leer como "seleccionar nombre de clientes y devolver tres filas en el resultado, comenzando en la fila número 2". Fíjese en que los números de fila están indexados en cero, es decir, que la primera fila del resultado es la fila número cero.

Esto resulta muy útil en aplicaciones Web. Por ejemplo, cuando el cliente está examinando los productos de un catálogo y quiere ver 10 por página.

Actualización de registros en la base de datos

Además de recuperar datos desde una base de datos, también resulta normal modificarlos. Por ejemplo, podemos aumentar los precios de los libros incluidos en la base datos. Para ello, podemos utilizar una instrucción UPDATE.

La forma habitual de una instrucción UPDATE es la siguiente:

```
UPDATE nombretabla
SET columna1=expresión1,columna2=expresión2,...
```

```
[WHERE condición]
[LIMIT número]
```

La idea consiste en actualizar la tabla llamada *nombretabla*, estableciendo cada columna mencionada a la expresión apropiada. Puede limitar la aplicación de la operación `UPDATE` a filas concretas mediante una cláusula `WHERE` y restringir el número total de filas con una cláusula `LIMIT`.

Vamos a examinar algunos ejemplos.

Si desea incrementar los precios de todos los libros un 10%, puede utilizar una instrucción UPDATE con una cláusula `WHERE`:

```
update books
set  price=price*1.1;
```

Si quiere cambiar una sola fila (por ejemplo, para actualizar una dirección de cliente), puede hacerlo de la siguiente forma:

```
update customers
set address = '250 Olsens Road'
where customerid = 4;
```

Alteración de tablas tras su creación

Además de actualizar filas, podemos alterar la estructura de las tablas dentro de una base de datos. Para ello, se puede utilizar la instrucción `ALTER TABLE`. La forma básica de esta instrucción es la siguiente:

```
ALTER TABLE nombretabla alteración [, alteración ...]
```

Fíjese en que en ANSI SQL sólo puede realizar una alteración por cada instrucción `ALTER TABLE`, pero MySQL permite realizar todas las que desee. Cada cláusula de alteración se puede utilizar para cambiar diferentes aspectos de la tabla.

En la tabla 9.4 se recogen los diferentes tipos de alteración que se puede realizar con esta instrucción.

Tabla 9.4. Cambios posibles con la instrucción ALTER TABLE

Sintaxis	Descripción
`ADD [COLUMN] descripción_columna [FIRST | AFTER columna ]`	Agrega una nueva columna en la ubicación especificada. Si no se especifica, la columna irá al final. Fíjese en que `descripción_columna` necesita un nombre y un tipo, como en una instrucción `CREATE`.
`ADD [COLUMN] (descripción_columna, descripción_columna,...)`	Agrega una o varias columnas nuevas al final de la tabla.

Sintaxis	Descripción
`ADD INDEX [índice] (columna,...)`	Agrega un índice a la tabla en la columna o columnas especificadas.
`ADD PRIMARY KEY (columna,...)`	Convierte la columna o columnas especificadas en la clave principal de la tabla.
`ADD UNIQUE [índice] (columna,...)`	Agrega un índice único a la tabla en una columna o columnas especificadas.
`ALTER [COLUMN] columna {SET DEFAULT valor | DROP DEFAULT}`	Agrega o elimina un valor predeterminado de una columna dada.
`CHANGE [COLUMN] columna descripción_nueva_columna`	Cambia la columna llamada `columna` para que incluya la descripción. Observe que se puede utilizar para cambiar el nombre de la columna porque `descripción_columna` incluye un nombre.
`MODIFY [COLUMN] descripción_columna`	Similar a `CHANGE`. Se puede utilizar para cambiar tipos de columna, no nombres.
`DROP [COLUMN] columna`	Elimina columnas con nombre.
`DROP PRIMARY KEY`	Elimina el índice principal (pero no la columna).
`DROP INDEX índice`	Elimina el índice con nombre.
`RENAME [AS] nombre_nueva_tabla`	Cambia el nombre de una tabla.

Vamos a examinar algunos usos comunes de `ALTER TABLE`. Algo que suele ocurrir con frecuencia es darse cuenta de que no se ha asignado el ancho suficiente a una columna para albergar los datos. Por ejemplo, en la tabla `Customers`, hemos asignado30 caracteres de longitud a los nombres. Al recibir datos, puede que descubramos que algunos nombres son demasiado largos y quedan cortados. Podemos solucionar este problema cambiando los tipos de datos de las columnas para aumentar su tamaño a 45 caracteres de largo.

```
alter table customers
modify name char(45) not null;
```

Otra cosa que suele ocurrir de manera habitual es la necesidad de agregar una columna. Imagine que se ha introducido un impuesto local y que Book-O-Rama necesita agregarlo al total del pedido y realizar su seguimiento por separado. Podemos agregar una columna para el impuesto en la tabla `Orders` de la siguiente forma:

```
alter table orders
add tax float(6,2) after amount;
```

Otra necesidad que suele surgir de forma habitual es la de eliminar una columna. Podemos suprimir la columna que acabamos de agregar de la siguiente forma:

```
alter table orders
drop tax;
```

Eliminación de registros de la base de datos

La operación de eliminar filas de la base de datos resulta sencilla.

Para ello, puede utilizar la instrucción DELETE, que suele presentar este aspecto:

```
DELETE FROM tabla
[WHERE condición] [LIMIT número]
```

Si escribe

```
DELETE FROM tabla;
```

en solitario, se eliminarán todas las filas de una tabla, por lo que conviene tener cuidado. Lo normal es eliminar filas concretas, para lo cual se pueden especificar en una cláusula WHERE.

Por ejemplo, podemos eliminar filas si un libro no estuviera disponible o un cliente concreto llevara mucho tiempo si realizar pedidos y quisiera limpiar un poco la base de datos.

```
delete from customers
where customerid=5;
```

La cláusula LIMIT se puede utilizar para limitar el número máximo de filas que se eliminan.

Eliminación de tablas

En ocasiones se necesita eliminar una tabla entera. Para ello se puede utilizar la instrucción DROP TABLE. Esta operación resulta muy sencilla y presenta este aspecto.

```
DROP TABLE tabla;
```

Esta instrucción eliminará todas las filas de la tabla y la propia tabla, por lo que es necesario tener cuidado.

Eliminación de una base de datos entera

Podemos ir incluso más allá y eliminar una base de datos completa con una instrucción DROP DATABASE, que presenta un aspecto parecido al siguiente:

```
DROP DATABASE base de datos;
```

Esta instrucción elimina todas las filas, todos los índices y la propia base de datos, por lo no es necesario indicar el cuidado que hay que poner al utilizarla.

Lecturas adicionales

En este capítulo, hemos pasado revista a los elementos más comunes de SQL que se utilizan al interactuar con una base de datos MySQL. En los siguientes dos capítulos, examinaremos cómo conectar MySQL y PHP para poder acceder a su base de datos desde la Web. También exploraremos algunas técnicas avanzadas de MySQL.

Si desea saber más sobre SQL, puede consultar el estándar ANSI SQL. Lo encontrará en:

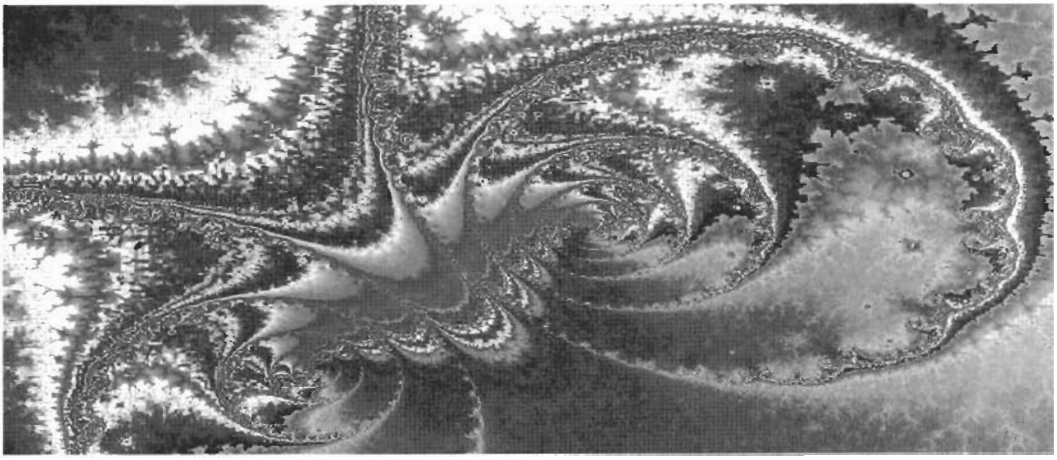
```
http://www.ansi.org/
```

Si desea obtener más detalles sobre las extensiones de MySQL para ANSI SQL puede consultar el sitio Web MySQL:

```
http://www.mysql.com
```

A continuación

En el siguiente capítulo veremos cómo permitir el acceso a Book-O-Rama desde la Web.



10
Acceso a la base de datos de MySQL desde la Web con PHP

En un capítulo anterior, al trabajar con PHP, utilizamos un archivo sin procesar para almacenar y recuperar datos. En dicho capítulo, indicamos que los sistemas de base de datos relacionales facilitan enormemente las tareas de almacenamiento y recuperación de datos y resultan mucho más eficientes en una aplicación Web. A continuación, tras trabajar con MySQL, podemos comenzar a conectar esta base de datos a una interfaz de usuario basada en la Web.

En este capítulo, explicaremos cómo acceder a la base de datos Book-O-Rama desde la Web utilizando PHP. Aprenderá a leer una base de datos y a escribir en ella, y a filtrar datos de entrada potencialmente peligrosos.

A continuación, se recogen los aspectos básicos que se abordarán:

- Funcionamiento de las arquitecturas de base de datos
- Pasos básicos para consultar una base de datos desde la Web
- Cómo configurar una conexión
- Cómo obtener información sobre bases de datos disponibles
- Selección de una base de datos que utilizar
- Cómo consultar una base de datos
- Recuperación de los resultados de consulta
- Desconexión de la base de datos
- Cómo colocar nueva información en la base de datos

- Otras funciones útiles de PHP y MySQL
- Uso de una interfaz genérica de base de datos: PEAR DB
- Otras interfaces de base de datos de PHP

Funcionamiento de las arquitecturas de base de datos

En un capítulo anterior se indicaron las líneas básicas del funcionamiento de las arquitecturas de bases de datos Web. A continuación se repiten dichos pasos para ayudarle a recordar:

1. Un usuario de navegador Web envía una petición HTTP solicitando una página Web dada. Por ejemplo, el usuario podría solicitar todos los libros escritos por Michael Morgan que tenga Book-O-Rama, utilizando un formulario HTML. Los resultados de la búsquedas se almacenan en results.php.
2. El servidor Web recibe la petición de results.php, recupera el archivo y lo pasa al motor de PHP para su procesamiento.
3. El motor de PHP comienza a analizar la secuencia de comandos. Dentro de la secuencia de comandos hay un comando que establece la conexión a la base de datos y ejecuta una consulta (realiza la búsqueda de libros). PHP abre una conexión al servidor MySQL y remite la consulta pertinente.
4. El servidor MySQL recibe la consulta de la base de datos y la procesa. A continuación, envía los resultados (una lista de libros) al motor de PHP.
5. El motor de PHP termina de ejecutar la secuencia de comandos, lo que suele implicar la aplicación de formato a los resultados en HTML. Seguidamente, devuelve el código HTML resultante al servidor Web.
6. El servidor Web devuelve el código HTML al navegador donde el usuario puede ver la lista de los libros solicitados.

Ahora tenemos una base de datos MySQL, por lo que podemos escribir el código PHP para realizar los pasos anteriores. Comenzaremos por el formulario de búsqueda.

Se trata de un sencillo formulario escrito en HTML. En el listado 10.1 se recoge su código:

Listado 10.1. search.html. Página de búsqueda de base de datos de Book-O-Rama

```
<html>
<head>
  <title>Book-O-Rama Catalog Search</title>
```

```
</head>

<body>
 <h1>Book-O-Rama Catalog Search</h1>

 <form action="results.php" method="post">
    Choose Search Type:<br />
    <select name="searchtype">
       <option value="author">Author</option>
       <option value="title">Title</option>
       <option value="isbn">ISBN</option>
    </select>
    <br />
    Enter Search Term:<br />
    <input name="searchterm" type="text">
    <br />
    <input type="submit" value="Search">
 </form>

</body>
</html>
```

El código de este formulario resulta bastante claro. En la figura 10.1 se muestra el resultado.

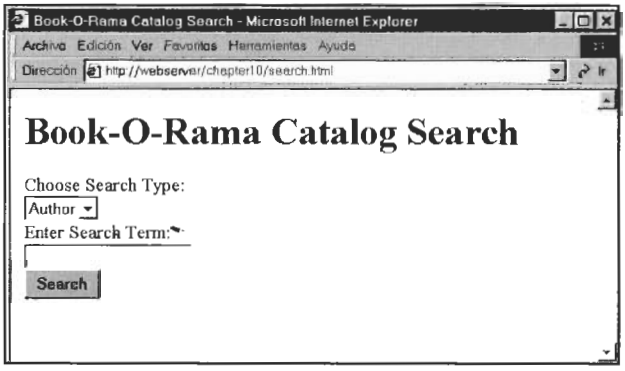


Figura 10.1. El formulario de búsqueda resulta bastante general por lo que se puede buscar un libro a partir de su título, autor o ISBN

La secuencia de comandos que se invocará al pulsar el botón **Search** es results.php. En el listado 10.2 se incluye el código completo.

En este capítulo veremos qué realiza esta secuencia de comandos y su funcionamiento.

Listado 10.2. result.php. Esta secuencia de comandos recupera los resultados de nuestra base de datos MySQL y les aplica formato para su visualización

```
<html>
```

```
<head>
  <title>Book-O-Rama  Search  Results</title>
</head>
<body>
<h1>Book-O-Rama Search Results</h1>
<?php
  // cree nombres de variables cortos
  $searchtype=$HTTP_POST_VARS['searchtype'];
  $searchterm=$HTTP_POST_VARS['searchterm'];

  $searchterm= trim($searchterm);

  if (!$searchtype || !$searchterm)
  {
    echo 'You have not entered search details.Please go back and try
again.';
    exit;
  }

  $searchtype  =  addslashes($searchtype);
  $searchterm = addslashes($searchterm);

  @ $db = mysql_pconnect('localhost', 'bookorama', 'bookorama123');

  if (!$db)
  {
    echo 'Error: Could not connect to database.  Please try again later.';
    exit;
  }

  mysql_select_db('books');
  $query = "select * from books where ".$searchtype." like
'%".$searchterm."%'";
  $result = mysql_query($query);

  $num_results = mysql_num_rows($result);

  echo '<p>Number of books found: '.$num_results.'</p>';

  for ($i=0; $i <$num_results; $i++)
  {
    $row = mysql_fetch_array($result);
    echo '<p><strong>'.($i+1).'. Title: ';
    echo htmlspecialchars(stripslashes($row['title']));
    echo '</strong><br />Author: ';
    echo stripslashes($row['author']);
    echo '<br />ISBN: ';
    echo stripslashes($row['isbn']);
    echo '<br />Price: ';
    echo stripslashes($row['price']);
    echo '</p>';
  }
?>

</body>
</html>
```

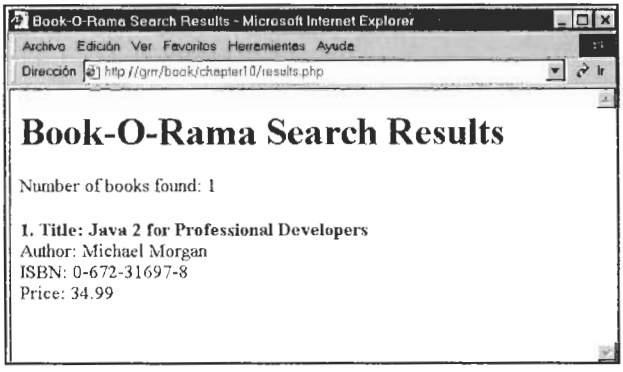


Figura 10.2. Los resultados de la búsqueda de la base de datos sobre libros de Java se presentan en la página Web utilizando la secuencia de comandos results.php

Pasos básicos para consultar una base de datos desde la Web

En cualquier secuencia de comandos utilizada para acceder a una base de datos desde la Web, se siguen algunos pasos básicos:

1. Comprobar y filtrar los datos procedentes del usuario.
2. Configurar una conexión a la base de datos pertinente.
3. Consultar la base de datos.
4. Recuperar los resultados.
5. Presentar los resultados al usuario.

Éstos son los pasos seguidos en la secuencia de comandos results.php y vamos a revisar cada uno de ellos por separado.

Comprobación y filtrado de datos entrantes

En primer lugar vamos a eliminar todos los espacios en blanco que pudiera haber introducido el usuario de manera involuntaria al principio o al final del término de búsqueda. Para ello, aplicamos la función `trim()` a `$searchterm`.

```
$searchterm=trim($searchterm);
```

El siguiente paso consiste en verificar que el usuario ha introducido un término de búsqueda y un tipo de búsqueda. Tenga en cuenta que este paso se realiza tras eliminar los espacios en blanco al final del término de búsqueda. Si hubiéramos ordenado

estas líneas en orden inverso, podría darse el caso de que el usuario hubiera introducido un término de búsqueda formado por espacios en blanco y de esta forma no se generaría un mensaje de error.

Por ello, se utiliza la instrucción `trim()`:

```
if  (!$searchtype  ||  !$searchterm)
{
   echo 'You have not entered search details.  Please go back and try again.';
   exit;
}
```

Como observará hemos comprobado la variable `$searchtype` aunque en este caso procede de una instrucción `SELECT` de HTML. Se estará preguntando por qué preocuparse de comprobar datos que tienen que introducirse. Es importante recordar que puede que exista más de una interfaz hasta la base de datos. Por ejemplo, Amazon incluye una gran cantidad de afiliados que utilizan su propia interfaz de búsqueda.

Además, resulta aconsejable filtrar los datos para evitar los problemas de seguridad que pueden surgir debido a usuarios procedentes de diferentes puntos de entrada.

Además, cuando se van a utilizar datos introducidos por un usuario, es importante filtrarlos adecuadamente para detectar cualquier carácter de control. En un capítulo anterior, hablamos de las funciones `addslashes()` y `stripslashes()`. La función `addslashes()` se utiliza al enviar los datos introducidos por el usuario a una base de datos como MySQL y `stripslashes()` al devolver el resultado al usuario que ha utilizado caracteres de control.

En este caso hemos utilizado `addslashes()` en los términos de búsqueda:

```
$searchterm  =  addslashes($searchterm);
```

También hemos utilizado `stripslashes()` sobre los datos procedentes de la base de datos. Ninguno de los datos introducidos a mano en la base de datos tenía barras ni caracteres de control. La llamada a la función `stripslashes()` no tendrá ningún efecto.

Al desarrollar una interfaz Web para la base de datos, es probable que necesitemos introducir nuevos libros y es posible que alguno de los detalles introducidos por los usuarios contenga estos caracteres.

Al introducirlos en la base de datos, llamaremos a la función `addslashes()`, lo que significa que debemos llamar a la instrucción `stripslashes()` al extraer los datos.

Se trata de una costumbre que resulta bastante aconsejable adoptar.

Estamos utilizando la función `htmlspecialchars()` para codificar caracteres que tengan significados especiales en HTML.

Nuestros datos de prueba actuales no incluyen ningún símbolo ampersand (`&`), menor que (<), mayor que (>) ni comillas dobles ("), pero hay títulos de libros en los que podrían aparecer.

Esta función nos permite evitar errores futuros.

Cómo configurar de una conexión

Utilizaremos la siguiente línea en nuestra secuencia de comandos para establecer una conexión al servidor de MySQL:

```
@ $db = mysql_pconnect('localhost', 'bookorama', 'bookorama123');
```

Hemos utilizado la función `mysql_pconnect()` para establecer una conexión a la base de datos. La sintaxis de esta función es la siguiente:

```
resource mysql_pconnect( [string host [:puerto] [:/rutasocket]  [,
          string usuario [, string contraseña]]] );
```

Por regla general, se le pasa el nombre del host en el que se ejecuta el servidor MySQL, el nombre de usuario utilizado para iniciar la sesión y la contraseña de dicho usuario. Todos estos parámetros son opcionales y si no se especifican la función utilizará los predeterminados: el host local para host, el nombre de usuario con el que se procese PHP y una contraseña en blanco.

La función devuelve un identificador de vínculo a la base de datos MySQL en caso de que el intento de conexión resulte satisfactorio (que debería guardar para otras ocasiones) o `false` en caso contrario. Es conveniente comprobar el resultado ya que el resto del código no funcionará sin una conexión válida a la base de datos. Para ello, hemos utilizado el siguiente código:

```
if (!$db)
{
   echo 'Error: Could not connect to database.  Please try again later.';
   exit;
}
```

Una función alternativa que realiza la misma labor que `mysql_pconnect()` es `mysql_connect()`. La diferencia está en que la primera devuelve una conexión permanente a la base de datos.

Una conexión normal a la base de datos se cerrará cuando la secuencia de comandos termine su ejecución o cuando se llame a la función `mysql_close()`. Las conexiones permanentes siguen abiertas cuando la secuencia de comandos termina la ejecución y no se pueden cerrar utilizando la función `mysql_close()`.

Es probable que se esté preguntado para qué comprobar si existe una conexión permanente. La respuesta está en que el establecimiento de una conexión a una base de datos implica una determinada carga y por lo tanto lleva cierto tiempo. Al llamar a `mysql_pconnect()`, antes de intentar establecer la conexión a la base de datos, se comprueba automáticamente si existe una conexión permanente abierta. Si así fuera, se utilizará dicha conexión en lugar de abrir una nueva, con el consiguiente ahorro de tiempo y carga del servidor.

También conviene señalar que las conexiones permanentes no resultan permanentes si se está ejecutando PHP como un CGI. (Cada llamada a una secuencia de comandos de PHP inicia una nueva instancia de PHP y la cierra cuando la secuencia

de comandos termina la ejecución. También se cierran todas las conexiones permanentes.)

Hay que tener en cuenta que el número de conexiones simultáneas a MySQL es limitado y viene determinado por el parámetro `max_connections`. El objetivo de este parámetro y del parámetro `MaxClients` de Apache es indicarle al servidor que rechace las solicitudes de nueva conexión en lugar de permitir el uso de todos los recursos del equipo en momentos de mucho tráfico o cuando se haya producido un error en el software.

Los valores predeterminados de estos parámetros se pueden modificar editando los archivos de configuración. Para establecer el parámetro `MaxClients` de Apache, modifique el archivo `httpd.conf` de su sistema. Para establecer el parámetro `max_connections` de MySQL, edite el archivo `my.conf`.

Si utiliza conexiones permanentes y prácticamente todas las páginas del sitio necesitan acceder a la base de datos, es probable que abra una conexión permanente para cada proceso de Apache.

Esta situación puede dar lugar a un problema si se mantienen los valores predeterminados. De manera predeterminada, Apache permite 150 conexiones, pero MySQL sólo permite 100. En fases de mucho tráfico, el número de conexiones puede resultar insuficiente.

Es aconsejable ajustar estos parámetros en función del hardware de manera que cada proceso de servidor Web disponga de una conexión.

Selección de una base de datos

Como recordará, al utilizar MySQL desde un interfaz de línea de comandos, necesitamos indicarle qué base de datos tenemos pensado utilizar con la ayuda de un comando como el siguiente:

```
use  books;
```

También necesitamos realizar esta tarea al establecer la conexión desde la Web. Desde PHP lo hacemos con una llamada a la función `mysql_select_db()`, como hemos en el siguiente caso:

```
mysql_select_db('books');
```

La función `mysql_select_db()` tiene la siguiente sintaxis:

```
bool mysql_select_db(string basededatos, [resource conexión_basededatos] );
```

Este código intentará utilizar la base de datos llamada *basededatos*. También se puede incluir el vínculo de base de datos sobre el que nos gustaría realizar esta operación (en este caso `$db`), pero si no se especifica, se utilizará el último vínculo abierto. Si no dispone de un vínculo abierto, se abrirá el predeterminado como si hubiera llamada a `mysql_connect()`.

Cómo consultar la base de datos

Para realizar una consulta, podemos utilizar la función `mysql_query()`. Sin embargo, antes de realizar esta tarea, conviene configurar la consulta que se desea ejecutar:

```
$query = "select * from books where ".$searchtype." like
'%".$searchterm."%'";
```

En este caso, buscamos el valor introducido por el usuario (`$searchterm`) en el campo especificado por el usuario (`$searchtype`). Como observará, hemos utilizado el operador `like` en lugar del signo igual para establecer la búsqueda (por regla general, conviene ser tolerante en las búsquedas sobre bases de datos).

Truco

Tenga en cuenta que no es necesario incluir un punto y coma al final de una consulta enviada a MySQL, a diferencia de las consultas realizadas a través del monitor de MySQL.

Ahora podemos ejecutar la consulta:

```
$result = mysql_query($query);
```

La función `mysql_query()` tiene la siguiente sintaxis:

```
resource mysql_query(string consulta, [resource conexión_basededatos] );
```

Se le pasa la consulta que se desea ejecutar y, opcionalmente, el vínculo a la base de datos (nuevamente, `$db`). Si no se especifica, la función utilizará el último vínculo abierto. Si no existiese ninguno, la función abriría el predeterminado como si hubiera llamado a `mysql_connect()`.

Esta función devuelve un identificador de resultados (que permite recuperar los resultados de la consulta) en caso de éxito y `false` en caso de contrario. Debería almacenarlo (como hemos hecho en este caso en `$result`) para poder hacer algo útil.

Recuperación de resultados de consulta

Existe un conjunto de funciones para dividir los resultados del identificador de resultados de varias formas. El identificador de resultados es la clave para acceder a las filas devueltas por la consulta.

En nuestro ejemplo, hemos utilizado dos funciones de este tipo: `mysql_num_rows()` y `mysql_fetch_array()`.

La función `mysql_num_rows()` indica el número de filas devueltas por la consulta. Debería pasárselo al identificador de resultados, de la siguiente forma:

```
$num_results = mysql_num_rows($result);
```

Resulta útil saber hacerlo. Si tenemos previsto procesar o mostrar los resultados, sabremos cuántos hay y podemos procesarlos en un bucle:

```
for ($i=0; $i <$num_results; $i++)
{
   // procese los resultados
}
```

En cada iteración de este bucle, llamamos a `mysql_fetch_array()`. El bucle no se ejecutará si no se devuelve ninguna fila.

Esta función toma cada fila del conjunto de resultados y devuelve la fila como una matriz asociativa, con cada clave en forma de nombre de atributo y cada valor con su valor correspondiente en la matriz:

```
$row = mysql_fetch_array($result);
```

Dada la matriz asociativa `$row`, podemos recorrer cada campo y mostrarlos adecuadamente, por ejemplo:

```
echo '<br />ISBN: ';
echo  stripslashes($row['isbn']);
```

Como se indicó previamente, hemos llamado a `stripslashes()` para limpiar el valor antes de mostrarlo.

Existe una serie de variaciones a la hora de obtener los resultados desde un identificador de resultados. En lugar de utilizar una matriz asociativa, podemos recuperar los resultados en una matriz enumerada con `mysql_fetch_row()`, de la siguiente forma:

```
$row = mysql_fetch_row($result);
```

Los valores de atributo se enumerarán en cada uno de los valores de matriz `$row[0]`, `$row[1]` etc.

También puede acceder a cada uno de los atributos mediante `$row->title`, `$row->author`, etc.

Cada uno de estas opciones recupera una fila por vez.

Para ello, debe especificar el número de fila (desde cero al número `rows[ms]1`) así como el nombre del campo.

Por ejemplo:

```
$row = mysql_result($result, $i, 'title');
```

Puede especificar el nombre del campo como una cadena (en forma de "título" o "libros.título") o en forma de número (como en `mysql_fetch_row()`). No debería usar `mysql_result()` con otras funciones de recuperación.

Las funciones de recuperación orientadas a filas resultan mucho más eficientes que la función `mysql_fetch_row()`, por lo que en general debería utilizar una de aquéllas.

Desconexión de una base de datos

Puede utilizar

```
mysql_close(conexión_basededatos);
```

para cerrar una conexión de base de datos no permanente. Esta opción no resulta estrictamente necesaria porque se cerrarán de todos modos cuando una secuencia de comandos finalice su ejecución.

Cómo colocar nueva información en la base de datos

La inserción de nuevos elementos en la base de datos resulta bastante similar a la obtención de elementos de la base de datos. Los pasos son prácticamente iguales: se establece una conexión, se envía una consulta y se comprueban los resultados. En este caso, la consulta enviada será `INSERT` en lugar de `SELECT`.

Aunque el proceso es similar, a veces resulta útil examinar un ejemplo. En la figura 10.3, se ilustra un formulario HMTL sencillo para introducir nuevos libros dentro de la base de datos.

El código HMTL de esta página se incluye en el listado 10.3.

Listado 10.3. newbook.html. Código HTML de la página de entrada de libros

```
<html>
 <head>
  <title>Book-O-Rama - New Book Entry</title>
</head>

<body>
  <h1>Book-O-Rama - New Book Entry</h1>

  <form action="insert_book.php" method="post">
    <table border="0">
      <tr>
        <td>ISBN</td>
        <td><input type="text" name="isbn" maxlength="13" size="13"><br
        /></td>
      </tr>
      <tr>
        <td>Author</td>
```

```
            <td> <input type="text" name="author" maxlength="30"
            size="30"><br /></td>
        </tr>
        <tr>
            <td>Title</td>
            <td> <input type="text" name="title" maxlength="60"
            size="30"><br></td>
        </tr>
        <tr>
            <td>Price $</td>
            <td><input type="text" name="price" maxlength="7"
            size="7"><br /></td>
        </tr>
        <tr>
            <td colspan="2"><input type="submit" value="Register"></td>
        </tr>
    </table>
</form>
</body>
</html>
```

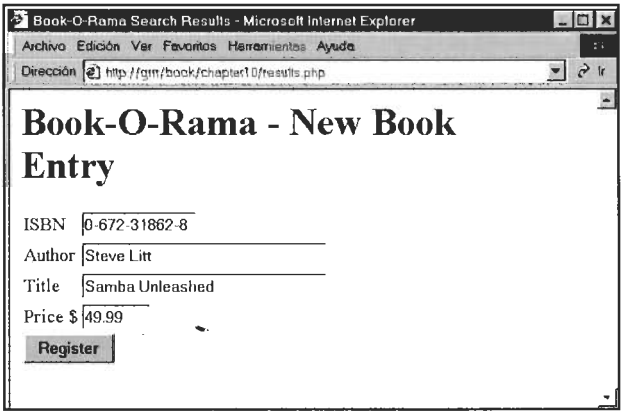


Figura 10.3. El personal de Book-O-Rama puede utilizar esta interfaz para la introducción de nuevos libros en la base de datos

Los resultados de este formulario se pasan a `insert_book.php`, una secuencia de comandos que toma los detalles, realiza algunas operaciones de validación e intenta escribir los datos dentro de la base de datos. El código correspondiente a esta secuencia de comandos se incluye en el listado 10.4.

Listado 10.4. insert_book.php. Esta secuencia de comandos escribe nuevos libros en la base de datos

```
<html>
<head>
    <title>Book-O-Rama Book Entry Results</title>
</head>
```

```
<body>
<h1>Book-O-Rama Book Entry Results</h1>
<?php
  // cree nombres de variable cortos
  $isbn=$HTTP_POST_VARS['isbn'];
  $author=$HTTP_POST_VARS['author'];
  $title=$HTTP_POST_VARS['title'];
  $price=$HTTP_POST_VARS['price'];

  if (!$isbn || !$author || !$title || !$price)
  {
     echo 'You have not entered all the required details.<br />'
          .'Please go back and try again.';
     exit;
  }

  $isbn = addslashes($isbn);
  $author = addslashes($author);
  $title = addslashes($title);
  $price = doubleval($price);

  @ $db = mysql_pconnect('localhost', 'bookorama', 'bookorama123');

  if (!$db)
  {
     echo 'Error: Could not connect to database.  Please try again later.';
     exit;
  }

  mysql_select_db('books');
  $query = "insert into books values
       ('".$isbn."', '".$author."', '".$title."', '".$price."')";
  $result = mysql_query($query);
  if ($result)
       echo mysql_affected_rows().' book inserted into database.';
?>

</body>
</html>
```

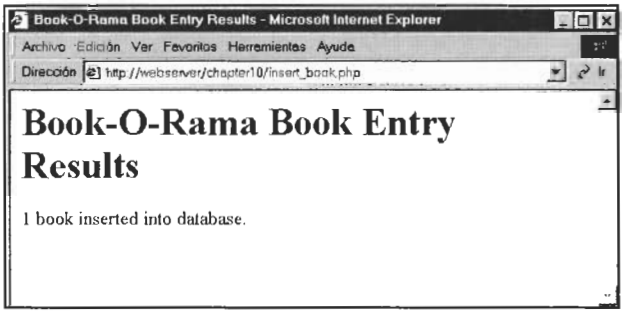


Figura 10.4. La secuencia de comandos se completa satisfactoriamente e indica que el libro se ha agregado a la base de datos

En la figura 10.4 se muestra el resultado cuando se inserta un libro satisfactoriamente.

Si examina el código de `insert_book.php`, verá que en gran parte resulta similar a la secuencia de comandos que escribimos para recuperar datos de la base de datos. Hemos comprobado que todos los campos del formulario se han rellenado y que se les ha aplicado formato correctamente para su inserción en la base de datos con `addslashes()`:

```
$isbn   =  addslashes($isbn);
$author  =  addslashes($author);
$title   =  addslashes($title);
$price   =  doubleval($price);
```

Como el precio se ha almacenado en la base de datos como una variable de tipo flotante, no queremos colocar barras en su interior. Para filtrar todos los caracteres extraños que se puedan introducir en este campo numérico, se puede llamar a la función `doubleval()`, comentada en un capítulo anterior. También se encargará de filtrar los símbolos de moneda que pueda haber introducido el usuario en el formulario.

Hemos vuelto a establecer la conexión a la base de datos por medio de `mysql_pconnect()`, y hemos configurado una consulta para enviarla a la base de datos. En este caso, la consulta es una instrucción `INSERT` de SQL:

```
$query = "insert into books values
     ('".$isbn."',  '".$author."',  '".$title."',  '".$price."')";
$result = mysql_query($query);
```

Ésta se ejecuta sobre la base de datos de la forma habitual llamando a `mysql_query()`.

Una diferencia significativa entre utilizar `INSERT` y `SELECT` está en el uso de

```
mysql_affected_rows():
echo  mysql_affected_rows()." book inserted into database.";
```

En la secuencia de comandos anterior, utilizamos `mysql_num_rows()` para determinar cuántas filas devolvía una instrucción `SELECT`. Al escribir consultas que cambien la base de datos como `INSERT`, `DELETE` y `UPDATE`, debería utilizar en su lugar `mysql_affected_rows()`.

Con esto, quedan vistos los elementos básicos del uso de bases de datos MySQL desde PHP.

A continuación, examinaremos brevemente algunas otras funciones útiles de las que no hemos hablado todavía.

Otras funciones útiles PHP y MySQL

Existen algunas otras funciones de PHP y MySQL útiles, que comentaremos brevemente.

Liberación de recursos

Si tiene problemas de memoria al ejecutar una secuencia de comandos, puede utilizar `mysql_free_result()`, cuya sintaxis es la siguiente:

```
bool mysql_free_result(resource resultado);
```

Se llama con un identificador de resultados, como el siguiente:

```
mysql_free_result($result);
```

Este código libera la memoria utilizada para almacenar el resultado. Obviamente no se invoca hasta terminar de trabajar con un conjunto de resultados.

Creación y eliminación de bases de datos

Para crear una nueva base de datos MySQL desde una secuencia de comandos de PHP, puede utilizar `mysql_create_db()` y para eliminar una base de datos MySQL existente, puede utilizar `mysql_drop_db()`.

Estas funciones constan de las siguientes sintaxis:

```
bool mysql_create_db(string basededatos, [resource conexión_basededatos] );
bool mysql_drop_db(string basededatos, [resource conexión_basededatos] );
```

Ambas funciones toman un nombre de base de datos y una conexión opcional. Si no se suministra ninguna conexión, se abrirá la última. Intentarán crear o eliminar la base de datos cuyo nombre se haya indicado. Ambas funciones devuelven `true` si el resultado es satisfactorio y `false` en caso contrario.

Otras interfaces de base de datos y PHP

PHP admite bibliotecas para establecer conexiones a una gran cantidad de bases de datos, incluidas Oracle, Microsoft SQL Server, mSQL y PostgreSQL.

Por regla general, los principios para establecer la conexión y consultar cualquiera de las bases de datos son prácticamente los mismos. Varían los nombres de las funciones individuales y las distintas bases de datos tienen funcionalidades ligeramente diferentes, pero si puede establecer una conexión a MySQL, no le costará adaptar esos conocimientos a cualquier situación.

Si desea utilizar una base de datos que no tenga una biblioteca específica disponible en PHP, puede utilizar las funciones ODBC genéricas. ODBC equivale a Conectividad de base de datos abierta y es un estándar para las conexiones a bases de datos. Su funcionalidad es la más restrictiva de cualquier conjunto, por razones bastante obvias. Si se necesita una compatibilidad completa, no se pueden explotar funciones especiales.

Además de las bibliotecas incluidas en PHP, también están disponibles clases de abstracción de base de datos como Metabase o PEAR::DB, que permiten utilizar los mismos nombres de función para cada tipo de base de datos.

Uso de una interfaz de base de datos genérica: PEAR DB

Vamos a examinar un breve ejemplo del uso de la capa de abstracción PEAR DB. Se trata de uno de los componentes fundamentales de PEAR y probablemente el más utilizado de todos.

Si tiene instalado PEAR, ya debería tener el componente DB. De lo contrario, consulte la sección correspondiente del apéndice A.

Para permitirle comparar, vamos a ver cómo hubiéramos escrito la secuencia de comandos de resultados de búsqueda utilizando DB.

Listado 10.5. results_generic.php. Recupera resultados de búsqueda desde la base de datos MySQL y les aplica formato para su visualización

```
<html>
<head>
   <title>Book-O-Rama  Search  Results</title>
</head>
<body>
<h1>Book-O-Rama Search Results</h1>
<?php
   // cree nombres de variables cortos
   $searchtype=$HTTP_POST_VARS['searchtype'];
   $searchterm=$HTTP_POST_VARS['searchterm'];

   $searchterm= trim($searchterm);

   if (!$searchtype || !$searchterm)
   {
      echo 'You have not entered search details.Please go back and try again.';
      exit;
   }

   $searchtype = addslashes($searchtype);
   $searchterm  =  addslashes($searchterm);

   // configuración para utilizar PEAR DB
   require('DB.php');
   $user = 'bookorama';
   $pass = 'bookorama123';
   $host = 'localhost';
   $db_name = 'books';

   // configure la cadena de conexión universal o DSN
```

```
    $dsn = "mysql://$user:$pass@$host/$db_name";

    // establezca una conexión a la base de datos
    $db = DB::connect($dsn, true);

    // compruebe si funciona la conexión
    if (DB::isError($db))
    {
       echo  $db->getMessage();
       exit;
    }

    // realice la consulta
    $query = "select * from books where ".$searchtype." like
'%".$searchterm."%'";
    $result = $db->query($query);
    // compruebe que el resultado es correcto
    if (DB::isError($result))
    {
       echo $db->getMessage();
       exit;
    }

    // obtenga el número de filas devuelto
    $num_results = $result->numRows();

    //  muestre  cada  fila  devuelta
    for ($i=0; $i <$num_results; $i++)
    {
       $row = $result->fetchRow(DB_FETCHMODE_ASSOC);
       echo '<p><strong>'.($i+1).'. Title: ';
       echo htmlspecialchars(stripslashes($row['title']));
       echo '</strong><br />Author: ';
       echo stripslashes($row['author']);
       echo '<br />ISBN: ';
       echo stripslashes($row['isbn']);
       echo '<br />Price: ';
       echo stripslashes($row['price']);
       echo '</p>';
    }

    // desconecte de la base de datos
    $db->disconnect();
?>

</body>
</html>
```

Vamos a examinar los elementos nuevos de esta secuencia de comandos. Para establecer la conexión utilizamos la línea

```
$db  =  DB::connect($dsn,  true);
```

Esta función acepta una cadena de conexión universal con todos los parámetros necesarios parar establecer la conexión a la base de datos.

Puede apreciarlo si examina el formato de la cadena de conexión:

```
$dsn = "mysql://$user:$pass@$host/$db_name";
```

El segundo parámetro de `connect()` determina si la conexión será permanente o no. El valor `true` la convertirá en permanente.

Tras ello, comprobamos si la conexión falló con ayuda del método `isError()` y, en caso afirmativo, devolvemos un mensaje de error y salimos:

```
if (DB::isError($db))
{
    echo $db->getMessage();
    exit;
}
```

Asumiendo que todo haya salido bien, establecemos una consulta y la ejecutamos de la siguiente forma:

```
$result = $db->query($query);
```

Podemos comprobar el número de filas devueltas:

```
$num_results = $result->numRows();
```

Recuperamos cada fila de la siguiente forma:

```
$row = $result->fetchRow(DB_FETCHMODE_ASSOC);
```

El método genérico `fetchRow()` permite recuperar una fila con muchos formatos diferentes. El parámetro `DB_FETCHMODE_ASSOC` le indica que queremos recuperar la fila en una matriz asociativa. Tras mostrar las filas devueltas, terminamos cerrando la conexión a la base de datos:

```
$db->disconnect();
```

Como puede apreciar, este ejemplo genérico resulta muy similar a la primera secuencia de comandos.

Las ventajas de usar DB son que sólo necesitamos recordar un conjunto de funciones de base de datos y que el código requerirá mínimos cambios si decidimos modificar el software de nuestra base de datos.

Como este libro trata de MySQL, utilizaremos las bibliotecas nativas de MySQL para obtener una mayor velocidad y flexibilidad, pero si lo desea puede utilizar el paquete DB en sus proyectos dado lo extremadamente útil que resulta acudir a él en determinadas ocasiones.

Lecturas adicionales

Si desea obtener más información sobre cómo combinar MySQL y PHP, puede leer las secciones pertinentes de manuales sobre PHP y MySQL.

Si desea obtener más información sobre ODBC, visite

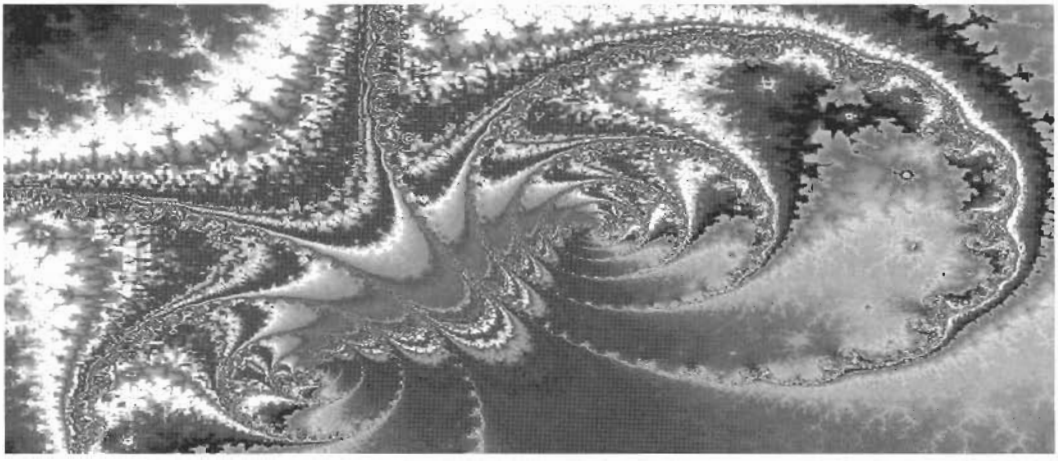
```
http://www.webopedia.com/TERM/O/ODBC.html
```

Metabase está disponible en

```
http://phpclasses.upperdesign.com/browse.html/package/20
```

A continuación

En el siguiente capítulo, profundizaremos en el estudio de la administración de MySQL y veremos formas de optimizar las bases de datos.



11

MySQL Avanzado

En este capítulo, vamos a examinar algunos aspectos avanzados de MySQL como los privilegios, la seguridad y la optimización.

En concreto abordaremos los siguientes temas:

- Análisis detallado del sistema de privilegios
- Protección de una base de datos MySQL
- Cómo obtener más información sobre la base de datos
- Agilización de los procesos por medio de índices
- Sugerencias de optimización
- Tipos diferentes de tablas
- Copias de seguridad y recuperación

Análisis detallado del sistema de privilegios

En un capítulo anterior vimos cómo configurar usuarios y concederles privilegios. Para ello, utilizamos el comando `GRANT`. Si va a administrar una base de datos MySQL, debe entender la función de este comando y su funcionamiento.

El uso de la instrucción `GRANT` afecta a las tablas de una base de datos especial llamada `mysql`. La información sobre privilegios se almacena en cinco tablas de

esta base de datos. Por lo tanto, al conceder privilegios sobre bases de datos, debe tener cuidado al otorgar permisos de acceso a la base de datos `mysql`.

El comando `GRANT` sólo está disponible a partir de la versión 3.22.11 de MySQL.

Si desea examinar los contenidos de la base de datos `mysql`, regístrese como administrador y escriba

```
use  mysql;
```

Tras hacerlo, podrá ver las tablas de la base de datos con la instrucción

```
show  tables;
```

como de costumbre.

Los resultados presentarán un aspecto semejante al siguiente:

```
+-----------------+
| Tables_in_mysql |
+-----------------+
| columns_priv    |
| db              |
| func            |
| host            |
| tables_priv     |
| user            |
+-----------------+
```

Cada una de estas tablas, a excepción de la tabla `func`, almacenan información sobre privilegios. (La tabla `func` almacena funciones definidas por el usuario.) También se conocen como tablas de permisos. Las funciones específicas de estas tablas varían pero todas ellas tiene un mismo objetivo general: determinar lo que los usuarios pueden y no pueden hacer. Estas tablas contienen dos tipos de campos: los campos de ámbito, que identifican al usuario, al host y a parte de una base de datos; y los campos de privilegio, que identifican las acciones que puede realizar el usuario en dicho ámbito.

La tabla `user` se utiliza para decidir si un usuario puede conectarse al servidor MySQL y si dispone de privilegios administrativos. Las tablas `db` y `host` determinan a qué bases de datos puede acceder el usuario. La tabla `tables_priv` determina qué tablas de una base de datos puede utilizar un usuario y la tabla `columns_priv` establece a qué columnas de la tabla dispone de acceso.

La tabla de usuario

Esta tabla contiene los detalles de los privilegios globales de usuario. Determina si un usuario puede conectarse al servidor MySQL y si dispone de privilegios globales, es decir, de privilegios que se aplican a toda las bases de datos del sistema.

Para ver la estructura de esta tabla, puede utilizar la instrucción `describe user;`.

En la tabla 11.1 se recoge el esquema de la tabla `user`.

Tabla 11.1. Esquema de la tabla user de la base de datos mysql

Campo	Tipo
Host	char(60)
User	char(16)
Password	char(16)
Select_priv	enum('N','Y')
Insert_priv	enum('N','Y')
Update_priv	enum('N','Y')
Delete_priv	enum('N','Y')
Create_priv	enum('N','Y')
Drop_priv	enum('N','Y')
Reload_priv	enum('N','Y')
Shutdown_priv	enum('N','Y')
Process_priv	enum('N','Y')
File_priv	enum('N','Y')
Grant_priv	enum('N','Y')
References_priv	enum('N','Y')
Index_priv	enum('N','Y')
Alter_priv	enum('N','Y')

Cada fila de esta tabla se corresponde con un conjunto de privilegios para un usuario procedente de un host y registrado con la contraseña `Password`. Se trata de los campos de ámbito de esta tabla, ya que describen el ámbito del resto de los campos, llamados campos de privilegio.

Los privilegios enumerados en esta tabla (y en las siguientes) se corresponden con los concedidos por medio de la instrucción `GRANT` en un capítulo anterior. Por ejemplo, `Select_priv` se corresponde con el privilegio necesario para ejecutar un comando `SELECT`.

Si un usuario tiene un privilegio concreto, el valor de dicha columna será `Y`. Por el contrario, si a un usuario no se le ha concedido dicho privilegio, el valor será `N`.

Los privilegios listados en la tabla de usuarios son globales, lo que quiere decir que se aplican a todas las bases de datos del sistema (incluida la base de datos `mysql`).

Los administradores tendrán varios valores `Y` pero la mayoría de los usuarios deberían tener todos `N`.

Los usuarios normales deberían disponer de derechos para acceder a determinadas bases de dates, pero no a todas las tablas.

Las tablas db y host

La mayor parte de los privilegios de los usuarios se almacenan en las tablas `db` y `host`.

La tabla `db` determina qué usuarios pueden acceder, a qué bases de datos y desde qué hosts. Los privilegios listados en esta tabla se aplican a cualquier base de datos designada en una fila dada.

La tabla `host` complementa a la tabla `db`. Si un usuario va a conectarse a una base de datos desde varios hosts, no se enumerará ningún host para dicho usuario en la tabla. En su lugar, se incluirá un conjunto de entradas en la tabla `host` para especificar los privilegios correspondientes a cada combinación de usuario y host.

En las tablas 11.2 y 11.3 se recogen los esquemas de estas dos tablas, respectivamente.

Tabla 11.2. Esquema de la tabla db de la base de datos mysql

Campo	Tipo
Host	char(60)
Db	char(64)
User	char(16)
Select_priv	enum('N','Y')
Insert_priv	enum('N','Y')
Update_priv	enum('N','Y')
Delete_priv	enum('N','Y')
Create_priv	enum('N','Y')
Drop_priv	enum('N','Y')
Grant_priv	enum('N','Y')
References_priv	enum('N','Y')
Index_priv	enum('N','Y')
Alter_priv	enum('N','Y')

Tabla 11.3. Esquema de la tabla host de la base de datos mysql

Campo	Tipo
Host	char(60)
Db	char(64)

Campo	Tipo
Select_priv	enum('N','Y')
Insert_priv	enum('N','Y')
Update_priv	enum('N','Y')
Delete_priv	enum('N','Y')
Create_priv	enum('N','Y')
Drop_priv	enum('N','Y')
Grant_priv	enum('N','Y')
References_priv	enum('N','Y')
Index_priv	enum('N','Y')
Alter_priv	enum ('N','Y')

Las tablas tables_priv y columns_priv

Estas dos tablas se utilizan para almacenar privilegios de nivel de tabla y de nivel de columna, respectivamente.

Funcionan como la tabla `db`, con la diferencia de que suministran privilegios para las tablas de una base de datos dada y para las columnas de una tabla concreta, respectivamente.

Estas tablas constan de una estructura ligeramente diferente a las tablas `user`, `db` y `host`. En la tabla 11.4 y 11.5 se incluyen los esquemas de las tablas `tables_priv` y `columns_priv`, respectivamente.

Tabla 11.4. Esquema de la tabla tables_priv de la base de datos mysql

Campo	Tipo
Host	char(60)
Db	char(64)
User	char(16)
Table_name	char(60)
Grantor	char(77)
Timestamp	timestamp(14)
Table_priv	set('Select', 'Insert', 'Update', 'Delete', 'Create', 'Drop', 'Grant', 'References', 'Index', 'Alter')
Column_priv	set ('Select', 'Insert', 'Update', 'References')

Tabla 11.5. Esquema de la tabla columns_priv de la base de datos mysql

Campo	Tipo
Host	char(60)
Db	char(64)
User	char(16)
Table_name	char(64)
Column_name	char(64)
Timestamp	timestamp(14)
Column_priv	set('Select', 'Insert', 'Update', 'References')

La columna `Grantor` de la tabla `tables_priv` almacena el usuario que concedió el privilegio a este usuario. La columna `Timestamp` de ambas tablas almacena la fecha y la hora en la que se concedió el privilegio.

Control de acceso: ¿cómo utiliza MySQL las tablas de concesión de privilegios?

MySQL utiliza las tablas `grant` para determinar qué puede hacer un usuario en un proceso de dos fases:

1. Verificación de conexión. En esta fase, MySQL comprueba si disponemos de permiso para establecer una conexión en función de la información de la tabla de usuario, como se indicó anteriormente. Para realizar esta operación, se utiliza el nombre de usuario, el nombre de host y la contraseña. Si se deja en blanco un nombre de usuario, equivaldrá a todos los usuarios. Los nombres de host se pueden especificar mediante el uso de un carácter comodín (`%`). Se puede utilizar en forma de campo completo (es decir, que `%` equivalga a todos los hosts) o como parte de un nombre de host, por ejemplo, `%.tangledweb.com.au`, que equivale a todos los hosts que terminen en `.tangledweb.com.au`. Si se deja en blanco el campo de contraseña, no será necesario utilizar una contraseña. Sin embargo, no conviene utilizar usuarios en blanco, comodines en hosts ni usuarios sin contraseñas por cuestiones de seguridad.
2. Verificación de la solicitud. Cada vez que se introduce una solicitud, tras establecer una conexión, MySQL comprueba si disponemos del nivel necesario de privilegios para realizar dicha petición. En primer lugar, el sistema comprueba los privilegios globales (en la tabla `user`) y si no resultan suficientes, comprueba las tablas `db` y `host`. Si no se dispone de los privilegios

necesarios en dicho nivel, MySQL examinará la tabla `tables_priv` y, si no son bastan, examinará la tabla `colums_priv`.

Actualización de privilegios: ¿cuándo surten efecto los cambios?

El servidor MySQL lee automáticamente las tablas de privilegios al iniciarse y al aplicar las instrucciones `GRANT` y `REVOKE`.

Sin embargo, ahora que ya sabemos dónde y cómo se almacenan dichos privilegios, podemos alterarlos manualmente. Al actualizarlos manualmente, el servidor MySQL no notará que han cambiado.

Es necesario indicarle al servidor que se ha producido un cambio y existen tres formas de hacerlo. Puede escribir

```
FLUSH PRIVILEGES;
```

en la línea de comandos de MySQL (necesitará haber iniciado la sesión como administrador). Ésta es la forma más habitual de actualizar los privilegios.

También puede ejecutar

```
mysqladmin flush-privileges
```

o

```
mysqladmin  reload
```

desde el sistema operativo.

Tras ello, se comprobarán los privilegios globales cuando un usuario se conecte, los privilegios de base de datos en la siguiente instrucción de uso y los privilegios de tabla y columna en la siguiente petición de usuario.

Cómo proteger la bases de datos MySQL

La seguridad es importante, en especial al establecer la conexión entre la base de datos MySQL y el sitio Web. En esta sección, vamos a repasar las precauciones que se deberían adoptar para proteger una base de datos.

MySQL desde el punto de vista del sistema operativo

No es aconsejable ejecutar el servidor MySQL (`mysqld`) como usuario raíz si está utilizando un sistema operativo de tipo UNIX ya que permitiría a un usuario de MySQL con un conjunto completo de privilegios leer y escribir archivos en cualquier

parte del sistema operativo. Es importante tener en cuenta esta circunstancia ya que se suele pasar fácilmente por alto y ya ha sido utilizada para piratear sitios Web basados en Apache. (Afortunadamente, se trataba de piratas buenos y lo único que hicieron fue reforzar la seguridad).

Es aconsejable configurar un usuario de MySQL de manera específica para ejecutar `mysqld`. Además puede configurar los directorios (donde se almacenan los datos físicamente) para limitar el acceso a dicho usuario. En muchas instalaciones, el servidor se configura para ejecutarse como userid mysql, en el grupo mysql.

También debería colocar su servidor MySQL tras un cortafuegos. De esta forma podrá evitar conexiones procedentes de equipos no autorizados. Compruebe si puede conectarse desde el exterior a su servidor a través del puerto número 3306; se trata del puerto predeterminado en el que se ejecuta MySQL y debería estar cerrado en el cortafuegos.

Contraseñas

Asegúrese de que todos los usuarios tienen contraseñas (en especial el usuario raíz) y que están bien elegidas y se cambian periódicamente, como en el caso de las contraseñas de sistemas operativos. Recuerde que no conviene utilizar contraseñas que contengan palabras o están formadas por palabras de un diccionario. Lo mejor es utilizar combinaciones de letras y números.

Si va a almacenar contraseñas en archivos de secuencias de comandos, asegúrese de que sólo el usuario cuya contraseña esté almacenada puede verlos. Los dos lugares principales en los puede suceder son:

1. En la secuencia de comandos `mysql.server`, es probable que necesite utilizar la contraseña raíz de UNIX. Si así fuera, asegúrese de que sólo el usuario raíz puede leer esta secuencia de comandos.

2. En las secuencias de comandos de PHP que se utilizan para establecer una conexión a la base de datos, necesitará almacenar la contraseña para dicho usuario. Esta operación se puede realizar de forma segura colocando el nombre de usuario y la contraseña en un archivo llamado, por ejemplo, `dbconnect.php`, que puede situar donde lo necesite. Esta secuencia de comandos se puede almacenar fuera de árbol de documentos Web y configurarla para que sólo resulte accesible al usuario adecuado. Recuerde que si coloca estos detalles en un archivo `.inc` o en cualquier otra extensión de archivo en el árbol Web, tendrá que asegúrese de comprobar que el servidor Web sabe que dichos archivos deben interpretarse como PHP, para que los detalles no puedan verse en un navegador Web.
   No almacene contraseñas en forma de texto dentro de la base de datos. Las contraseñas de MySQL no se almacenan de esa forma, pero resulta habitual en aplicaciones Web en las que se desea almacenar los nombres de usuario y contraseñas de los miembros del sitio. Puede cifrar las contraseñas utilizando las funciones `PASSWORD()` o `MD5` de MySQL. Recuerde que si inserta una contraseña con el comando `INSERT` utilizando una de estas funciones, al

ejecutar una selección (para registrar un usuario), tendrá que utilizar la misma función de nuevo para comprobar la contraseña utilizada por el usuario.
En la parte dedicada a la implementación de proyectos utilizaremos esta funcionalidad.

Privilegios de usuario

Asegúrese de tener claro el sistema de privilegios de MySQL y las consecuencias de la concesión de los distintos privilegios. No conceda más privilegios a un usuario de los necesarios. Para verificarlos, examine las tablas de concesión de privilegios.

En concreto, no conceda los privilegios `PROCESS`, `FILE`, `SHUTDOWN` y `RELOAD` a ningún usuario que no sea un administrador a menos que resulte completamente necesario. El privilegio `PROCESS` se puede utilizar para ver qué están haciendo o escribiendo otros usuarios, incluidas sus contraseñas. El privilegio `FILE` se puede utilizar para leer y escribir archivos desde y hasta el sistema operativo (incluyendo, por ejemplo, el archivo `/etc/password` en un sistema UNIX).

El privilegio `GRANT` debería concederse con precaución ya que permite a los usuarios compartir sus privilegios con otros usuarios.

Asegúrese de que al configurar los usuarios, sólo les concede acceso desde los hosts que utilizan para conectarse. El usuario `jane@localhost` es correcto pero el usuario `jane` es bastante común y podría registrarse desde cualquier lugar (y puede que no se trate de la persona que está pensando). Evite utilizar comodines en los nombres de host por razones similares.

Puede aumentar más la seguridad utilizando direcciones IP en lugar de nombres de dominio en la tabla `host`. Esta medida evita errores o intrusos en su DNS. Para ello, inicie el demonio de MySQL con la opción `-skip-name-resolve`, que exige que todos los valores de columna de host sean una dirección IP o un host local.

Como alternativa puede iniciar `mysqld` con la opción `-secure`. Esta opción comprueba las IP resueltas para determinar si se corresponden con los nombres de host suministrados. (Esta opción sólo está activada de manera predeterminada desde la versión 3.22 en adelante).

No debería permitir que los usuarios no administrativos dispongan de acceso al programa `mysqladmin` en el servidor Web. Este programa se ejecuta desde la línea de comandos por lo que su acceso se convierte en una aspecto relacionado con los privilegios del sistema operativo.

Aspectos relacionados con la Web

Al conectar la base de datos MySQL a la Web, surgen una serie de temas de seguridad especiales.

Conviene, en primer lugar, configurar un usuario especial para las operaciones relacionadas con las conexiones Web. Puede concederle los privilegios mínimos necesario excluyendo, `DROP`, `ALTER` o `CREATE`. Puede concederle el privilegio `SELECT` sólo sobre las tablas de catálogos y el privilegio `INSERT` sobre las tablas de pedidos.

Se trata de un nuevo ejemplo de la aplicación del principio de concesión de la menor cantidad posible de privilegios.

Advertencia

En el último capítulo hablamos del uso de las funciones `addslashes()` y `stripslashes()` para eliminar caracteres problemáticos en cadenas. Es importante no olvidarse de aplicar estas funciones y de limpiar los datos antes de enviar nada a MySQL. Como recordará, utilizamos la función`doubleval()` para comprobar que los datos numéricos eran numéricos. Resulta habitual pasar por alto esta comprobación.

Debería comprobar siempre todos los datos procedentes de un usuario. Aunque el formulario HTML esté formado de cuadros de selección y botones de opción, alguien podría haber modificado el URL para piratear la secuencia de comandos. También conviene comprobar el tamaño de los datos entrantes.

Si los usuarios escriben contraseñas o datos confidenciales para su almacenamiento en la base de datos, recuerde que se transmitirán como texto sin procesar desde el navegador al servidor a menos que utilice SSL (del inglés Secure Sockets Layer, Nivel de sockets seguro). En una sección posterior se ampliará este tema.

Cómo obtener más información sobre bases de datos

Por el momento, hemos utilizado `SHOW` y `DESCRIBE` para determinar las tablas que incluye la base de datos y las columnas de las tablas.

A continuación, vamos a analizar otras formas de utilizar estas instrucciones así como la instrucción `EXPLAIN` para obtener más información sobre cómo se realiza una operación de selección.

Cómo obtener información con SHOW

Antes hemos utilizado

```
show tables;
```

para obtener una lista de tablas de la base de datos
La instrucción

```
show databases;
```

muestra una lista de bases de datos disponibles.

A continuación puede utilizar la instrucción `SHOW TABLES` para ver la lista de tablas de una de esas bases de datos:

```
show tables from books;
```

Si se aplica la instrucción `SHOW TABLES` sin especificar una base de datos, se utilizará la base de datos actual de manera predeterminada.

Si sabe cuáles son las tablas, puede obtener una lista de las columnas:

```
show columns from orders from books;
```

Si no activa el parámetro correspondiente a la base de datos, la instrucción `SHOW COLUMNS` utilizará la base de datos actual. También puede utilizar la notación tabla.columna:

```
show columns from books.orders;
```

Existe una variación de la instrucción `SHOW` para ver los privilegios que tiene un usuario. Por ejemplo, si ejecutamos la siguiente instrucción, obtendremos el resultado ilustrado en la figura 11.1.

```
show grants for bookorama;
```

Las instrucciones `GRANT` que se muestran no son necesariamente las que se ejecutaron para conceder privilegios a un usuario dado, sino más bien instrucciones de resumen equivalentes que generan el nivel actual de privilegios del usuario.

```
+-----------------------------------------------------------------------------------------------------------------+
| Grants for bookorama@%                                                                                          |
+-----------------------------------------------------------------------------------------------------------------+
| GRANT USAGE ON *.* TO 'bookorama'@'%' IDENTIFIED BY PASSWORD '6a87b6810cb073de'                                 |
| GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, INDEX, ALTER ON books.* TO 'bookorama'@'%'                  |
+-----------------------------------------------------------------------------------------------------------------+
```

Figura 11.1. El resultado de la instrucción SHOW GRANTS

Nota

La instrucción SHOW GRANTS se agregó en la versión 3.23.4 de MySQL. Si su versión es inferior no funcionará.

Existen muchas variantes de la instrucción `SHOW`. En la tabla 11.6 se recoge un resumen de todas ellas.

Tabla 11.6. Sintaxis de la instrucción SHOW

Variante	Descripción
`SHOW DATABASES [LIKE` *`base_de_datos`*`]`	Enumera todas las bases de datos. Puede sustituir el parámetro `base_de_datos` por sus nombres.

Variante	Descripción
`SHOW TABLES [FROM base_de_datos] [LIKE tabla]`	Enumera las tablas de la base de datos utilizada actualmente o de la base de datos llamada `base_de_datos` si se especifica. Puede sustituir `tabla` por el nombre de tablas.
`SHOW COLUMNS FROM tabla [FROM base_de_datos] [LIKE columna]`	Enumera todas las columnas de una tabla de la base de datos actualmente en uso o de una base de datos especificada. Puede sustituir `columna` por nombres de columna. Puede utilizar `SHOW FIELDS` en lugar de `SHOW COLUMNS`.
`SHOW INDEX FROM tabla [FROM base_de_datos]`	Muestra los detalles de todos los índices de una tabla dada de la base de datos actualmente en uso o de otra base de datos llamada `base_de_datos` si se especifica. También puede utilizar `SHOW KEYS` en su lugar.
`SHOW STATUS [LIKE estado_elemento]`	Devuelve información sobre una serie de elementos del sistema, como el número de subprocesos que se están ejecutando. La cláusula `LIKE` se utiliza para buscar coincidencias en los nombres de estos elementos; por ejemplo, `'Thread%'` recupera los elementos `'Threads_cached'`, `'Threads_connected'` y `'Threads_running'`.
`SHOW VARIABLES [LIKE nombre_de_variable]`	Muestra los nombres y los valores de las variables del sistema MySQL, como el número de versión. La cláusula `LIKE` se puede utilizar para buscar correspondencias de forma similar a `SHOW STATUS`.
`SHOW [FULL] PROCESSLIST`	Muestra todos los procesos que se están ejecutando en el sistema. La mayor parte de los usuarios verán sus subprocesos pero si disponen del privilegio `PROCESS` también verán los procesos de todos los demás usuarios, incluidas sus contraseñas si aparecen en las consultas. Las consultas se dividen en secciones de 100 caracteres de manera predeterminada. La palabra clave `FULL` muestra las consultas completas.
`SHOW TABLE STATUS [FROM base_de_datos] [LIKE base_de_datos]`	Muestra información sobre cada una de las tablas de la base de datos actualmente en uso o de la base de datos llamada `base_de_datos` si se especifica, con la posibilidad de buscar coincidencias con comodines. Entre la información se incluye el tipo de tabla y cuándo se actualizó por última vez.
`SHOW GRANTS FOR usuario`	Muestra la instrucción `GRANT` requerida para recuperar el nivel actual de privilegio asociado al `usuario`.

Cómo obtener información con DESCRIBE

Como alternativa a la instrucción `SHOW COLUMNS`, se puede utilizar la instrucción `DESCRIBE`, que es similar a la instrucción `DESCRIBE` de Oracle (otro RDBMS).

Su sintaxis básica es

```
DESCRIBE tabla [columna];
```

Esta instrucción devolverá información sobre todas las columnas de la tabla o sobre una columna dada. Puede utilizar comodines en el nombre de la columna.

Compresión del funcionamiento de las consultas con EXPLAIN

La instrucción `EXPLAIN` se puede utilizar de dos formas. En primer lugar, puede utilizar

```
EXPLAIN   tabla;
```

El resultado de esta instrucción es bastante similar al devuelto por las instrucciones `DESCRIBE tabla` o `SHOW COLUMNS FROM tabla`.

La segunda forma de utilizar `EXPLAIN` es mucho más interesante ya que permite ver cómo evalúa MySQL una consulta `SELECT`.

Para utilizarla de esta forma, basta con poner la palabra `explain` delante de una instrucción `SELECT`.

Puede utilizar la instrucción `EXPLAIN` cuando esté intentando probar una consulta compleja y el resultado no es el esperado o si tarda más de lo esperado en completarse. Si está escribiendo una consulta compleja, puede probarla antes ejecutando el comando `EXPLAIN`. Con el resultado obtenido, puede modificar la consulta para optimizarla si resultara necesario. Se trata también de una práctica herramienta de aprendizaje.

Por ejemplo, intente ejecutar la siguiente consulta sobre la base de datos Book-O-Rama. El resultado se recoge en la figura 11.2.

```
explain
select  customers.name
from  customers,  orders,  order_items,  books
where  customers.customerid  =  orders.customerid
and  orders.orderid  =  order_items.orderid
and  order_items.isbn  =  books.isbn
and  books.title  like  '%Java%';
```

```
+-------------+--------+--------------+---------+---------+-----------------+------+-------------+
| table       | type   | possible_keys | key     | key_len | ref             | rows | Extra       |
+-------------+--------+--------------+---------+---------+-----------------+------+-------------+
| orders      | ALL    | PRIMARY      | NULL    |    NULL | NULL            |    4 |             |
| order_items | ref    | PRIMARY      | PRIMARY |       4 | orders.orderid  |    1 | Using index |
| customers   | ALL    | PRIMARY      | NULL    |    NULL | NULL            |    3 | where used  |
| books       | eq_ref | PRIMARY      | PRIMARY |      13 | order_items.isbn |    1 | where used  |
+-------------+--------+--------------+---------+---------+-----------------+------+-------------+
```

Figura 11.2. El resultado de la instrucción EXPLAIN

El resultado puede parecer confuso al principio pero resulta muy útil. Vamos a examinar las columnas de esta tabla por separado.

La primera columna, `table`, enumera simplemente las tablas utilizadas para responder a la consulta. Cada fila del resultado ofrece información sobre cómo se ha utilizado dicha tabla en la consulta. En este caso, las tablas utilizadas son `orders`, `order_items`, `customers` y `books`.

La columna `type` explica cómo se está utilizando la tabla en las combinaciones de la consulta. En la tabla 11.7 se recogen los distintos valores que puede tener esta columna. Estos valores se enumeran del más rápido al más lento en términos de ejecución de la consulta y proporciona una idea de la cantidad de filas que es necesario leer de cada tabla para ejecutar una consulta.

Tabla 11.7. Posibles tipos de combinaciones mostradas por EXPLAIN

Tipo	Descripción
`const` o `system`	La tabla se lee una sola vez. Esto ocurre cuando la tabla consta de una sola fila. El tipo `system` se utiliza cuando se trata de una tabla del sistema y el tipo `const` en el resto de los casos.
`eq_ref`	Para cada conjunto de filas del resto de tablas de la combinación, leemos una fila de esta tabla. Este tipo se utiliza cuando la combinación usa todas las partes del índice de la tabla y el índice es exclusivo o es la clave principal.
`ref`	Para cada conjunto de filas del resto de tablas de la combinación, leemos un conjunto de filas de esta tabla que coinciden en su totalidad. Este tipo se utiliza cuando la combinación no puede seleccionar una única fila en función de la condición de combinación, es decir, si sólo se utiliza parte de la clave en la combinación o si no es exclusiva o una clave principal
`range`	Para cada conjunto completo de filas del resto de las tablas de la combinación, se lee un conjunto de filas de esta tabla incluidas en un rango dado.
`index`	Se examina el índice completo.
`ALL`	Se examinan todas las filas de la tabla.

En el ejemplo anterior, puede ver que una de las tablas se combina con `eq_ref (books)`, que otra utiliza el tipo `ref(order_items)` y que las otras dos (`orders` y `customers`) se combinan con `ALL`; es decir, examinando todas las filas de la tablas.

La columna `rows` confirma estas combinaciones: lista aproximadamente el número de filas de cada tabla que deben examinarse para realizar la combinación. Si multiplica estos valores, obtendrá el número total de filas que se examinan al llevar a cabo una consulta. Estos números se multiplican porque una combinación es como el producto de las filas de diferentes tablas (si desea obtener más detalles al

respecto, consulte un capítulo anterior). Recuerde que se trata del número de filas examinado, no el número de filas devuelto y es una estimación. MySQL no puede saber el número exacto sin llevar a cabo la consulta.

Obviamente, cuanto más pequeño sea este número mejor. En estos momentos el conjunto de datos de nuestra base de datos resulta insignificante, pero cuando empiece a crecer, el tiempo de ejecución se disparará. Volveremos sobre este tema en un instante.

La columna `possible_keys` recoge las claves que MySQL podría utilizar para combinar la tabla. En este caso, se trata de las claves principales.

La columna `key` es la clave desde la que se utiliza la tabla MySQL o `NULL` si no se ha utilizado ninguna clave. Como observará, aunque las tablas `orders` y `customers` incluyen claves principales, no se utilizan en esta consulta. Veremos cómo resolver este problema en un instante.

La columna `key_len` indica la longitud de la clave utilizada. Puede utilizarla para determinar si sólo se está utilizando parte de una clave, lo cual resulta pertinente cuando se tienen claves compuestas de varias columnas. En este caso, se utilizan las claves completas allí donde se utilizan claves (`order_items` y `books`).

La columna `ref` muestra las columnas utilizadas con la clave al seleccionar filas de la tabla.

Por último, la columna `Extra` indica cualquier otra información sobre cómo se establece la combinación. En la tabla 11.8 se recogen los valores posibles que pueden aparecer en esta columna.

Tabla 11.8. Los valores posibles de la columna devueltos por la instrucción EXPLAIN

Valor	Significado
Not exists	La consulta se ha optimizado para utilizar `LEFT JOIN`.
Range checked for each record	Intenta buscar el mejor índice, si existiese, para cada fila del conjunto de filas del resto de las tablas de la combinación.
Using filesort	Serán necesarias dos barridos para ordenar los datos. (Por lo tanto, la operación resultará el doble de larga).
Using index	Toda la información de la tabla procede del índice, es decir, no se examinan las filas.
Using temporary `WHERE` used	Se utiliza una cláusula `WHERE` para seleccionar las filas.

Existen varias formas de solucionar los problemas descubiertos por `EXPLAIN`.

En primer lugar, compruebe los tipos de columnas y asegúrese de que son iguales. En concreto, fíjese en el ancho de las columnas. Los índices no se pueden utilizar para comparar columnas si tienen diferentes anchos. Para resolver este problema, cambie los tipos de columna de forma que coincidan o intégrelos en su diseño desde el principio.

En segundo lugar, puede pedirle al optimizador de combinaciones que examine las distribuciones de clave y que optimice las combinaciones de forma más eficaz por medio de la utilidad `myisamchk`. Para invocarla, escriba

```
myisamchk  -analyze   ruta_a_la_base_de_datos_mysql/tabla
```

Puede comprobar varias tablas si las incluye todas en la línea de comandos o si utiliza

```
myisamchk  -analyze   ruta_a_la_base_de_datos_mysql/*.MYI
```

Puede comprobar todas las tablas de todas las bases de datos ejecutando la siguiente línea, lo que generará el resultado ilustrado en la figura 11.3:

```
myisamchk  -analyze   ruta_al_la_directorio_de_datos_mysql/*/*.MYI
```

table	type	possible_keys	key	key_len	ref	rows	Extra
books	ALL	PRIMARY	NULL	NULL	NULL	4	where used
order_items	index	PRIMARY	PRIMARY	17	NULL	5	where used; Using index
orders	eq_ref	PRIMARY	PRIMARY	4	ordsr_items.orderid	1	
customers	eq_ref	PRIMARY	PRIMARY	4	orders.customerid	1	

Figura 11.3. Resultado devuelto por EXPLAIN tras ejecutar myisamchk

Como observará, la forma en la que se evalúa la consulta ha variado bastante. Ahora sólo utilizamos todas las filas de una de las tablas (`books`). En concreto, utilizamos `eq_ref` para dos de las tablas e `index` para las otras. MySQL también utiliza ahora la clave completa para `order_items` (17 caracteres en lugar de los 4 que utilizaba antes).

También ha aumentado el número de filas utilizado, lo que probablemente se debe a que la base de datos incluye pocos datos. Recuerde que el número de filas indicado no es más que una estimación. Puede realizar la consulta y comprobar si coinciden. Si los números varían mucho, el manual de MySQL sugiere utilizar una combinación directa y enumerar las tablas en la cláusula `FROM` en orden diferente.

En tercer lugar, podemos considerar la posibilidad de agregar un nuevo índice a la tabla, si la consulta es a) lenta y b) común.

Si se trata de una consulta especial que es probable que no vuelva a utilizar, no merecerá la pena el esfuerzo ya que ralentizará otros procesos. En la siguiente sección veremos cómo agilizar las consultas.

Cómo agilizar consultas con índices

Si se encuentra en una situación como la descrita anteriormente, en la que la columna `possible_keys` devuelta por una instrucción `EXPLAIN` contiene valores `NULL`, podría mejorar su rendimiento si agrega un índice a la tabla en cuestión. Si la columna que está utilizando en la cláusula `WHERE` resulta idónea para su indexación,

puede crear un nuevo índice para ella con ayuda de la instrucción ALTER TABLE de la siguiente forma:

```
ALTER TABLE tabla ADD INDEX (columna);
```

Trucos de optimización general

Además de los trucos vistos antes para la optimización de consultas, existen otros que permiten incrementar el rendimiento de las bases de datos MySQL.

Optimización del diseño

Es aconsejable que todos los elementos de la base de datos tengan el menor tamaño posible. Para ello, se debe minimizar la redundancia en la fase de diseño. También se puede lograr el mismo objetivo asignando los tipos de datos más pequeños a las columnas.

También debería reducir al mínimo los valores NULL y seleccionar una clave principal que sea lo más corta posible.

Evite el uso de columnas de longitud variable siempre que pueda (como VARCHAR, TEXT y BLOB).

Si sus tablas tienen campos de longitud fija, resultarán más rápidas de utilizar aunque absorban un poco más de espacio.

Permisos

Además de aplicar las sugerencias propuestas en la sección dedicada a EXPLAIN, puede mejorar la velocidad de las consultas simplificando los permisos. Ya indicamos anteriormente la forma en la que se comprueban las consultas con el sistema de permisos antes de su ejecución. Cuanto más sencillo resulte este proceso, más rápido se ejecutará la consulta.

Optimización de tablas

Si lleva utilizando una tabla durante un periodo largo de tiempo, los datos pueden fragmentarse al procesar las actualizaciones y las eliminaciones. Como consecuencia, el proceso de búsqueda puede alargarse. Para solucionar este problema puede utilizar la instrucción

```
OPTIMIZE TABLE nombre_de_tabla;
```

o escribir

```
myisamchk -r table
```

en la línea de comandos.

También puede recurrir a la utilidad `myisamchk` para ordenar un índice de tabla y los datos en función de dicho índice, de la siguiente forma:

```
myisamchk  -sort-index  -sort-records=1
ruta_al_la_directorio_de_datos_mysql/*/*.MYI
```

Uso de índices

Utilice índices siempre que lo necesite para aumentar la velocidad de las consultas. Intente que resulten sencillos y no cree índices que sus consultas no vayan a utilizar. Para comprobar los índices que se están utilizando ejecute la instrucción `EXPLAIN` como se indicó anteriormente.

Uso de valores predeterminados

Siempre que le resulte posible, utilice valores predeterminados e inserte únicamente los datos si difieren de los predeterminados. De esta forma, se reducirá el tiempo que tarda la instrucción `INSERT` en ejecutarse.

Uso de conexione permanentes

Este consejo resulta especialmente de aplicación para las bases de datos Web. Nuestra intención aquí es recodárselo ya que se comentó en una sección anterior

Otras sugerencias

Existen muchos otros trucos que puede aplicar para mejorar el rendimiento en situaciones concretas y para resolver necesidades particulares. En el sitio Web de MySQL encontrará una gran cantidad de ellos. Diríjase a

```
http://www.mysql.com
```

Tipos diferentes de tabla

Antes de terminar con el estudio de MySQL, es necesario analizar los distintos tipos de tablas. Para seleccionar el tipo de tabla al crearla utilice

```
CREATE TABLE tabla TYPE=tipo ....
```

Los posibles tipos de tabla son

- `MyISAM`. Éste es el tipo predeterminado y el que hemos estado utilizando hasta ahora. Se basa en `ISAM`, que equivale en inglés a Método de acceso secuencial indexado, un método estándar para almacenar los registros y los archivos.

- ISAM. Descrito arriba.
- HEAP. Las tablas de este tipo se almacenan en memoria y sus índices se cifran. Como resultado las tablas HEAP son extremadamente rápidas pero presentan el inconveniente de que si tiene lugar un fallo en el sistema, los datos se perderán. Las tablas HEAP resultan ideales para almacenar datos temporales. Es aconsejable especificar el número máximo de filas en la instrucción CREATE TABLE o las tablas pueden absorber toda la memoria. Por último, las tablas HEAP no pueden incluir columnas de tipo BLOB, TEXT o AUTO INCREMENT.
- BDB. Estas tablas garantizan la seguridad de las transacciones, es decir, incluyen las funciones COMMIT y ROLLBACK. Son más lentas que las tablas MyISAM, pero proporcionan todas las ventajas del uso de transacciones. Estas tablas se basan en Berkeley DB.
- InnoBD. Estas tablas también garantizan la seguridad de las transacciones y se les aplican los mismos comentarios de las tabas BDB.

Estos tipos de tablas pueden resultar útiles cuando se necesita una mayor velocidad o garantizar la seguridad de las transacciones.

Si desea utilizar los tipos BDB o InnoDB, debería seleccionar el tipo binario MySQL-Max que se incluye en la distribución de MySQL, en lugar del tipo binario MySQL normal.

Carga de datos desde un archivo

Una función útil de MySQL de la que todavía no hemos hablado es la instrucción LOAD DATA INFILE.

Esta instrucción se puede utilizar para cargar datos en la tabla procedentes de un archivo.

Se ejecuta de forma muy rápida.

Se trata de un comando flexible con muchas opciones, pero se suele utilizar de la siguiente forma:

```
LOAD DATA INFILE "newbooks.txt" INTO TABLE books;
```

Esta instrucción lee las filas del archivo newbooks.txt dentro de la tabla books. De manera predeterminada, los campos de datos del archivo deben separarse mediante tabuladores y encerrarse en comillas simples y cada fila debe separarse mediante un carácter de nueva línea (\n). Los caracteres especiales deben marcarse con la barra inclinada (\). Todas estas características se puede configurar mediante varias opciones de la instrucción LOAD (si desea obtener más detalles consulte el manual de MySQL).

Para usar la instrucción LOAD DATA INFILE, es necesario disponer del privilegio FILE comentado anteriormente.

Cómo realizar una copia de seguridad de la base de datos MySQL

MySQL incorpora dos formas de realizar una copia de seguridad.

La primera consiste en bloquear las tablas mientras se copian los archivos físicos, utilizando el comando `LOCK TABLES`. Su sintaxis es la siguiente.

```
LOCK TABLES tabla tipo_de_bloqueo [, tabla tipo_de_bloqueo...]
```

El parámetro *tabla* debería sustituirse por el nombre de una tabla y el *tipo_de_bloqueo* debería ser `READ` o `WRITE`. Para realizar una copia de seguridad sólo se necesita un bloqueo `READ`. Los usuarios y las secuencias de comandos pueden ejecutar consultas de sólo lectura durante el volcado. Si tiene un conjunto razonable de consultas que alteren la base de datos, como pedidos de cliente, esta solución no resulta práctica.

El segundo método es más avanzado y utiliza el comando `mysql_dump`. Se suele utilizar de la siguiente forma:

```
mysqldump -opt -all-databases > all.sql
```

Este comando volcará un conjunto de todo el SQL necesario para volver a construir la base de datos en el archivo `all.sql`.

A continuación, debería detener el proceso `mysqld` durante un instante y volver a reiniciarlo con la opción `-log-update[=archivo_de_registro]`. Las actualizaciones almacenadas en el archivo de registro indicarán los cambios realizados desde la realización del volcado. (Obviamente, debería volcar los archivos de registros en un archivo de copia de seguridad normal.)

Restablecimiento de la base de datos MySQL

Si necesita restablecer una base de datos MySQL, dispondrá nuevamente de dos opciones. Si el problema es una tabla de datos dañada, puede ejecutar `myisamchk` con la opción `-r` (reparar).

Si ha utilizado el primer método de volcado de seguridad, puede copiar los archivos de datos en la misma ubicación dentro de una nueva instalación de MySQL.

Si ha utilizado el segundo método de volcado, el proceso de restablecimiento constará de dos fases. En primer lugar, tendrá que ejecutar las consultas en el archivo de volcado para reconstruir la base de datos hasta el punto de volcado del archivo. En segundo lugar, deberá actualizar la base de datos desde el punto almacenado en los archivos de registro. En UNIX puede ejecutar un comando como el siguiente:

```
ls -l -t -r hostname.[0-9]* | xargs cat | mysql
```

para procesar los archivos de registro en el orden correcto.

Si desea obtener más información sobre el proceso de volcado y recuperación de MySQL, consulte el sitio Web de MySQL en:

```
http://www.mysql.org
```

Lecturas adicionales

En estos capítulos sobre MySQL, nos hemos centrado en los usos y las partes del sistema más relevantes para el desarrollo Web y en la combinación de MySQL con PHP.

Si desea saber más al respecto, en especial en relación a aplicaciones que no sean Web, o sobre la administración de MySQL, puede visitar el sitio Web de MySQL en

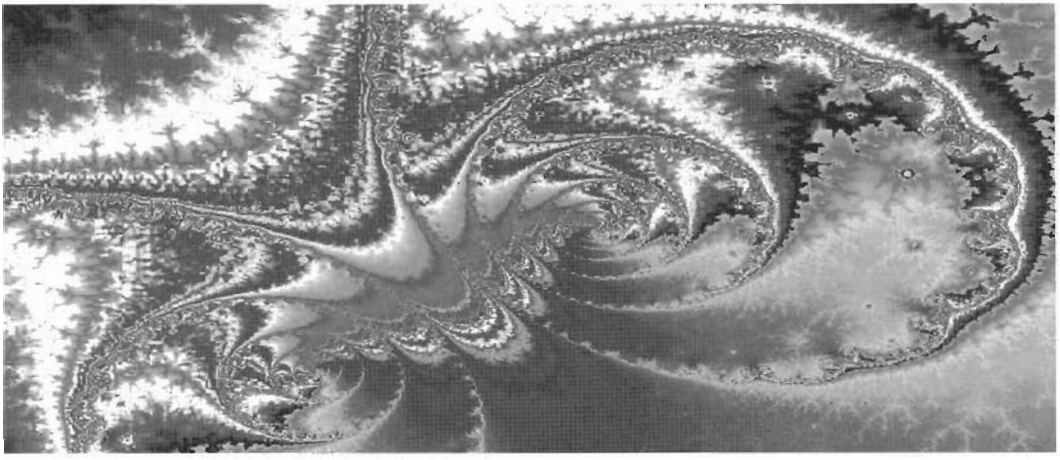
```
http://www.mysql.com
```

A continuación

Ya hemos analizado los conceptos básicos de PHP y MySQL. En el siguiente capítulo examinaremos los aspectos de seguridad y comercio electrónico relacionados con la configuración de sitios Web basados en bases de datos.

Parte III

Comercio electrónico y seguridad



12
Creación de un sitio Web de comercio electrónico

En este capítulo se presentan algunos aspectos relacionados con la especificación, diseño, construcción y mantenimiento de un sitio Web de comercio electrónico de manera eficaz. Examinaremos el plan, los posibles riesgos y varias formas de lograr que un sitio Web resulte autosuficiente.

Examinaremos los siguientes aspectos:

- Qué se puede conseguir con un sitio de comercio electrónico
- Tipos de sitios Web comerciales
- Riesgos y amenazas
- Por qué estrategia optar

¿Cuál es nuestro objetivo?

Antes analizar los detalles de implementación de su sitio Web, debería tener claro sus objetivos y disponer de un plan razonablemente pormenorizado para alcanzarlos.

En este libro, partimos del supuesto de que está desarrollando un sitio Web comercial y que, por lo tanto, uno de sus objetivos es ganar dinero.

Existen muchos enfoques comerciales aplicables a Internet. Puede que desee anunciar sus servicios tradicionales o vender en línea un producto del mundo real.

Quizás tenga un producto que se puede vender y distribuir por la red. O puede que no tenga intención de utilizar el sitio Web para generar ingresos, sino como medio para fomentar las actividades fuera de línea o como una alternativa menos costosa a las actividades presentes.

Tipos de sitios Web comerciales

Los sitios Web comerciales realizan una o varias de las actividades siguientes:

- Publicar información sobre la compañía a través de folletos en línea
- Recoger pedidos de artículos o servicios
- Suministrar servicios o artículos digitales
- Añadir valor a los artículos o servicios
- Reducir costes

Las secciones de muchos sitios Web encajan en varias categorías. A continuación, se incluye una descripción de cada una de ellas y la forma habitual en la que generan ingresos u obtienen otros beneficios para una organización.

La intención de esta sección del libro es ayudarle a formular sus objetos. ¿Por qué quiere un sitio Web? ¿Cómo va a contribuir cada función de su sitio Web al negocio?

Medios publicitarios en línea

Prácticamente todos los sitios Web comerciales de principios de los años 90 no eran más que una folleto en línea o una herramienta de venta. Este tipo de sitios sigue siendo la forma más habitual adoptada por los sitios Web comerciales. Ya sea como incursión en la Web o como ejercicio de publicidad de bajo coste, este tipo de sitios siguen teniendo su lógica para muchos negocios.

La presencia publicitaria en Internet puede adoptar distintas formas, desde una tarjeta de visita convertida en página Web a una colección de información publicitaria. En cualquier caso, el objetivo del sitio y la razón financiera de su existencia es atraer a los clientes para que entren en contacto con nuestro negocio.

Este tipo de sitios no genera ningún ingreso directamente, pero puede contribuir a los beneficios obtenidos a través de los medios tradicionales.

El desarrollo de un sitio de este tipo presenta pocos retos técnicos. Los aspectos sobre los que centrarse son similares a los de otros ejercicios de publicidad. Entre los errores más habituales que se suelen cometer al desarrollar este tipo de sitios se incluyen los siguientes:

- No suministrar información importante
- Mala presentación

- No responder a la información generada por el sitio
- No actualizar el sitio
- No rastrear el éxito del sitio

No suministrar información importante

¿Qué es lo que buscan los visitantes que acuden a su sitio? Todo depende de lo que ya sepan. Puede que busquen información técnica detallada sobre el producto o información muy básica como datos de contacto.

La información que suministran muchos sitios Web no resulta útil o no incluyen los datos más importantes. Como mínimo, el sitio debe indicar a los visitantes a qué nos dedicamos, en qué áreas geográficas del mundo se proporcionan nuestros servicios y cómo pueden ponerse en contacto con nosotros.

Mala presentación

En Internet, una empresa pequeña puede parecer de mayor tamaño y resultar muy llamativa y, al contrario, una empresa de gran tamaño puede parecer pequeña, poco profesional y poco llamativa si el sitio Web no está bien diseñado.

Con independencia del tamaño que tenga su empresa, asegúrese de que su sitio Web sea de calidad. El texto debe estar escrito y revisado por alguien con un buen conocimiento del lenguaje correcto. Los gráficos deben estar limpios, deben ser claros y deben tardar poco tiempo en descargarse. En un sitio comercial, debería tener cuidado al elegir los gráficos y los colores de manera que encajen con la imagen que se desea transmitir. Si utiliza efectos de animación y sonido procure hacerlo con cuidado.

Aunque no logrará que su sitio se genere de la misma forma en todos los equipos, sistemas operativos y navegadores, asegúrese de que se visualiza y de que no produzca errores generalizados.

No responder a la información generada por el sitio Web

En la Web, un buen servicio de asistencia al cliente resulta tan importante para atraer y mantener clientes como en el mundo exterior. Las compañías, ya sean pequeñas o grandes, son responsables de colocar una dirección de correo electrónico en una página Web y de no examinar o responder a los mensajes de correo rápidamente.

La gente suele tener diferentes expectativas sobrelos tiempos de respuesta a los mensajes de correo electrónico con respecto al correo postal. Si no examina y responde al correo diariamente, los usuarios creerán que no se concede la importancia necesaria a sus consultas.

Las direcciones que se incluyen en páginas Web deberían ser genéricas e ir dirigidas a un puesto o departamento en lugar de a una persona concreta. ¿Qué ocurrirá con el correo enviado a `fred.smith@ejemplo.com` cuando se vaya su titular? Es más probable que el correo dirigido a `ventas@ejemplo.com` pase a su

sucesor. También podría distribuirse a un grupo de personas, lo que contribuiría a garantizar su pronta contestación.

No actualizar el sitio

Debe procurar que su sitio Web esté actualizado. Para ello, es necesario cambiar los contenidos periódicamente. Los cambios en la organización deben reflejarse en el sitio. Un sitio Web con telarañas desanimará a los visitantes a volver y contribuirá a hacerles creer que gran parte de la información podría ser incorrecta.

Una forma de evitar que un sitio parezca obsoleto consiste en actualizar sus páginas manualmente. También puede utilizar un lenguaje de secuencia de comandos como PHP para crear páginas dinámicas. Si su secuencia de comandos dispone de acceso a información actualizada, podrá generar páginas que estén al día de manera constante.

No realizar el seguimiento del éxito del sitio

Crear un sitio Web es muy buena idea, pero ¿cómo justificar el esfuerzo y el gasto? Llegará un momento, en especial si se trata del sitio de una gran compañía, en el que se le pedirá que demuestre o cuantifique su valor para la organización.

En las campañas de marketing tradicionales, las grandes organizaciones invierten mucho dinero en estudios de mercado, antes y después de lanzar la campaña, para comprobar su eficacia. En función de la escala y del presupuesto asignado a la iniciativa Web, estos indicadores podrían resultar igualmente apropiados para el diseño del sitio o para cuantificar su valor. Entre las opciones disponibles, más caras o más baratas, se incluyen las siguientes:

- **Examinar los registros del servidor**: los servidores Web almacenan una gran cantidad de datos sobre cada petición realizada al servidor. Gran parte de estos datos resultan de poca utilidad y su formato sin procesar tampoco es de gran ayuda. Para destilar los archivos de registros y obtener un resumen significativo de los datos, es necesario utilizar un analizador de archivos de registros. Dos de los más conocidos y gratuitos son Analog, disponible en la `http://www.statslab.cam.ac.uk/~sret1/analog`, y Webalizer, disponible en `http://www.mrunix.net/webalizer/`. También existen programas comerciales como `Summary`, disponible en `http://summary.net`, que podrían resultar más completos. Los analizadores muestran cómo varía el tráfico de un sitio Web con el tiempo y qué paginas se ven.

- **Monitorización de las ventas**: el objetivo de los sitios Web con funciones publicitarias es generar ventas. Debería poder estimar el efecto sobre las ventas comparándolo con los niveles de ventas antes del lanzamiento del sitio. Esta operación se complica si se han aplicado otras acciones de marketing en el mismo periodo.

- **Solicitar información a los usuarios**: si pregunta a sus usuarios, éstos le dirán lo que piensan del sitio. Puede obtener información útil mediante un formula-

rio o una dirección de correo electrónico con la que recoger los comentarios de los usuarios. Para animar a los usuarios a enviar información puede incluir un incentivo, como la participación en un sorteo.

- **Análisis de usuarios representativos**: la creación de grupos de opinión puede convertirse en una técnica eficaz para evaluar un sitio o incluso en un prototipo del sitio previsto. Para dirigir un grupo de opinión sólo necesitará reunir algunos voluntarios, animarles a valorar el sitio y entrevistarles para obtener y registrar sus opiniones.

Los grupos de opinión pueden resultar caros si su dirección se delega en moderadores profesionales encargados de valorar y filtrar a los participantes con el objetivo de garantizar una representación demográfica y de personalidad exacta dentro de una comunidad amplia y , a continuación, entrevistar hábilmente a los participantes. Pero el coste de los grupos de opinión también puede ser nulo, si su dirección se deja en manos de un amateur y si se utiliza un conjunto de personas cuya relevancia con respecto al mercado de destino sea desconocida.

Una forma de obtener un grupo de opinión bien dirigido consiste en recurrir a los servicios de una compañía especializada en investigaciones de mercado, pero no es la única. Si dirige su propio grupo de opinión, procure seleccionar un moderador hábil. Se deben valorar especialmente sus cualidades de comunicación e imparcialidad ante los resultados. Limite el tamaño de los grupos a conjuntos de entre seis y diez personas. El moderador debería estar apoyado por una persona encargada de registrar los datos o por una secretaria que le permita centrarse en la entrevista. La validez de los resultados obtenidos de los grupos se corresponderá con al muestra de gente utilizada. Si utiliza a amigos o a los familiares de sus empleados para valorar el producto, es poco probable que representen a la comunidad general.

Recogida de pedidos de artículos y servicios

Si la publicidad del sitio resulta atrayente, el siguiente paso lógico consistirá en permitir que sus clientes realicen el pedido por Internet. Los profesionales de las ventas saben que es importante lograr que los clientes tomen una decisión de manera inmediata. Cuanto más tiempo se conceda al cliente para reconsiderar la decisión de compra, mayor será la probabilidad de que mire otras ofertas y cambie de opinión. Si un cliente quiere su producto, la mejor estrategia es permitirle realizar la compra de la forma más fácil y rápida posible. Si obligamos a la gente a desconectar el módem para llamar por teléfono y realizar un pedido o a acudir a una tienda, lo que estaremos haciendo es poner obstáculos a la compra. Si la publicidad en línea ha convencido a un visitante a comprar un producto, permítale que realice la compra de inmediato sin abandonar el sitio Web.

La recogida de pedidos a través de un sitio Web es una buena idea para muchas iniciativas comerciales. Todos los negocios quieren pedidos. La posibilidad de realizar pedidos en línea permite aumentar las ventas o reducir la carga sobre el personal de ventas. Obviamente, también se incurrirá en unos costes. La construcción de un

sitio Web dinámico, la organización de la infraestructura de pago y el suministro de un servicio de atención al cliente suponen gastos. Intente determinar si sus productos resultan adecuados para su comercialización a través de un sitio de comercio electrónico.

Entre los productos que se suelen comprar a través de Internet se incluyen libros y revistas, programas y equipos informáticos, música, ropa, viajes y entradas a eventos de entretenimiento.

Si su producto no se incluye en ninguna de estas categorías, no desespere. Estas categorías están atestadas de marcas establecidas. Sin embargo, resulta aconsejable considerar alguno de los factores que explican por qué estos productos se venden tan bien por Internet.

De manera ideal, los productos que se venden en un comercio electrónico son imperecederos, sencillos de transportar y lo suficientemente caros como para que los gastos de envíos resultan razonables aunque no demasiado caros como para que el comprador sienta la necesidad de examinarlos físicamente antes de comprarlos.

Los mejores productos para su comercialización a través de un comercio electrónico son los bienes de consumo. Si quiere comprar un aguacate, es probable que desee examinarlo visualmente antes e incluso tocarlo. No todos los aguacates son iguales. La copia de un libro, de un CD o de un programa informático suele ser idéntica a otras copias con el mismo título. Los compradores no necesitan ver el artículo concreto antes de comprarlo.

Además, los productos comercializados a través de comercios electrónicos deben atraer a la gente que utiliza Internet. En el momento de escribir este libro, esta audiencia se compone fundamentalmente de adultos con empleo, jóvenes, con ingresos medios altos y asentados en zonas urbanas. Con el tiempo, sin embargo, los usuarios de Internet abarcarán a toda la población.

Algunos productos nunca aparecerán reflejados en los estudios sobre ventas por Internet, pero son un éxito. Si tiene un producto que atrae únicamente a un nicho del mercado, Internet puede ser un medio fantástico para obtener comparadores.

Algunos productos tienen pocas probabilidades de convertirse en categorías de comercio electrónico. Los artículos baratos y perecederos, como los comestibles, son una mala opción, aunque esto no ha disuadido a las compañías a intentarlo, aunque sin mucho éxito. Existen categorías que encajan perfectamente en sitios publicitarios pero que resultan óptimas para su venta en línea. Por ejemplo, artículos grandes y caros, como vehículos o propiedades inmobiliarias que requieren mucha investigación antes de su compra, pero que resultan demasiado caros para servirlos en forma de pedidos y difíciles de transportar.

Existen varios obstáculos para convencer a un comprador potencial de que rellene un pedido. A saber:

- Preguntas sin respuesta
- Confianza
- Facilidad de uso
- Compatibilidad

Si un usuario se ve frustrado por cualquiera de estos obstáculos, es probable que abandone el sitio sin comprar.

Preguntas sin respuesta

Si un usuario potencial no encuentra una respuesta inmediata a una pregunta, es probable que abandone el sitio. Este hecho trae consigo una serie de implicaciones. Asegúrese de que el sitio está bien organizado. ¿Resulta sencillo buscar lo que se desea cuando se visita el sitio por primera vez? Asegúrese de que el sitio resulta completo pero sin abrumar a los visitantes. En la Web, los usuarios suelen examinar rápidamente el contenido en lugar de leerlo atentamente, por lo que es aconsejable ser conciso. En la mayor parte de los medios publicitarios, existen límites prácticos en cuanto a la cantidad de información que se puede incluir. No ocurre así en el caso de un sitio Web. En este caso, los dos límites principales son el coste de crear y actualizar la información y las restricciones impuestas por la capacidad para organizar, distribuir y conectar la información sin abrumar a los visitantes.

Resulta tentador imaginar un sitio Web como un comercial que recoge los pedidos automáticamente, que nunca duerme y que no reciben ningún salario. Sin embargo, el servicio de asistencia al cliente sigue siendo importante. Anime a los visitantes a formular preguntas. Intente suministrar respuestas inmediatas o prácticamente inmediatas por teléfono, por correo electrónico o a través de cualquier otro medio oportuno.

Confianza

Si un visitante no está familiarizado con una marca, ¿por qué debería fiarse de nosotros? Todo el mundo puede crear un sitio Web. La gente no tiene por qué fiarse de lo escrito en el sitio Web y la realización de un pedido requiere una determinada confianza. ¿Cómo distingue el visitante entre una organización con buena reputación o una empresa sin escrúpulos?

A la gente le preocupa una serie de cosas al comprar en la red:

- ¿Qué se va a hacer con su información personal? ¿Se va a vender a otros, se va a utilizar para enviar enormes cantidades de publicidad o se va almacenar en un lugar poco seguro para que otros puedan acceder a ella? Es importante indicar a la gente qué es lo que se va a hacer con sus datos. Es lo que se conoce como política de privacidad y debería resultar fácil de acceder a ella.
- ¿Su negocio es respetable? Si su empresa está registrada en una entidad con autoridad relevante, tiene una dirección física y un número de teléfono, y lleva varios años funcionando, es menos probable que se trate de un engaño que un negocio compuesto únicamente por un sitio Web y quizás una dirección postal. Asegúrese de mostrar estos detalles.
- ¿Qué ocurre si un comprador no está satisfecho con su compra? ¿En qué circunstancias se procederá a realizar un reembolso? ¿Quién se hace cargo de los gastos de envío? Tradicionalmente, las empresas dedicadas a la venta por

correo aplican políticas de reembolso más liberales que los establecimientos tradicionales. Muchos ofrecen una garantía de satisfacción sin condiciones. Tenga en cuenta los costes de las devoluciones con respecto al aumento de las ventas que generará una política general de devolución. En cualquier caso, asegúrese de indicar la política aplicada en el sitio.

- ¿Deberían los clientes confiarle la información de sus tarjetas de crédito? El aspecto más importante relacionado con la confianza de los compradores de Internet es el temor a enviar los datos de su tarjeta de crédito por la red. Por esta razón, debe administrar la información sobre tarjetas de crédito de manera segura y recalcar su preocupación al respecto. Esto significa que, como mínimo, debería aplicar el sistema SSL (Secure Sockets Layer) para transmitir los detalles desde el navegador del usuario al servidor Web y asegurarse de que su servidor Web está administrado de manera competente y segura. En una sección posterior ampliaremos estos temas.

Facilidad de uso

Los consumidores varían enormemente en cuanto a su experiencia con los ordenadores, idioma, nivel de alfabetización, memoria y visión. Debe intentar que su sitio resulte lo más sencillo posible de utilizar. Aunque existe una gran cantidad de libros dedicados al diseño de la interfaz de usuario, a continuación se recogen varias directrices:

- **Intente mantener la sencillez del sitio al máximo.** Cuantas más opciones, anuncios o elementos de distracción se incluyan en la pantalla, más probable será confundir al usuario.
- **Intente que el texto resulte claro.** Utilice fuentes claras y sencillas. No utilice tamaños demasiado reducidos y tenga en cuenta que en cada equipo se representará con tamaños diferentes.
- **Asegúrese de simplificar al máximo el proceso de realización de pedidos.** La intuición y las pruebas disponibles avalan la idea de que cuantos más clics del ratón tenga que aplicar el usuario para realizar un pedido mayor será la posibilidad de que no complete el proceso. Intente reducir al mínimo lo pasos, pero tenga en cuenta que Amazon.com tiene la patente en EEUU sobre un proceso que utiliza un único clic, denominado 1-Click. Esta patente es impugnada enérgicamente por muchos propietarios de sitios Web.
- **Procure que los usuarios no se pierdan.** Incluya puntos de referencia y pistas de navegación para indicar a los usuarios dónde se encuentran. Por ejemplo, si un usuario se encuentra dentro de una subsección del sitio, resalte la señal de navegación correspondiente a dicho sitio.

Si está utilizando la metáfora del carro de la compra en la que se suministra un contenedor en el que los clientes pueden acumular sus compras antes de finalizar la venta, mantenga un vínculo visible al carro en la pantalla constantemente.

Compatibilidad

Asegúrese de probar su sitio en varios navegadores y sistemas operativos. Si el sitio no funciona en un navegador o sistema operativo extendido, el sitio parecerá poco profesional y perderá un porcentaje importante del mercado.

Si su sitio ya es operativo, los registros de su servidor Web pueden indicar qué navegadores utilizan sus visitantes. Como regla general, si prueba su sitio en las dos últimas versiones de Microsoft Internet Explorer y Netscape/Mozilla en un PC que ejecute Microsoft Windows, en las dos últimas versiones de Netscape/Mozilla en un Apple Mac, la versión actual de Netscape/Mozilla de Linux y un navegador de sólo texto como Lynx, su sitio resultará visible para la mayor parte de los usuarios.

Intente evitar funciones y mecanismos nuevos, a menos que le apetezca escribir y mantener varias versiones del sitio.

Suministro de servicios y artículos digitales

Muchos productos y servicios se puede vender a través de la Web y enviar a los clientes por correo. Sin embargo, existen servicios que se pueden entregar de manera inmediata a través de Internet. Si un servicio o artículo se puede trasmitir hasta un módem, se podrá pedir, pagar y entregar de manera instantánea, sin interacción humana.

El servicio de este tipo más obvio es la información. En ocasiones la información resulta completamente gratuita o se alimenta de publicidad. En algunos casos la información se suministra a través de una suscripción o se paga por unidades.

Entre los artículos digitales de este tipo se incluyen libros electrónicos y música en formato electrónico como MP3. Las bibliotecas de imágenes se pueden digitalizar y descargar. El software informático no tiene por qué incluirse siempre en CD. Se puede comprimir y descargar.

Entre los servicios que se pueden vender de esta forma se incluyen el acceso a Internet o servicios de alojamiento Web así como servicios profesionales que se pueden sustituir por un sistema experto.

El envío físico de un artículo solicitado desde un sitio Web presenta ventajas e inconvenientes sobre los artículos y servicios digitales.

El envío de un artículo físicamente tiene un coste mientras que las descargas digitales son prácticamente gratuitas. Esto significa que si tiene algo que se pueda duplicar y vender digitalmente, el coste para su sitio será el mismo si vende un unidad que si vende mil. Por supuesto, existen límites: si el nivel de ventas y el tráfico aumenta, se verá obligado a invertir en hardware o ancho de banda.

Los productos o servicios digitales se pueden vender con facilidad como artículos inmediatos. Si una persona realiza un pedido de un artículo físico tendrá que esperar uno o varios días para recibirlo. Las descargas se suelen medir en segundos o minutos. La inmediatez puede convertirse en una carga para los comerciantes. Si distribuye un producto digitalmente, tendrá que hacerlo de inmediato. No puede supervisar

el proceso manualmente ni distribuir los picos de actividad durante el día. Los sistemas de distribución inmediata resultan por tanto más proclives al fraude y suponen una carga sobre los recursos informáticos.

Los artículos y los servicios digitales son ideales para el comercio electrónico pero, como es obvio, el abanico de productos que se puede distribuir de esta forma es limitado.

Cómo añadir valor a los artículos y servicios

Algunas secciones de sitios Web comerciales no venden ningún artículo ni servicio. Los servicios de seguimientos que incorporan las compañías de mensajería (UPS en `www.ups.com` o Fedex en `www.fedex.com`) no están diseñados para generar ingresos directamente. Añaden valor a los servicios existentes ofrecidos por la organización. Las compañías obtienen una ventaja competitiva al permitir que los clientes puedan saber dónde se encuentra su paquete o acceder a sus cuentas bancarias.

Dentro de esta categoría se incluyen los foros de ayuda. Existen fundadas razones comerciales para ofrecer a los clientes un área de debate en la que compartir sugerencias para resolver problemas sobre los productos de su compañía. Los clientes pueden resolver sus problemas consultando las soluciones dadas a otros, los clientes internacionales puede obtener ayuda sin tener que realizar llamadas internacionales y pueden responder a preguntas de otros clientes fuera de los horarios de oficina. Este tipo de servicios puede contribuir a aumentar la satisfacción de los clientes con un coste muy bajo.

Recorte de costes

Internet se suele utilizar para recortar costes. El ahorro puede proceder de la distribución de la información en línea, de la facilidad para el establecimiento de la comunicación, de la sustitución de servicios o de la centralización de operaciones.

Si necesita distribuir información a una gran cantidad de usuarios, probablemente pueda realizar la misma tarea a través de un sitio Web de manera más económica. Independientemente de que se trate de suministrar listas de precios, catálogos, procedimientos documentados, especificaciones técnicas o cualquier otra información, es probable que resulte más barato colocar la información en la Web que imprimirla y distribuirla en papel, en especial si la información cambia de manera regular. El resultado es el mismo, ya se trate de distribuir ofertas y responder a ellas rápidamente o de facilitar la comunicación directa de cliente con el mayorista o el fabricante, eliminando intermediarios: los precios bajarán o los ingresos crecerán.

La sustitución de servicios que supongan un gasto por su versión electrónica puede reducir los costes. Un buen ejemplo es el de Egghead.com. Los responsables de esta empresa decidieron cerrar sus almacenes de ordenadores y centrarse en actividades de comercio electrónico. Aunque la construcción de un sitio de comercio electrónico de cierta entidad cuesta dinero, el coste de mantener una cadena de más de 70 tiendas

es mucho mayor. La sustitución de un servicio existente tiene sus riesgos. Entre ellos, la pérdida de clientes que no utilicen Internet.

La centralización también puede contribuir a reducir costes. El mantenimiento de una gran cantidad de sedes físicas requiere el pago de rentas y gastos indirectos, el personal que trabaja en ellas y los costes del mantenimiento de inventario en cada sede. Un negocio en Internet puede situarse en un único punto y resultar accesible a todo el mundo.

Riesgos y amenazas

Todos los negocios deben hacer frente a riesgos, competidores, robos, cambios en las preferencias del público y a desastres naturales, entre otros. La lista es interminable. Sin embargo, muchos de los riesgos a los que deben hacer frente las compañías de comercio electrónico son muy inferiores o resultan irrelevantes con respecto a otras iniciativas.

Estos riesgos incluyen:

- Piratas informáticos
- Fracaso en la atracción de suficiente negocio
- Fallos de hardware
- Fallos de alimentación, de comunicación y de red
- Fiabilidad de los servicios de distribución
- Competencia
- Errores de software
- Cambios en las políticas e impuestos gubernamentales
- Límites de la capacidad sistema

Piratas informáticos

Las amenazas más conocidas para los sitios de comercio electrónico proceden de los usuarios de ordenadores malintencionados conocidos como piratas informáticos. Todos los negocios corren el riesgo de convertirse en el objetivo de los criminales, pero no hay duda de que los comercios electrónicos atraen la atención de los piratas informáticos con una variedad de intenciones y habilidades.

Los piratas pueden atacar por el desafío o la fama que supone sabotear un sitio, para robar o para obtener artículos y servicios gratuitos.

La protección de un sitio implica una combinación de tareas:

- Mantener copias de seguridad de la información importante

- Aplicar políticas de contratación que atraigan a personal honesto y mantener su fidelidad (los ataques más peligrosos proceden del interior)
- Adoptar precauciones basadas en software, como seleccionar software seguro o actualizarlo
- Formar al personal para identificar objetivos y puntos débiles
- Auditar y registrar las actividades para detectar las intrusiones o los intentos de intrusión

Los ataques más efectivos sobre sistemas informáticos aprovechan puntos débiles conocidos, como contraseñas fáciles de averiguar, malas configuraciones y versiones antiguas de software.

La adopción de unas cuantas medidas puede desanimar a los atacantes con poca experiencia y garantizar la disponibilidad de una copia de seguridad si ocurre lo peor.

Fracaso en la atracción de suficiente negocio

Aunque los ataques de los piratas informáticos son muy temidos, la razón de la mayor parte de los fracasos en el ámbito de los comercios electrónicos viene determinado por factores económicos tradicionales.

La creación y comercialización de un sitio de comercio electrónico de gran tamaño exige fuertes inversiones. Las compañías aceptan la pérdida de dinero a corto plazo si ello conlleva un refuerzo de la marca en el mercado y un incremento de clientes e ingresos futuros.

En el momento de escribir estas líneas, Amazon.com, que es con pocas dudas el establecimiento minorista más conocido de la Web, lleva perdiendo dinero durante cinco años consecutivos (las pérdidas en el primer trimestre de 2000 ascienden a 99 millones de dólares).

En la larga lista de fracasos sonados destaca la empresa europea `boo.com`, que agotó sus fondos y fue absorbida tras invertir más 120 millones de dólares en seis meses.

No es que Boo no vendiera, sino que gastaban mucho más de lo que obtenía.

Fallos de hardware

No hace falta decir que si su negocio depende de un sitio Web y falla una parte esencial de uno de los equipos, el resultado se verá afectado.

En los sitios Web con mucho tráfico o cuyo mantenimiento activo resulte esencial está justificada la existencia de sistemas redundantes para que el fallo de uno afecte al funcionamiento de todo el sistema.

Como con todas las amenazas, debe determinar si la posibilidad de desactivar su sitio Web por un día a la espera de que se realicen las reparaciones pertinentes justifica los gastos en equipo redundante.

Fallos de alimentación, comunicación, red y distribución

Internet se basa en una compleja red de proveedores de servicios. Si su conexión con el resto del mundo falla, no podrá hacer mucho más que esperar a que el proveedor vuelva a instalar el servicio. Otro tanto se puede decir con respecto a las cortes del fluido eléctrico, a las huelgas y a otro tipo de parones relacionados con las compañías de reparto.

Existe una solución que depende del presupuesto y consiste en contratar varios servicios a proveedores distintos.

El coste se incrementará pero si uno de los proveedores falla, podrá recurrir a otro. Los cortes breves de suministro eléctrico se pueden superar invirtiendo en generadores.

Competencia

Si va a abrir una tienda en una calle, no le costará mucho hacer un estudio exacto sobre la competencia con la que se va a enfrentar. Sus competidores serán principalmente negocios que vendan artículos similares y que estén situados en los alrededores. Además surgirán otros. En el ámbito del comercio electrónico, el terreno es más incierto.

Los competidores dependerán de los costes de envío y pueden encontrarse en cualquier parte del mundo.

Además, estarán sujetos a las fluctuaciones de las divisas y a los costes de la mano de obra.

En Internet la competencia es encarnizada y cambia a gran velocidad. Si trabaja en una categoría popular, pueden aparecer competidores todos los días.

No hay mucho que pueda hacer para eliminar el riesgo de la competencia, pero si se mantiene al tanto de los nuevos desarrollos, puede estar seguro de que su iniciativa resultará competitiva.

Errores de software

Los negocios basados en software resultan vulnerables a los errores de las aplicaciones informáticas.

Para reducir la probabilidad de errores vitales, seleccione software que resulte fiable, asigne tiempo suficiente para probarlo tras cambiar partes de su sistema, disponga de un proceso de prueba formal y no permita que se realicen cambios en el sistema activo sin probarlos antes en otra parte.

Para reducir la gravedad de los resultados, realice copias de seguridad actualizadas de todos los datos, mantenga configuraciones operativas de software al realizar un cambio y supervise el funcionamiento del sistema para detectar problemas rápidamente.

Cambios en las políticas e impuestos gubernamentales

La legislación sobre las empresas basadas en Internet varía según los países. Las posibilidades van desde que no exista legislación alguna a que se encuentre en fase de desarrollo o que esté suficientemente desarrollada. Esta situación es poco probable que continúe así. Como resultado, algunos modelos empresariales podrían verse amenazados, regulados o eliminados por la legislación futura. Además, surgirán impuestos.

No hay forma de evitar estas cuestiones. La única solución consiste en estar atento a la nueva legislación y mantener el sitio dentro de la ley. Considere la posibilidad de integrarse en un grupo de presión cuando vayan surgiendo estas cuestiones.

Límites de la capacidad del sistema

Una cuestión que debería tener presente al diseñar su sistema es el crecimiento. Lo normal es que su sistema reciba cada vez más tráfico por lo que debería diseñarse de forma se pueda escalar para hacer frente a la nueva demanda.

Si el crecimiento es limitado puede incrementar la capacidad del sistema adquiriendo hardware más rápido. Pero los equipos tienen un límite en cuanto a la velocidad. ¿Está escrito su software para que al alcanzar ese punto, se puedan separar sus partes para compartir la carga entre varios sistemas? ¿Puede su base de datos procesar varias peticiones simultáneas procedentes de diferentes equipos?

Hay muy pocos sistemas que puedan hacer frente a un crecimiento ingente si esfuerzos, pero si diseña su sistema teniendo en cuenta el factor de la escalabilidad, podrá identificar y eliminar los cuellos de botella cuando crezca su base de clientes.

Por qué estrategia optar

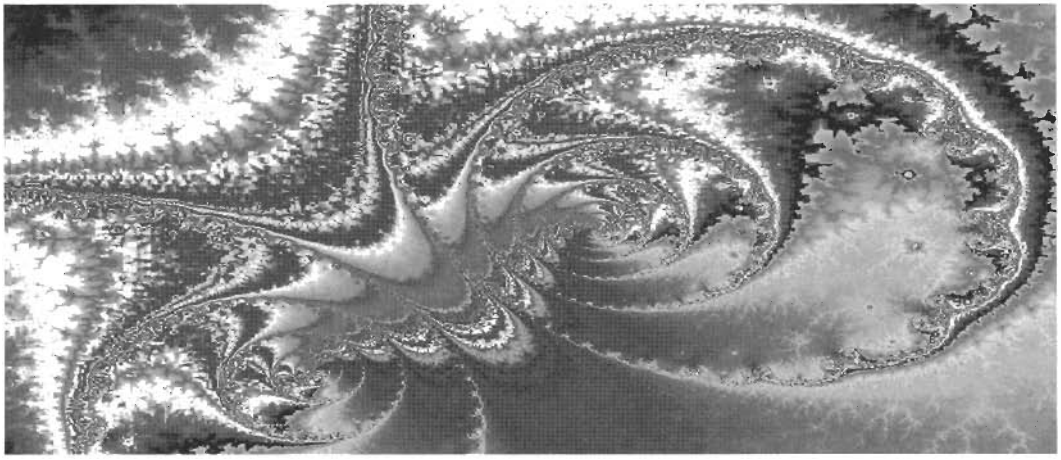
Algunos creen que Internet cambia demasiado rápido como para poder realizar planes que resulten eficaces. Pero esa mutabilidad convierte en fundamental la labor de planificación. Si no establece metas y selecciona una estrategia, se verá obligado a improvisar ante los cambios en lugar de anticiparse a ellos.

Tras examinar algunos de los objetivos típicos de un sitio Web comercial y algunas de las amenazas principales a las que hacer frente, es probable que ya tenga pensada alguna estrategia para el suyo.

Esta estrategia debe identificar un modelo de negocios. En general, el modelo de negocios suele ser una iniciativa cuyo funcionamiento ya se ha probado, pero en ocasiones puede tratarse de una idea en la que tenga fe. ¿Adaptará su modelo de negocios actual a la Web, imitará a un competidor o creará un servicio pionero?

A continuación

En el siguiente capítulo, abordaremos el tema de la seguridad en el comercio electrónico y suministraremos una descripción general de los términos, amenazas y técnicas de seguridad.



13
Aspectos de seguridad relacionados con el comercio electrónico

En este capítulo se analiza el papel de la seguridad en el comercio electrónico. Veremos quién puede estar interesado en nuestra información y cómo podrían obtenerla, los principios implicados en la creación de una política para evitar este tipo de problemas y algunas tecnologías disponibles para salvaguardar la seguridad de un sitio Web incluida la criptografía, la autenticación y el rastreo.

En este capítulo se examinarán los siguientes aspectos:

- Importancia de la información
- Amenazas contra la seguridad
- Creación de una política de seguridad
- Equilibrio entre usabilidad, rendimiento, coste y seguridad
- Principios de autenticación
- Uso de autenticación
- Fundamentos de la criptografía
- Criptografía de clave pública
- Firmas digitales
- Certificados digitales
- Servidores Web seguros
- Auditorías y registros

- Cortafuegos
- Copia de seguridad de datos
- Seguridad física

Importancia de la información

Al analizar el tema de la seguridad, lo primero que tenemos que valorar es la importancia de los datos que estamos protegiendo, tanto para nosotros como para los atacantes potenciales.

Puede resultar tentador creer que siempre debería aplicarse el nivel más alto seguridad a todos los sitios, pero la protección tiene un coste. Antes de decidir el esfuerzo o los gastos que se desean invertir en el sistema de seguridad, es necesario determinar el valor de la información.

El valor de la información almacenada en el equipo de un usuario que utilice su ordenador para divertirse, el de una empresa, el de un banco y el de una organización militar es diferente, al igual que varían los esfuerzos que invertirá un atacante para obtener acceso a dicha información. Tenemos que determinar el valor que tienen los contenidos almacenados en nuestros equipos para un visitante malintencionado.

Los usuarios que utilizan el ordenador como pasatiempo dispondrán de tiempo limitado para profundizar en el tema de la seguridad de sus sistemas o en aumentarla. Como la información que almacenarán en sus equipos es poco probable que interese a otros usuarios, los ataques serán poco frecuentes y el esfuerzo invertido en ellos será limitado. Sin embargo, todos los usuarios de equipos con conexión a la red deberían tomar una serie de precauciones ya que incluso los equipos con los datos menos interesantes pueden utilizarse como plataformas para lanzar ataques a otros sistemas.

Los equipos militares son un objetivo obvio tanto para individuos como para gobiernos extranjeros. Para atacar a los gobiernos es necesario disponer de muchos medios, por lo que resulta aconsejable invertir en personal y otro tipo de recursos para garantizar la aplicación de todas las precauciones posibles.

Si es responsable de un sitio de comercio electrónico, su atractivo para los piratas informáticos se situará entre estos dos extremos.

Amenazas contra la seguridad

¿Cuáles son los riesgos que se ciernen sobre su sitio? ¿Qué amenazas existen?

En un capítulo anterior se analizaron algunas de las amenazas a las que se expone un comercio electrónico. Muchas de ellas están relacionadas con la seguridad.

Las amenazas variarán en función del tipo de sitio Web pero se pueden citar las siguientes:

- Exposición de datos confidenciales
- Pérdida o destrucción de datos
- Modificación de datos
- Denegación de servicio
- Errores en el software
- Repudio

En las siguientes secciones se analiza cada una de ellas.

Exposición de datos confidenciales

La información almacenada en los equipos, o la transmitida de un equipo a otro, puede ser confidencial. Podría tratarse de información destinada únicamente a determinadas personas, como listas de precios de mayoristas, información suministrada por un cliente, como su contraseña, datos de contacto o el número de una tarjeta de crédito.

No debería guardar la información que no desee compartir en su servidor Web. Los servidores Web no son el lugar más aconsejable en el que guardar información secreta. Si almacena los datos sobre su nómina o su plan para la dominación del mundo en un ordenador, procure que no sea el mismo que el utilizado como servidor Web. Los servidores Web son equipos diseñados para su acceso público por lo que sólo deberían contener información que se ofrezca públicamente o que se acabe de recoger del público.

Para reducir el riesgo de la exposición de datos, debe limitar los métodos de acceso a la información y los usuarios que disponen de dicho acceso, lo cual implica desarrollar las labores de diseño teniendo presente la seguridad, configurar el servidor y el software de manera correcta, programar con atención, realizar pruebas exhaustivas, eliminar los servicios innecesarios del servidor Web y exigir autenticación.

Diseñe, configure, codifique y pruebe atentamente su sitio Web para reducir los riesgos de un ataque y, lo que es más importante, para reducir la probabilidad de que un error exponga accidentalmente la información.

Suprima los servicios innecesarios de su servidor Web para reducir el número de puntos débiles potenciales. Cada servicio que esté ejecutando puede resultar vulnerable. Es necesario mantener todos los servicios actualizados para eliminar los puntos vulnerables conocidos. Los servicios no utilizados pueden resultar peligrosos. Si no utiliza el comando `rcp`, ¿porqué tenerlo instalado? Si indica al instalador que su equipo es un host de red, las distribuciones más importantes de Linux y Windows NT instalarán una gran cantidad de servicios que no necesita y que debería eliminar.

La autenticación implica solicitar a los usuarios que demuestren su identidad. Cuando un sistema sabe quién realiza una petición, puede decidir si dicha persona dispone de acceso. Existen varios métodos de autenticación de los cuales sólo se utilizan dos de forma generalizada: las contraseñas y las firmas digitales. Los analizaremos más adelante.

CD Universe ofrece un buen ejemplo del coste, tanto en dinero como en reputación, que supone permitir la exposición de información confidencial. A finales de 1999, un pirata informático, que se hacía llamar Maxus, se puso en contacto con CD Universe para indicarles que disponía de 300.000 números de tarjetas de crédito que había sustraído su sitio Web. Por la destrucción de esta información pedió una recompensa que CD Universe se negó a pagar. Como resultado, CD Universe acabó en las primeras páginas de todos los periódicos de mayor tirada cuando Maxus comenzó a hacer públicos los números de las tarjetas de crédito de sus usuarios.

Los datos también corren el riesgo de exposición en sus trayectos por la red. Las redes TCP/IP incorporan una gran cantidad de funciones que han contribuido a convertirlas en el estándar de facto para la conexión de diversas redes como Internet. Sin embargo, la seguridad no se incluye entre ellas. Los protocolos TCP/IP dividen los datos en paquetes y los reenvían de equipo en equipo hasta que alcanzan su destino. Por lo tanto, los datos pasarán a través de una gran cantidad de ordenadores hasta llegar a su destinatario, como se ilustra en la figura 13.1, y cualquiera de ellas puede ver los datos en su tránsito.

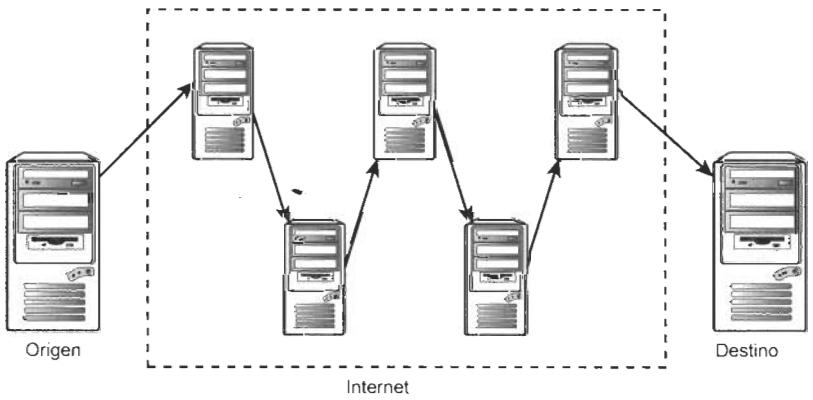


Figura 13.1. Los datos enviados a través de Internet recorren una gran cantidad de equipos potencialmente inseguros

Para ver la ruta que siguen los datos desde un equipo hasta el equipo de destino, puede utilizar el comando `traceroute` (en un equipo UNIX). Este comando devuelve las direcciones de los equipos a través de los cuales han pasado los datos para alcanzar el host de destino. Si los datos van dirigidos a un equipo de su propio país, podrían atravesar una decena de equipos diferentes. Si fueran destinados a un equipo situado fuera del país, el número de intermediarios podría superar la veintena. Si

su organización es de gran tamaño y consta de una red compleja, los datos podrían recorrer hasta cinco equipos antes de abandonar el edificio.

Para proteger la información confidencial, puede cifrar los datos antes de enviarlos a través de una red y descifrarlos en el otro extremo. Los servidores Web suelen utilizar el sistema SSL (Secure Socket Layer), desarrollado por Netscape para realizar esta tarea mientras los datos navegan entre los servidores Web y los navegadores. Se trata de una forma bastante barata en términos de costes y de esfuerzos de proteger las transmisiones, pero como el servidor necesita cifrar y descifrar los datos en lugar de enviarlos y recibirlos simplemente, el número de visitantes por segundo al que puede dar servicio el equipo se reduce drásticamente.

Pérdida o destrucción de datos

La pérdida de datos puede resultar más costosa que su apropiación indebida. Si lleva meses construyendo su sitio, recogiendo información sobre usuarios y pedidos, ¿sería capaz de determinar el coste, en tiempo, reputación y dinero, resultante de perder toda esa información? Si no ha realizado ninguna copia de seguridad de esos datos, se verá obligado a volver a escribir el sitio Web y a comenzar desde cero.

Un pirata informático podría colarse en su sistema y formatear su disco duro. También podría ocurrir que un programador o un administrador poco atento eliminen el disco sin querer o que un disco se estropee. Los discos duros giran miles de veces por minuto y en ocasiones fallan. Según la ley de Murphy, fallará el disco más importante y mucho después de realizar una copia de seguridad.

Puede adoptar varias medidas para reducir la probabilidad de perder datos. Proteja sus servidores contra ataques informáticos. Reduzca al mínimo los empleados con acceso a su equipo. Contrate únicamente personal competente y atento. Compre discos duros de calidad. Utilice la arquitectura RAID para permitir el uso de varios discos duros como un disco más rápido fiable.

Independientemente del causante de las pérdidas de datos sólo hay una forma de protección real: las copias de seguridad. La realización de copias de seguridad no es exactamente una tarea divertida y con suerte no necesitará recurrir a ellas, pero su función es vital. Asegúrese de volcar sus datos de manera periódica y de probar el procedimiento para asegurarse de que resultan recuperables. Asegúrese de almacenar las copias de seguridad lejos de los equipos habituales. Aunque es poco probable que los locales de su empresa se quemen o sufran alguna otra catástrofe, el almacenamiento de una copia de seguridad fuera de ellos es una medida de seguridad barata y eficaz.

Modificación de los datos

Aunque la pérdida de los datos puede causar muchos daños, su modificación puede resultar aún peor. ¿Qué ocurriría si alguien logra acceder a sus datos y los modifica? La eliminación completa de los datos no tardaría mucho tiempo en detectarse, pero pregúntese cuánto tiempo llevaría detectar su modificación.

Entre las formas de modificación se incluyen cambios en los archivos de datos o ejecutables. La intención de un pirata informático puede ser modificar el aspecto de la Web u obtener beneficios fraudulentos. Mediante la sustitución de archivos ejecutables con versiones saboteadas, un pirata informático que haya logrado entrar una vez en el sistema podría crear una puerta trasera secreta para futuras visitas.

Puede proteger los datos contra su modificación al viajar por la red mediante una firma. Esta técnica no impide la modificación de los datos pero, si el receptor comprueba que la firma coincide al recibir los archivos, sabrá que el archivo no se ha modificado. Si cifra los datos para protegerlos contra la visualización no autorizada, resultarán muy difíciles de modificar durante el trayecto sin su detección.

La protección de los archivos almacenados en un servidor frente a su modificación exige el uso de las funciones de permisos de acceso a archivos que incorpore su sistema operativo y protege el sistema de accesos no autorizados. La aplicación de permisos de archivos permite autorizar el uso del sistema, pero no modificar los archivos del sistema ni de otros usuarios. La falta de un sistema de permisos es una de las razones que convertían a los sistemas operativos Windows 95 y Windows 98 en poco aptos para su uso con servidores.

Detectar las modificaciones puede resultar difícil. Si descubrimos que alguien ha penetrado en nuestro sistema de seguridad ¿cómo podremos saber si se han modificado archivos importantes? Algunos archivos, como los archivos de datos que se almacenan en una base de datos, están ideados para cambiar con el paso del tiempo.

Sin embargo, la intención de otros es permanecer invariables desde su instalación a menos que se actualicen deliberadamente. La modificación de programas y datos puede resultar engañosa, y aunque los programas puedan reinstalarse si se sospecha que han sido modificados, puede que no sepa qué versión de los datos está "intacta".

Existen aplicaciones que permiten comprobar la integridad de los archivos, como Tripware. Esta aplicación registra información sobre archivos en un estado seguro, probablemente inmediatamente después de la instalación, y permite utilizarla en un momento posterior para verificar si los archivos han variado. Puede descargar la versión comercial o gratuita de

```
http://www.tripwire.com
```

Errores en el software

Es posible que el software que haya comprado, obtenido o escrito incluya errores graves. Dado los cortos plazos de desarrollo asignados a los proyectos Web, es muy probable que el software incluya errores.

Todas las iniciativas comerciales basadas en procesos informáticos resultan vulnerables a software con errores.

Los errores en el software pueden causar todo tipo de comportamientos impredecibles incluida la indisponibilidad del servicio, lagunas de seguridad, pérdidas financieras y servicios deficientes.

Entre las causas habituales de los errores se pueden citar las malas especificaciones técnicas, suposiciones erróneas realizadas por los desarrolladores y pruebas incompletas.

Malas especificaciones técnicas

Cuanto más escasa y ambigua resulte su documentación de diseño, más probable es que el producto final incluya errores. Aunque puede que resulte obvio especificar la anulación de un pedido si se rechaza la tarjeta de crédito de un cliente, puedo hablarles de un sitio con un alto presupuesto que incluía este error. Cuanta menos experiencia tengan los desarrolladores con el tipo de sistemas con el que están trabajando, más precisas deberían ser las especificaciones técnicas.

Suposiciones erróneas hechas por los desarrolladores

Los diseñadores y los programadores de un sistema necesitan realizar una gran cantidad de suposiciones. Es de esperar que las documenten y que resulten acertadas. Sin embargo, las suposiciones pueden resultar erróneas. Por ejemplo, un desarrollador podría asumir que los datos de entrada serán válidos, que no incluirán caracteres inusuales o que la cantidad de caracteres introducidos será inferior a un tamaño dado. También podría asumir que no tendrán lugar dos acciones simultáneas que entren en conflicto o que una tarea de procesamiento compleja lleve más tiempo que una tarea sencilla.

Suposiciones como ésta se pueden deslizar porque suelen ser ciertas. Un pirata informático puede aprovecharse de un desbordamiento de búfer generado por una suposición hecha por un programador con respecto a la longitud máxima de los datos de entrada o un usuario legítimo puede obtener mensajes de error confusos o abandonar la visita porque a los desarrolladores no se les ocurrió prever que el nombre de una persona pueda incluir un apóstrofe. Este tipo de errores se pueden detectar y solucionar mediante una combinación de pruebas y revisión detallada del código.

Históricamente, el sistema operativo o las debilidades del nivel de aplicación explotadas por los piratas informática se relacionan con los desbordamientos de búfer o con los errores de sincronización.

Pruebas incompletas

Resulta prácticamente imposible verificar todas las posibles entradas de los usuarios en todos dispositivos de hardware existentes, con todos los sistemas operativos posibles y utilizando todos los parámetros de usuario disponibles. Esto es especialmente cierto en el caso de los sistemas basados en la Web.

Es necesario implementar un plan de pruebas bien diseñado que verifique todas las funciones del software en un conjunto representativo de equipos. Un conjunto de pruebas bien planeado debería probar todas las líneas de código del proyecto al menos una vez.

De manera ideal, estas pruebas deberían automatizarse para poder ejecutarlas en los equipos seleccionados con muy poco esfuerzo.

El problema más importante de esta operación es que se trata de un trabajo repetitivo muy poco atractivo. Aunque hay gente a la que le gusta romper cosas, hay muy pocas a las que le guste romper la misma cosa una y otra vez. Es importante que en este proceso participen otras personas además de los desarrolladores originales. Uno de los objetivos principales de las pruebas es detectar las suposiciones erróneas realizadas por los desarrolladores. Es probable que una persona no implicada realice suposiciones diferentes. Además, los profesionales no suelen mostrarse muy inclinados a buscar errores dentro de su trabajo.

Repudio

El último riesgo que considerar es el repudio. El repudio tiene lugar cuando una parte implicada en una transacción niega haber tomado parte en ella. En el comercio electrónico puede tratarse de una persona que realice un pedido a través del sitio Web y que rechace la autorización para realizar el cargo sobre su tarjeta de crédito, o una persona que acepte algo procedente del correo electrónico y que después afirme que alguien falsificó dicho correo.

De manera ideal, las transacciones financieras deberían garantizar la seguridad de que ninguna de las partes las repudiará. Ninguna parte podría denegar su participación en una transacción o, para ser más exacto, ambas partes podrían demostrar de manera terminante las acciones realizadas por la otra parte ante un tercero, como por ejemplo un tribunal. En la práctica, esto no suele ocurrir.

La autenticación ofrece ciertas garantías sobre la parte con la que se está tratando. Los certificados digitales de autenticación emitidos por una organización de confianza brindan una gran fiabilidad.

También deberían poder certificarse los contenidos de lo mensajes enviados por cualquiera de las partes. No sirve de mucho poder demostrar que Corp Pty Ltd nos ha enviado un mensaje si no podemos probar que hemos recibido exactamente los que los han enviado. Como se indicó anteriormente, la firma o el cifrado de los mensajes dificulta su alteración de manera subrepticia.

Para las transacciones entre las partes con una relación ya establecida, los certificados digitales y el uso de comunicaciones cifradas o firmadas proporcionan una forma efectiva de limitar el repudio.

Si las transacciones son aisladas, como el contacto inicial entre un sitio de comercio electrónico y un usuario que utilice una tarjeta de crédito, esta opción no resulta tan práctica.

Las compañías de comercio electrónico deben estar dispuestas a probar su identidad y a contratar los servicios de una autoridad de certificación como VeriSign (`http://www.verisign.com/`) o Thawte (`http://www.thawte.com/`) para garantizar la autenticidad de la compañía. ¿Estará dispuesta esa misma compañía a rechazar a todos los clientes que no estén dispuestos a adoptar la misma medida para demostrar su identidad? En las pequeñas transacciones, los comerciantes están dispuestos a admitir un determinado nivel de fraude o riesgo de repudio a cambio de mantener el nivel de transacciones.

Desde 1997 existe una alianza entre VISA, un determinado número de organizaciones financieras y compañías de software para promover un estándar llamado SET (del inglés Secure Electronic Transaction, Transacción electrónica segura). Este sistema permite a los titulares de tarjetas de crédito obtener certificados digitales de los emisores de sus tarjetas. Si esta iniciativa sigue adelante, podría reducirse el riesgo de repudio así como otros tipos de fraude en las transacciones por Internet.

Por desgracia, aunque la especificación lleva en pie desde hace muchos años, parece que los bancos no se animan a emitir certificados compatibles con SET a los titulares de tarjetas. Los comerciantes no parecen dispuestos a rechazar a los clientes sin software SET y los usuarios muestran poco entusiasmo en adoptarlo. No hay muchas razones que obliguen a los clientes a hacer cola en sus bancos y a instalar el monedero digital en sus equipos si los comerciantes no van a rechazarlo si no lo utilizan.

Equilibrio entre usabilidad, rendimiento, coste y seguridad

Por su propia naturaleza, la Web es un lugar peligroso. Este medio está diseñado para permitir que numerosos usuarios soliciten servicios desde sus equipos. La mayor parte de estas peticiones realizan peticiones de páginas Web perfectamente legítimas, pero al conectar sus equipos a Internet también existe la posibilidad de que la gente realice otros tipos de conexiones.

Aunque resulta tentador asumir que es apropiado establecer el nivel de seguridad más alto, no suele ser así. Si desea estar completamente protegido, tendrá que colocar todos los equipos en una caja fuerte sin conexión a la red. Para que los equipos resulten accesibles y se puedan utilizar, es necesario rebajar el nivel de seguridad de alguna manera.

Debemos buscar un equilibrio entre la seguridad, la usabilidad, el coste y el rendimiento. Si aumentamos la seguridad de un servicio podemos reducir su capacidad de uso, por ejemplo, al limitar lo que puede hacer la gente o pedir que se identifiquen. El incremento de la seguridad también puede reducir el nivel de rendimiento de sus equipos. La ejecución de software para lograr que el sistema resulte más seguro (como sistemas de encriptación y detección de intrusos, escáneres de virus y operaciones de registro de usuarios) absorbe recursos. Se necesita mucho más potencial de procesamiento para suministrar una sesión cifrada, como una conexión SSL a un sitio Web, que una sesión normal. La pérdida de rendimiento se puede compensar con la adquisición de equipos o hardware más rápido que esté especialmente diseñado para operaciones de encriptación.

El rendimiento, la usabilidad, el coste y la seguridad se pueden considerar como objetivos contrapuestos. Deberá examinar los pros y los contras de cada opción y tomar una decisión que logre un compromiso en función del valor de la información, del presupuesto, de la cantidad de visitas esperadas y de los obstáculos que considere que aceptarán los usuarios legítimos.

Creación de una política de seguridad

Una política de seguridad es un documento que describe

- La filosofía general sobre seguridad en su organización
- Qué se debe proteger: software, hardware y datos
- Quién es responsable de la protección de estos elementos
- Estándares de seguridad e indicadores para medir el cumplimiento de dichos estándares

A la hora de desarrollar la política de seguridad se pueden aplicar las mismas reglas utilizadas para escribir un conjunto de requisitos de software. En esta política no debería hablarse de implementaciones o soluciones, sino de requisitos de seguridad y de objetivos dentro del entorno. Y no debería actualizarse a menudo.

Debería crear un documento en el que se establezcan las directrices utilizadas para medir el cumplimiento de la política de seguridad en un entorno dado. Puede establecer directrices diferentes para las distintas partes de la organización. Sería como un documento de diseño o un procedimiento manual en el que se recoja qué se está haciendo para garantizar el nivel de seguridad requerido.

Principios de autenticación

El objetivo de la autenticación es demostrar que alguien es quién dice ser. Existen muchas formas de suministrar autenticación pero como en el caso de muchas medidas de seguridad, cuanto más seguros son los métodos más problemáticos resultan de utilizar.

Las técnicas de autenticación incluyen el uso de contraseñas, firmas digitales, medidas biométricas mediante escáneres de huellas y medidas que implican el uso de hardware como tarjetas inteligentes. En la Web sólo se utilizan dos de manera extendida: las contraseñas y las firmas digitales.

Las medidas biométricas y la mayor parte de las soluciones de hardware implican el uso de dispositivos de entrada especiales, lo que limitaría el acceso a aquellos usuarios autorizados con ordenadores equipados. Esta opción podría resultar aceptable o incluso deseable para obtener acceso a los sistemas internos de una organización, pero reduce gran parte de las ventajas de permitir el acceso al sistema en la Web.

Las contraseñas resultan fáciles de implementar, sencillas de utilizar y no requieren el uso de dispositivos de entrada especiales. Suministran un determinado nivel de autenticación pero es posible que resulten insuficientes para sistemas de alta seguridad.

El concepto de una contraseña es sencillo. El usuario y el sistema conocen la contraseña. Si otro visitante afirma ser un usuario y conoce su contraseña, el sistema

lo aceptará. Este sistema resulta seguro siempre y cuando nadie más sepa o pueda adivinar la contraseña. Las contraseñas presentan en sí mismas varios puntos débiles y no suministran autenticación segura.

Muchas contraseñas resultan fáciles de averiguar. Si permite que los usuarios seleccionen sus propias contraseñas, un 50% escogerá contraseñas fáciles de descubrir, como vocablos de un diccionario o el nombre de usuario. Puede obligar a los usuarios a incluir números o signos de puntuación en sus contraseñas, aunque esta medida afectará a la usabilidad ya que los usuarios tendrán dificultades para recordarlas. Otra opción consiste en mostrar a los usuarios cómo seleccionar mejores contraseñas pero de todas maneras el 25% de ellos seguirá utilizando contraseñas sencillas de adivinar. Puede implementar políticas de contraseñas que impidan a los usuarios seleccionar combinaciones fáciles comparando las contraseñas introducidas con un diccionario o exigiendo el uso de números o signos de puntuación o una combinación de letras en mayúsculas y minúsculas. El peligro de una política de este tipo es que los usuarios legítimos no logren recordar las contraseñas escogidas.

Las contraseñas difíciles de recordar aumentan la probabilidad de que los usuarios realicen una acción poco segura como escribirlas en una nota y pegarla en sus monitores.

Es necesario educar a los usuarios para que no escriban sus contraseñas ni las comuniquen por teléfono a alguien que afirme estar trabajando en el sistema.

Las contraseñas sólo deberían capturarse electrónicamente. Meditante el uso de programas para capturar pulsaciones del teclado en una terminal o la información que circula por la red, los piratas informáticos pueden capturar pares de nombres de usuario y contraseñas. Puede limitar las oportunidades de capturar contraseñas cifrando el tráfico de red.

A pesar de sus defectos potenciales, las contraseñas son una forma sencilla y relativamente efectiva de autenticar usuarios. Puede que el nivel de anonimato no resulte adecuado para la seguridad nacional, pero son una solución perfecta para comprobar el estado del pedido de un cliente.

Uso de la autenticación

La mayor parte de los navegadores y servidores Web incorporan mecanismos de autenticación. Los servidores Web se pueden configurar para solicitar un nombre de usuario y una contraseña para permitir el acceso a los archivos de determinados directorios del servidor.

Cuando se necesite introducir un número de usuario y una contraseña, el navegador presentará un cuadro de diálogo parecido al de la figura 13.2.

Tanto Apache como el servidor IIS de Microsoft permiten proteger de manera sencilla una parte del sitio o el sitio en su totalidad de esta forma. PHP y MySQL incorporan muchas otras formas de lograr el mismo objetivo. MySQL resulta más rápido que la autenticación integrada. PHP suministra una autenticación más flexible o presenta la solicitud de forma más atractiva.

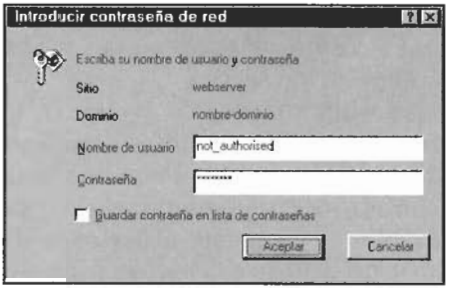


Figura 13.2. Los navegadores Web solicitan a los usuarios que se autentiquen al intentar visitar un directorio de acceso restringido en un servidor Web

En un capítulo posterior veremos algunos ejemplos de autenticación.

Fundamentos de la encriptación

Un algoritmo de encriptación es un proceso matemático que transforma información en una cadena aparentemente aleatoria de datos.

Los datos de los que se parte son texto sin procesar, aunque es importante para el proceso que represente la información (que se trate de texto o de otro tipo de datos). La información cifrada se conoce como texto encriptado, aunque se parezca poco a un texto en la mayor parte de los casos. La figura 13.3 muestra el proceso de encriptado en forma de diagrama de flujo. El texto que se desea cifrar se introduce en un motor de encriptación, que podría ser un dispositivo mecánico, como los equipos Enigma de la segunda guerra mundial, auque en la actualidad presenta prácticamente siempre el aspecto de un programa informático. El motor devuelve el texto encriptado.

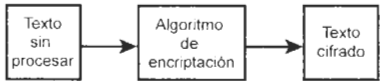


Figura 13.3. La encriptación toma texto sin procesar y lo transforma en texto encriptado

Para crear el directorio protegido cuya solicitud de autenticación se muestra en la figura 13.2, utilizamos el tipo de autenticación más básica de Apache. (En el siguiente capítulo le mostraremos cómo utilizarlo). Este método cifra las contraseñas antes de almacenarlas. Creamos un usuario con la contraseña `password`. Ésta se cifró y se almacenó como `aWDuA3X3H.mc2`. Como puede observar, el texto de partida y la cadena cifrada no se parecen en nada.

Este sistema de cifrado no es reversible. Muchas contraseñas se almacenan utilizando un algoritmo de encriptación de una sola dirección. Para poder determinar si una contraseña introducida resulta correcta, no es necesario descifrar la contraseña

almacenada. En su lugar, bastará con cifrar la contraseña introducida y compararla con la versión almacenada.

Muchos procesos de encriptación se pueden invertir, aunque no todos. El proceso inverso al encriptación se conoce como *decriptar*. La figura 13.4 muestra un proceso de encriptación de doble sentido.

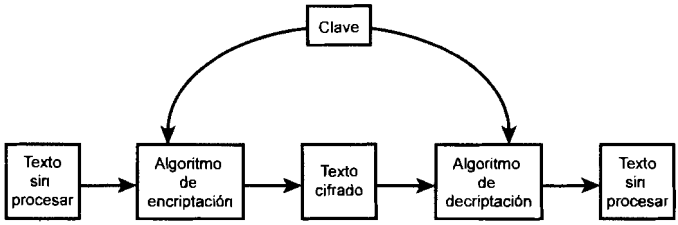


Figura 13.4. El proceso de encriptación toma texto sin procesar y lo transforma en texto cifrado con aspecto aleatorio. El proceso de decriptado toma el texto encriptado y lo transforma en el texto de partida

La criptografía tiene unos 4000 años de antigüedad, pero alcanzó la mayoría de edad en la segunda guerra mundial. Su crecimiento desde entonces ha seguido de cerca al desarrollo de las redes de ordenadores. En un principio sólo era utilizada por las instituciones militares y organizaciones financieras. En los años 70 se extendió su uso y en los años 90 se hizo omnipresente. En los últimos años, la encriptación ha pasado de ser un concepto que la gente sólo conocía por las películas de espías y de la segunda guerra mundial a convertirse en algo de lo que se habla en los periódicos y se utiliza cada vez que se compra algo a través de navegadores Web. Existen muchos algoritmos de encriptación disponibles. Algunos, como DES, utilizan una clave secreta o privada; otros, como RSA, utilizan una clave pública y una clave privada distinta.

Encriptación de clave privada

La encriptación de clave privada se basa en usuarios autorizados que conocen o disponen de acceso a una clave. Esta clave debe mantenerse en secreto ya que de lo contrario los mensajes cifrados podrían ser leídos por personas no autorizadas. Como se muestra en la figura 13.4, tanto el remitente (la persona que cifra el mensaje) como el receptor (el que descifra el mensaje) tienen la misma clave.

El algoritmo de clave secreta más utilizado es DES (del inglés Data Encryptation Standard, Estándar de encriptación de datos). Este método fue desarrollado por IBM en los años 70 y fue adoptado como el estándar americano para comunicaciones comerciales y gubernamentales no clasificadas. La velocidad de los ordenadores ha aumentado enormemente desde entonces y DES ha quedado obsoleto desde al menos 1998.

Otros sistemas de claves secretas conocidos son RC2, RC4, RC5, triple DES e IDEA. El sistema triple DES resulta bastante seguro. Utiliza el mismo algoritmo que DES, pero se aplica tres veces con hasta tres claves diferentes. El mensaje de texto se cifra con la primera clave, se descifra con la segunda y se vuelve a cifrar con la tercera.

Uno de los defectos de la encriptación con clave secreta es que para enviar un mensaje seguro a alguien, es necesario disponer de un forma segura de poderles enviar la clave secreta. Si dispone de una forma segura de enviar una clave, ¿por qué no utilizarla para enviar el mensaje?

Afortunadamente, en 1976 se produjo un avance significativo cuando Diffie y Hellman publicaron el primer método de clave pública.

Encriptación de clave pública

La encriptación de clave pública se basa en dos claves, una clave pública y una clave privada. Como se ilustra en la figura 13.5, la clave pública se utiliza para cifrar mensajes y la clave privada para descifrarlos.

La ventaja de este sistema es que la clave pública, como su nombre indica, se puede distribuir públicamente. Todo el mundo al que le entreguemos nuestra clave pública puede enviarnos un mensaje seguro. Mientras mantengamos en secreto nuestra clave privada, sólo nosotros podremos descifrar el mensaje.

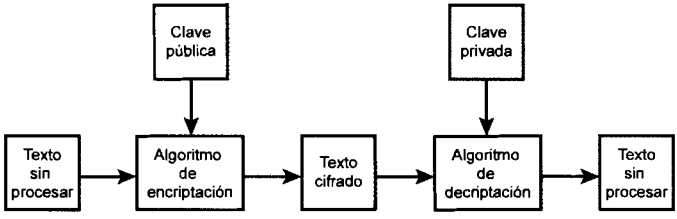


Figura 13.5. La encriptación de clave pública utiliza claves distintas para cifrar y descifrar mensajes

El algoritmo de clave pública más utilizado es RSA, desarrollado por Rivest, Shamir y Adelman en MIT y publicado en 1978. RSA era un sistema propietario pero la patente venció en septiembre de 2000.

La capacidad para transmitir una clave pública de forma abierta y sin necesidad de preocuparse por que resulte visible para una tercera parte supone una gran ventaja. Un sistema de clave pública se utiliza para transmitir la clave para un sistema de clave secreta que se utilizará durante el resto de una comunicación de sesión. Esta complejidad añadida resulta tolerable porque los sistemas de clave secreta son unas 1000 veces más rápidos que los sistemas de clave pública.

Firmas digitales

Las firmas digitales están relacionados con la criptografía de clave pública, pero invierten los papeles de las claves públicas y privadas. Un remitente puede cifrar y

firmar digitalmente un mensaje con su clave secreta. Cuando se recibe el mensaje, el receptor puede descifrarlo con la clave secreta del remitente. Como el remitente es la única persona con acceso a la clave secreta, el receptor puede estar bastante seguro de la procedencia del mensaje y de que no se ha alterado.

Las firmas digitales resultan muy útiles. Garantizan la procedencia de los mensajes y dificultan el rechazo del envío por parte del remitente.

De todos modos, es importante tener en cuenta que aunque el mensaje se ha cifrado, cualquier persona que tenga la clave pública puede leerlo. Si bien se utilizan las mismas técnicas y claves, la finalidad del uso de la encriptación en este caso es impedir la modificación de los mensajes y su repudio, no su lectura.

Como el cifrado de clave pública resulta bastante lento en el caso de mensajes de gran tamaño, se suele utilizar otro tipo de algoritmo, llamado función hash, para mejorar la eficacia.

La función hash calcula un mensaje implícito o valor hash para cualquier mensaje que se le indique. El valor generado por el algoritmo no es importante. Lo importante es que el resultado sea fijo, es decir, que sea el mismo cada vez que se utiliza una entrada dada, que sea pequeño y que el algoritmo sea rápido.

Las funciones hash más habituales son MD5 y SHA.

Una función hash genera un mensaje implícito que coincide con un mensaje dado. Si disponemos de ambos mensajes, podremos verificar si el texto se ha variado, siempre y cuando estemos seguros de que no se ha tocado el mensaje implícito.

La forma más habitual de crear una firma digital consiste en crear un mensaje implícito para todo el mensaje utilizando una función hash rápida y, a continuación, cifrar únicamente el mensaje implícito utilizando una algoritmo de clave pública. Seguidamente se puede enviar la firma con el mensaje a través de cualquier método normal no seguro.

Cuando se recibe un mensaje, se puede realizar su comprobación. La firma se descifrará utilizando la clave pública del remitente. Se genera un valor hash para el mensaje utilizando el mismo método utilizado por el remitente. Si el valor hash descifrado coincide con el valor hash generado, podemos estar seguros de que el mensaje procede del remitente y no se ha alterado.

Certificados digitales

La posibilidad de verificar que un mensaje no se ha alterado y que procede de un usuario o de un equipo dado es un gran avance. Para las interacciones comerciales, resultaría incluso mejor disponer de la posibilidad de vincular a dicho usuario o servidor a una entidad legal real como una persona o compañía.

Un certificado digital combina una clave pública con los detalles de una organización o individuo en un formato digital firmado. En un certificado, nosotros tenemos la clave pública de la otra parte, por si queremos enviar un mensaje cifrado y tendremos los detalles de la otra parte, que podemos estar seguros de que no se han alterado.

El problema en este caso es que la fiabilidad de la información depende de lo fiable que sea el remitente ya que todo el mundo puede generar y firmar un certificado utilizando otra identidad. En las transacciones comerciales, resulta útil utilizar una tercera parte de confianza que acredite la identidad de los participantes y los detalles de sus certificados.

Estas terceras partes se conocen como CA (del inglés Certifying Authorities, Autoridades de certificación). Las autoridades de certificación expiden certificados digitales a individuos y compañías sujetas a una serie de comprobaciones de identidad. Las dos CA más conocidas son VeriSign (`http://www.verisign.com/`) y Thawte (`http://www.thawte.com/`), pero existen otras. VeriSign y Thawte pertenecen a la misma compañía y hay muy poca diferencia entre ambas. Algunas menos conocidas, como Equifax Secure (`www.equifaxsecure.com`), resultan bastante más baratas. Las autoridades firman un certificado para acreditar que se ha comprobado la identidad de una persona o compañía. Tenga en cuenta que el certificado no es una referencia ni una declaración de solvencia. No garantiza que se está tratando con una entidad fiable. Lo único que garantiza es que en caso de que nos engañen, es probable que obtengamos una dirección física y una entidad o persona contra la que dirigir la demanda.

Los certificados suministran una estructura de confianza. Si confiamos en la CA, podemos confiar en aquellas partes en las que confía la CA y a su vez en aquellas partes en las que confía la parte certificada. En la figura 13.6 se ilustra la ruta de certificados que muestra Internet Explorer para un certificado concreto. En ella, puede observar que `www.equifaxsecure.com` tiene un certificado emitido por Equifax Secure E-Business Certifying Authority. Esta CA, a su vez, dispone de un certificado emitido por Thawte Server Certifying Authority.

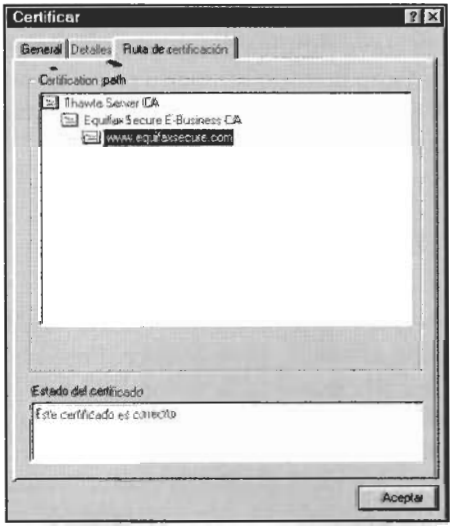


Figura 13.6. La ruta de certificados de www.equifaxsecure.com muestra la estructura de confianza que permite confiar en este sitio

Los certificados digitales se suelen utilizar para conferir un aire de respetabilidad al sitio Web de un comercio electrónico. Con un certificado emitido por una CA conocida, los navegadores Web pueden establecer conexiones SSL a nuestro sitio sin generar cuadros de diálogo de advertencia. Los servidores Web que permite conexiones SSL se suelen llamar servidores Web seguros.

Servidores Web seguros

Puede utilizar el servidor Web Apache, el servidor IIS de Microsoft o cualquier otro servidor Web comercial para establecer comunicaciones seguras con navegadores a través de SSL. Apache permite utilizar un sistema operativo de tipo UNIX, que suelen resultar más fiables pero más difíciles de configurar que IIS. El servidor Apache se puede utilizar en una plataforma Windows.

El uso de SSL sobre IIS implica sencillamente instalar IIS, generar un par de claves e instalar el certificado. El uso de SSL en Apache requiere la instalación de tres paquetes diferentes: Apache, Mod_SSL y OpenSSL.

También puede obtener lo mejor de los dos mundos comprando Stronghold. Stronghold es un producto comercial disponible desde `www.c2.net` por unos 1000 dólares. Está basado en Apache pero tiene forma de archivo binario autoinstalable y lleva preconfigurado el sistema SSL. De esta forma, se obtiene la fiabilidad de UNIX así como un producto de fácil instalación con asistencia técnica del proveedor.

Las instrucciones de instalación para los dos servidores Web más populares, Apache e IIS, se incluyen en el apéndice A. Puede comenzar a utilizar SSL de forma inmediata si genera sus propios certificados digitales pero los visitantes que acudan a su sitio recibirán una advertencia de sus navegadores indicándoles que hemos firmado nuestros propios certificados. Para poder utilizar SSL con eficacia, necesitará un certificado publicado por una autoridad de certificación.

El proceso exacto para lograrlo varía según la autoridad de certificación, pero por regla general deberá demostrar que su empresa está reconocida legalmente, que consta de una dirección física y que es el titular del nombre de dominio pertinente.

Necesitará generar una solicitud de firma de certificado. El proceso para lograrlo varía de un servidor a otro. Encontrará las instrucciones en los sitios Web de las autoridades de certificación. Stronghold e IIS incorporan un proceso basado en cuadros de diálogo mientras que Apache requiere la introducción de comandos. Sin embargo, el proceso es esencialmente el mismo para todos los servidores. El resultado final es una solicitud de firma de certificado (CSR) cifrada. Las CSR presentan un aspecto parecido al siguiente:

```
--BEGIN NEW CERTIFICATE REQUEST---
MIIBuwIBAAKBgQCLn1XX8faMHhtzStp9wY6BVTPuEU9bpMmhrb6vgaNZy4dTe6VS
84p7wGepq5CQjfOL4Hjda+g12xzto8uxBkCDO98Xg9q86CY45HZk+q6GyGOLZSOD
8cQHwh1oUP65s5Tz0180FBcpI3bHxfO6aYelWYziDiFKp1BrUdua+pK4SQIVAPLH
```

```
SV9FSz8Z7IHOg1Zr5H82oQOlAoGAWSPWyfVXPAF8h2GDb+cf97k44VkHZ+Rxpe8G
ghlfBn9L3ESWUZNOJMfDLlny7dStYU98VTVNekidYuaBsvyEkFrny7NCUmiuaSnX
4UjtFDkNhX9j5YbCRGLmsc865AT54KRu31O2/dKHLo6NgFPirijHy99HJ4LRY9Z9
HkXVzswCgYBwBFH2QfK88C6JKW3ah+6cHQ4Deoiltxi627WN5HcQLwkPGn+WtYSZ
jG5tw4tqqogmJ+IP2F/5G6FI2DQP7QDvKNeAU8jXcuijuWo27S2sbhQtXgZRTZvO
jGn89BC0mIHgHQMkI7vz35mx1Skk3VNq3ehwhGCvJlvoeiv2J8X2IQIVAOTRp7zp
En7QlXnXw1s7xXbbuKP0
--END NEW CERTIFICATE REQUEST---
```

Tras adquirir la CSR, pagar la cuota correspondiente, conseguir la documentación que prueba nuestra existencia y verificar que el nombre de dominio que estamos utilizando es el mismo que el incluido en la documentación corporativa, estará en disposición de firmar un certificado con una autoridad de certificación.

Cuando la autoridad de certificados emita el suyo, deberá almacenarlo en su sistema e indicarle a su servidor Web dónde encontrarlo. El certificado final es un archivo de texto parecido al CSR mostrado anteriormente.

Auditorías y registros

Su sistema operativo le permitirá registrar todo tipo de eventos. Los eventos en los que puede estar interesado desde el punto de vista de la seguridad incluyen errores de red, acceso a archivos de datos concretos como los de configuración o el registro de NT y llamadas a programas como `su` (que se utiliza para convertirse en otro usuario, por regla general el usuario raíz, en un sistema UNIX).

Los archivos de registro pueden servir de ayuda para detectar comportamientos erróneos o malintencionados cuando tienen lugar. También se puede utilizar para determinar cómo tuvo lugar un problema o una intrusión si se examina tras observar problemas. Los archivos de registro presentan dos problemas principales: el tamaño y la veracidad.

Si establece los criterios para detectar y registrar problemas en el nivel de mayor paranoia, terminará obteniendo registros gigantescos que resultan muy difíciles de examinar. Para trabajar con archivos de registro de gran tamaño, tendrá que recurrir a una herramienta existente o derivar algunas secuencias de comandos de la política de seguridad establecida para realizar búsquedas de eventos "interesantes" dentro de los registros. El proceso de auditar los registros puede realizarse en tiempo real o de manera periódica.

Los archivos de registros resultan vulnerables a ataques. Si un intruso dispone de acceso de usuario raíz o de administrador a nuestro sistema, podrá alterar los archivos de registros para borrar sus huellas. Unix permite registrar eventos en un equipo diferente, lo que implica que el intruso deberá acceder a dos equipos al menos para cubrir sus huellas. NT dispone de funcionalidad similar, pero no resulta tan sencilla de implementar.

El administrador de sistema puede realizar auditorías periódicas, pero además, podríamos contratar un servicio de auditorías externo para que controlara el comportamiento de los administradores.

Cortafuegos

El objetivo de los cortafuegos es alejar una red del mundo exterior. De la misma forma que los cortafuegos de un edificio o de un coche impiden que el fuego se extienda a otros compartimentos, los cortafuegos de red impiden que el caos se extienda por una red.

Los cortafuegos están diseñados para proteger los equipos de una red del mundo exterior. Filtran y rechazan el tráfico que no cumple las reglas establecidas en ellos. Restringe las actividades de la gente y de los equipo situados fuera del cortafuegos.

En ocasiones, los cortafuegos también se pueden utilizar para restringir las actividades de los equipos incluidos en su interior. Un cortafuegos puede limitar los protocolos de red que se pueden utilizar, los hosts a los que se pueden conectar o forzar el uso de un servidor proxy para reducir costes en términos de ancho de banda.

Un contrafuegos puede ser un dispositivo de hardware, como un enrutador con reglas de filtrado o un programa de software que se ejecute en un equipo. En cualquier caso, el cortafuegos necesita interfaces para las dos redes y un conjunto de reglas. Su misión consiste en supervisar todo el tráfico que intenta pasar de una red a otra. Si el tráfico cumple las reglas establecidas, será dirigido a otra red; de lo contrario, se detendrá o se rechazará.

Los paquetes se pueden filtrar por su tipo, dirección de origen, dirección de destino o puerto. En algunos casos, los paquetes pueden rechazarse simplemente y en otros pueden desencadenar entradas en el registro o activar alarmas.

Copia de seguridad de los datos

No se puede subestimar la importancia de las copias de seguridad en ningún plan de recuperación de desastres. El hardware y los inmuebles se puede asegurar y sustituir, o los sitios alojados en cualquier host, pero si su software Web personal desaparece, ninguna entidad aseguradora podrá sustituirlo.

Debe hacer una copia de seguridad de todos los componentes de su sitio Web (páginas estáticas, secuencias de comandos y bases de datos) periódicamente. La frecuencia depende del dinamismo del sitio. Si se trata de un sitio completamente estático, bastará con hacer una copia cuando se modifique. Sin embargo, el tipo de sitios que estamos tratando en este libro es probable que cambien con frecuencia, en especial si se reciben pedidos en línea.

La mayor parte de los sitios con un tamaño razonable deben alojarse en un servidor con sistema RAID (del inglés Redundant Array of Inexpensive Disks, Matriz redundante de discos independientes) que admiten funciones de réplica. De esta forma, queda cubierta la probabilidad de que ocurra un fallo en el disco duro. Considere, sin embargo, qué ocurriría en una situación que afecta a la matriz, al equipo o al edificio entero.

La frecuencia con la que realizar los volcados de seguridad debería corresponderse con el volumen de actualización. Estas copias de seguridad deberían almacenarse en un soporte diferente y a ser posible en un lugar distinto y seguro contra incendios, robos o desastres naturales.

Existen muchas aplicaciones para realizar funciones de copia de seguridad y recuperaciones. En nuestro caso, nos centraremos en cómo volcar un sitio construido con PHP y equipado de una base de datos MySQL.

Copia de seguridad de los archivos generales

Los volcados de archivos HTML, PHP, imágenes y otros archivos que no sean de base de datos se puede realizar con facilidad en la mayor parte de los sistemas utilizando software para la realización de copias de seguridad.

Entre las herramientas gratuitas más utilizadas se puede mencionar AMANDA desarrollada por la University of Maryland. Esta utilidad se incorpora en la mayor parte de las distribuciones de UNIX y también se puede utilizar para realizar copias de seguridad en equipos Windows a través de SAMBA. Si desea saber más sobre AMANDA, diríjase a

```
http://www.amanda.org/
```

Copia de seguridad y restauración de bases de datos MySQL

El proceso de volcado de seguridad de una base de datos en funcionamiento resulta más complicado ya que no queremos copiar ningún dato mientras la base de datos se está modificando.

En un capítulo anterior encontrará instrucciones sobre cómo volcar y restaurar una base de datos MySQL.

Seguridad física

Las amenazas de seguridad consideradas hasta el momento hacen referencia a elementos intangibles como el software, pero no debería descuidar la seguridad física de su sistema. Debe prever un sistema de aire acondicionado y sistemas de protección contra el fuego, personas (torpes o criminales), fallos en el suministro eléctrico y fallos en la red.

Su sistema debería estar protegido bajo llave en un lugar seguro. El lugar concreto dependerá de cada caso pero puede ser una habitación, una jaula o un armario. Sólo se debería conceder permiso de acceso a esta habitación al personal pertinente. El personal no autorizado podría, voluntaria o involuntariamente, desconectar los cables o intentar eludir los mecanismos de seguridad utilizando un disco de reinicio.

Los rociadores de agua pueden causar tanto daño a los componentes electrónicos como el fuego. En el pasado, se utilizaron sistemas de extinción de incendios alimentados por gas halón para evitar este problema pero en la actualidad la producción de este gas está prohibida por el Protocolo de Montreal sobre sustancias que reducen la capa de ozono, por lo que los sistemas para la extinción de incendios deben utilizar otras alternativas menos dañinas como el gas argón o el dióxido de carbono. Puede leer más al respecto en

```
http://epa.gov/ozone/title6/snap
```

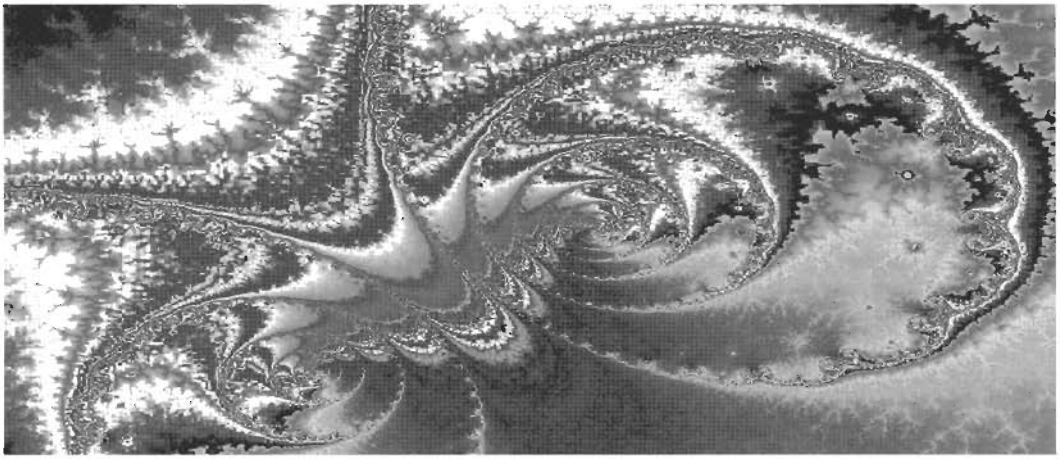
En determinados lugares, se producen breves cortes de fluido electrónico de manera habitual. En sitios con climatologías extremas o cables enterrados se suelen producir cortes de fluido de larga duración. Si el funcionamiento continuo del sistema es un factor importante debería invertir en un SAI (sistema de alimentación ininterrumpido). El coste de un SAI capaz de mantener el fluido eléctrico para un equipo durante 10 minutos asciende a nos 300 dólares. Si se necesita un sistema para cubrir cortes de suministro más largos o más equipos, el coste se elevará. Los cortes de suministro de larga duración requieren un generador que alimente el sistema de refrigeración y los equipos.

Al igual que los fallos de suministro eléctrico, los fallos de red, que pueden durar minutos o horas, quedan fuera de nuestro control y se producen irremediablemente de tiempo en tiempo. Si el mantenimiento de la red es vital, es aconsejable contratar los servicios de varios proveedores de servicios. El coste será superior, pero en caso de fallo, su sitio seguirá siendo visible, aunque se vea afectada su capacidad.

Este tipo de cuestiones podrían llevarle a considerar la opción de ubicar sus equipos dentro de una infraestructura dedicada. En un negocio de tamaño medio puede que no resulte justificable el uso de un SAI que mantenga el fluido eléctrico durante varios minutos, varias conexiones de red redundantes y sistemas antiincendios. Sin embargo, en una infraestructura de calidad en la que se alojen cientos de equipos similares sí resultaría justificable.

A continuación

En el siguiente capítulo, nos centraremos de manera específica en la autenticación, función que permite a los usuarios acreditar su identidad. Examinaremos varios métodos diferentes, incluido el uso de PHP y MySQL para autenticar a nuestros visitantes.



14
Implementación de la autenticación con PHP y MySQL

En este capítulo veremos cómo implementar varias técnicas de PHP y MySQL para autenticar a un usuario.

Entre los aspectos que analizaremos se incluyen los siguientes:

- Identificación de visitantes
- Implementación del control de acceso
- Autenticación básica
- Uso de la autenticación básica de PHP
- Uso de la autenticación básica .htaccess de Apache
- Uso de la autenticación básica con IIS
- Uso de la autenticación con `mod_auth_mysql`
- Creación de procesos de autenticación personalizados

Identificación de visitantes

La Web es un medio relativamente anónimo, pero a menudo resulta útil saber quién está visitando nuestro sitio. Afortunadamente para la privacidad de los visitantes, los datos que se pueden obtener sin su colaboración son bastante escasos.

Con un poco de trabajo, los servidores pueden averiguar bastantes datos sobre los equipos y las redes que se conectan a ellos. Los navegadores suelen identificarse, para lo cual indican al servidor qué navegador, versión y sistema operativo se está utilizando. Se puede determinar qué resolución y profundidad de color está utilizando el visitante en su pantalla así como el tamaño asignado a la ventana del navegador Web.

Cada equipo que se conecta a Internet tiene una dirección IP exclusiva. Esta dirección permite deducir algunos datos sobre el visitante. Se puede determinar a quién pertenece la IP y, aproximadamente, la ubicación geográfica del visitante. Algunas direcciones resultan más útiles que otras.

Por regla general, los usuarios con conexiones permanentes a Internet tendrán una dirección permanente. A los clientes que llamen a un proveedor de acceso a Internet se les suelen asignar direcciones para su uso temporal.

Si vuelve a ver dicha dirección es probable que esté siendo utilizada por un equipo distinto y si vuelve a encontrar al visitante es probable que esté utilizando una dirección IP diferente.

Por suerte para los usuarios Web, ninguno de los datos que ofrece el navegador sirve para identificarles. Si desea saber el nombre u otra información sobre un visitante, tendrá que preguntarles.

Muchos sitios Web utilizan razones convincentes para obtener datos de los usuarios. El New York Times (`http://www.nytimes.com`) ofrece su contenido gratis pero sólo a aquellas personas que estén dispuestas a suministrar datos como su nombre, sexo e ingresos.

El sitio Slashdot (`http://www.slashdot.org`) permite a los usuarios registrados participar en los debates con un apodo y personalizar la interfaz que ven.

La mayor parte de los sitios de comercio electrónico registran información sobre sus usuarios cuando realizan el primer pedido. De esta forma no tendrán que volver a hacerlo en los pedidos siguientes.

Tras solicitar y recabar información de los visitantes, necesitamos una forma de asociar dicha información con sus usuarios correspondientes cuando vuelvan a visitarnos. Si partimos del supuesto de que cada persona visitará su sitio desde una cuenta determinada y desde un equipo dado, y que cada visitante utiliza únicamente un equipo, podemos almacenar una cookie en el equipo del usuario para identificarle. Sin embargo, no ocurre siempre es así.

Con frecuencia, la gente comparte un equipo y muchos usuarios utilizan varios equipos para navegar. Por lo tanto, no quedará más remedio que preguntar al visitante quién es de nuevo. Además de solicitar que se identifique, tendrá que pedirle alguna prueba que confirme que es quién dice ser.

Como se indicó en un capítulo anterior el proceso de pedir a los usuarios que demuestren su identidad se conoce como autenticación. El método habitual de autenticación utilizado en los sitios Web consiste en pedir a los usuarios que introduzcan un nombre de usuario y una contraseña.

El proceso de autenticación se suele utilizar para conceder o rechazar el acceso a páginas o recursos concretos, pero puede resultar adecuado o útil para otros fines como la personalización.

Implementación del control de acceso

Un control de acceso sencillo no resulta difícil de implementar. El código incluido en el listado 14.1 devuelve tres posibles resultados. Si el archivo se carga sin parámetros, mostrará un formulario HMTL solicitando un nombre de usuario y una contraseña. En la figura 14.1 se muestra este tipo de formulario.



Figura 14.1. Nuestro formulario HTML pide a los visitantes que introduzcan un nombre de usuario y una contraseña para poder acceder

Si se han introducido los parámetros pero no son correctos, se mostrará un mensaje de error. En al figura 14.2 se muestra dicho mensaje.

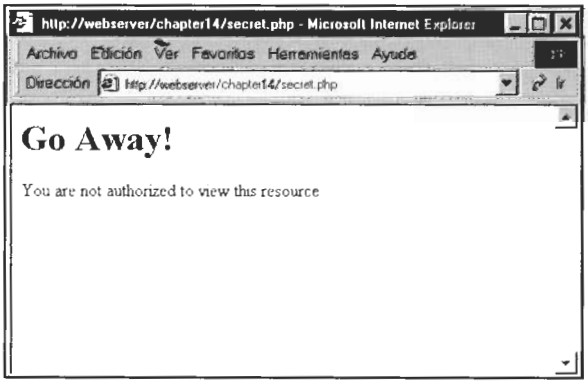


Figura 14.2. Cuando los usuarios introducen información incorrecta, tenemos que devolver un mensaje de error

Si los parámetros se han introducido y son correctos, se mostrará el contenido secreto, como se ilustra en la figura 14.3.

El código para crear la funcionalidad ilustrada en las figuras 14.1, 14.2 y 14.3 se muestra en el listado 14.1.



Figura 14.3. Cuando se introducen los detalles correctos, la secuencia de comandos muestra el contenido.

Listado 14.1. secret.php. Código PHP y HTML para proporcionar un sencillo mecanismo de autenticación

```
<?php
   //cree nombres cortos para las variables
@  $name = $HTTP_POST_VARS['name'];
@  $password = $HTTP_POST_VARS['password'];

   if(empty($name)||empty($password))
   {
      //El visitante tiene que introducir un nombre y una contraseña
?>
      <h1>Please Log In</h1>
      This page is secret.
      <form method="post" action="secret.php">
      <table border="1">
      <tr>
         <th> Username </th>
         <td> <input type="text" name="name"> </td>
      </tr>
      <tr>
         <th> Password </th>
         <td> <input type="password" name="password"> </td>
      </tr>
      <tr>
         <td colspan="2" align="center">
            <input type="submit" value="Log In">
         </td>
      </tr>
      </table>
      </form>
<?php
```

```
    }
    else if($name=='user'&&$password=='pass')
    {
       // el nombre y la contraseña del visitante son correctos
       echo '<h1>Here it is!</h1>';
       echo 'I bet you are glad you can see this secret page.';
    }
    else
    {
       // el nombre y la contraseña del visitante no son correctos
       echo '<h1>Go Away!</h1>';
       echo 'You are not authorized to view this resource.';
    }
?>
```

El código del listado 14.1 constituye un sencillo mecanismo de autenticación para conceder acceso a una página a los usuarios autorizados, pero presenta algunos problemas importantes.

Esta secuencia de comandos

- Incluye un nombre de usuario y una contraseña en su interior
- Almacena la contraseña en forma de texto sin procesar
- Sólo protege una página
- Transmite la contraseña en forma de texto sin procesar

Estas cuestiones se pueden resolver con diferentes grados de esfuerzo y éxito.

Almacenamiento de contraseñas

Existen muchos lugares en los que almacenar nombres de usuario y contraseñas mejores que dentro de una secuencia de comandos. Los datos introducidos dentro de una secuencia de comandos resultan difíciles de modificar. Se podría escribir una secuencia de comandos que se modificara a sí misma pero no es una buena idea ya que supondría tener una secuencia de comandos en el servidor, que se ejecutaría en él pero que otros podrían modificar o escribir sobre ella. Si almacenamos los datos en otro archivo dentro del servidor resultará más sencillo escribir un programa para agregar y eliminar usuarios y alterar contraseñas.

El uso de secuencias de comandos u otros archivos de datos con funciones de almacenamiento presenta un límite en cuanto al número de usuarios que se pueden incluir sin que se vea afectada la velocidad de la secuencia de comandos.

Si tiene previsto almacenar y realizar búsquedas en un archivo con una gran cantidad de elementos, debería considerar la opción de utilizar una base de datos, como se comentó anteriormente.

Como regla, si el número de elementos que necesita almacenar y buscar supera los 100, utilice una base de datos en lugar de un archivo.

El uso de una base de datos para almacenar nombres de usuario y contraseñas complicará la secuencia de comandos, pero a cambio permitirá autenticar a muchos

usuarios rápidamente. También permitirá escribir con facilidad una secuencia de comandos para agregar nuevos usuarios, para eliminar usuarios y para permitir que los usuarios cambien sus contraseñas.

En el listado 14.2 se incluye una secuencia de comandos para autenticar visitantes utilizando una base de datos.

Listado 14.2. secretdb.php. Hemos utilizado MySQL para mejorar nuestro sencillo mecanismo de autenticación

```
<?php
   if(!isset($HTTP_POST_VARS['name'])&&!isset($HTTP_POST_VARS['password']))
   {
      //El visitante tiene que introducir un nombre y una contraseña
?>
      <h1>Please  Log  In</h1>
      This page is secret.
      <form method="post" action="secretdb.php">
      <table border="1">
      <tr>
         <th> Username </th>
         <td> <input type="text" name="name"> </td>
      </tr>
      <tr>
         <th> Password </th>
         <td> <input type="password" name="password"> </td>
      </tr>
      <tr>
         <td colspan="2" align="center">
            <input type="submit" value="Log In">
         </td>
      </tr>
      </table>
      </form>
 <?php
   }
   else
   {
      // establezca conexión a mysql
      $mysql = mysql_connect( 'localhost', 'webauth', 'webauth' );
      if(!$mysql)
      {
         echo 'Cannot connect to database.';
         exit;
      }
      //  seleccione  la  base  de  datos  pertinente
      $mysql = mysql_select_db( 'auth' );
      if(!$mysql)
      {
         echo 'Cannot select database.';
         exit;
      }

      // consulte la base de datos para determinar si existe un registro que
      //coincida
```

```
        $query = "select count(*) from auth where
              name = '$name' and
              pass = '$password'";

        $result = mysql_query( $query );
        if(!$result)
        {
           echo 'Cannot run query.';
           exit;
        }

        $count = mysql_result( $result, 0, 0 );

        if ( $count > 0 )
        {
           // el nombre y la contraseña del visitante son correctos
           echo '<h1>Here it is!</h1>';
           echo 'I bet you are glad you can see this secret page.';
        }
        else
        {
           // el nombre y la contraseña del visitante no son correctos
           echo '<h1>Go Away!</h1>';
           echo 'You are not authorized to view this resource.';
        }
    }
?>
```

La base de datos que estamos utilizando se puede crear conectando a MySQL como usuario raíz de MySQL y ejecutando los contenidos del listado 14.3.

Listado 14.3. createauthdb.sql. Estas consultas MySQL crean la base de datos auth, la tabla auth y dos usuarios de ejemplo

```
create database auth;

use auth;

create table auth (
       name      varchar(10) not null,
       pass      varchar(30) not null,
       primary key (name)
);

insert into auth values
   ('user', 'pass');

insert into auth values
   ( 'testuser', password('test123') );

grant select, insert, update, delete
on auth.*
to webauth@localhost
identified by 'webauth';
```

Cifrado de contraseñas

Independientemente de dónde almacenemos las contraseñas (en una base de datos o en un archivo), no conviene hacerlo como texto sin procesar porque corremos un riesgo innecesario. Podemos utilizar un algoritmo de comprobación aleatoria (hash) de una dirección para obtener una mayor seguridad con muy poco esfuerzo.

La función de PHP `crypt()` proporciona una función de comprobación aleatoria de una dirección. Su sintaxis es la siguiente:

```
string crypt (string str [, string salt])
```

Dada la cadena `str`, la función devolverá una cadena pseudoaleatoria. Por ejemplo, dada la cadena "pass" y el parámetro salt "xx", la función `crypt()` devolverá "xxkT1mYjlikoII". Esta cadena no se puede descifrar y convertir en "pass" ni siquiera por su creador, lo que puede que no resulte de gran utilidad a primera vista. Lo que hace que la función `crypt()` resulte útil es que su resultado es siempre el mismo. Por lo tanto, si utilizamos la misma cadena y el mismo parámetro `salt`, la función `crypt()` devolverá el mismo resultado siempre que se ejecute.

En lugar de utilizar el siguiente código de PHP

```
if( $username == 'user' && $password == 'pass' )
{
   //Aceptado, las contraseñas coinciden

}
```

podemos utilizar código como éste

```
if( $username == 'user' && crypt($password,'xx') == 'xxkT1mYjlikoII'
)
{
   //Aceptado, las contraseñas coinciden
}
```

No necesitamos saber qué aspecto presenta "xxkT1mYjlikoII" antes de aplicarle la función `crypt()`. Tan sólo necesitamos saber si la contraseña escrita coincide como la cifrada inicialmente con la función `crypt()`.

Como se mencionó anteriormente, no conviene incluir los nombres de usuario y las contraseñas en una secuencia de comandos de manera expresa. Utilizaremos un archivo independiente o una base de datos para almacenar esta información.

Si estamos utilizando una base de datos MySQL para almacenar los datos de autenticación, podemos utilizar la función `crypt()` de PHP o la función `PASSWORD()` de MySQL. Estas funciones no generan el mismo resultado, pero están diseñadas con el mismo fin. Ambas toman una cadena y aplican un algoritmo de comprobación aleatoria no reversible.

Para utilizar `PASSWORD()`, podríamos escribir la consulta SQL del listado 14.2 como

```
select count(*) from auth where
```

```
name = '$name' and
pass = password('$password')
```

La consulta contará el número de filas de la tabla auth que tengan un nombre igual a los contenidos de `$name` y un valor `pass` igual al resultado producido por `PASSWORD()` aplicado a los contenidos de `$password`. Si los usuarios deben utilizar nombres de usuario únicos, el resultado de esta consulta será 0 o 1.

Si examina de nuevo el listado 14.3, observará que hemos creado un usuario ('user') con una contraseña sin cifrar y otro usuario con una contraseña cifrada ('testuser') para ilustrar estos dos enfoques.

Protección de páginas múltiples

La tarea de crear una secuencia de comandos como ésta para proteger más de una página resulta un poco más complicada. Como HTTP no tiene estado, no existe un enlace o asociación automática entre las solicitudes consecutivas de un mismo usuario. Este hecho dificulta la tarea de mantener datos de una página a otra, como la información de autenticación introducida por un usuario.

La forma más sencilla de proteger varias páginas consiste en utilizar los mecanismos de control de acceso incorporados en su servidor Web, como veremos en breve.

Si queremos crear esta funcionalidad, podemos incluir partes de la secuencia de comandos ilustrada en el Listado 14.1 dentro de las páginas que queremos proteger. Utilizando `auto_prepend_file` y `auto_append_file`, podemos adjuntar al principio o después el código requerido para cada archivo en directorios concretos. El uso de estos directorios se analizó en un capítulo anterior.

Si utilizamos este enfoque, ¿qué ocurrirá cuando nuestros visitantes recorran varias páginas de nuestro sitio? No podemos obligarles a volver a introducir sus nombres y contraseñas para cada página que deseen ver.

Podemos adjuntar los detalles introducidos a cada hipervínculo de la página. Como los nombres de usuario pueden incluir espacios u otros caracteres no permitidos en los URL, deberíamos utilizar la función `urlencode()` para codificar de forma segura estos caracteres.

Sin embargo, este enfoque presenta todavía algunos problemas. Como los datos se incluirán en páginas Web enviadas al usuario así como los URL que visiten, las páginas protegidas que visiten resultarán visibles para todos los usuarios que utilicen el mismo equipo y retrocedan por las páginas almacenadas en caché o examinen el historial del navegador. La contraseña se envía entre el navegador y el servidor cada vez que se solicita una página, lo cual no resulta muy recomendable.

Existen dos buenas formas de resolver estos problemas: la autenticación básica de HTTP y las sesiones. La autenticación básica resuelve el problema de las páginas almacenadas en caché, pero el navegador seguirá enviando la contraseña al servidor con cada petición. El control de sesiones resuelve ambos problemas. En primer lugar examinaremos el proceso de autenticación básica de HTTP y dejaremos para capítulos posteriores el análisis del control de sesión.

Autenticación básica

La autenticación de usuarios es una tarea habitual, de ahí que HTTP incorpore funciones que se ocupan de ello. Las secuencias de comandos o los servidores Web pueden solicitar autenticación desde un navegador Web. El navegador Web se encargará de mostrar un cuadro de diálogo o un elemento similar para obtener la información necesaria del usuario.

Aunque el servidor Web solicite nuevos detalles de autenticación para cada petición del usuario, el navegador Web no necesitará pedir los detalles del usuario para cada página. Por regla general el navegador almacena esta información mientras el usuario tenga una ventana del navegador abierta y la reenvía automáticamente al servidor Web cuando éste la solicite sin interacción con el usuario.

Esta función de HTTP se denomina autenticación básica. Puede desencadenarla utilizando PHP o mecanismos incorporados al servidor Web. Examinaremos el método utilizado por PHP, por Apache y por IIS.

El proceso de autenticación básica transmite el nombre de usuario y la contraseña de un usuario en forma de texto sin procesar, por lo que no resulta muy seguro. HTTP 1.1 contiene un método algo más seguro conocido como autenticación de texto implícita, que utiliza un algoritmo de comprobación aleatoria (habitualmente MD5) para disfrazar los detalles de la transacción. Muchos servidores Web admiten la autenticación de texto implícita, pero no ocurre así con un número significativo de navegadores Web. Microsoft Internet Explorer admite la autenticación de texto implícita desde su versión 5.0. En el momento de escribir estas líneas, ni Netscape ni Mozilla la admiten.

Además del problema de compatibilidad con los navegadores Web, la autenticación de texto implícita resulta todavía poco segura. Tanto la autenticación básica como su variante por medio de resúmenes ofrecen un bajo nivel de seguridad. Ninguna garantiza a los usuarios que se está interactuando con el equipo deseado. Ambas permiten que un pirata informático pueda reproducir la misma petición en el servidor. Como la autenticación básica transmite la contraseña del usuario en forma de texto sin procesar, cualquier pirata informático podría capturar paquetes y suplantar al usuario para realizar una petición.

La autenticación básica ofrece un nivel bajo de seguridad similar al de las conexiones a través de telnet o ftp, ya que las contraseñas se transmiten en forma de texto sin procesar. La autenticación de texto implícita resulta un poco más segura, ya que cifra las contraseñas antes de transmitirlas. Para proteger todas las partes de una transacción Web de forma segura se puede utilizar SSL y los certificados digitales.

Consulte el siguiente capítulo, si desea implementar un sistema de protección seguro. Sin embargo, en muchas ocasiones, resulta apropiado utilizar un sistema rápido aunque relativamente inseguro, como la autenticación básica.

La autenticación básica protege un dominio con nombre y exige el uso de un nombre de usuario y una contraseña válidas. Los dominios reciben nombres para poder alojar más de uno en el mismo servidor. Se puede dar el caso de que distintos

archivos o directorios del mismo directorio formen parte de distintos dominios y que cada uno de ellos esté protegido por un conjunto diferente de nombres y contraseñas. Los dominios con nombre también permiten agrupar varios directorios en un host físico o virtual como dominio y utilizar una contraseña para protegerlos todos.

Uso de la autenticación básica en PHP

Por regla general, las secuencias de comandos son multiplataforma pero el uso de la autenticación básica se basa en variables de entorno establecidas por el servidor. Para que una secuencia de comandos HTTP se pueda ejecutar en Apache utilizando PHP, en forma de módulo Apache o en IIS utilizando PHP como módulo ISAPI, es necesario detectar el tipo de servidor y comportarse de forma ligeramente diferente. La secuencia de comandos del Listado 14.4 se ejecutará en ambos servidores, pero a diferencia de la mayor parte de las secuencias de comandos utilizadas en este libro, es necesario que el parámetro `register_globlas` del archivo php.ini esté activado.

Listado 14.4. PHP puede desencadenar autenticación HTTP básica

```
<?php

// Si el usuario está utilizando IIS, necesitamos establecer $PHP_AUTH_USER
// y $PHP_AUTH_PW
if (substr($SERVER_SOFTWARE, 0, 9) == 'Microsoft' &&
    !isset($PHP_AUTH_USER) &&
    !isset($PHP_AUTH_PW) &&
    substr($HTTP_AUTHORIZATION, 0, 6) == 'Basic '
    )
{
   list($PHP_AUTH_USER, $PHP_AUTH_PW) =
      explode(':', base64_decode(substr($HTTP_AUTHORIZATION, 6)));
}

// Sustituya esto si se trata de una instrucción con una base de
//datos o similar
if ($PHP_AUTH_USER != 'user' || $PHP_AUTH_PW != 'pass')
{
   // el visitante no ha introducido sus datos o su
   // nombre y contraseña no son correctos

   header('WWW-Authenticate: Basic realm="Realm-Name"');
   if (substr($SERVER_SOFTWARE, 0, 9) == 'Microsoft')
      header('Status: 401 Unauthorized');
   else
      header('HTTP/1.0 401 Unauthorized');

   echo  '<h1>Go  Away!</h1>';
   echo 'You are not authorized to view this resource.';
}
```

```
else
{
   // el visitante ha introducido los datos correctos
   echo '<h1>Here it is!</h1>';
   echo '<p>I bet you are glad you can see this secret page.</p>';
}
?>
```

El código del listado 14.4 actúa de forma muy similar a la de los listados anteriores del capítulo. Si un usuario no ha introducido la información de actualización, se le pedirá. Si los datos introducidos son incorrectos, se devolverá un mensaje rechazándolos. Si incluye un nombre de usuario y una contraseña correctos, se devolverán los contenidos de la página.

El usuario verá una interfaz algo diferente a la devuelta por los listados anteriores. No suministramos un formulario HTML para la información de inicio de sesión. El navegador del usuario mostrará un cuadro de diálogo. Para algunas personas, será una mejora mientras que otros preferirán disponer de control completo sobre los aspectos visuales de la interfaz. En la figura 14.4 se presenta el cuadro de diálogo de inicio de sesión que devuelve Internet Explorer.

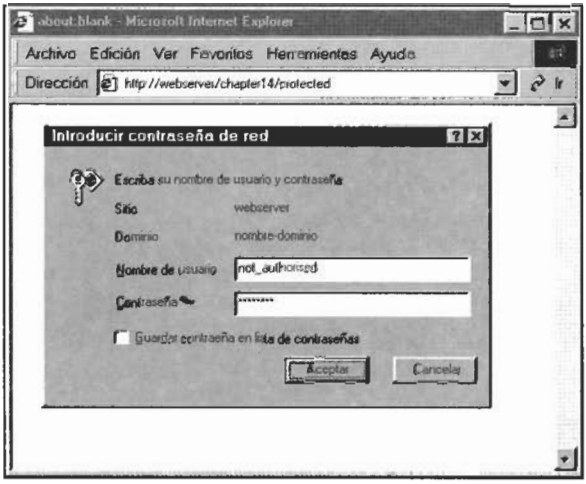


Figura 14.4. El navegador del usuario se encarga de la apariencia del cuadro de diálogo al utilizar la autenticación HTTP

Como el proceso de autenticación se delega en las funciones integradas en el navegador, éste opta por ejercer cierta discreción al procesar los intentos de autorización fallidos. Internet Explorer permite al usuario realizar la autenticación tres veces antes de mostrar la página de rechazo. Netscape permitirá que el usuario intente autenticarse las veces que lo desee (entre los distintos intentos, mostrará una cuadro de diálogo indicando que la autorización ha fallado y preguntará al usuario si desea intentarlo de nuevo). Netscape sólo muestra la página de rechazo si el usuario hace clic en **Cancelar**.

Como en el caso del código de los listados 14.1 y 14.2, podemos incluir el código del listado 14.4 en las páginas que deseemos proteger o adjuntarlos al principio de todos los archivos de un directorio.

Uso de la autenticación básica con los archivos .htaccess de Apache

Podemos obtener resultados muy similares a los de la secuencia de comandos anterior sin escribir una secuencia de comandos de PHP.

El servidor Web Apache incorpora una gran cantidad de módulos de autenticación diferentes que se pueden utilizar para determinar la validez de los datos introducidos por un servidor. El más sencillo de utilizar es `mod_auth`, que compara pares de nombre y contraseña con las líneas de un archivo de texto incluido en el servidor.

Para lograr el mismo resultado de la secuencia de comandos anterior, necesitamos crear dos archivos HMLT distintos, uno para el contenido y otro para la página de rechazo. En los ejemplos anteriores nos saltamos algunos elementos de HTML, entre ellos las etiquetas `<html>` y `<body>`, que deberíamos incluir al generar HTML.

El listado 14.5 recoge el contenido que ven los usuarios autorizados. Hemos llamado a este archivo `content.html`. El listado 14.6 recoge la página de error, que hemos llamado `rejection.html`. La visualización de esta página es opcional, pero incorpora un toque profesional si se incluye información útil en ella. Como la página se mostrará cuando un usuario intente entrar en un área protegida, el contenido útil podría incluir instrucciones sobre cómo registrarse para obtener una contraseña o cómo recuperarla y que nos la envíen por correo electrónico si la hemos olvidado.

Listado 14.5. content.html. Contenido de ejemplo

```
<html><body>
<h1>Here it is!</h1>
<p>I bet you are glad you can see this secret page.</p>
</body></html>
```

Listado 14.6. rejection.html. Página de error de ejemplo

```
<html><body>
<h1>Go Away!</h1>
<p>You are not authorized to view this resource.</p>
</body></html>
```

Estos archivos no contienen nada nuevo. El código verdaderamente interesante para este ejemplo se recoge en el listado 14.7. Este archivo se puede denominar .htaccess y controlará el acceso a los archivos y a todos los subdirectorios incluidos dentro de su directorio.

Listado 14.7. .htaccess. Un archivo .htaccess puede establecer muchos parámetros de configuración de Apache, incluyendo la activación del proceso de autenticación

```
ErrorDocument  401  /chapter14/rejection.html
AuthUserFile /home/book/.htpass
AuthGroupFile /dev/null
AuthName "Realm-Name"
AuthType Basic
require valid-user
```

El listado 14.7 es un archivo .htaccess para activar un proceso de autenticación básico en un directorio. En un archivo .htaccess se pueden configurar muchos parámetros, pero las seis líneas del utilizado en este ejemplo se relacionan con el proceso de autenticación.

La primera línea

```
ErrorDocument  401  /chapter14/rejection.html
```

indica a Apache qué documento mostrar para aquellos visitantes que no logren autenticarse.

Puede utilizar otras directivas `ErrorDocument` para suministrar páginas personalizadas para otros errores HTTP como el 404.

Su sintaxis es la siguiente:

```
ErrorDocument   número_de_error  URL
```

Para que una página procese el error 401, es importante que el URL dado esté disponible públicamente. No resulta muy útil suministrar una página de error personalizada para indicar a los usuarios que ha fallado su autorización si la página está bloqueada en un directorio en el que necesitan autenticarse para poder verlo.

La línea

```
AuthUserFile /home/book/.htpass
```

indica a Apache dónde encontrar el archivo que contiene las contraseñas autorizadas de los usuarios. Este archivo se suele denominar `.htpass` pero puede asignarle el nombre que desee. Lo importante no es el nombre sino el lugar utilizado para almacenarlo.

No debería almacenarse dentro del árbol Web ya que se podría descargar a través del servidor Web.

En el listado 14.8 se recoge el archivo .htpass de ejemplo.

Además de especificar los usuarios individuales que están autorizados, es posible especificar que sólo aquellos autorizados que se incluyan en determinados grupos dispongan de acceso a los recursos. En nuestro caso no lo hemos hecho, por lo que la línea

```
AuthGroupFile  /dev/null
```

establece que el archivo `AuthGroupFile` apunte a `/dev/null`, un archivo especial de los sistemas Unix cuyo valor queda garantizado como nulo.

Como el ejemplo PHP, para utilizar autenticación HTTP, necesitamos aplicar un nombre a nuestro dominio de la siguiente forma:

```
AuthName  "Nombre-Dominio"
```

Puede seleccionar el nombre de dominio que desee, pero recuerde que este nombre se mostrará a los visitantes.

Como se admiten varios métodos de autenticación, necesitamos especificar el que estamos utilizando.

Vamos a utilizar la autenticación de tipo `Basic` como se especifica en la siguiente directiva:

```
AuthType Basic
```

Necesitamos especificar quién dispone de acceso. Podemos especificar usuarios o grupos concretos, o como hemos hecho, todos los accesos de usuarios autenticados.

La línea

```
require valid-user
```

especifica que se conceda acceso a cualquier usuario válido.

Listado 14.8. htpass. El archivo de contraseñas almacena nombres de usuarios y la contraseña cifrada de cada usuario

```
user1:0nRp9M80GS7zM
user2:nC13sOTOhp.ow
user3:yjQMCPWjXFTzU
user4:LOmlMEi/hAme2
```

Cada línea del archivo .htpass contiene un nombre de usuario, dos puntos y la contraseña cifrada del usuario.

Los contenidos exactos del archivo `.htpass` variarán. Para crearlo, se utiliza un pequeño programa llamado `htpasswd` que se incluye en la distribución de Apache.

El programa `htpasswd` se utiliza de una de las siguientes formas:

```
htpasswd [-cmdps] archivo_de_cotraseñas nombre_de_usuario
```

o

```
htpasswd -b[cmdps] archivo_de_cotraseñas nombre_de_usuario constreñía
```

El único modificador que necesita utilizar es `-c`. Este modificador indica a htpasswd que cree el archivo. Debe utilizarlo para el primer usuario que agregue. Debe tener cuidado de no utilizarlo para el resto de los usuarios porque si el archivo existiese, htpasswd lo eliminaría y crearía uno nuevo. Los modificadores opcionales `m`, `d`, `p` o `s` se pueden utilizar si desea especificar el algoritmo de cifrado que desea utilizar (incluida la opción de no utilizar cifrado alguno).

El modificador `b` indica al programa que espere la contraseña como parámetro, en lugar de solicitarla, lo cual resulta útil si desea llamar a `htpasswd` de forma no

interactiva como parte de un proceso por lotes, pero no debería utilizarse si está llamando a `htpasswd` desde la línea de comandos.

Los siguientes comandos crean el archivo que se muestra en el listado 14.8:

```
htpasswd -bc /home/book/.htpass user1 pass1
htpasswd -b /home/book/.htpass user2 pass2
htpasswd -b /home/book/.htpass user4 pass3
htpasswd -b /home/book/.htpass user4 pass4
```

Este tipo de autenticación resulta sencillo de configurar, pero el uso de un archivo `.htaccess` de esta forma plantea una serie de problemas.

Los usuarios y las contraseñas se almacenan en un archivo de texto. Cada vez que un navegador solicita un archivo protegido por el archivo `.htaccess` el servidor debe analizar dicho archivo y, a continuación, el archivo de contraseñas, para determinar la coincidencia del nombre de usuario y la contraseña. En lugar de utilizar un archivo `.htaccess`, podríamos especificar las mismas acciones en el archivo `httpd.conf`, el archivo de configuración principal para el servidor Web. Cada vez que se solicita un archivo se analiza un archivo `.htaccess`. El archivo `httpd.conf` sólo se analiza cuando el servidor se abre inicialmente. Este método resulta más rápido pero obliga a detener y reiniciar el servidor si necesita realizar cambios.

Independientemente de dónde almacene las directivas de servidor, será necesario comprobar la contraseña en el archivo de contraseñas de cada petición. Esta técnica, como en el caso de las vistas anteriormente en las que se utiliza un archivo sin procesar, no resulta adecuada cuando se manejan cientos o miles de usuarios.

Autenticación básica con IIS

Como Apache, IIS admite autenticación HTTP. Apache utiliza el enfoque de UNIX y se controla mediante la edición de archivos de texto. En el caso de IIS, la configuración se realiza a través de cuadros de diálogo.

Windows 2000 permite cambiar la configuración de Internet Information Server 5 (IIS5) utilizando el Administrador de servicios de Internet. Esta utilidad se incluye dentro de la sección **Herramientas administrativas** del **Panel de control**.

El Administrador de servicios de Internet presenta un aspecto parecido al que se recoge en la figura 14.5. El control de árbol situado en la parte izquierda muestra que el servidor `windows-server` incluye varios servicios. En el que estamos interesados es el sitio Web predeterminado. Dentro de estos sitio Web, tenemos un directorio llamado `protected`. Dentro de este directorio hay un archivo llamado `content.html`.

Para agregar autenticación básica al directorio `protected`, haga clic con el botón derecho del ratón sobre él y seleccione **Propiedades** en el menú contextual.

El cuadro de diálogo **Propiedades** permite cambiar una gran cantidad de parámetros del directorio. Las dos fichas en las que estamos interesados son **Seguridad de directorios** y **Errores personalizados**. Una de las opciones de la ficha de **Seguridad de directorios** es **Control de autenticación y acceso anónimo**. Si hace clic sobre el botón **Modificar** se abrirá el cuadro de diálogo que se muestra en la figura 14.6.

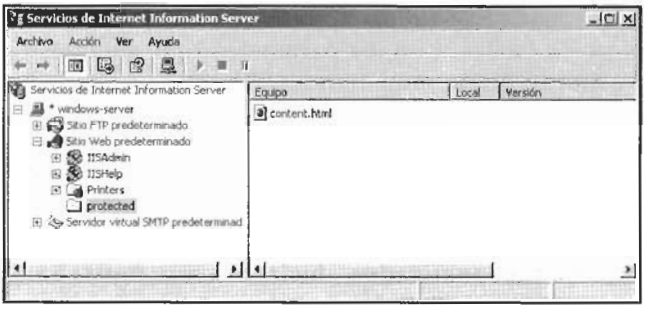


Figura 14.5. La consola de administración de Microsoft permite configurar Internet Information Server 5

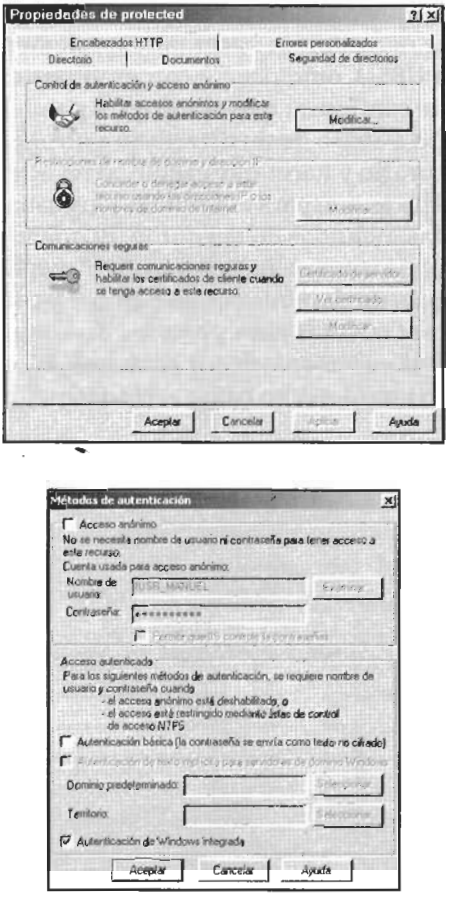


Figura 14.6. IIS5 permite el acceso anónimo de manera predeterminada, pero nos permite activar la autenticación

Dentro de este cuadro de diálogo, podemos deshabilitar el acceso anónimo y activar la autenticación básica. Con los parámetros que se muestran en la figura 14.6, sólo aquellos usuarios que suministren un nombre y una contraseña adecuados pueden ver los archivos de este directorio.

Para poder duplicar el comportamiento de los ejemplos anteriores, incluiremos una página para indicar a los usuarios que sus detalles de autenticación no son correctos. Al cerrar el cuadro de diálogo de los métodos de autenticación podremos seleccionar la ficha **Errores personalizados**.

La ficha **Errores personalizados**, ilustrada en la figura 14.7, asocia errores con mensajes de error. En este caso, hemos almacenado el mismo archivo de rechazo utilizado anteriormente, `rejection.html`, ilustrado el listado 14.6. IIS permite incluir mensajes de error más específicos que los que permite Apache, con el código de error HTTP que tuvo lugar y una explicación. Podemos incluir diferentes mensajes para cada caso, pero en nuestro ejemplo hemos optado por sustituir los dos que van a tener lugar en este ejemplo con nuestra página de error.

Esto es todo lo que necesitamos hacer para implementar la autenticación de este directorio utilizando IIS5. Como ocurre con gran parte del software de Windows, resulta más sencillo de configurar que el software paralelo de UNIX, pero más difícil de copiar de un equipo a otro o de un directorio a otro. También resulta más sencillo equivocarse al configurarlo y hacer que el equipo resulte inseguro.

El peor defecto de IIS es que para autenticar a los usuarios Web compara los datos de registro con cuentas incluidas en el equipo. Si queremos permitir que el usuario `"john"` se registre con la contraseña `"password"`, necesitamos crear una cuenta de usuario en el equipo o en un dominio con su nombre y contraseña. Es necesario poner mucha atención al crear cuentas para procesos de autenticación Web de forma que los usuarios dispongan únicamente de los derechos de cuenta que necesiten para ver páginas Web y ningún otro derecho como acceso telnet.

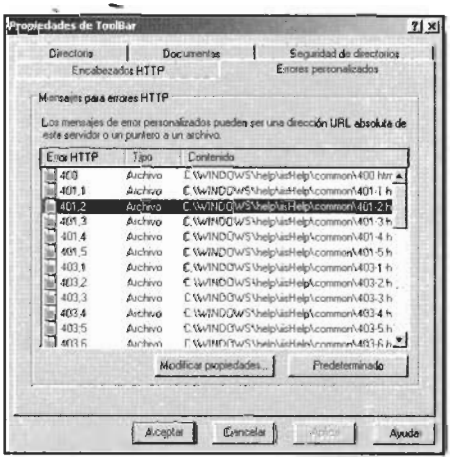


Figura 14.7. La ficha Errores personalizados permite asociar páginas de error con eventos de error

Uso de la autenticación mod_auth_mysql

Como ya se ha indicado, el uso de `mod_auth` con Apache es un método sencillo de configurar y eficaz. Sin embargo, como almacena los sitios en un archivo de texto, no resulta práctico para sitios con tráfico que reciban la visita de una gran cantidad de usuarios.

Afortunadamente, puede disfrutar de la misma facilidad de uso que `mod_auth` y de la velocidad de una base de datos utilizando `mod_auth_mysql`. Este módulo funciona prácticamente igual que `mod_auth`, pero como utiliza una base de datos MySQL en lugar de un archivo de texto, permite buscar sobre una larga lista de usuarios rápidamente.

Para utilizar este módulo, es necesario compilarlo e instalarlo en su sistema o pedirle a su administrador de sistema que lo instale.

Instalación de mod_auth_mysql

Para poder utilizar `mod_auth_mysql`, necesitará configurar Apache y MySQL siguiendo las instrucciones incluidas en el Apéndice A con una serie de pasos adicionales. Las instrucciones de los archivos `README` y `USAGE` de esta distribución resultan bastante buenas. A continuación, incluimos un resumen de las mismas.

1. Obtenga el archivo de distribución correspondiente al módulo. Se encuentra incluido en el CD-ROM que se adjunta al libro, pero puede obtener la última versión de

```
http://www.mysql.com/doc/en/Contrib.html
```

o de

```
http://www.mysql.com/Downloads/Contrib/
```

2. Descomprima el código fuente.
3. Cambie al directorio `mod_auth_mysql` y ejecute `configure`. Necesitará indicarle dónde encontrar la instalación MySQL y el código fuente de Apache. Por ejemplo:

```
./configure -with-mysql=/var/mysql -with-apache=../src/apache_1.x.xx
```

pero puede ser diferente.

4. Ejecute `make` y, a continuación, `make install`. Necesitará agregar

```
-activate-module=src/modules/auth_mysql/libauth_mysql.a
```

a los parámetros indicados en `configure` al configurar Apache.

Por ejemplo:

```
./configure  -enable-module=ssl  \
```

```
-activate-module=src/modules/php4/libphp4.a  \
-enable-module=php4 -prefix=/usr/local/apache -enable-shared=ssl \
-activate-module=src/modules/auth_mysql/libauth_mysql.a
```

5. Tras aplicar el resto de los pasos del apéndice A, tendrá que crear una base de datos y una tabla en MySQL para incluir la información de autenticación. No tiene por qué tratarse de una base de datos o tabla diferente; puede utilizar una tabla existente como la base de datos `auth` del ejemplo utilizado anteriormente.
6. Agregue una línea al archivo `httpd.conf` para suministrar al módulo `mod_auth_mysql` los parámetros que necesita para establecer la conexión a MySQL. La directiva presentará este aspecto:

```
Auth_MySQL_Info  hostname  user  password
```

¿Funcionó?

La forma más sencilla de comprobar si la compilación funciona es ver si se inicia el servidor Apache. Para iniciar Apache, escriba lo siguiente en caso de incorporar compatibilidad SSL:

```
/usr/local/apache/bin/apachectl  startssl
```

Si no incorpora compatibilidad SSL, escriba

```
/usr/local/apache/bin/apachectl  start
```

Si se inicia con la directiva `Auth_MySQL_Info` en el archivo `httpd.conf`, el módulo `mod_auth_mysql` se agregará satisfactoriamente.

Uso de mod_auth_mysql

Tras instalar el módulo, su uso resultará más complicado que utilizar `mod_auth`. El listado 14.9 muestra un ejemplo del archivo `.htaccess` que autenticará usuarios con contraseñas cifradas almacenadas en la base de datos creada anteriormente en este capítulo.

Listado 14.9. El archivo .htaccess autentica usuarios utilizando una base de datos MySQL

```
ErrorDocument  401  /chapter14/rejection.html

AuthName "Realm Name"
AuthType Basic

Auth_MySQL_DB  auth
Auth_MySQL_Encryption_Types  MySQL
Auth_MySQL_Password_Table auth
```

```
Auth_MySQL_Username_Field  name
Auth_MySQL_Password_Field  pass

require  valid-user
```

Como puede observar la mayor parte del listado 14.9 es igual a la del listado 14.7. Queda por especificar un documento de error para cuando falla la autenticación (error 401). También especificamos autenticación básica y el nombre del dominio. Como ocurre con el listado 14.7, permitimos el acceso de cualquier usuario válido y autenticado. Como estamos utilizando el módulo `mod_auth_mysql` y no queremos utilizar todos los parámetros predeterminados, disponemos de algunas directivas para especificar cómo debería funcionar. `Auth_MySQL_DB`, `Auth_MySQL_Password_Table`, `Auth_MySQL_Username_Field` y `Auth_MySQL_Password_Field` especifican el nombre de la base de datos, la tabla, el campo de nombre de usuario y el campo de contraseña, respectivamente.

Hemos incluido la directiva `Auth_MySQL_Encryption_Types` para especificar que queremos utilizar el cifrado de contraseñas MySQL. Los valores aceptables son `Plaintext`, `Crypt_DES` o `MySQL`. `Crypt_DES` (el predeterminado), que utiliza el sistema de contraseñas cifradas DES estándar de UNIX.

Desde la perspectiva del usuario, este ejemplo del módulo `mod_auth_mysql` funciona exactamente igual que el ejemplo `mod_auth`. El navegador Web le presentará un cuadro de diálogo. Si la autenticación resulta satisfactoria, se mostrará el contenido. Si la autenticación falla, se devolverá la página de error.

Para muchos sitios Web, el módulo `mod_auth_mysql` es ideal. Es rápido, relativamente sencillo de implementar y permite utilizar cualquier mecanismo que resulte práctico para agregar entradas de base de datos para nuevos usuarios. Para obtener más flexibilidad y la posibilidad de lograr un control más exhaustivo sobre secciones de las páginas, puede implementar su propio sistema de autenticación utilizando PHP y MySQL.

Creación de su propio sistema de autenticación

Hemos visto cómo crear nuestros propios métodos de autenticación incluyendo algunos defectos y exige ciertos compromisos, y hemos utilizado métodos de autenticación incorporados, que resultan menos flexibles que escribir nuestro propio código. En un capítulo posterior, cuando abordemos el tema del control de sesiones, podremos escribir nuestro propio sistema de autenticación con menos compromisos que en este capítulo.

En un capítulo posterior, desarrollaremos un sistema de autenticación simple de usuario que evite algunos de los problemas vistos aquí utilizando sesiones para realizar el seguimiento de variables entre páginas.

En otro capítulo posterior, aplicaremos este enfoque a un proyecto del mundo real y veremos cómo se puede utilizar para implementar un sistema de autenticación más exhaustivo.

Lecturas adicionales

Los detalles de la autenticación HTTP se especifican en la RCF 2617, que está disponible en

```
http://www.rfc-editor.org/rfc/rfc2617.txt
```

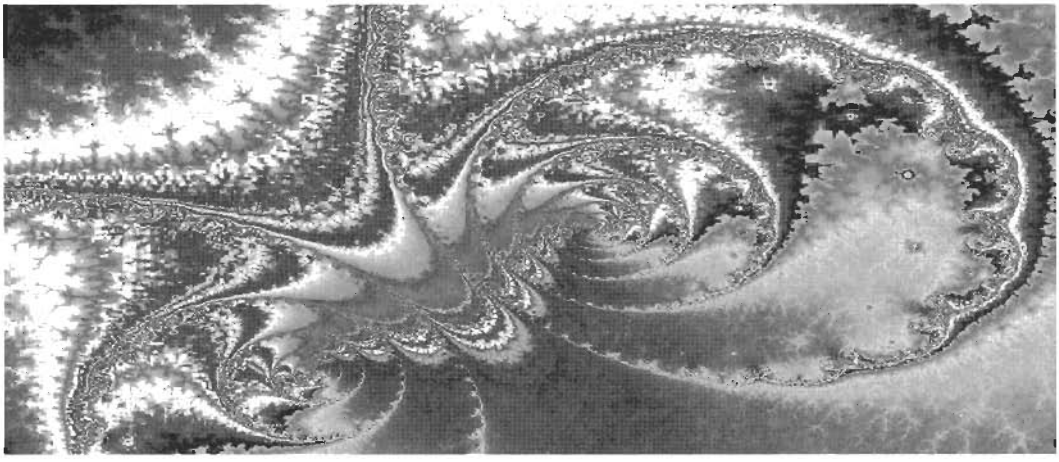
La documentación de `mod_auth`, que controla la autenticación básica en Apache, puede encontrarla en

```
http://www.apache.org/docs/mod/mod_auth.html
```

La documentación del módulo `mod_auth_mysql` se incluye en el archivo de descarga. El tamaño de este archivo es pequeño, por lo que merece la pena descargarlo.

A continuación

En el siguiente capítulo se explica cómo salvaguardar los datos en todas las fases de procesamiento, desde su entrada, hasta su transmisión y almacenamiento. Incluye el uso de SSL, certificados digitales y cifrado.



15
Implementación de transacciones seguras con PHP y MySQL

En este capítulo, vamos a explicar cómo tratar de forma segura los datos de usuario desde la introducción y transmisión hasta la fase de almacenamiento, lo que nos permitirá implementar una transacción entre nosotros y un usuario de forma segura de extremo a extremo. En este capítulo nos centraremos en los siguientes aspectos:

- Cómo suministrar transacciones seguras
- Uso de SSL (Secure Sockets Layer, Capa de sockets segura)
- Almacenamiento seguro
- Por qué almacenar números de tarjeta de crédito
- Uso de la encriptación en PHP

Cómo suministrar transacciones seguras

Para garantizar el suministro de transacciones seguras a través de Internet es necesario examinar el flujo de información de nuestro sistema y garantizar la seguridad de la información en cada punto. En el tema de la seguridad de Internet, no hay conceptos absolutos. Ningún sistema tiene garantizada su impenetrabilidad para siempre. El término seguro indica que el nivel de esfuerzo aplicado a un sistema o transmisión es alto comparado con el valor de la información.

Si queremos dirigir nuestros esfuerzos de seguridad de manera eficaz, necesitamos examinar el flujo de la información a través de todos las partes del sistema. En la figura 15.1 se ilustra el flujo que sigue la información de usuario en una aplicación típica escrita utilizando PHP y MySQL.

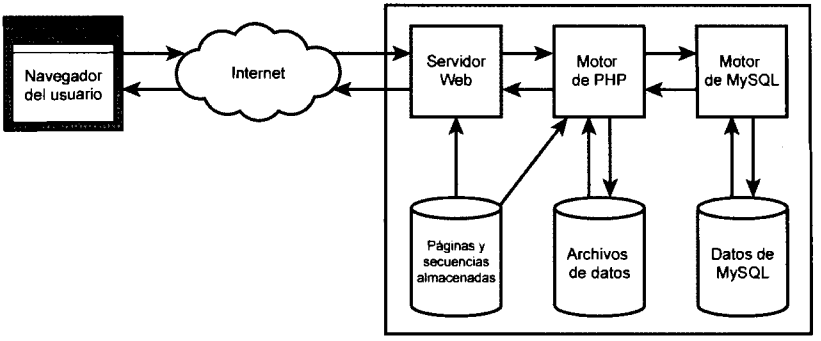


Figura 15.1. La información de usuario es almacenada o procesada por los siguientes elementos de un entorno de aplicación Web típico.

Los detalles de cada transacción que tengan lugar en su sistema pueden variar en función del diseño, de los datos del usuario y de las acciones que desencadenaron la transacción. Puede examinar todos estos elementos de una manera similar. Cada transacción entre una aplicación Web y un usuario comienza con el envío de una solicitud desde el navegador del usuario a través de Internet hasta el servidor Web. Si la página es una secuencia de comandos de PHP, el servidor Web delegará el procesamiento de la página en el motor PHP.

La secuencia de comandos de PHP puede leer y escribir datos en el disco. También puede incluir o requerir otros archivos de PHP o HTML. Así mismo puede enviar consultas de SQL al demonio de MySQL y recibir respuestas. El motor de MySQL es responsable de leer y escribir sus propios datos en el disco.

Este sistema tiene tres partes fundamentales:

- El equipo del usuario
- Internet
- El sistema

A continuación, analizaremos los aspectos de seguridad relacionados con cada una de estas partes de manera separada, pero obviamente el equipo del usuario e Internet son las que quedan más lejos de nuestro control.

El equipo del usuario

Desde nuestro punto de vista, el equipo del usuario ejecuta un navegador Web. No disponemos de control sobre factores como la seguridad aplicada al equipo. Debemos

tener en cuenta que el equipo puede resultar muy poco seguro o que se trate de un terminal compartida en una biblioteca, escuela o café.

Existen muchos navegadores diferentes en el mercado y cada uno de ellos incorpora funciones diferentes. Si consideremos únicamente las versiones más recientes de los dos navegadores más populares, las innumerables diferencias entre ellos afectarán a la forma de generar y representar el código HTML, a lo que hay que añadir las cuestiones de seguridad y funcionalidad.

Debemos tener en cuenta que muchos usuarios deshabilitarán funciones que consideren como un riesgo para su seguridad o privacidad, como las funciones de Java, las cookies o JavaScript. Si utilizamos estas funciones, deberíamos comprobar que nuestra aplicación no pierde demasiado atractivo para los usuarios sin ellas o considerar el uso de una interfaz menos completa que les permita utilizar nuestro sitio.

Los usuarios situados fuera de los EEUU y Canadá puede que utilicen navegadores que sólo admitan un sistema de encriptación de 40 bits. Aunque el gobierno de los EEUU ha cambiado la ley en enero de 2002 para permitir la exportación de encriptación segura (excluyendo a aquellos países sobre los que pese un embargo) y las versiones de 128 bits ya están disponibles para la mayoría de los usuarios, es probable que muchos no hayan actualizado sus sistemas. Si no incluye garantías de seguridad a los usuarios de manera textual en su sitio, este hecho no debería ser motivo de preocupación para los desarrolladores Web ya que SSL se encargará de procesar automáticamente el nivel de seguridad más alto que pueda establecer entre el servidor y el navegador del usuario.

No podemos estar seguros de que la conexión a nuestro sitio Web se produzca desde un navegador Web utilizando la interfaz desarrollada para tal efecto. Las peticiones que llegan a nuestro sitio pueden proceder de otro sitio que esté utilizando imágenes o contenidos sin permiso de su propietario, o de una persona que esté utilizando software como cURL para evitar las medidas de seguridad.

En un capítulo posterior examinaremos la biblioteca de cURL, que se puede utilizar para simular conexiones desde un navegador. Como desarrolladores, esta biblioteca nos resulta de utilidad pero recuerde que se puede utilizar de manera malintencionada.

Aunque no podemos cambiar o controlar la forma en la que están configurados los equipos de nuestros usuarios, debemos tenerlos en cuenta. La variedad de equipos utilizados por los usuarios para conectarse a Internet puede convertirse en un factor importante a la hora de decidir cuánta funcionalidad suministrar a través de secuencias de comandos del lado del servidor (como PHP) y cuánta a través de secuencias de comandos del lado del cliente (como JavaScript).

La funcionalidad suministrada por PHP puede ser compatible con el navegador de todos los usuarios ya que el resultado final no es más que una página HMTL. El uso de cualquier otro elemento que no sea código JavaScript básico exigirá tener en cuenta las diferentes funciones de cada versión de los navegadores.

Desde la perspectiva de la seguridad, conviene utilizar secuencias de comandos del lado del servidor para operaciones como la validación de datos porque, de esta forma, nuestro código fuente no resultará visible para el usuario. Si validamos los datos en JavaScript, los usuarios podrán ver el código y quizás evitarlo.

Los datos que resulte necesario guardar podemos almacenarlos en nuestros propios equipos, como archivos o registros de base de datos o en los equipos de nuestros usuarios en forma de cookies. En un capítulo posterior veremos cómo utilizar cookies para almacenar datos restringidos (como una clave de sesión).

La mayor parte de los datos que almacenamos deberían alojarse en el servidor Web o en nuestra base de datos. Existen buenas razones para almacenar la menor cantidad posible de datos en el equipo del usuario. Si la información no se almacena en nuestro equipo, no dispondremos de control sobre la seguridad de su almacenamiento, no podremos estar seguros de que el usuario no la eliminará ni podremos impedir que la modifique para intentar confundir al sistema.

Internet

Al igual que ocurre con el equipo del usuario, tenemos muy poco control sobre las características de Internet, pero eso no significa que tengamos que ignorarlas al diseñar el sistema.

Internet incorpora una gran cantidad de funciones interesantes, pero es un lugar inseguro de manera inherente. Al enviar información desde un punto a otro, es necesario tener en cuenta que otros podrían verla o alterarla, como se comentó en un capítulo anterior. Con esto en mente, puede decidir qué acción tomar.

Podría optar por:

- Transmitir la información de todas formas, sabiendo que podría leerse.
- Cifrar o firmar la información antes de transmitirla para mantener su privacidad o protegerla ante posibles manipulaciones.
- Decidir que la información es demasiado valiosa para arriesgarse a divulgarla y buscar otra forma de distribuirla.

Internet es también un medio bastante anónimo. Resulta difícil determinar si una persona con la que mantiene relación resulta ser quién dice ser. Incluso en el caso de que podamos asegurarnos de que un usuario es quién dice ser, resultaría difícil demostrarlo sin ninguna duda en un foro público como un tribunal, lo que puede dar lugar a problemas como el del repudio, visto en un capítulo anterior.

En definitiva, las transacciones en Internet presentan dos problemas fundamentales, la privacidad y el repudio.

Existen al menos dos formas diferentes de garantizar el flujo de la información desde nuestro servidor Web hasta Internet y viceversa:

- SSL (Capa segura de sockets)
- S-HTTP (Protocolo de transferencia de hipertexto seguro)

Ambas tecnologías ofrecen mensajes privados y no manipulables, y autenticación, pero SSL está más extendido y resulta más sencillo de implementar mientras que S-HTTP no acaba de calar. En este capítulo examinaremos SSL.

Su sistema

La parte del universo sobre la que disponemos de control es nuestro sistema. Nuestro sistema viene representado por los componentes que se incluyen dentro del cuadro ilustrado en la figura 15.1. Estos componentes pueden estar separados físicamente en una red o incluirse en un único equipo físico.

Resulta bastante seguro despreocuparse de la seguridad de la información si delegamos esta tarea en productos de terceros. Es probable que los autores de estas aplicaciones hayan dedicado más tiempo al tema que nosotros. Mientras utilice una versión actualizada de un producto reconocido, podrá descubrir cualquier problema conocido mediante la acertada aplicación de Google o su motor de búsqueda preferido. Debería asignar máxima prioridad a la tarea de mantener esta información al día.

Si es responsable de los procesos de instalación y configuración, deberá preocuparse por la forma de instalar y configurar el software. Muchos errores de seguridad surgen como resultado de no seguir las advertencias de la documentación o implican aspectos relativos a la administración general del sistema. Cómprese un buen libro sobre la administración del sistema operativo que tenga la intención de utilizar o contrate los servicios de un administrador de sistema con experiencia.

Al instalar PHP debería tener en cuenta que resulta más seguro y más eficiente hacerlo en forma de un módulo SAPI para el servidor Web que ejecutarlo a través de la interfaz CGI.

Un aspecto que debe considerar es qué pueden hacer sus secuencias de comandos.

Pregúntese qué datos potencialmente delicados trasmite su aplicación al usuario a través de Internet y qué datos estamos pidiendo a nuestros usuarios que nos envíen. Si estamos transmitiendo información entre nosotros y nuestros usuarios cuya privacidad debamos garantizar o cuya manipulación debamos dificultar, deberíamos considerar la opción de utilizar SSL.

Ya hemos hablado del uso de SSL entre el equipo del usuario y el servidor. También deberíamos prever la situación en la que se transmitan datos desde un componente de nuestro sistema a otro a través de la red. Un ejemplo típico sería si nuestra base de datos estuviera alojada en un equipo diferente a nuestro servidor Web. PHP se conectaría al servidor MySQL a través de TCP/IP y esta conexión no estaría encriptada. Si los equipos estuvieran en una red de área local privada, debería asegurarse de que la red es segura. Si los equipos se conectan a través de Internet, el sistema probablemente resulte lento y necesite tratar la conexión de la misma forma que otras conexiones sobre Internet.

Es importante asegurarse de que cuando los usuarios crean estar interactuando con nosotros que así sea. La obtención de un certificado digital protegerá a nuestros visitantes contra prácticas de personificación y nos permitirá utilizar SSL sin que los usuarios vean aparecer un mensaje de aviso además de aportar un aire de respetabilidad a nuestra aventura en línea.

¿Comprueban minuciosamente nuestras secuencias de comandos los datos introducidos por los usuarios?

¿Tenemos cuidado de almacenar la información de manera segura?
En las siguientes secciones responderemos a estas preguntas.

Uso SSL

El protocolo SSL fue diseñado originalmente por Netscape para facilitar la seguridad de las comunicaciones entre servidores y navegadores Web. Desde entonces ha sido adoptado como el método estándar de manera no oficial por los navegadores y servidores para intercambiar información delicada.

Existe una amplia compatibilidad para la versión 2 y 3 de SSL. La mayor parte de los servidores Web incluyen funcionalidad SSL o pueden aceptarla como módulo complementario. Internet Explorer y Netscape incorporan compatibilidad SSL desde la versión 3.

Los protocolos de red y el software que los implemente se suelen organizar en forma de pila de capas. Cada capa puede pasar datos a la capa superior e inferior, y solicitar servicios de la capa situada por encima o por debajo. La figura 15.2 muestra una pila de protocolos.

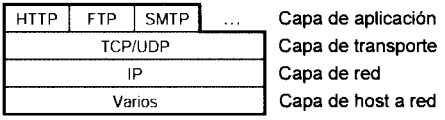


Figura 15.2. La pila de protocolos utilizada por un protocolo de capa de aplicación como el Protocolo de transferencia de hipertexto

Cuando se utiliza HTTP para transferir información, el protocolo HTTP llama al Protocolo de control de transmisión (TCP) que a su vez se basa en el Protocolo de Internet. Este protocolo necesita un protocolo adecuado para que el hardware de red pueda utilizarse para tomar paquetes de datos y enviarlos como señal eléctrica a nuestro destino. HTTP es un protocolo de capa de aplicación. Existen muchos otros protocolos de capa de aplicación, como FTP, SMTP y telnet (como se muestra en la figura) y otros como POP e IMAP. TCP es uno de los dos protocolos de capa de transporte utilizados en redes TCP/IP. IP es el protocolo de la capa de red. El host hasta la capa de red es responsable de conectar nuestro host (equipo) a una red. La pila de protocolos TCP/IP no especifica los protocolos utilizados para esta capa, ya que necesitamos diferentes protocolos para los diferentes tipos de redes.

Al enviar datos, éstos se dirigen a través de la pila desde una aplicación a un dispositivo físico de red. Al recibir los datos, éstos se dirigen desde la red física, a través de la pila, hasta la aplicación.

El uso de SSL agrega una capa transparente adicional a este modelo. La capa SSL existe entre la capa de transporte y la capa de aplicación, como se ilustra en la figura 15.3. La capa SSL modifica los datos desde nuestra aplicación HTTP antes de pasarlos a la capa de transporte para enviarlos a su destino.

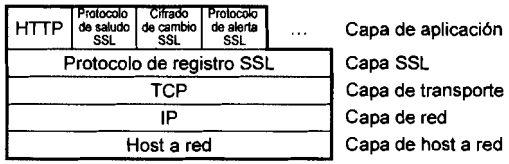


Figura 15.3. SSL agrega una capa adicional a la pila de protocolos así como protocolos de capa de aplicación para controlar sus propias operaciones

- SSL es teóricamente capaz de crear un entorno de transmisión seguro para otros protocolos aparte de HTTP, pero se suele utilizar para HTTP. Se pueden utilizar otros protocolos porque la capa SSL es básicamente transparente. Esta capa proporciona la misma interfaz a los protocolos situados por encima que la capa de transporte subyacente. A continuación se encarga, de forma transparente, de los saludos, de la encriptación y de la decriptación.
  Cuando un navegador Web se conecta a un servidor Web seguro a través de HTTP, los dos necesitan seguir un protocolo de saludo para llegar a un acuerdo sobre elementos como la autenticación y la encriptación.

La secuencia de saludo implica los siguientes pasos:

1. El navegador se conecta a un servidor con SSL habilitado y solicita que se autentique.
2. El servidor envía su certificado digital.
3. El servidor puede solicitar opcionalmente (aunque no suele ser lo normal) al navegador que se autentique.
4. El navegador presenta una lista con los algoritmos de encriptación y funciones hash que admite. El servidor selecciona la encriptación más segura que admite.
5. El navegador y el servidor generan claves de sesión:
   5.1. El navegador obtiene la clave pública desde su certificado digital y la utiliza para encriptar un número generado de forma aleatoria.
   5.2. El servidor responde con más datos aleatorios en forma de texto sin procesar (a menos que el navegador haya suministrado un certificado digital en la solicitud del servidor en cuyo caso el servidor utiliza la clave pública del navegador).
   5.3. Las claves de encriptación para la sesión se generan a partir de estos datos aleatorios utilizando funciones hash.

El proceso de saludo no es instantáneo porque la generación de datos aleatorios de calidad, la decriptación de los certificados digitales, la generación de las claves y el uso de la criptografía de clave pública lleva su tiempo. Afortunadamente, los resultados se guardan en caché, de manera que si el mismo navegador y servidor desean intercambiar varios mensajes seguros, el proceso de saludo y el tiempo de procesamiento requerido sólo tendrán lugar una vez.

Cuando los datos se envían a través de una conexión SSL, tienen lugar los siguientes pasos:

1. Se dividen en paquetes que resulten sencillos de procesar.
2. Cada paquete se comprime (opcionalmente).
3. Cada paquete lleva asignado un código de autenticación de mensaje (MAC) calculado utilizando un algoritmo hash.
4. El código MAC y los datos comprimidos se combinan y se encriptan.
5. Los paquetes encriptados se combinan con información de cabecera y se envían a la red.

En la figura 15.4 se ilustra todo el proceso.

Como observará en el diagrama, el encabezado TCP se agrega tras encriptar los datos, lo que significa que la información dirigida podría manipularse y aunque los husmeadores no puedan ver la información que estamos intercambiando sí podrían ver quién lo está haciendo.

La razón por la que SSL realiza la compresión antes de la encriptación responde a que, aunque el tráfico de red pueda (como ocurre a menudo) comprimirse antes de transmitirse por una red, los datos encriptados no se comprimen bien.

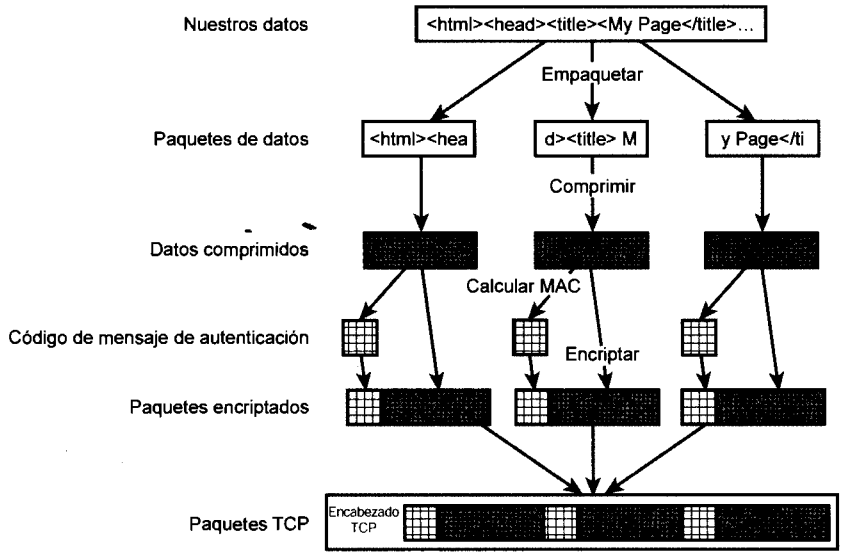


Figura 15.4. SSL divide, comprime, aplica una función hash a los datos y los encripta antes de enviarlos

Los sistemas de compresión se basan en la identificación de repeticiones o patrones dentro de los datos. El intento de aplicar un algoritmo de compresión tras convertir los datos en un conjunto aleatorio de bits mediante un proceso de encriptación no

tiene mucho sentido. Sería una pena si SSL, que fue diseñado para incrementar la seguridad de la red, tuviera el efecto secundario de incrementar el tráfico de red.

Aunque SSL es relativamente complejo, los usuarios y los desarrolladores resultarán ajenos a gran parte de lo que ocurre, ya que sus interfaces externas simulan protocolos existentes.

En el futuro, SSL 3.0 podría ser sustituido por TLS 1.0 (Seguridad de capa de transporte) pero en el momento de escribir estas líneas, TLS se encuentra todavía en la fase de estándar de borrador y no resulta compatible con la mayoría de los servidores y navegadores. La intención de TLS es la de convertirse en un verdadero estándar abierto, en lugar de un estándar definido por una organización que se ha puesto a disposición de otras. TLS se basa directamente en SSL 3.0 pero contiene mejoras pensadas para superar los puntos débiles de SSL.

Filtrado de la entrada de los usuarios

Uno de los principios de la construcción de una aplicación Web segura es que no se debe confiar nunca en la información introducida por el usuario. Filtre siempre los datos introducidos por el usuario antes de colocarlos en un archivo o base de datos, o de pasarlos a través de un comando de ejecución del sistema.

A lo largo de este libro hemos hablado varias veces de las técnicas que se pueden utilizar para filtrar las entradas de los usuarios. A continuación, los recogemos brevemente para recordarlas.

- La función `addslashes()` debería utilizarse para filtrar los datos del usuario antes de pasarlos a una base de datos. Esta función evitará todos los caracteres especiales que podrían presentar problemas en una base de datos. Puede utilizar la función `stripslashes()` para devolver los datos a su forma original.
- Comillas mágicas. Puede activar las directivas `magic_quotes_gpc` y `magic_quotes_runtime` en su archivo php.ini. Estas directivas agregan y quitan las barras invertidas automáticamente. La directiva `magic_quotes_gpc` aplicará este formato a las variables GET, POST y cookies entrantes y la directiva `magic_quotes_runtime` lo aplicará a los datos dirigidos o procedentes de bases de datos.
- La función `escapeshellcmd()` debería utilizarse al pasar datos del usuario a llamadas `system()` o `exec()` o a apóstrofes invertidos. Esta función evitará todos los metacaracteres que se puedan utilizar para obligar al sistema a ejecutar comandos arbitrarios introducidos por un usuario malintencionado.
- Puede utilizar la función `strip_tags()` para limpiar de etiquetas HTML y PHP una cadena. Esta función evitará que los usuarios introduzcan secuencias de comandos malintencionados en los datos que puedan representarse en el navegador.

- Puede utilizar la función `htmlspecialchars()`, que convierte los caracteres en sus equivalentes de entidad HTML. Por ejemplo, < se convertirá en `<`. Esta función convierte todas las etiquetas de secuencia de comandos en caracteres inofensivos.

Suministro de un almacenamiento seguro

Los tres tipos diferentes de datos almacenados (archivos HTML o PHP, datos relacionados con secuencias de comandos y datos MySQL) se almacenarán en áreas diferentes del mismo disco, aunque se muestran de forma separada en la figura 15.1. Cada tipo de almacenamiento requiere adoptar distintas precauciones y se examinará de manera individual.

El tipo de datos más peligroso que almacenaremos es el contenido ejecutable. En un sitio Web, suelen adoptar la forma de secuencias de comandos. Es necesario poner mucho cuidado al establecer los permisos de archivo dentro de la jerarquía Web. En concreto, el árbol de directorios cuyo primer elemento es `htdocs` en un servidor Apache y `inetpub` en un servidor IIS. Otros necesitan disponer de permisos para leer nuestras secuencias de comandos para poder ver su salida, pero no deberían tener permiso para escribir o editarlos.

La misma advertencia se aplica a los directorios incluidos dentro de la jerarquía Web. Sólo nosotros deberíamos disponer de permiso para escribir en dichos directorios. Otros usuarios, incluido el usuario utilizado por el servidor Web para ejecutarse, no deberían disponer de permiso para escribir o crear nuevos archivos en directorios que se puedan cargar desde el servidor Web. Si permite que otros escriban archivos en este punto, podrían crear una secuencia de comandos malintencionada y ejecutarla al cargarla a través del servidor Web.

Si sus secuencias de comandos necesitan disponer de permiso para escribir en los archivos, cree un directorio fuera del árbol Web para ello, en especial si se trata de secuencias de comandos para la carga de archivos en el servidor. No deberían mezclarse las secuencias de comandos y los datos que escriben.

Al escribir datos delicados, es probable que se vea tentado a encriptarlos como primer paso. En este enfoque, sin embargo, no resultará de mucho valor.

Veamos por qué: si tememos un archivo llamado `númerostajetascrédito.txt` en nuestro servidor Web y un pirata informático obtiene acceso a nuestro servidor y logra leerlo, ¿qué podría leer? Para poder encriptar y decriptar los datos necesitamos un programa que los cifre y un programa que los descifre, y uno o varios archivos de claves. Si el pirata puede leer los datos, nada le impedirá, con toda probabilidad, leer los archivos de clave así como otros archivos.

La encriptación de los datos puede resultar valiosa en un servidor Web, pero sólo si el software y las claves para decriptar los datos no se almacenan en el servidor Web sino en otro equipo. Una forma de tratar datos delicados de forma segura sería encriptarlos en el servidor y transmitirlos a otro equipo, por correo electrónico, por ejemplo.

Los datos de la base de datos son similares a los archivos de datos. Si se configura MySQL de manera correcta, sólo MyQSL podrá escribir en sus archivos de datos, lo cual significa que sólo necesitamos preocuparnos por los accesos de los usuarios dentro de MySQL. Ya se ha comentado el sistema de permisos de MySQL, que asigna derechos concretos a nombres de usuarios concretos en hosts concretos.

También hay que destacar el hecho de que a menudo necesitaremos escribir una contraseña de MySQL en una secuencia de comandos de PHP. Por regla general, las secuencias de comandos de PHP se pueden cargar públicamente. Este hecho no es tan peligroso como pudiera parecer a primera vista ya que si no se penetra en la configuración del servidor Web, el código PHP no resultará visible desde el exterior.

Si el servidor Web se configura para analizar archivos con la extensión .php usando el intérprete de PHP, no se podrá ver desde fuera el código sin interpretar. Sin embargo, conviene tener cuidado al utilizar otras extensiones. Si coloca archivos .inc en sus directorios Web, cualquier persona que los solicite recibirá el código sin analizar. Por lo tanto, será necesario colocar los archivos de inclusión fuera del árbol Web, configurar el servidor para que entregue archivos en esta extensión o utilizar .php también como extensión en éstos. Si está compartiendo un servidor Web con otros, las contraseñas de MySQL podrían resultar visibles a otros usuarios del mismo equipo que puedan ejecutar secuencias de comandos a través del mismo servidor Web. Esta situación puede resultar inevitable en función de la configuración del sistema. Para evitarla, puede configurar el servidor Web de manera que ejecute secuencias de comandos como usuarios individuales o que cada usuario ejecute su propia instancia del servidor Web. Si no es el administrador del servidor Web (como ocurrirá con toda probabilidad si comparte un servidor), es aconsejable que comente esta situación con el administrador y busque opciones de seguridad.

¿Por qué almacenar números de tarjeta de crédito?

Tras comentar el tema del almacenamiento seguro de datos delicados, debemos detenernos en un tipo de datos que merece especial atención. Los usuarios de Internet están especialmente preocupados por sus números de tarjeta de crédito. Si va a almacenar estos números, debe prestar especial cuidado. Pregúntese para qué los necesita y si resulta verdaderamente necesario almacenarlos.

¿Qué va hacer con un número de tarjeta? Si el tipo de transacciones con el usuario va a ser esporádico y su sistema realiza el procesamiento de tarjetas en tiempo real, convendría aceptar el número de tarjeta del cliente y enviarlo directamente a la pasarela de procesamiento de transacciones sin almacenarlo.

Si necesita aplicar cargos cada cierto tiempo, por ejemplo aplicar una cantidad mensual a cada tarjeta por una suscripción, esta opción puede que no resulte la más acertada. En este caso, debería considerar la posibilidad de almacenar los números en algún lugar distinto al servidor Web.

Si va a almacenar grandes cantidades de detalles relativos a las tarjetas de clientes, asegúrese de que el administrador del sistema tenga experiencia en el tema y que disponga de tiempo para comprobar las actualizaciones de la información de seguridad del sistema operativo y de otros productos que utilice.

Uso de encriptación en PHP

Una acción sencilla pero útil a la que podemos recurrir para demostrar el uso de la encriptación es enviar un mensaje de correo electrónico cifrado. El estándar de facto utilizado para encriptar correos electrónicos durante muchos años ha sido PGP, que equivale a Pretty Good Privacy (Privacidad bastante buena). Este estándar fue escrito por Philip R. Zimmermann especialmente para agregar privacidad al correo electrónico.

Existen versiones gratuitas de PGP, pero debe tener en cuenta que no se trata de software gratuito. La versión gratuita sólo se puede utilizar de manera legal con fines no comerciales.

Para obtener una licencia de PGP fuera de los EEUU y Canadá, consulte la lista de sitios de descarga internacional en la página de PGP: `http://www.pgpi.org`.

Recientemente ha surgido una alternativa de código abierto a PGP. GPG (Gnu Privacy Guard) no contiene algoritmos patentados y se puede utilizar comercialmente sin restricción. Ambos productos realizan la misma tarea de forma similar. Si tiene pensado utilizar herramientas de línea de comandos, la elección no importa, pero cada una incorpora interfaces diferentes como complementos para programas de correo electrónico que descifran los mensajes recibidos automáticamente.

GPG está disponible en `http://www.gnupg.org`.

Puede utilizar los dos productos de manera conjunta. Por ejemplo, puede crear un mensaje encriptado con GPG y dirigirlo a alguien que utilice PGP (siempre que sea una versión reciente) para descifrarlo. Como nosotros estamos interesados en la creación de mensajes en el servidor Web, vamos a ver un ejemplo del uso de GPG. Si se utiliza PGP, la operación no requerirá muchos cambios.

Como en el caso de otros ejemplos de este libro, necesitará disponer de GPG para que el código funcione. Puede que ya tenga instalada esta herramienta en su sistema. De no ser así, no se inquiete, ya que el procedimiento de instalación resulta sencillo (aunque la configuración puede complicarse).

Instalación de GPG

Para agregar GPG a nuestro equipo de Linux, descargamos el archivo pertinente de `www.gnupg.org`. Necesitaremos utilizar `gunzip` y `tar` o extraer los archivos del archivo comprimido, dependiendo de su tipo, .tar.gz o .tar.bz2.

Para compilar e instalar el programa, utilice los mismos comandos que en la mayor parte de las versiones de Linux:

```
configure (or ./configurar en función de su sistema)
```

```
make
make install
```

Si no es el usuario raíz, tendrá que configurar la secuencia de comandos con la opción `–prefix` de la siguiente forma:

```
./configure  –prefix=/path/to/your/directory
```

ya que los usuarios distintos al usuario raíz no disponen de acceso al directorio predeterminado de GPG.

Si todo discurre satisfactoriamente, GPG se compilará y el ejecutable se copiará en `/usr/local/bin/gpg` o en el directorio que especifiquemos. Puede cambiar muchas opciones.

Consulte la documentación de GPG si desea obtener más detalles.

Para un servidor Windows, el proceso resulta incluso más sencillo. Descargue el archivo zip, descomprímalo y coloque el archivo gpg.exe en algún punto de su PATH. (C:\Windows\, por ejemplo). Cree un directorio en C:\gnupg. Abra el símbolo de comandos y escriba `gpg`.

También necesitamos instalar GPG o PGP y generar un par de claves en el sistema desde las que tengamos previsto comprobar el correo.

En el servidor Web, existen muy pocas diferencias entre las versiones de línea de comandos de GPG y PGP, por lo que podemos utilizar GPG ya que resulta gratuito. En el equipo desde el que leer el correo, puede que prefiera utilizar una versión comercial de PGP por el complemento de interfaz gráfico que incorpora al lector de correo electrónico.

Si no tiene un par de claves, genérelas en el equipo en el que vaya a leer el correo electrónico. Recuerde que un par de claves se compone de una clave pública que otros usuarios (así como su secuencia de comandos de PHP) utilizarán para encriptar el correo antes de enviárnoslo y una clave privada que se utilizará para descifrar los mensajes recibidos o firmar el correo saliente.

Es importante que el proceso de generación de claves se realice en el equipo en el que se va a leer el correo y no en el servidor Web ya que la clave privada no debe almacenarse en el servidor Web.

Si está utilizando la versión de línea de comandos de GPG para generar las claves, introduzca el siguiente comando:

```
gpg  --gen-key
```

Deberá responder a una serie de preguntas. La mayor parte de ellas incluyen una respuesta predeterminada que puede aceptar. Además, se le pedirá su nombre real, una dirección de correo electrónico y un comentario, que se utilizará para designar a la clave.

Por ejemplo, `'Luke Welling <luke@tangledweb.com.au>'`. Como puede apreciar el patrón resulta claro. Si se incluyera un comentario, se colocaría entre el nombre y la dirección.

Para exportar la clave pública del par de claves, puede utilizar el comando:

```
gpg  –export  >   nombredearchivo
```

Como resultado se generará un archivo binario para su importación en el archivo de claves de GPG o PGP en otro equipo. Si quiere enviar por correo electrónico esta clave a otros usuarios, para que puedan importarla en sus propios archivos de claves, puede crear una versión ASCII como la siguiente:

```
gpg -export -a > nombredearchivo
```

Tras extraer la clave pública, puede cargar el archivo en su cuenta en el servidor Web. Para ello, puede utilizar FTP.

Los siguientes comandos asumen que estamos utilizando UNIX. Los pasos son los mismos que para Windows, pero los nombres de los directorios y los comandos del sistema serán diferentes.

Regístrese en el servidor con su cuenta y cambie los permisos sobre el archivo para que los usuarios puedan leerlo. Escriba:

```
chmod 644 nombredearchivo
```

Necesitará crear un archivo de claves para que el usuario utilizado para ejecutar las secuencias de comandos de PHP pueda usar GPG. Este usuario dependerá de cómo esté configurado el servidor. A menudo suele ser el usuario "nobody", pero puede ser cualquier otro.

Cambie al usuario del servidor Web. Necesitará disponer de acceso de usuario raíz para ello. En la mayor parte de los sistemas, el servidor Web se ejecuta como `nobody`. En los siguientes ejemplos partiremos de este supuesto. (Puede cambiarlo por el usuario adecuado en su sistema.) Si este fuera en caso en su sistema, escriba:

```
su root
su nobody
```

Cree un directorio para el usuario `nobody` y almacene el archivo de claves y otra información de configuración. Este archivo debería incluirse en el directorio raíz del usuario `nobody`.

El directorio raíz de cada usuario se especifica en `/etc/passwd`. En muchos sistemas Linux, el directorio del usuario `nobody` es el directorio predeterminado `/`, sobre el que dicho usuario no tiene permiso para escribir. En muchos sistemas BSD, el directorio predeterminado para el usuario `nobody` es `/noexistente`, que como no existe, no se puede escribir en él. En nuestro sistema, el usuario `nobody` se ha asignado al directorio raíz `/tmp`. Deberemos asegurarnos de que el usuario de servidor Web dispone de un directorio raíz sobre el que tenga permiso para escribir.

Escriba

```
cd ~
mkdir .gnupg
```

El usuario `nobody` necesitará una clave de firma propia. Para crearla, ejecute el siguiente comando de nuevo:

```
gpg -gen-key
```

Como el usuario `nobody` es probable que no reciba ningún correo personal, podemos crear una clave de sólo firma.

El único objetivo de esta clave es permitirnos confiar en la clave pública extraída anteriormente. Para importar la clave pública exportada anteriormente, utilice el siguiente código:

```
gpg -import nombredearchivo
```

Para indicarle a GPG que queremos confiar en esta clave, necesitamos editar sus propiedades de la siguiente forma:

```
gpg -edit-key 'Luke Welling <luke@tangledweb.com.au>'
```

En esta línea, el texto entre comillas es el nombre de la clave. Obviamente, el nombre de la clave no será 'Luke Welling <luke@tangledweb.com.au>', sino una combinación del nombre, el comentario y la dirección de correo electrónico suministrada al generarlo.

Este programa incluye la opción `help`, que describirá los comandos disponibles `trust`, `sign` y `save`.

Escriba **`trust`** para indicarle a GPG que confía en la clave completamente. Escriba **`sign`** para firmar esta clave pública utilizando la clave privada del usuario `nobody`.

Finalmente, utilice **`save`** para salir del programa manteniendo los cambios.

Cómo probar GPG

Ya tenemos configurado GPG y listo para su uso.

La creación de un archivo que contenga alguna secuencia de texto y su almacenamiento como test.txt nos permitirá utilizarlo para guardarlo.

Escriba el siguiente comando (modificado para utilizar el nombre de su clave).

```
gpg -a -recipient 'Luke Welling <luke@tangledweb.com.au>' -encrypt
test.txt
```

debería devolver la siguiente advertencia

```
gpg: Warning: using insecure memory!
```

y crear un archivo llamado test.txt.asc. Si abre este archivo, debería ver un mensaje encriptado como el siguiente:

```
--BEGIN PGP MESSAGE---
Version: GnuPG v1.0.3 (GNU/Linux)
Comment: For info see http://www.gnupg.org

hQEOA0DU7hVGgdtnEAQAhr4HgR7xpIBsK9CiELQw85+k1QdQ+p/FzqL8tICrQ+B3
0GJTEehPUDErwqUw/uQLTds0r1oPSrIAZ7c6GVkh0YEVBj2MskT81IIBvdo95OyH
K9PUCvg/rLxJ1kxe4Vp8QFET5E3FdII/1y8VP5gSTE7gAgm0SbFf3S91PqwMyTkD
/2oJEvL6e3cP384s0i81rBbDbOUAAhCjjXt2DX/uX9q6P18QW56UICUOn4DPaW1G
/gnNZCkcVDgLcKfBjbkB/TCWWhpA7o7kX4CIcIh7K1IMHY4RKdnCWQf271oE+8i9
```

```
cJRSCMsFIoI6MMNRCQHY6p9bfxL2uE39IRJrQbe6xoEe0nkB0uTYxiL0TG+FrNrE
tvBVMS0nsHu7HJey+oY4Z833pk5+MeVwYumJwlvHjdZxZmV6wz46GO2XGT17b28V
wSBnWOoBHSZsPvkQXHTOq65EixP8y+YJvBN3z4pzdH0Xa+NpqbH7q3+xXmd30hDR
+u7t6MxTLDbgC+NR
=gfQu
--END PGP MESSAGE---
```

Debería poder transferir este archivo al sistema en que generó la clave inicialmente y ejecutar

```
gpg test.txt.asc
```

para ver el texto original de nuevo.

Para colocar el texto en un archivo, en lugar de mostrarlo en la pantalla, puede utilizar el modificador -o y especificar un archivo de salida de la siguiente forma:

```
gpg -do test.out test.txt.asc
```

Si tiene configurado GPG para que el usuario utilizado para ejecutar las secuencias de comandos PHP pueda usarlo desde al línea de comandos.

Si no funcionara, consulte a su administrador del sistema o la documentación de GPG. El listado 15.1 y 15.2 permite enviar correo electrónico encriptado utilizando PHP para llamar a GPG.

Listado 15.1. private_mail.php. Nuestro formulario HTML para enviar correo electrónico encriptado

```
<html>
<body>
<h1>Send Me Private Mail</h1>

<?php
   // puede que necesite cambiar esta línea, si no utiliza
   //los puertos predeterminados; 80 para el tráfico normal y 443 para SSL
   if($HTTP_SERVER_VARS['SERVER_PORT']!=443)
      echo '<p><font color="red">
         WARNING: you have not connected to this page using SSL.
         Your message could be read by others.</font></p>';
?>

<form method="post" action="send_private_mail.php"><br />
Your email address:<br />
<input type="text" name="from" size"="38"><br />
Subject:<br />
<input type="text" name="title" size="38"><br />
Your message:<br />
<textarea name="body" cols="30" rows="10">
</textarea><br />
<input type="submit" value="Send!">
</form>
</body>
</html>
```

Listado 15.2. send_private_mail.php. Nuestra secuencia de comandos de PHP para llamar a GPG y enviar correos electrónicos encriptados

```
<?php
  //cree nombres de variables cortos
  $from = $HTTP_POST_VARS['from'];
  $title = $HTTP_POST_VARS['title'];
  $body = $HTTP_POST_VARS['body'];

  $to_email = 'luke@localhost';

  // Indique a gpg dónde encontrar el archivo de claves
  // En este sistema, el directorio raíz del usuario nobody es /tmp/
  putenv('GNUPGHOME=/tmp/.gnupg');

  //cree un nombre de archivo único
  $infile = tempnam('', 'pgp');
  $outfile = $infile.'.asc';

  //escriba el texto del usuario en el archivo
  $fp = fopen($infile, 'w');
  fwrite($fp, $body);
  fclose($fp);

  //configure el comando
  $command ="/usr/local/bin/gpg -a \\
            -recipient 'Luke Welling <luke@tangledweb.com.au>' \\
            -encrypt -o $outfile $infile";

  // ejecute el comando gpg
  system($command, $result);

  //elimine el archivo temporal no encriptado
  unlink($infile);

  if($result==0)
  {
     $fp = fopen($outfile, 'r');
     if(!$fp||filesize ($outfile)==0)
     {
        $result = -1;
     }
     else
     {
        //lea el archivo encriptado
        $contents = fread ($fp, filesize ($outfile));
        //elimine el archivo temporal encriptado
        unlink($outfile);

        mail($to_email, $title, $contents, "From: $from\n");
        echo '<h1>Message Sent</h1>
              <p>Your message was encrypted and sent.</p>
              <p>Thank you.'</p>';
     }
  }

  if($result!=0)
```

```
    {
        echo  '<h1>Error:</h1>
                <p>Your message could not be encrypted, so has not been
                    sent.</p>
                <p>Sorry.</p>';
    }
?>
```

Para que este código funcione, necesitará cambiar algunos elementos. El correo electrónico se enviará a la dirección incluida en `$to_email`.

Tendrá que modificar la línea

```
putenv('GNUPGHOME=/tmp/.gnupg');
```

para reflejar la ubicación del archivo de claves GPG. En nuestro sistema, el servidor Web se ejecutará como usuario `nobody` y el directorio raíz será `/tmp/`.

Estamos utilizando la función `tmpnam()` para crear un nombre de archivo temporal único. Puede especificar el directorio y un prefijo de nombre de archivo. Vamos a crear y a eliminar estos archivos en un segundo, por lo que no resulta muy importante cómo los llamemos. Vamos a especificar el prefijo 'pgp', pero vamos a permitir que PHP utilice el directorio temporal del usuario.

La instrucción

```
$command =  '/usr/local/bin/gpg -a '.
            '-recipient 'Luke Welling <luke@tangledweb.com.au>' '.
            '-encrypt -o $outfile $infile';
```

configura el comando y los parámetros que se utilizarán para llamar a gpg. Deberá modificarlo para adaptarlo a sus necesidades.

Al utilizarlo en la línea de comandos tendrá que indicar a GPG qué clave utilizar para encriptar el mensaje.

La instrucción

```
system($command, $result);
```

ejecuta los comandos almacenados en `$command` y almacena el valor devuelto en `$result`. Podemos ignorar el valor devuelto, pero nos permite tener una instrucción `if` e indicar al usuario que algo ha ido mal.

Cuando hayamos terminado con los archivos temporales utilizados, los eliminamos con la función `unlink()`. Por lo tanto, el correo no encriptado del usuario se almacenará muy poco tiempo en el servidor.

Y si el servidor fallara durante la ejecución, el archivo podría quedarse en el servidor.

Al considerar la seguridad de nuestra secuencia de comandos, es importante tener en cuenta todos los flujos de información dentro del sistema. GPG encriptará nuestra secuencia de comandos y permitirá que nuestro destinatario la descifre, pero cabe preguntarse cómo llegó la información originalmente desde el remitente. Si suministramos una interfaz Web para enviar correos GPG encriptados, el flujo de la información seguiría la ruta ilustrada en la figura 15.5.

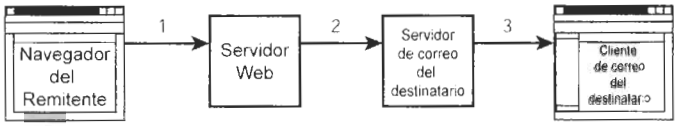


Figura 15.5. En nuestra aplicación de correo electrónico cifrado, el mensaje se envía a través de Internet tres veces

En esta figura, cada flecha representa el mensaje enviado desde un equipo a otro. Cada vez que se envía el mensaje, viaja a través de Internet y puede pasar por una gran cantidad de redes y equipos intermedios.

La secuencia de comandos que estamos examinando aquí existe en el equipo denominado Servidor Web en el diagrama. En el servidor Web, el mensaje se encriptará utilizando la clave pública del destinatario. A continuación se enviará a través de SMPT al servidor de correo del destinatario. Éste se conectará a su servidor de correo electrónico, utilizando POP o IMAP probablemente, y descargará el mensaje utilizando un lector de correo. En este punto descifraremos el mensaje utilizando su clave privada. Los datos transferidos en la figura 15.5 se etiquetan como 1, 2 y 3. Para las fases 2 y 3, la información transmitida es un mensaje GPG encriptado y no tienen ningún valor para nadie que no tenga la clave privada. Para la transferencia 1, el mensaje transmitido es el texto que el remitente introduce en el formulario.

Si nuestra información es lo suficientemente importante como para encriptarla en la segunda y tercera etapa del viaje, es un poco tonto enviarla sin encriptar en la primera fase. Por lo tanto, esta secuencia de comandos pertenece a un servidor que utiliza SSL.

Si conectamos a nuestra secuencia de comandos utilizando un puerto distinto al 443, se generará una advertencia. Este puerto es el puerto utilizado por SSL de manera predeterminada. Si su servidor utiliza un puerto distinto para SSL, necesitará modificar este código.

En lugar de suministrar un mensaje de error, podemos procesar esta situación de otras formas. Podemos redirigir al usuario al mismo URL a través de una conexión SSL. Podemos optar por ignorarlo porque por regla general no resulta importante si el formulario se entrega utilizando una conexión segura. Lo que si suele ser importante es que los detalles que el usuario ha introducido en el formulario se nos envíen de forma segura. Bastaría con indicar un URL completo como acción en nuestro formulario.

En la actualidad, la etiqueta de apertura del formulario presenta este aspecto:

```
<form method="post" action="send_private_mail.php">
```

Podríamos modificarla para enviar datos a través de SSL incluso si el usuario se conecta sin SSL, de la siguiente forma:

```
<form method="post" action="https://webserver/send_private_mail.php">
```

Si codificamos de manera específica el URL completo de esta forma, nos aseguraremos de que los datos del visitante se envíen utilizado SSL, pero necesitaremos mo-

dificar el código cada vez que lo utilicemos en otro servidor o incluso en otro directorio.

Aunque en este caso y en muchos otros, no es importante enviar el formulario al usuario a través de SSL, resulta una buena idea hacerlo así. Si aparece el símbolo de un candado en la barra de estado de sus navegadores, los usuarios podrán estar seguros de que la información se está enviando de forma segura. No necesitarán examinar el código HTML para determinar el atributo de acción.

Lecturas adicionales

La especificación de la versión 3 de SSL podrá encontrarla en Netscape:
`http://home.netscape.com/eng/ssl3/`

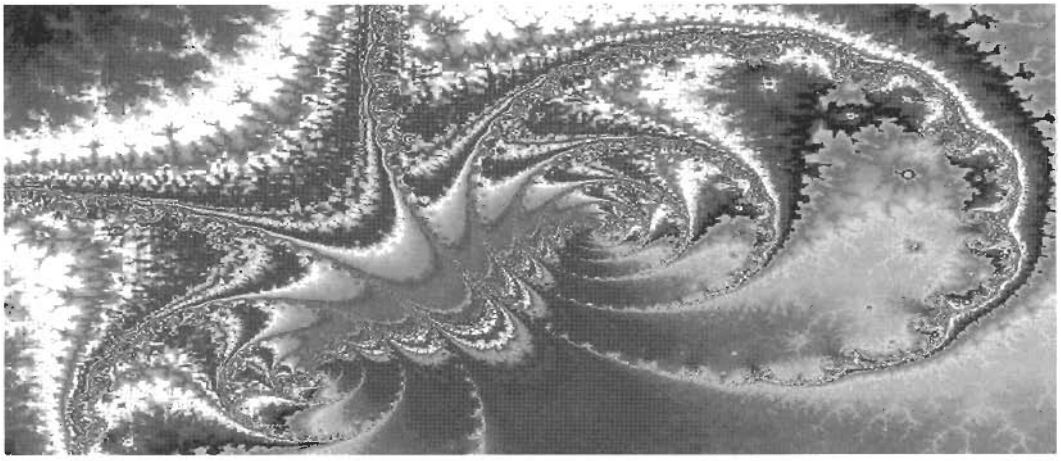
Si desea saber más sobre cómo funcionan las redes y los protocolos de redes, puede consultar un texto de introducción ya clásico *Computer Networks* de Andrew S. Tanenbaum.

A continuación

Con esto terminamos el análisis de los aspectos relacionados con el comercio electrónico y la seguridad. En la siguiente sección, examinaremos técnicas más avanzadas de PHP, como la interacción con otros equipos en Internet, la generación de imágenes de forma instantánea y el uso del control de sesión.

Parte IV

Técnicas avanzadas de PHP



16
Interacción con el sistema de archivos y el servidor

En un capítulo anterior, vimos cómo leer y escribir datos en archivos del servidor Web. En este capítulo, vamos a examinar otras funciones de PHP que nos permiten interactuar con el sistema de archivos en el servidor Web.

En este capítulo analizaremos los siguientes aspectos:

- Carga de archivos con PHP
- Uso de funciones de directorio
- Interacción con archivos en el servidor
- Ejecución de programas en el servidor
- Uso de las variables de entorno de servidor

Para analizar los usos de estas funciones, examinaremos un ejemplo.

Suponga que queremos permitir que nuestros clientes puedan actualizar parte de los contenidos de un sitio Web, por ejemplo, las noticias sobre nuestra compañía. (O que queremos incorporar una interfaz más sencilla que la de FTP para su uso.)

Una posibilidad podría consistir en permitir a los clientes que carguen archivos de contenido en forma de texto sin procesar. Estos archivos estarían disponibles en el sitio, a través de una plantilla diseñada con PHP, como hicimos en un capítulo anterior.

Antes de profundizar en las funciones del sistema de archivos, vamos a examinar brevemente el funcionamiento de la carga de archivos.

Introducción a la carga de archivos

PHP incorpora una funcionalidad de gran utilidad que permite realizar cargas HTTP. En lugar de recorrer el camino del servidor al navegador utilizando HTTP, los archivos realizan el trayecto contrario, es decir, van desde el navegador al servidor. Por regla general, esta función se implementa mediante una interfaz de formulario HTML.

En la figura 16.1 se ilustra la que utilizaremos en nuestro ejemplo.

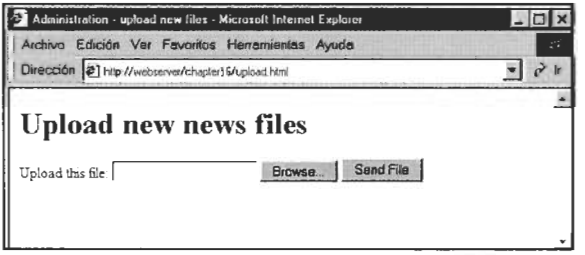


Figura 16.1. El formulario HTML utilizado para la carga de este archivo consta de campos y tipos de campos diferentes de los incluidos en un formulario HMTL normal

Como puede ver, el formulario incluye un cuadro en el que el usuario puede introducir un nombre de archivo o hacer clic sobre el botón **Browse** para examinar los archivos disponibles localmente.

Es posible que no haya visto un formulario de carga antes. En un instante veremos cómo implementarlo.

Tras introducir un nombre de archivo, el usuario puede hacer clic sobre **Send File** para cargar el archivo en el servidor, donde le espera una secuencia de comandos de PHP.

HTML para la carga de archivos

Para poder implementar la carga de archivos, necesitamos utilizar sintaxis de HTML disponible para dicho fin. En el listado 16.1 se recoge el código HMTL correspondiente a este formulario.

Listado 16.1. upload.html. Formulario HTML para la carga de archivos

```
<html>
<head>
 <title>Administration - upload new files</title>
</head>
<body>
<h1>Upload new news files</h1>
 <form enctype="multipart/form-data" action="upload.php"
method="post">
```

```
<input type="hidden" name="MAX_FILE_SIZE" value="1000000">
Upload this file: <input name="userfile" type="file">
<input type="submit" value="Send File">
</form>
</body>
</html>
```

Fíjese en que este formulario utiliza `POST`. Las cargas de archivo también se pueden realizar con el método `PUT` admitido por Netscape Composer y Amaya. No funcionan con `GET`.

Las funciones adicionales de este formulario son las siguientes:

- En la etiqueta `<form>`, debemos establecer el atributo `enctype="multipart/form-data"` para permitir que el servidor sepa que viene un archivo con la información habitual del formulario.
- Debemos incluir un campo de formulario que establezca el archivo de tamaño máximo que se puede cargar. Se trata de un campo oculto y se muestra a continuación:

```
<input type="hidden" name="MAX_FILE_SIZE" value="1000000">
```

  El nombre de este campo de formulario debe ser `MAX_FILE_SIZE`. El valor es el tamaño máximo (en bytes) de archivos que permitiremos que se carguen. Por el momento lo hemos establecido en 1000000 bytes (aproximadamente un megabyte). Puede aumentar o disminuir este tamaño en función de su aplicación.

- Necesitamos una entrada para el archivo de tipo, que en el ejemplo se corresponden con la línea

```
<input name="userfile" type="file">
```

  Puede seleccionar el nombre que desee para el archivo, pero téngalo en cuenta cuando vaya a utilizarlo para acceder a su archivo desde la secuencia de comandos PHP de recepción.

Un inciso sobre seguridad

Antes de seguir adelante, conviene recordar que algunas versiones de PHP presentan ciertos puntos débiles en el código para la carga de archivos. Si decide utilizar la función de carga de archivos en un servidor de producción, es aconsejable que utilice la versión más actual de PHP y que se mantenga atento a los posibles parches que puedan surgir.

Este hecho no debería disuadirle de utilizar una tecnología tan buena. Basta con poner atención al escribir el código y considerar la posibilidad de restringir el acceso a la función de carga de archivos a los administradores del sitio y a los encargados de administrar el contenido, por ejemplo.

Código PHP para procesar la tarea de carga del archivo

La tarea de escribir PHP para capturar el archivo resulta bastante sencilla.

Al cargar un archivo, se incluirá en una ubicación temporal en el servidor Web. Se trata del directorio temporal predeterminado del servidor Web. Si no mueve o cambia el nombre del archivo antes de que termine la ejecución de la secuencia de comandos, se eliminará.

Como nuestro formulario HTML consta de un campo llamado `userfile`, pasaremos cinco variables a PHP. Existen varias formas de acceder a estas variables. Puede utilizar la matriz superglobal `$_FILES`, disponible en PHP 4.1.0. Éste es el método recomendado. También puede acceder a las variables a través de la matriz `$HTTP_POST_FILES` o, si tiene activado el parámetro `register_globals`, puede acceder a ellas directamente.

Sin embargo, éste es el ámbito en el que resulta más importante mantener desactivado el parámetro `register_globals`, por lo que le recomendamos acceder a las variables de la siguiente forma:

- El valor almacenado en `$_FILE['userfile']['tmp_name']` o `$HTTP_POST_FILES['userfile']['tmp_name']` es el lugar en el que el archivo se ha almacenado temporalmente en el servidor Web.
- El valor almacenado en `$_FILE['userfile']['name']` o `$HTTP_POST_FILES['userfile']['name']` es el nombre del archivo en el sistema del usuario.
- El valor almacenado en `$_FILE['userfile']['size']` o `$HTTP_POST_FILES['userfile']['size']` es el tipo MIME del archivo, por ejemplo `text/plain` o `image/gif`.
- El valor almacenado en `$_FILE['userfile']['error']` o `$HTTP_POST_FILES['userfile']['error']` devolverá cualquier código de error asociado con la carga de archivos. Éste se añadió en PHP 4.2.0.

En el ejemplo de este capítulo vamos a utilizar `$HTTP_POST_FILES` para garantizar la compatibilidad con versiones anteriores, pero tenga en cuenta que si tiene previsto utilizar la función de carga de archivos en su servidor Web, debería utilizar la versión más actualizada de PHP.

Como sabemos dónde se encuentra el archivo y cómo se llama, podemos copiarlo en cualquier ubicación que nos resulte útil. Al final de la ejecución de la secuencia de comandos, se eliminará el archivo temporal. Por lo tanto, debe trasladarlo de ubicación o asignarle otro nombre si desea guardarlo.

En nuestro ejemplo, vamos a utilizar los archivos cargados como artículos de noticias recientes para eliminar todas las etiquetas que puede incluir y a trasladarlo a un directorio más útil. En el listado 16.2 se recoge una secuencia de comandos que realiza esta operación.

Listado 16.2. upload.php. PHP para capturar archivos desde el formulario HTML

```
<html>
<head>
  <title>Uploading...</title>
</head>
<body>
<h1>Uploading file...</h1>
<?php

  // $userfile es donde se colocó el archivo en el servidor Web
  $userfile = $HTTP_POST_FILES['userfile']['tmp_name'];

  // $userfile_name es el nombre original del archivo
  $userfile_name  =  $HTTP_POST_FILES['userfile']['name'];

  // $userfile_size es el tamaño en bytes
  $userfile_size = $HTTP_POST_FILES['userfile']['size'];

  // $userfile_type es el tipo mime, por ejemplo image/gif
  $userfile_type = $HTTP_POST_FILES['userfile']['type'];

  // $userfile_error es cualquier error encontrado
  $userfile_error = $HTTP_POST_FILES['userfile']['error'];


// userfile_error se introdujo en PHP 4.2.0
// use este código con versiones nuevas
  if ($userfile_error > 0)
  {
    echo 'Problem: ';
    switch ($userfile_error)
    {
      case 1: echo 'File exceeded upload_max_filesize'; break;
      case 2: echo 'File exceeded max_file_size'; break;
      case 3: echo 'File only partially uploaded'; break;
      case 4: echo 'No file uploaded'; break;
    }
    exit;
  }
// final del código para la versión 4.2.0

// para las versiones anteriores a la 4.2.0 use la comprobación manual de
//errores como se muestra a
// continuación
/*
  if ($userfile=='none')
  {
    echo 'Problem: no file uploaded';
    exit;
  }

  if ($userfile_size==0)
  {
    echo 'Problem: uploaded file is zero length';
    exit;
```

```
    }
*/
// final de la comprobación de errores para las versiones antiguas

// una comprobación más: ¿lleva asignado el archivo el tipo MIME correcto?

    if ($userfile_type != 'text/plain')
    {
       echo 'Problem: file is not plain text';
       exit;
    }

// coloque el archivo donde desee
    $upfile  =  '/uploads/'.$userfile_name;

// is_uploaded_file y move_uploaded_file se agregaron en la versión 4.0.3
    if (is_uploaded_file($userfile))
    {
        if (!move_uploaded_file($userfile, $upfile))
        {
             echo 'Problem: Could not move file to destination directory';
             exit;
        }
    }
    else
    {
       echo 'Problem: Possible file upload attack. Filename: '.$userfile_name;
       exit;
    }

// código para versiones antiguas según recomienda el manual de PHP
/*
    function is_uploaded_file($filename) {
       if (!$tmp_file = get_cfg_var('upload_tmp_dir')) {
            $tmp_file = dirname(tempnam('', ''));
       }
       $tmp_file .= '/' . basename($filename);
       // El usuario podría llevar una barra inclinada al final en php.ini...
       return (ereg_replace('/+', '/', $tmp_file) == $filename);
    }

    if (is_uploaded_file($userfile))
    {
       copy($userfile, $upfile);
    } else
    {
       echo 'Problem: Possible file upload attack. Filename: '.$userfile_name';
    }
*/
// final del código para la versión antigua

    echo 'File uploaded successfully<br /><br />';

// volver a aplicar formato a los contenidos del archivo
    $fp = fopen($upfile, 'r');
    $contents = fread ($fp, filesize ($upfile));
```

```
    fclose ($fp);

    $contents = strip_tags($contents);
    $fp = fopen($upfile, 'w');
    fwrite($fp, $contents);
    fclose($fp);

//  mostrar qué se ha cargado
    echo 'Preview of uploaded file contents:<br /><hr />';
    echo $contents;
    echo '<br /><hr />';

?>
</body>
</html>
```

Llama la atención que la mayor parte del código de esta secuencia de comandos esté dirigido a la comprobación de errores. La carga de archivos implica riesgos potenciales de seguridad que deben evitarse siempre que resulte posible. Necesitamos validar el archivo cargado con mucho cuidado para asegurarnos de que resulta seguro distribuirlo a nuestros visitantes.

Vamos a examinar las partes fundamentales de esta secuencia de comandos.

Comencemos por la comprobación del código de error devuelto por `$HTTP_POST_FILES['userfile']['error']`. Este código de error se introdujo en PHP 4.2.0. Desde PHP 4.3 también existe una constante asociada a cada uno de estos códigos. Las constantes y los valores posibles son los siguientes:

- `UPLOAD_ERROR_OK`, valor 0, significa que no se produjo ningún error.
- `UPLOAD_ERR_INI_SIZE`, valor 1, significa que el tamaño del archivo cargado supera el máximo valor especificado en el archivo php.ini con la directiva `upload_max_filesize`.
- `UPLOAD_ERR_FORM_SIZE`, valor 2, significa que el tamaño del archivo cargado supera el valor máximo especificado en el formulario HTML del elemento `MAX_FILE_SIZE`.
- `UPLOAD_ERR_PARTIAL`, valor 3, significa que el archivo se ha cargado parcialmente.
- `UPLOAD_ERR_NO_FILE`, valor 4, significa que no se ha cargado ningún archivo.

Si está utilizando una versión antigua de PHP, puede crear una versión manual de estas comprobaciones como se indica a continuación. Puede comprobar si `$userfile` es "none". Éste es el valor establecido por PHP si no se cargó ningún archivo. También probamos si el archivo tiene contenido (para ello, verificamos si `$userfile_size` es mayor que 0).

Por último e independientemente de la versión, decidimos que sólo queremos que se carguen archivos de texto por lo que probamos el tipo MIME verificando `$userfile_type`.

Seguidamente, comprobamos que el archivo que estamos intentando abrir se ha cargado y no se trata de un archivo local como `/etc/passwd`. Volveremos sobre este tema en un instante.

Si todo esto funciona, copiamos el archivo en nuestro directorio de inclusión. En este ejemplo, hemos utilizado `/uploads/`, que se encuentra fuera del árbol de documentos Web y resulta, por lo tanto, un buen lugar en que almacenar los archivos que deban dirigirse a otras ubicaciones.

A continuación, abrimos el archivo, limpiamos las etiquetas de HTML o PHP que puedan haber quedado con la función `strip_tags()` y volvemos a guardarlo.

Por último, mostramos los contenidos del archivo para que el usuario pueda ver que se ha cargado satisfactoriamente.

En la figura 16.2 se muestran los resultados de la ejecución (satisfactoria) de esta secuencia de comandos.

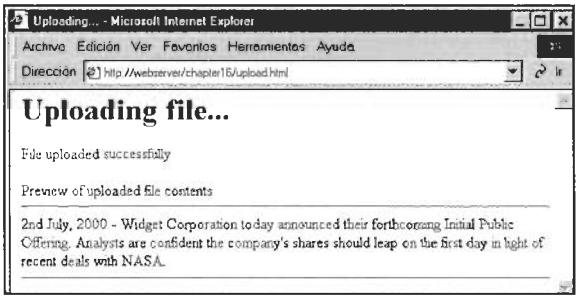


Figura 16.2. Tras copiar el archivo y aplicarle formato, se muestra el archivo cargado para indicar al usuario que la operación de carga se ha realizado correctamente

En septiembre de 2000, se anunció que un pirata informático podía hacer que la secuencia de comandos para la carga de archivos procesara un archivo local como si se hubiera cargado en el servidor.

Este hecho se documentó en la lista de correo BUGTRAQ. Si lo desea, puede consultar los documentos sobre seguridad oficiales en uno de los muchos compendios BUGTRAQ, como `http://lists.insecure.org/bugtraq/2000/Sep/0237.html`.

En nuestro caso hemos utilizado las funciones `is_uploaded_file()` y `move_uploaded_file()` para asegurarnos de que el archivo que estamos procesando se ha cargado y que no se trata de un archivo local como `/etc/passwd`. Esta función está disponible a partir de la versión 4.0.3 de PHP.

Hemos incluido código de ejemplo por si estuviera utilizando una versión anterior con funcionalidad equivalente (indicada en el código por medio de comentarios).

Si no se escribe con cuidado la secuencia de comandos para procesar la carga de archivos, un visitante malintencionado podría utilizar un nombre de archivo temporal y convencer a la secuencia de comandos para que procese dicho archivo como si se tratara de un archivo cargado.

Como muchas secuencias de comandos para la carga de archivos devuelven los datos cargados al usuario o los almacenan en algún lugar desde los que puedan cargarse, se podría lograr acceder a cualquier archivo que el servidor Web pueda leer, como los archivos del directorio `/etc/passwd` y código fuente de PHP, incluidas las contraseñas de la base de datos.

Problemas habituales

Al realizar cargas de archivo en un servidor es necesario tener en cuenta una serie de aspectos.

- En el ejemplo anterior se asume que los usuarios han sido autenticados en algún momento. No es aconsejable permitir que todo el mundo pueda cargar archivos en un sitio.
- Si va a permitir que los usuarios no autenticados o no fiables carguen archivos en el servidor, conviene examinar exhaustivamente su contenido para evitar que se cargue o ejecute una secuencia de comandos malintencionada. Debería estar atento, no sólo al tipo y a los contenidos del archivo, sino también al nombre del archivo en sí mismo. Es aconsejable cambiar el nombre a los archivos cargados utilizando instancias que resulten seguras.
- Si está utilizando un equipo basado en Windows, asegúrese de que utilizar `\\` o `/` en lugar de `\` en las rutas de archivo.
- Si el proceso de carga no funciona, compruebe el archivo `php.ini`. Verifique que la directiva `upload_tmp_dir` apunta a algún directorio al que disponga de acceso. Puede que también necesite ajustar la directiva `memory_limit` para cargar archivos de mayor tamaño. Esta directiva determina el tamaño máximo de los archivos que puede cargar.
- Si se está ejecutando PHP en modo seguro, obtendrá un mensaje de error indicando que no se puede acceder al archivo temporal. Para solucionar problema, sólo se puede desactivar el modo seguro o escribir una secuencia de comandos que no sea de PHP para que copie el archivo en una ubicación accesible. Tras ello, puede ejecutar esta secuencia de comandos desde la secuencia de comandos de PHP. Al final del capítulo, analizaremos cómo ejecutar programas en el servidor desde PHP.

Uso de las funciones de directorio

Una vez cargados los archivos, resultará útil permitir ver lo que se ha cargado y manipular los archivos de contenido.

PHP consta de una serie de funciones de sistema de archivos y directorios que resultan de utilidad al respecto.

Lectura desde directorios

En primer lugar, implementaremos una secuencia de comandos que permita examinar el contenido cargado. La operación de examinar directorios resulta muy sencilla en PHP. En el listado 16.3, se incluye una sencilla secuencia de comandos que se puede utilizar con este fin.

Listado 16.3. browserdir.php. Un listado de directorios de los archivos cargados

```
<html>
<head>
   <title>Browse Directories</title>
</head>
<body>
<h1>Browsing</h1>
<?php
   $current_dir = '/uploads/';
   $dir  =  opendir($current_dir);

   echo "Upload directory is $current_dir<br />";
   echo 'Directory Listing:<br /><hr /><br />';
   while ($file = readdir($dir))
   {
        echo "$file<br />";
   }
   echo '<hr /><br />';
   closedir($dir);
?>
</body>
</html>
```

Esta secuencia de comandos utiliza las funciones `opendir()`, `closedir()` y `readdir()`.

La función `opendir()` se utiliza para abrir un directorio para su lectura. Su uso es muy similar al de `fopen()` para leer desde archivos. En lugar de pasarle un nombre de archivo, debería pasarle un nombre de directorio:

```
$dir  =  opendir($current_dir);
```

La función devuelve un indicador de directorio, de la misma forma que `fopen()` devuelve un indicador de archivo.

Cuando el directorio está abierto, puede leer un nombre de archivo desde él llamando a `readdir($dir)`, como se muestra en el ejemplo. Este elemento devuelve `false` cuando no hay archivos que leer. (Tenga en cuenta que también devolverá `false` si lee un archivo llamado `"0"`; conviene verificarlos si es probable que ocurra.) Los archivos no se ordenan en ningún orden concreto, por lo que si necesita una lista ordenada, debería leerlos en una matriz y ordenarlos después.

Cuando haya terminado de leer desde un directorio, deberá llamar a `closedir($dir)` para terminar. De nuevo, la operación es similar a `fclose()` para un archivo.

El resultado de ejemplo de la secuencia de comandos de exploración de directorios se muestra en la figura 16.3.

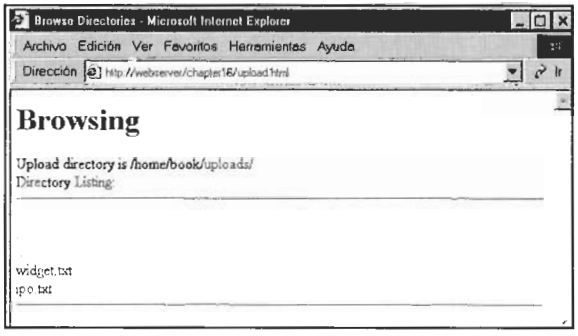


Figura 16.3. El listado de directorio muestra todos los archivos del directorio seleccionado, incluidos el directorio . (directorio actual) y el directorio .. (el directorio de nivel superior). Puede filtrarlos si lo desea

Si está utilizando este mecanismo para permitir que el usuario examine los directorios, es aconsejable limitar los directorios que se pueden explorar para evitar el acceso a áreas normalmente restringidas. Una función asociada que resulta de utilidad en ocasiones es la función `rewinddir($dir)`. Esta función restablece la lectura de los nombres de archivo al principio del directorio.

Como alternativa a estas funciones, puede utilizar la clase `dir` de PHP. Ésta consta de las propiedades `handle` y `path`, y de los métodos `read()`, `close()` y `rewind()`, que funcionan igual que las alternativas no incluidas en clases.

Cómo obtener información sobre el directorio actual

Podemos obtener información adicional utilizando una ruta a un archivo.

Las funciones `dirname($path)` y `basename($path)` devuelven la parte de una ruta y la parte de un archivo de nombre de la ruta, respectivamente. Esto podría resultar útil para nuestro explorador de directorios, en especial si comenzamos a construir una estructura compleja de directorios de contenido basada en nombres de directorios y nombres de archivos descriptivos.

También podemos agregar al listado de directorios una indicación del espacio libre para cargas de archivos con la función `disk_free_space($path)`. Si pasamos una ruta a un directorio, devolverá el número de bytes libres en el disco (Windows) o el sistema de archivos (UNIX) en el que se encuentra el directorio.

Creación y eliminación de directorios

Además de leer de manera pasiva la información sobre directorios, podemos utilizar las funciones `mkdir()` y `rmdir()` para crear y eliminar directorios. Sólo

podrá crear y eliminar directorios en rutas a las que disponga de acceso el usuario utilizado para ejecutar la secuencia de comandos.

El uso de la función `mkdir()` resulta más complicado de lo que podría parecer a primera vista. Esta función toma dos parámetros, la ruta al directorio deseado (incluyendo el nuevo nombre de directorio) y el permiso que le gustaría que tuviera dicho directorio, por ejemplo:

```
mkdir("/tmp/testing",  0777);
```

Sin embargo, los permisos que se enumeren no tienen por qué ser los permisos que vaya a obtener. La función `umask` actual será `ANDed` (como resta) con este valor para obtener los permisos actuales. Por ejemplo, si la función `umask` es 022, obtendremos el permiso 0755.

Si lo desea puede restablecer la `umask` antes de crear un directorio para sustituir este efecto mediante el siguiente código:

```
$oldumask  =  umask(0);
mkdir("/tmp/testing", 0777);
umask($oldumask);
```

Este código utiliza la función `umask()`, que se puede utilizar para comprobar y cambiar la `umask` actual. Se sustituirá la `umask` por el valor pasado y se devolverá la `umask` antigua o, si se llama sin parámetros, se devolverá únicamente la `umask` actual. Tenga en cuenta que la función `umask()` no tiene ningún efecto en los sistemas Windows. La función `rmdir()` elimina un directorio. Por ejemplo:

```
rmdir("/tmp/testing");
```

o

```
rmdir("c:\\tmp\\testing");
```

El directorio que se desee eliminar debe estar vacío.

Interacción con el sistema de archivos

Además de ver y obtener información sobre directorios, podemos interactuar y obtener información sobre archivos incluidos en el servidor Web. Ya hemos visto cómo escribir y leer archivos, pero existen muchas otras funciones de archivo disponibles.

Cómo obtener información de archivo

Podemos alterar la parte de la secuencia de comandos para explorar directorios que leen archivos de la siguiente forma:

```
while  ($file  =  $dir->read())
{
```

```
    echo '<a href="filedetails.php?file='.$file.'">'.$file.'</a><br />';
}
```

Podemos crear la secuencia de comandos filedetails.php para suministrar información adicional sobre un archivo. En el listado 16.4 se muestran los contenidos de este archivo.

Tenga en cuenta que parte de las funciones utilizadas aquí no son compatibles con Windows, entre las que se incluyen `posix_getpwuid()`, `fileowner()` y `filegroup()`, o su compatibilidad no es fiable.

Listado 16.4. filedetails.php. Las funciones de estado de archivo y sus resultados

```
<html>
<head>
  <title>File Details</title>
</head>
<body>
<?php
  $current_dir = '/uploads/';
  $file = basename($file); // filtrar la información de directorio
  //por seguridad

  echo '<h1>Details of file: '.$file.'</h1>';
  $file = $current_dir.$file;

  echo '<h2>File data</h2>';
  echo 'File last accessed: '.date('j F Y H:i', fileatime($file)).'<br />';
  echo 'File last modified: '.date('j F Y H:i', filemtime($file)).'<br />';

  $user = posix_getpwuid(fileowner($file));
  echo 'File owner: '.$user['name'].'<br />';

  $group = posix_getgrgid(filegroup($file));
  echo 'File group: '.$group['name'].'<br />';

  echo 'File permissions: '.decoct(fileperms($file)).'<br />';

  echo 'File type: '.filetype($file).'<br />';

  echo 'File size: '.filesize($file).' bytes<br />';


  echo '<h2>File tests</h2>';

  echo 'is_dir: '.(is_dir($file)? 'true' : 'false').'<br />';
  echo 'is_executable: '.(is_executable($file)? 'true' : 'false').'<br />';
  echo 'is_file: '.(is_file($file)? 'true' : 'false').'<br />';
  echo 'is_link: '.(is_link($file)? 'true' : 'false').'<br />';
  echo 'is_readable: '.(is_readable($file)? 'true' : 'false').'<br />';
  echo 'is_writable: '.(is_writable($file)? 'true' : 'false').'<br />';

?>
</body>
</html>
```

En la figura 16.4 se muestran los resultados de ejecutar el listado 16.4.



Figura 16.4. La vista de los detalles de archivo muestra información del sistema de archivos sobre un archivo. Como puede observar los permisos se muestra en formato octal

Vamos a analizar qué hace cada una de las funciones utilizada en el listado 16.4.

Como se mencionó anteriormente, la función `basename()` obtiene el nombre del archivo sin el directorio. (También puede utilizar la función `dirname()` para obtener el nombre del directorio sin el nombre de archivo).

Las funciones `fileatime()` y `filemtime()` devuelven la marca de tiempo indicando el momento en el que se accedió por última vez al archivo y la última vez que se modificó, respectivamente. Hemos modificado el formato de la marca de tiempo con la función `date()` para que resulte más legible. Estas funciones devolverán el mismo valor en algunos sistemas operativos (como en el ejemplo) según qué información almacene el sistema.

Las funciones `fileowner()` y `filegroup()` devuelven el Id. del usuario (`uid`) y el Id. de grupo (`gid`) del archivo. Éstos se pueden convertir en nombres utilizando las funciones `posix_getpwuid()` y `posix_getgrgid()` para que resulten más sencillos de leer. Estas funciones toman el `uid` o `gid` como parámetro y devuelven una matriz asociativa de información sobre el usuario o grupo, incluido sus nombres, como hemos hecho en esta secuencia de comandos.

La función `fileperms()` devuelve los permisos sobre el archivo. Les hemos aplicado un nuevo formato utilizando la función `decoct()` para que resulte más familiar a los usuarios de UNIX.

La función `filetype()` devuelve información sobre el tipo de archivo que se va a examinar. Los resultados posibles son `fifo`, `char`, `dir`, `block`, `link`, `file` y `unknown`.

La función `filesize()` devuelve el tamaño del archivo en bytes.

El segundo conjunto de funciones (`is_dir()`, `is_executable()`, `is_file()`, `is_link()`, `is_readable()` y `is_writable()`) prueba el atributo con nombre de un archivo y devuelve `true` o `false`.

También podríamos utilizar la función `stat()` para recopilar una gran cantidad de información del mismo tipo. Al pasar un archivo, devuelve una matriz que contiene datos similares a los de estas funciones. La función `lstat()` es similar pero para su uso con vínculos simbólicos.

Las funciones de estado de archivo tardan mucho tiempo en procesarse. Por esta razón, sus resultados se almacenan en caché. Si desea comprobar información sobre el archivo antes o después de un cambio, deberá llamar a

```
clearstatcache();
```

para poder vaciar los resultados anteriores. Si desea utilizar la secuencia de comandos previa antes o después de cambiar datos del archivo, debería llamar a esta función como primer paso para asegurarse de que los datos generados están actualizados.

Cómo cambiar las propiedades de archivo

Además de ver las propiedades de archivo, podemos alterarlas.

Las funciones `chgrp(archivo, grupo)`, `chmod(archivo, permiso)` y `chown(archivo, usuario)` se comportan de forma similar a sus equivalentes de UNIX. Ninguna de ellas funcionará en los sistemas basados en Windows, aunque `chown()` se ejecutará y devolverá siempre `true`.

La función `chgrp()` se utiliza para cambiar el grupo de un archivo. Sólo se puede utilizar para cambiar el grupo al que pertenece el usuario, a menos que se trate del usuario raíz.

La función `chmod()` se utiliza para cambiar los permisos sobre un archivo. Los permisos se pasan con la forma `chmod` habitual de UNIX (debe anteponerles un "`0`" para indicar que su formato es octal, por ejemplo).

```
chmod('somefile.txt', 0777);
```

La función `chown()` se utiliza para cambiar el titular de un archivo. Sólo se puede utilizar si la secuencia de comandos se está ejecutando como raíz, lo que no debería ocurrir nunca.

Creación, eliminación y desplazamiento de archivos

Puede utilizar las funciones de sistema de archivos para crear, trasladar y eliminar archivos.

En primer lugar, y de forma sencilla, puede crear un archivo o cambiar la hora de la última modificación utilizando la función `touch()`. Su funcionamiento es similar al comando `touch` de UNIX. Su sintaxis es la siguiente:

```
int touch (string archivo, [int hora [, int hora]])
```

Si el archivo ya existiese, su hora de modificación se sustituiría por la actual o por la indicada en el segundo parámetro si se especifica.

Si desea utilizar esta última opción, debería incluirse en formato de marca de tiempo.

Si el archivo no existiese se crearía. La hora de acceso del archivo también se modificaría: de manera predeterminada la hora del sistema actual o la marca de tiempo especificada en parámetro opcional.

Puede eliminar archivos utilizando la función `unlink()`. (Tenga en cuenta que esta función no se llama `delete`: no existe dicha función.) Se utiliza de la siguiente forma:

```
unlink($filename);
```

Esta función no resulta operativa con las versiones antiguas de Windows. Si éste fuera su caso, puede utilizar la siguiente función para eliminar un archivo en Windows:

```
system("del   filename.ext");
```

Puede copiar y mover archivos con las funciones `copy()` y `rename()` de la siguiente forma:

```
copy($source_path, $destination_path);
rename($oldfile,   $newfile);
```

Como puede haber observado, hemos utilizado la función `copy()` en el listado 16.2.

La función `rename()` se encarga también de la tarea de trasladar archivos de un lugar a otro porque PHP no consta de la función `move`.

La posibilidad de mover archivos entre sistemas de archivos y la de sobrescribir archivos con `rename()` depende del sistema operativo, por lo que debe comprobar los efectos en su servidor.

Así mismo, tenga cuidado con la ruta que utilizará hasta el nombre de archivo. Si fuera relativa, lo sería con respecto a la ubicación de la secuencia de comandos, no al archivo original.

Uso de funciones de ejecución de programas

A continuación, dejaremos a un lado las funciones relacionadas con el sistema de archivos y pasaremos a examinar las funciones disponibles para ejecutar comandos en el servidor.

Estas funciones resultan útiles cuando se quiere suministrar una interfaz de usuario basada en la Web a un sistema existente basado en línea de comandos. Por ejemplo, hemos utilizado estos comandos para crear una interfaz de comandos para el administrador de listas de correos `ezmlm`. Volveremos a utilizarlas en capítulos posteriores.

Existen cuatro técnicas principales para ejecutar un comando en el servidor Web. Aunque son bastante parecidas, presentan pequeñas diferencias.

1. `exe()`

La función `exec()` tiene la siguiente sintaxis:

```
string exec (string comando [, array resultado [, int valor_devuelto]])
```

Se le pasa el comando que le gustaría que se ejecutase, por ejemplo,

```
exec("ls -la");
```

La función `exec()` no genera un resultado directo.

Devuelve la última línea del resultado del comando.

Si se pasa en una variable como resultado, se obtendrá un matriz de cadenas que representa cada línea del resultado. Si se pasa en una variable como `valor_devuelto`, se obtendrá el código de devolución.

2. `passthru()`

La función `passthru()` tiene la siguiente sintaxis:

```
void passthru (string comando [, int valor_devuelto])
```

La función `passthru()` repite el resultado en el navegador, lo cual resulta útil si el resultado es binario (por ejemplo, datos de una imagen).

No devuelve nada.

Los parámetros funcionan de la misma forma que los de la función `exec()`.

3. `system()`

La función `system()` tiene la siguiente sintaxis:

```
string system (string comando [, int valor_devuelto])
```

Esta función repite el resultado del comando en el navegador. Se diferencia de la función anterior en que intenta eliminar el resultado tras cada línea (si se ejecuta PHP como módulo de servidor).

Devuelve la última línea de resultado (si tiene éxito) o `false` (en caso de fallo).

Sus parámetros funcionan de la misma forma que los parámetros de las funciones anteriores.

4. Apóstrofes invertidos

Se trata de operadores de ejecución como ya se indicó en un capítulo anterior.

No pueden tener una salida directa. El resultado de ejecutar el comando se devuelve en una cadena, que se puede dirigir a un dispositivo o utilizar para realizar cualquier otra operación deseada.

Si sus necesidades resultan más complicadas, puede utilizar `popen()`, `proc_open()` y `proc_close()`. Estas funciones se utilizan para bifurcar procesos externos y enviar datos entre ellos. Las últimas dos funciones se agregaron en la versión 4.3 de PHP.

La secuencia de comandos que se recoge en el listado 16.5 ilustra cómo utilizar las cuatro técnicas de forma equivalente.

Listado 16.5. Funciones de estado de archivo y sus resultados

```
<?php

   chdir('/uploads/');

///// versión exec
   echo '<pre>';

   // unix
   exec('ls -la', $result);
   // windows
   // exec('dir', $result);
   foreach ($result as $line)
      echo "$line\n";

   echo '</pre>';
   echo '<br /><hr /><br />';

///// versión passthru
   echo '<pre>';

   // unix
   passthru('ls -la');
   // windows
   // passthru('dir');

   echo '</pre>';
   echo '<br /><hr /><br />';

///// versión system

   echo '<pre>';
   // unix
   $result = system('ls -la');
   // windows
   // $result = system('dir');
   echo '</pre>';
   echo '<br /><hr /><br />';

/////versión con apóstrofes invertidos

   echo '<pre>';
   // unix
   $result = 'ls -al';
   // windows
   // $result = 'dir';
```

```
    echo  $result;
    echo '</pre>';

?>
```

Podríamos haber utilizado estos enfoques como alternativa a la secuencia de comandos de exploración de directorios escrita anteriormente. Uno de los efectos secundarios del uso de funciones externas queda ampliamente demostrado aquí: su código dejará de ser portable. En este caso hemos utilizado comandos de Unix y el código no se ejecutará en Windows.

Si tiene previsto incluir datos enviados por el usuario como parte del comando que va a ejecutar, debería ejecutarlos siempre a través de la función `escapeshellcmd()` en primer lugar. De esta forma evitará que los usuarios puedan ejecutar comandos malintencionados en su equipo. Puede llamar a esta función de la siguiente forma:

```
system(escapeshellcmd($command));
```

También debería utilizar esta función para convertir cualquier argumento especial que tenga previsto pasar a su comando de núcleo.

Interacción con el entorno: getenv() y putenv()

Antes de dar por concluida esta sección, vamos a ver cómo podemos utilizar variables de entorno desde PHP. Para ello, disponemos de dos funciones: `getenv()`, que permite recuperar variables de entorno y `putenv()` que permite establecer variables de entorno.

Tenga en cuenta que estamos hablando del entorno en el que PHP se ejecuta en el servidor.

Puede obtener una lista de todas las variables de entorno de PHP ejecutando `phpinfo()`.

Algunas resultan más útiles que otras, por ejemplo

```
getenv("HTTP_REFERER");
```

devolverá el URL de la página desde la que el usuario llegó hasta la página actual.

También puede establecer las variables de entorno como desee con `putenv()`, por ejemplo,

```
$home = "/home/nobody";
putenv  ("  HOME=$home  ");
```

Si fuera un administrador de sistemas y quisiera limitar las variables de entorno que pueden establecer los programadores, podría utilizar la directiva `safe_mode_allowed_env_vars` de php.ini. Cuando PHP se ejecuta en modo seguro, los usuarios sólo pueden establecer las variables de entorno cuyos prefijos se enumeran en esta directiva.

Si desea obtener información adicional sobre qué representan las variables de entorno, puede consultar la especificación de CGI.

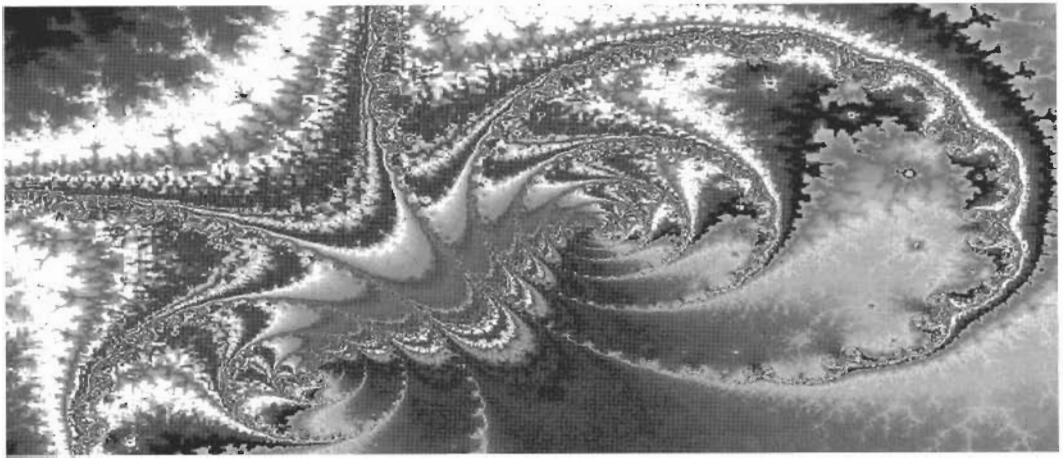
```
http://hoohoo.ncsa.uiuc.edu/cgi/env.html
```

Lecturas adicionales

La mayor parte de las funciones de sistemas de archivo de PHP están asociadas con las funciones subyacentes de los sistemas operativos. Si está utilizando UNIX y desea obtener más información, consulte las páginas man.

A continuación

En el siguiente capítulo, veremos cómo utilizar las funciones de red y protocolo de PHP para interactuar con sistemas distintos a los de nuestro servidor Web, lo cual contribuirá a ampliar el horizonte de posibilidades para nuestras secuencias de comandos.



17
Uso de funciones de red y de protocolo

En este capítulo, examinaremos las funciones orientadas a la red de PHP que permiten a nuestras secuencias de comandos interactuar con el resto de Internet. En Internet, encontrará una ingente cantidad de recursos y una gran variedad de protocolos para su uso. En esta sección examinaremos los siguientes aspectos:

- Una descripción general de los protocolos disponibles
- Envío y lectura de correo electrónico
- Uso de otros sitios Web a través de HTTP
- Uso de funciones de búsqueda de red
- Uso de FTP
- Uso de comunicaciones de red genéricas con cURL

Descripción general de protocolos

Los protocolos son reglas de comunicación para una situación dada. Por ejemplo, ya conocemos el protocolo que hay que seguir cuando nos encontramos a alguien. Decimos hola, estrechamos la mano y decimos adiós. Los protocolos de red son similares. Como en el caso de los protocolos humanos, se utilizan diferentes protocolos informáticos para cada situación y aplicaciones. Por ejemplo, utilizamos HTTP (Pro-

tocolo de transferencia de hipertexto) para enviar y recibir páginas Web. Es probable que también haya utilizado FTP (Protocolo de transferencia de archivos) para transferir archivos entre equipos de una red.

Existen muchos otros.

Los protocolos, así como otros estándares de Internet, se describen en las RFC (Solicitudes de comentarios). Estos protocolos son definidos por el IETF (Grupo de trabajo de ingeniería de Internet).

En Internet existen muchos lugares en los que encontrar RFC. La fuente básica es el editor de RCF situado en

```
http://www.rfc-editor.org/.
```

Si tiene problemas al trabajar con un protocolo dado, las RFC son la fuente oficial y suelen resultar útiles para solucionar los problemas con el código. Sin embargo, la cantidad de detalles es muy grande y suelen incluir cientos de páginas.

Entre las RFC más conocidas se puede citar la RFC2616, que describe el protocolo HTTP/1.1 y la RFC822, que describe el formato de los mensajes de correo electrónico por Internet.

En este capítulo, vamos a examinar los aspectos de PHP que utilizan algunos protocolos. Específicamente, veremos cómo enviar correo con SMTP, cómo leer correo con POP e IMAP, cómo conectar con otros servidores Web a través de HTTP y HTTPS, y cómo transferir archivos con FTP.

Envío y recepción de correos electrónicos

La forma principal de enviar correos electrónico en PHP consiste en utilizar la función `mail()`, ya comentada en un capítulo anterior. Esta función utiliza el protocolo SMTP (Protocolo simple para la transferencia de correo) para enviar correo electrónico.

Es posible utilizar toda una variedad de clases gratuitas para añadir funcionalidad a `mail()`. En un capítulo posterior, utilizaremos una clase complementaria para enviar archivos adjuntos HTML en un mensaje de correo. SMTP sólo se utiliza para enviar correos. Los protocolos IMAP (Protocolo de acceso a mensajes de Internet, descrito en la RFC2060) y POP (Protocolo de oficina de correos, descrito en la RFC1939 o STD0053) se utilizan para leer el correo de un servidor de correos. Estos protocolos no pueden enviar correo.

IMAP se utiliza para leer y manipular mensajes de correo almacenados en un servidor y es más avanzado que POP, el cual se suele utilizar para descargar mensajes de correo electrónico a un cliente y eliminarlos del servidor.

PHP incorpora una biblioteca IMAP. Esta biblioteca también se puede utilizar para realizar conexiones POP, NNTP (Protocolo de transferencia de noticias de red) e IMAP.

Examinaremos con detenimiento el uso de la biblioteca IMAP en el proyecto incluido en un capítulo posterior.

Uso de otro sitio Web

La Web permite utilizar, modificar e incrustar servicios e información existente en nuestras propias páginas. PHP convierte esta tarea en algo muy sencillo, como se ilustra en el ejemplo que se comenta a continuación.

Imagine que la compañía para la que trabaja quiere mostrar en la página principal el precio de sus acciones. Esta información está disponible en varios sitios dedicados a temas bursátiles, pero ¿cómo acceder a ella?

Tenemos que empezar por buscar un URL en el que se incluya esta información de primera mano. Tras ello, cuando un usuario se dirija a nuestra página principal, abriremos una conexión a dicho URL para recuperar la página y extraer la información deseada.

Como ejemplo, hemos creado una secuencia de comandos que recupera y aplica formato a la cotización extraída del sitio Web de AMEX. En concreto, recuperaremos el precio de cotización de las acciones de Amazon. (Puede incluir otra información ya que el principio es el mismo.) En el listado 17.1 se recoge la secuencia de comandos.

Listado 17.1. lookup.php. Secuencia de comandos que recupera la cotización de una acción del NASDAQ con el símbolo listado en $symbol

```
<html>
<head>
  <title>Stock Quote from NASDAQ</title>
</head>
<body>
<?php
  // seleccione qué acción examinar
  $symbol='AMZN';
  echo "<h1>Stock Quote for $symbol</h1>";

  $theurl='http://www.amex.com/equities/listCmp/EqLCDetQuote.jsp?
Product_Symbol=AMZN';


  if (!($fp = fopen($theurl, 'r')))
  {
     echo 'Could not open URL';
     exit;
  }
  $contents = fread($fp, 1000000);
  fclose($fp);

  //represente $contents;

  // busque la parte de la página deseada y devuélvala como
  //resultado
  $pattern = "(\\$[0-9 ]+\\.[0-9]+)";
  if (eregi($pattern, $contents, $quote))
  {
```

```
        echo "$symbol was last sold at: ";
        echo $quote[1];
    } else
    {
        echo 'No quote available';
    };

    // reconozca la fuente
    echo '<br />'
        .'This information retrieved from <br />'
        ."<a href=\"$theurl\">$theurl</a><br />"
        .'on '.(date('l jS F Y g:i a T'));
?>
</body>
</html>
```

En la figura 17.1 se recoge el resultado del listado 17.1.

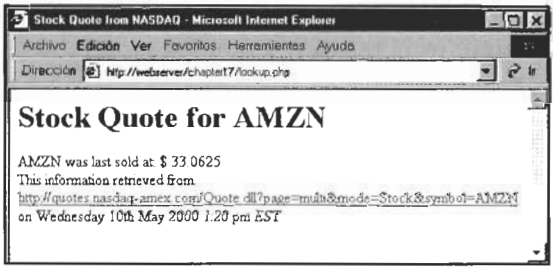


Figura 17.1. Esta secuencia de comandos utiliza una expresión regular para extraer la cotización de una acción de la información recuperada de la bolsa de valores

La secuencia de comandos resulta bastante clara. De hecho, no utiliza ninguna función nueva, sólo nuevas aplicaciones de dichas funciones.

Puede que recuerde de un capítulo anterior que podemos utilizar las funciones de archivo para leer desde un URL. Eso es lo que hemos hecho en este caso. La llamada a `fopen()`

```
$fp = fopen($theurl, 'r')
```

devuelve un puntero al inicio de página del URL suministrado.

A partir de ahí se trata simplemente de leer el URL desde la página y cerrarlo de nuevo.

```
$contents = fread($fp, 1000000);
fclose($fp);
```

Como observará, el número utilizado para indicarle a PHP cuánto leer del archivo es muy alto. Si se tratara de un archivo en el servidor, lo normal sería utilizar `filesize($file)` pero esta función no se puede utilizar con un URL.

Tras ello, tenemos todo el texto de la página Web de dicho URL almacenado en la variable `$contents`.

Seguidamente, podemos utilizarlo como expresión regular y recurrir a la función `eregi()` para buscar la parte de la página deseada:

```
$pattern = "(\\$[0-9 ]+\\.[0-9]+)";
if  (eregi($pattern,  $contents,  $quote))
{
   echo "$symbol was last sold at: ";
   echo $quote[1];
}
```

Y eso es todo.

Podemos utilizar esta técnica parar realizar otras tareas. Por ejemplo, podemos recuperar la información meteorológica local e incluirla en nuestra página.

El mejor uso de esta técnica consiste en combinar información procedente de varias fuentes para agregarle algún valor.

Un buen ejemplo es la secuencia de comandos creada por Philip Greenspun que genera la página Bill Gates Wealth Clock:

```
http://www.webho.com/WealthClock
```

Esta página toma la información de dos fuentes. Obtiene la población actual de los EEUU de la oficina del censo estadounidense.

A continuación, examina el valor de una acción de Microsoft y combina los dos datos, agrega una dosis saludable de la opinión del autor y genera nueva información: el valor aproximado de Bill Gates.

Si va a utilizar una fuente de información externa como ésta con fines comerciales, conviene comprobar antes si puede hacerlo ya que en algunos casos es necesario tener en cuenta aspectos relativos a la propiedad intelectual.

Si está creando una secuencia de comandos como ésta, es probable que desee pasarle algunos datos.

Por ejemplo, si está conectándose a un URL externo, es probable que desee pasar algunos parámetros escritos por el usuario. Si éste fuera el caso, conviene utilizar la función `urlencode()`.

Esta función toma una cadena y la convierte al formato correcto para un URL. Por ejemplo, transforma los espacios en signos más. Puede llamarla de la siguiente manera:

```
$encodedparameter  =  urlencode($parameter);
```

Este enfoque presenta un problema: que el sitio del que se obtiene la información varíe el formato de los datos, lo que impediría el funcionamiento de nuestra secuencia de comandos.

Una alternativa a este sistema son los servicios Web.

Son como objetos remotos a los que podemos conectarnos para recuperar datos como cotizaciones bursátiles.

La compatibilidad no oficial de PHP para servicios Web aumenta día a día y ya hay una extensión SOAP en fase de construcción. En un capítulo posterior ampliaremos este tema.

Uso de funciones de búsqueda de red

PHP incorpora un conjunto de funciones de "búsqueda" que se pueden utilizar para comprobar información sobre nombres de host, direcciones IP e intercambios de correo electrónico. Por ejemplo, si está creando un directorio como Yahoo!, cuando se remitan nuevas direcciones podría comprobar automáticamente que el host de un URL y la información de contacto de dicho sitio es válida.

De esta forma puede contribuir a reducir la labor posterior de los revisores al examinar el sitio si descubren que no existe o que la dirección de correo electrónico no es válida.

El listado 17.2 muestra el código HTML correspondiente a un formulario de envío para un directorio como éste.

Listado 17.2. directory_submit.html. Código HTML para el formulario de envío

```
<head>
  <title>Submit your site</title>
</head>
<body>
<h1>Submit site</h1>
<form method="post" action="directory_submit.php">
URL: <input type="text" name="url" size="30" value="http://"><br />
Email contact: <input type="text" name="email" size="23"><br />
<input type="submit" name="Submit site">
</form>
</body>
</html>
```

Se trata de un formulario bastante sencillo. En la figura 17.2 se recoge la versión resultante, con datos de ejemplo introducidos.

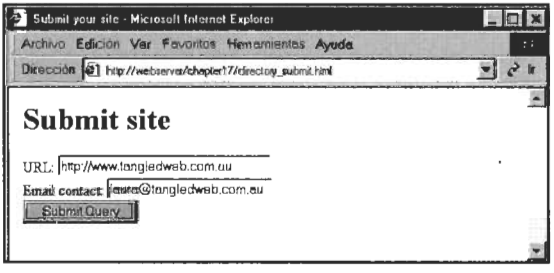


Figura 17.2. El envío de direcciones a un directorio suele requerir la inclusión de un URL y detalles de contacto para que los administradores del directorio puedan avisarnos del momento en el que se agrega nuestra dirección al directorio

Al pulsar el botón de envío, en primer lugar tenemos que comprobar que el URL está alojado en un equipo real y, después, que la parte del host de la dirección de correo electrónico también está en un equipo real.

Hemos escrito una secuencia de comandos para comprobar estos elementos y en la figura 17.3 se recoge el resultado.

La secuencia de comandos encargada de realizar estas comprobaciones utiliza dos funciones del conjunto de funciones de red de PHP: `gethostbyname()` y `getmxrr()`. En el listado 17.3 se incluye el código de la secuencia completa.



Figura 17.3. Esta versión de la secuencia de comandos muestra los resultados de comprobar los nombres de host de los URL y direcciones de correo electrónico. Es probable que una versión de producción no muestre estos resultados, pero es interesante ver la información devuelta por las comprobaciones

Listado 17.3. directory_submit.php. Secuencia de comandos para el URL y la dirección de correo electrónico

```
<html>
 <head>
  <title>Site submission results</title>
 </head>
 <body>
 <h1>Site submission results</h1>
 <?php

  // Extraiga los campos del formulario

  $url = $HTTP_POST_VARS["url"];
  $email = $HTTP_POST_VARS['email'];

  // Compruebe el URL

  $url = parse_url($url);
  $host = $url["host"];
  if(!($ip = gethostbyname($host)))
  {
    echo "Host for URL does not have valid IP";
    exit;
  }

  echo "Host is at IP $ip <br />";

  // Compruebe la dirección de correo electrónico

  $email = explode("@", $email);
```

```
    $emailhost = $email[1];

    // fíjese en que la función getmxrr() *no se implementa* en
    // las versiones de Windows de PHP
    if (!getmxrr($emailhost, $mxhostsarr))
    {
       echo 'Email address is not at valid host';
       exit;
    }

    echo 'Email is delivered via: ';
    foreach ($mxhostsarr as $mx)
       echo "$mx ";

    // Si se abre, todo perfecto

    echo '<br />All submitted details are ok.<br />';
    echo 'Thank you for submitting your site.<br />'
          .'It will be visited by one of our staff members soon.'

    // En un caso real, agregue a la base de datos de sitios en
     //espera...
?>
</body>
</html>
```

Vamos a examinar las partes interesantes de esta secuencia de comandos.

En primer lugar, tomamos el URL y le aplicamos la función `parse_url()`. Esta función devuelve una matriz asociativa de las diferentes partes de un URL. Las partes disponibles de la información son `scheme`, `user`, `pass`, `host`, `port`, `path`, `query` y `fragment`.

Por regla general, no vamos a necesitar todas ellas, pero a continuación se incluye un ejemplo de cómo forman un URL.

Dado un URL como el siguiente

```
http://nobody:secret@bigcompany.com:80/script.php?variable=value#anchor
```

los valores de cada una de las partes de la matriz serán

- scheme: http://
- user: nobody
- pass: secret
- host: bigcompany.com
- port: 80
- path: script.php
- query: variable=value
- fragment: anchor

En nuestra secuencia de comandos, sólo queremos información sobre `host` por lo que la extraemos de la siguiente forma:

```
$url = parse_url($url);
$host = $url['host'];
```

Tras ello, podemos obtener la dirección IP del host, si se encuentra en el DNS. Para ello podemos utilizar la función `gethostbyname()`, que devuelve la dirección IP si la hubiera o `false` en caso contrario:

```
$ip = gethostbyname($host);
```

También puede seguir el otro camino utilizando la función `gethostbyaddr()`, que toma una IP como parámetro y devuelve el nombre de host. Si llama a estas funciones de manera sucesiva, podría obtener un nombre de host distinto al utilizado al principio, lo que implica que el sitio está utilizando un servicio de host virtual.

Si el URL es válido, podemos comprobar la dirección de correo electrónico. En primer lugar la dividimos en nombre de usuario y nombre de host con una llamada a `explode()`:

```
$email = explode('@', $email);
$emailhost = $email[1];
```

Tras obtener la parte del host de la dirección, podemos comprobar si existe un lugar al que enviar dicho correo utilizando la función `getmxrr()`:

```
getmxrr($emailhost, $mxhostsarr);
```

Esta función devuelve un conjunto de registros MX (del inglés Mail Exchange, Intercambio de correo) para una dirección de la matriz suministrada en `$mxhostarr`.

Los registros MX se almacenan en el DNS y se buscan como un nombre de host. El equipo listado en el registro MX no tiene por qué ser necesariamente el equipo en el que acabará finalmente el correo.

En su lugar podría tratarse de un equipo que sepa dónde dirigir el correo. (Puede haber más de uno, en cuyo caso la función devolverá una matriz en lugar de una cadena con el nombre de usuario.) Si no tiene un registro MX en el DNS, el correo no tendrá dónde dirigirse.

Tenga en cuenta que la función `getmxrr()` no se implementa en las versiones de Windows de PHP.

Si todas estas comprobaciones resultan correctas, podemos colocar los datos del formulario en una base de datos para su posterior revisión por un miembro del personal.

Además de las funciones que acabamos de utilizar, puede utilizar la función genérica `checkdnsrr()`, que toma un nombre de host y devuelve `true` si incluye algún registro en el DNS.

Uso de FTP

El Protocolo de transferencia de archivos, o FTP, se utiliza para transferir archivos entre hosts de una red. En PHP, puede utilizar la función `fopen()` así como varias funciones de archivo con FTP también disponibles con conexiones HTTP, para conectar y transferir archivos desde y hacia un servidor FTP. Sin embargo, también se incluye un conjunto de funciones específicas de FTP que vienen con la instalación PHP estándar.

Estas funciones no se incluyen en la instalación estándar de manera predeterminada. Para poder utilizarlas en UNIX, es necesario ejecutar el programa `configure` de PHP con la opción `–enable-ftp` y volver a ejecutar `make`.

Si va a utilizar la instalación estándar de Windows, las funciones de FTP se habilitan de manera automática.

(Si desea obtener más detalles sobre la configuración de PHP, consulte el apéndice A.)

Uso de FTP para realizar una copia o reproducir un archivo

Las funciones de FTP son útiles para mover o copiar los archivos desde o hasta otros hosts. Se suelen utilizar para hacer copias de seguridad del sitio Web o para reproducir archivos situados en otra ubicación. Examinaremos un sencillo ejemplo que utiliza funciones FTP para reproducir un archivo. En el listado 17.4 se incluye la secuencia de comandos correspondiente a este ejemplo.

Listado 17.4. ftpmirror.php. Secuencia de comandos para descargar nuevas versiones de un archivo desde un servidor FTP

```
<html>
<head>
   <title>Mirror  update</title>
</head>
<body>
<h1>Mirror update</h1>
<?php
// configure variables: modifíquelas para ajustarlas a su aplicación
$host = 'ftp.cs.rmit.edu.au';
$user = 'anonymous';
$password = 'laura@tangledweb.com.au';
$remotefile = '/pub/tsg/teraterm/ttssh14.zip';
$localfile = '/tmp/writable/ttssh14.zip';

// conecte al host
$conn = ftp_connect("$host");
if (!$conn)
{
```

```
    echo 'Error: Could not connect to ftp server<br />';
    exit;
}
echo "Connected to $host.<br />";

// regístrese en el host
@ $result = ftp_login($conn, $user, $pass);
if (!$result)
{
    echo "Error: Could not log on as $user<br />";
    ftp_quit($conn);
    exit;
}
echo "Logged in as $user<br />";

// compruebe la información de fecha de los archivos para determinar si
// necesita actualizarlos
echo 'Checking file time...<br />';
if (file_exists($localfile))
{
    $localtime = filemtime($localfile);
    echo 'Local file last updated ';
    echo date('G:i j-M-Y', $localtime);
    echo '<br />';
}
else
    $localtime=0;
$remotetime = ftp_mdtm($conn, $remotefile);
if (!($remotetime >= 0))
{
     // Esto no significa que el archivo no esté ahí; puede que el servidor
    //  no admita el tiempo de  // modificación
     echo 'Can\'t access remote file time.<br />';
     $remotetime=$localtime+1;  // asegúrese de que existe una actualización

}
else
{
    echo 'Remote file last updated ';
    echo  date('G:i  j-M-Y',  $remotetime);
    echo '<br />';
}
if (!($remotetime > $localtime))
{
     echo 'Local copy is up to date.<br />';
     exit;
}

// descargue el archivo
echo 'Getting file from server...<br />';
$fp = fopen ($localfile, 'w');
if (!$success = ftp_fget($conn, $fp, $remotefile, FTP_BINARY))
{
    echo 'Error: Could not download file';
    ftp_quit($conn);
    exit;
```

```
}
fclose($fp);
echo 'File downloaded successfully';

// cierre la conexión al host
ftp_quit($conn);

?>
</body>
</html>
```

En la figura 17.4 se muestra el resultado de ejecutar esta secuencia de comandos una vez.

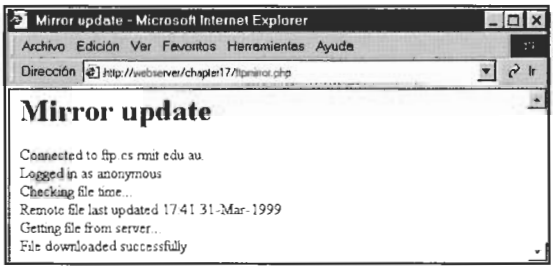


Figura 17.4. La secuencia de comandos para el sitio FTP de réplicas comprueba si la versión local de un archivo está actualizada y descarga una nueva versión en caso de que no lo esté

Se trata de una secuencia de comandos bastante genérica. Comienza por establecer algunas variables:

```
$host = 'ftp.cs.rmit.edu.au';
$user = 'anonymous';
$password = 'laura@tangledweb.com.au';
$remotefile = '/pub/tsg/teraterm/ttssh14.zip';
$localfile = "/tmp/writable/ttssh14.zip";
```

La variable `$host` debería contener el nombre del servidor FTP al que desee conectarse y las variables `$user` y `$password` se corresponden al nombre de usuario y contraseña que le gustaría utilizar para iniciar la sesión.

Muchos sitios FTP admiten lo que se conoce como sesiones anónimas en las que se puede utilizar un nombre de usuario puesto a disposición de todo el mundo para poder establecer la conexión. No es necesario utilizar ninguna contraseña, pero es habitual suministrar la dirección de correo electrónico en su lugar para que el administrador del sistema pueda saber de dónde proceden los usuarios. Ésta es la convención que hemos seguido aquí.

La variable `$remotefile` contiene la ruta hasta el archivo que queremos descargar. En este caso, vamos a descargar y reproducir una copia local del Tera Term SSH y el cliente SSH para Windows. (SSH equivale a "secure shell", núcleo seguro. Se trata de una forma de telnet encriptada.)

La variable `$localfile` contiene la ruta a la ubicación en la que vamos a almacenar el archivo descargado en nuestro equipo. En este caso, hemos creado un directorio llamado `/tmp/writable` con permisos establecidos para que PHP pueda escribir un archivo.

Debería poder cambiar estas variables para adaptar la secuencia de comandos a sus necesidades.

Los pasos básicos seguidos en esta secuencia de comandos son los mismos que si quisiéramos enviar por FTP el archivo desde una interfaz de línea de comandos:

1. Conectar al servidor remoto de FTP.
2. Iniciar la sesión (bien como un usuario normal o como anónimo).
3. Comprobar si el archivo remoto se ha actualizado.
4 Si se ha actualizado, descargarlo.
5. Cerrar la conexión a FTP.

Vamos a examinar cada uno de estos pasos uno a uno.

Conexión al servidor FTP remoto

Este paso equivale a escribir

```
ftp  hostname
```

en la línea de comandos en la plataforma Windows o en la plataforma UNIX. Este paso se realiza en PHP con el siguiente código:

```
$conn  =  ftp_connect("$host");
if (!$conn)
{
   echo 'Error: Could not connect to ftp server<br />';
   exit;
}
echo "Connected to $host.<br />";
```

La llamada a la función aquí es `ftp_connect()`. Esta función toma un nombre de host como parámetro y devuelve un identificador a una conexión o `false` si la conexión no pudo establecerse. La función también puede tomar el número de puerto del host al que conectarse como segundo parámetro opcional. (Aquí no hemos utilizado este parámetro.) Si no especifica un número de puerto, se utilizará el puerto 21 que es el puerto FTP predeterminado.

Inicio de sesión en el servidor FTP

El siguiente paso consiste en registrarse como usuario dado con un contraseña concreta. Para ello, puede utilizar la función `ftp_login()`:

```
@  $result  =  ftp_login($conn,  $user,  $pass);
if  (!$result)
```

```
{
   echo "Error: Could not log on as $user<br />";
   ftp_quit($conn);
   exit;
}
echo "Logged in as $user<br />";
```

La función toma tres parámetros: una conexión FTP (obtenida desde `ftp_connect()`), un nombre de usuario y una contraseña. Devolverá `true` si el usuario puede iniciar la sesión y `false` si no puede hacerlo. Como observará hemos colocado un símbolo @ al principio de la línea para eliminar errores. De esta forma, si el usuario no puede registrarse obtendremos una advertencia de PHP en nuestra ventana del navegador. Podemos capturar el error como hemos hecho aquí probando la variable `$result` y suministrando un mensaje de error más explicativo.

Tenga en cuenta que si el intento de inicio de sesión fallara, se cerraría la conexión FTP utilizando `ftp_quit()` (en un instante ampliaremos este punto).

Comprobación del momento temporal de la actualización de archivos

Como estamos actualizando una copia local de un archivo, resulta lógico comprobar primero si es necesario actualizar el archivo ya que no queremos volver a descargarlo si ya lo estuviera. La intención es evitar tráfico de red innecesario. Vamos a examinar el código que se encarga de esta tarea.

En primer lugar, comprobamos si tenemos una copia local del archivo con ayuda de la función `file_exists()`. En caso negativo, tendríamos que proceder a descargar el archivo. Si existiese, obtenemos la última fecha de modificación utilizando la función `filemtime()` y la almacenamos en la variable `$localtime`. Si no existiese, establecemos la variable `$localtime` en `0` para que resulte más antigua que cualquier hora de modificación de archivos remotos:

```
echo 'Checking file time...<br />';
if (file_exists($localfile))
{
   $localtime = filemtime($localfile);
   echo 'Local file last updated ';
   echo date('G:i j-M-Y', $localtime);
   echo '<br />';
}
else
   $localtime=0;
```

(Puede leer más acerca de las funciones `file_exists()` y `filemtime()` en varios capítulos anteriores.)

Tras establecer la hora local, necesitamos obtener la hora de modificación del archivo remoto. Para ello podemos utilizar la función `ftp_mdtm()`:

```
$remotetime = ftp_mdtm($conn, $remotefile);
```

Esta función toma dos parámetros (el identificador de la conexión FTP y la ruta al archivo remoto) y devuelve la marca de tiempo de UNIX de la hora a la que se

modificó por última vez el archivo o -1 si se produjo un error de algún tipo. No todos los servidores FTP admiten esta función, por lo que es posible que no obtengamos un resultado útil de esta función. En este caso, hemos optado por establecer artificialmente la variable `$remotetime` para que sea más reciente que la variable `$localtime` agregándole una unidad. De esta forma, nos aseguramos de que se intentará descargar el archivo una vez:

```
if (!($remotetime >= 0))
{
   // No significa que el archivo no esté ahí, sino que el servidor puede
   //no admitir la función
   echo 'Can't access remote file time.<br />';
   $remotetime=$localtime+1;  // asegúrese una actualización
}
else
{
   echo 'Remote file last updated ';
   echo date('G:i j-M-Y', $remotetime);
   echo '<br />';
}
```

Cuando tenemos ambas horas, las comparamos para comprobar si necesitamos descargar el archivo:

```
if (!($remotetime > $localtime))
{
   echo 'Local copy is up to date.<br />';
   exit;
}
```

Descarga del archivo

En esta fase intentamos descargar el archivo desde el servidor:

```
echo 'Getting file from server...<br />';
$fp = fopen ($localfile, 'w');
if (!$success = ftp_fget($conn, $fp, $remotefile, FTP_BINARY))
{
   echo 'Error: Could not download file';
   fclose($fp);
   ftp_quit($conn);
   exit;
}
fclose($fp);
echo 'File downloaded successfully';
```

Abrimos un archivo local utilizando `fopen()` como hicimos anteriormente. Tras ello, llamamos a la función `ftp_fget()` e intentamos descargar el archivo y almacenarlo en una ubicación local. Esta función toma cuatro parámetros. Los tres primeros son claros: la conexión FTP, el identificador de archivo local y la ruta hasta el archivo remoto. El cuarto es el modo FTP.

Existen dos modos de realizar una transferencia FTP: ASCII y binario. El modo ASCII se utiliza para transferir archivos de texto (es decir, archivo que se compo-

nen únicamente de caracteres ASCII) y el modo binario, se utiliza para transferir el resto. La biblioteca FTP de PHP incluye dos constantes predefinidas, `FTP_ASCII` y `FTP_BINARY`, que representan estos dos modos.

Necesita decidir qué modos encajan en su tipo de archivo y pasar la constante correspondiente a `ftp_fget()` como cuarto parámetro.

En este caso vamos a transferir un archivo zip, por lo que hemos utilizado el modo `FTP_BINARY`.

La función `ftp_fget()` devuelve `true` si todo va bien o `false` si aparece un error.

Almacenamos el resultado en `$success` e indicaremos al usuario cómo fue.

Tras intentar la descarga, cerramos el archivo local utilizando la función `fclose()`.

Como alternativa a `ftp_fget()`, podríamos haber utilizado `ftp_get()`, que tiene la siguiente sintaxis:

```
int ftp_get (int conexión_ftp, string ruta_archivo_local,
   string ruta_archivo_remoto, int modo)
```

Esta función funciona prácticamente de la misma forma que `ftp_fget()`, con la diferencia de que no requiere que se abra el archivo local. Se le pasa el nombre de archivo del sistema correspondiente al archivo local que le gustaría escribir en lugar de un identificador a un archivo.

Tenga en cuenta que no hay equivalente al comando `mget` de FTP, el cual se puede utilizar para descargar varios archivos a la vez. En su lugar debe utilizar varias llamada a `ftp_fget()` o `ftp_get()`.

Cierre de la conexión

Tras terminar con la conexión FTP, debería cerrarla utilizando la función `ftp_quit()`:

```
ftp_quit($conn);
```

Debería pasar a esta función el identificador correspondiente a la conexión FTP.

Carga de archivos

Si desea realizar la operación inversa, es decir, copiar archivos desde nuestro servidor a un equipo remoto, puede utilizar dos funciones que resultan básicamente opuestas a `ftp_fget()` y `ftp_get()`.

Estas funciones se denominan `ftp_fput()` y `ftp_put()`, y su sintaxis es la siguiente:

```
int ftp_fput (int conexión_ftp, string ruta_archivo_remoto, int fp,
int modo)
int ftp_put (int conexión_ftp, string ruta_archivo_remoto,
      string ruta_archivo_local, int modo)
```

Los parámetros son los mismos que los equivalentes de `_get`.

Cómo evitar los tiempos de espera

Uno de los problemas que pueden surgir al utilizar FTP para servir archivos es superar el tiempo máximo de ejecución. Sabrá cuándo ocurre porque PHP devuelve un mensaje de error. Este problema suele producirse si su servidor se ejecuta en una red lenta o con mucho tráfico o si se está descargando un archivo de gran tamaño, como un vídeo musical.

El valor predeterminado asignado al tiempo de ejecución máximo para todas las secuencias de comandos PHP se define en el archivo `php.ini`. De manera predeterminada es de 30 segundos. El objetivo es capturar secuencias de comandos que se ejecuten fuera de control. Sin embargo, al servir archivos con FTP, la transferencia de archivos podría superar este intervalo de tiempo, si su conexión es lenta o si el archivo es grande.

Afortunadamente, podemos modificar el tiempo máximo de ejecución para una secuencia de comandos dada con la función `set_time_limit()`.

La llamada a esta función restablece el número máximo de segundos que se puede ejecutar la secuencia de comandos, a partir del momento en el que se llamó a la función.

Por ejemplo, si llama a

```
set_time_limit(90);
```

la secuencia de comandos podrá ejecutarse otros 90 segundos a partir del momento en el que se llamó a la función.

Uso de otras funciones de FTP

PHP incorpora otras funciones útiles de FTP.

La función `ftp_size()` puede indicarle el tamaño de un archivo en el servidor remoto. Su sintaxis es la siguiente:

```
int ftp_size(int conexión_ftp, string ruta_archivo_remoto)
```

Esta función devuelve el tamaño del archivo remoto en bytes o -1 si surge un error. No todos los servidores FTP admiten esta función.

Esta función resulta útil para determinar el tamaño de ejecución máximo que establecer para una transferencia dada. Dado el tamaño del archivo y la velocidad de su conexión, se puede determinar aproximadamente el tiempo que tardará la transferencia en realizarse para utilizar la función `set_time_limit()` en consecuencia.

Puede obtener y mostrar una lista de archivos de un directorio en un servidor FTP remoto utilizando el siguiente código:

```
$listing = ftp_nlist($conn, "$directory_path");
foreach ($listing as $filename)
   echo "$filename <br />";
```

Este código utiliza la función `ftp_nlist()` para obtener una lista de los nombres de archivo de un directorio concreto.

En términos de otras funciones FTP, prácticamente todo lo que hagamos desde una línea de comandos FTP, se puede hacer utilizando funciones FTP. Si desea conocer las funciones específicas de cada comando FTP, consulte el manual en línea de PHP en `http://php.net/manual/en/ref.ftp.php`.

La excepción es `mget` (varios get), pero puede utilizar `ftp_nlist()` para obtener una lista de archivos y, a continuación, tomar los deseados.

Comunicaciones de red genéricas con cURL

PHP (desde la versión 4.0.2 en adelante) consta de un conjunto de funciones que actúan como interfaz a cURL, las funciones de biblioteca URL cliente de libcurl, escrita por Daniel Stenberg.

En una sección anterior de este capítulo, examinamos el uso de la función `fopen()` y de las funciones de lectura de archivos para leer un archivo remoto utilizando HTTP. Éste es más o menos el límite de la función `fopen()`. También vimos cómo establecer conexiones FTP con ayuda de funciones FTP.

Las funciones cURL permiten establecer conexiones utilizando FTP, HTTP, HTTPS, Gopher, Telnet, DICT, FILE y LDAP. También puede utilizar certificados para HTTPS, enviar parámetros HTTP POST y HTTP GET, cargar archivos mediante cargas FTP o HTTP, trabajar a través de proxies, establecer cookies y realizar funciones de autenticación de usuario HTTP simples.

En otras palabras, cURL permite realizar prácticamente cualquier tipo de conexión de red que desee.

Para usar cURL con PHP, necesitará descargar libcurl, compilarla y ejecutar la secuencia de comandos `configure` de PHP con la opción `–with-curl=[ruta]`. El directorio incluido en el parámetro `ruta` debería ser el que contenga los directorios `lib` e `include` en su sistema. Puede descargar la biblioteca de

```
http://curl.haxx.se/
```

Para poder trabajar con PHP necesitará utilizar una versión de cURL posterior a la 7.0.2-beta.

Para poder utilizar todo el potencial de cURL, sólo es necesario dominar el uso de unas cuantas funciones sencillas. El procedimiento habitual es el siguiente:

1. Establezca una sesión cURL con una llamada a la función `curl_init()`.
2. Establezca cualquier parámetro para realizar la transferencia con llamadas a la función `curl_setopt()`. Aquí es donde se establecen las opciones como el URL al que conectarse, los parámetros que enviar a dicho URL o el destino de la salida procedente del URL.
3. Cuando esté todo establecido, se llama a `curl_exec()` para establecer la conexión.

4. Para cerrar la sesión cURL se llama a la función a `curl_close()`.

Lo único que cambia con la aplicación de cURL es el URL al que se establece la conexión y los parámetros establecidos con `curl_opt()`. Se pueden establecer un gran número de parámetros.

A continuación, se recogen algunas aplicaciones de cURL:

- Para descargar páginas de un servidor que utilice HTTPS (porque `fopen()` no se puede utilizar con este fin en las versiones anteriores a PHP 4.3.)
- Para establecer la conexión a una secuencia de comandos que suela esperar datos procedentes de un formulario HTML utilizando `POST`
- Para escribir una secuencia de comandos para enviar varios conjuntos de datos de prueba a sus secuencias de comando y comprobar el resultado

Vamos a examinar un ejemplo (se trata de un sencilla aplicación que no se puede realizar de otra forma).

Este ejemplo, ilustrado en el listado 17.5, se conectará a Equifax Secure Server a través de HTTPS y escribirá el archivo que encuentra allí en un archivo de nuestro servidor Web.

Listado 17.5. https-curl.pha. Secuencia de comandos para establecer conexiones HTTPS

```
<?php
echo '<h1>HTTPS transfer with cURL</h1>';

$outputfile = '/tmp/writable/ssl-download.html';
$fp = fopen($outputfile, 'w+');

echo 'Initializing cURL session...<br />';
$ch = curl_init();

echo 'Setting cURL options...<br />';
curl_setopt ($ch, CURLOPT_URL, 'https://www.verisign.com/');
curl_setopt ($ch, CURLOPT_FILE, $fp);

echo 'Executing cURL session...<br />';
curl_exec ($ch);

echo 'Ending cURL session...<br />';
curl_close ($ch);

fclose($fp);

echo 'Here is the content of the file:<hr />';
readfile($outputfile);
?>
```

Vamos a examinar esta secuencia de comandos. Comenzaremos por abrir un archivo local utilizando `fopen()`. Aquí es donde vamos a almacenar la página que transferiremos desde la conexión segura.

Al hacerlo, necesitamos crear una sesión cURL utilizando la función `curl_init()`:

```
$ch = curl_init();
```

Esta función devuelve un identificador para la sesión cURL. Puede llamarla así, sin ningún parámetro, o puede pasarla en una cadena que contenga el URL al que conectarse. También puede establecer el URL utilizando la función `curl_setopt()`, que es lo que hemos hecho en este caso:

```
curl_setopt ($ch, CURLOPT_URL, 'https://www.equifaxsecure.com');
curl_setopt ($ch, CURLOPT_FILE, $fp);
```

La función `curl_setopt()` toma tres parámetros. El primero es el identificador de sesión, el segundo es el nombre del parámetro que establecer y el tercero es el valor que le gustaría establecer en el parámetro.

En este caso estamos estableciendo dos opciones. La primera es el URL al que queremos establecer la conexión. Se trata del parámetro `CURLOPT_URL`. El segundo es el archivo al que queremos que vayan los datos de la conexión. Si no se especifica un archivo, los datos de la conexión se dirigirán a la salida estándar (por regla general, el navegador). En este caso hemos especificado el identificador del archivo del archivo de salida que acabamos de abrir.

Tras establecer las opciones, indicamos a cURL que establezca la conexión:

```
curl_exec ($ch);
```

Esta instrucción abrirá una conexión al URL especificado, descargará el archivo y lo almacenará en el archivo al que apunte `$fp`.

Tras efectuar la conexión, debemos cerrar la sesión cURL y el archivo en el que se ha escrito:

```
curl_close ($ch);
fclose($fp);
```

Con esto termina el sencillo ejemplo utilizado.

Puede que le interese examinar la clase Snoopy, disponible en `http://snoopy.sourceforge.net/`.

Esta clase proporciona funcionalidad de cliente Web a través de cURL.

Lecturas adicionales

En este capítulo hemos cubierto muchos temas y, como cabría esperar, se han dejado muchos aspectos en el tintero.

Si desea obtener más información sobre protocolos concretos y su funcionamiento, puede consultar sus RFC en

`http://www.rfc-editor.org/`.

Puede que también encuentre interesante la información sobre protocolos recogida por el consorcio de la World Wide Web en:

```
http://www.w3.org/Protocols/
```

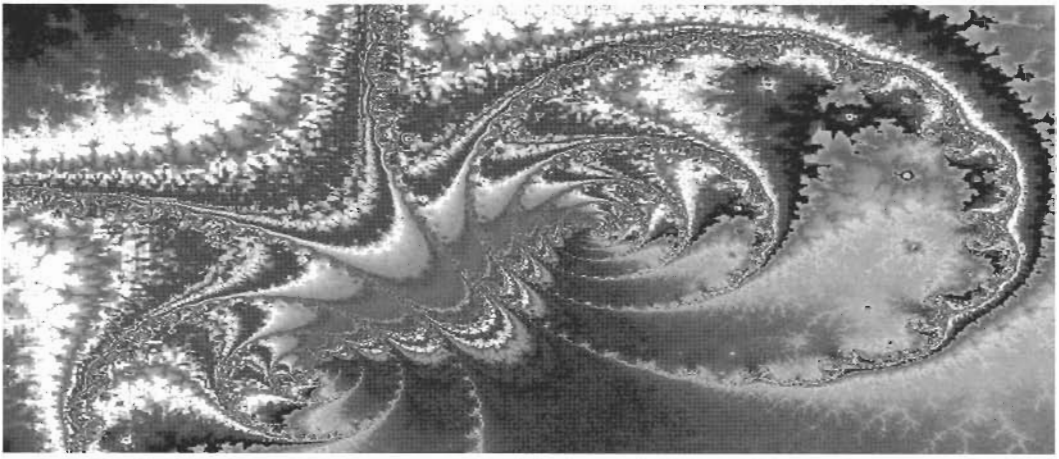
También puede consultar una obra sobre TCP/IP como *Computer Networks* escrita por Andrew Tanenbaum.

El sitio Web cURL incluye varias sugerencias sobre cómo utilizar las versiones de línea de comandos de las funciones cURL y resulta bastante fácil traducirlas a sus versiones de PHP:

```
http://curl.haxx.se/docs/httpscripting.shtml
```

A continuación

En el siguiente capítulo examinaremos las bibliotecas de PHP relacionadas con funciones de fecha y calendario. Veremos cómo convertir formatos de PHP introducidos por el usuario a formatos de MySQL y viceversa.



18 Administración de la fecha y la hora

En este capítulo, veremos cómo comprobar y aplicar formato a la fecha y a la hora, y cómo pasar de un formato de fecha a otro. Esta operación resulta especialmente importante al convertir formatos de fechas entre MySQL y PHP, entre Unix y PHP, y en fechas introducidas por el usuario a través de un formulario HTML.

En este capítulo, examinaremos los siguientes aspectos:

- Cómo obtener la fecha y la hora en PHP
- Conversión entre formatos de fecha de PHP y MySQL
- Cálculo de fechas
- Uso de funciones de calendario

Cómo obtener la fecha y la hora en PHP

En un capítulo anterior hablamos sobre el uso de la función `date()` para obtener y aplicar formato a la fecha y a la hora desde PHP. En este capítulo abordaremos estos aspectos y algunas otras funciones de fecha y hora de PHP.

Uso de la función date()

Como es posible que recuerde, la función `date()` toma dos parámetros, uno de ellos opcional. El primero es una cadena de formato y el segundo, de carácter opcio-

nal, es una marca de tiempo de UNIX. Si no se especifica una marca de tiempo, la función `date()` tomará la fecha y la hora actual de manera predeterminada. Esta función devuelve una cadena con formato en la que se representa la fecha pertinente.

Una llamada típica a la función de fecha podría ser:

```
echo date('S F Y');
```

El resultado de esta función será "27th August 2000".

En la tabla 18.1 se recogen los códigos de formato aceptados por la fecha.

Tabla 18.1. Códigos de formato para la función date() de PHP

Código	Descripción
a	Mañana o tarde, representado por los caracteres en minúsculas, "am" o "pm".
A	Mañana o tarde, representado por los caracteres en mayúsculas "AM" o "PM".
B	Hora de Internet de Swatch, un sistema horario universal. Si desea más información al respecto, diríjase a `http://www.swatch.com/`.
d	Día del mes en formato de dos dígitos con un cero a la izquierda. Los valores posibles van de "01" a "31".
D	Día de la semana en formato abreviado de dos caracteres. Los valores posibles van de "Lun" a "Dom".
F	Mes del año en formato de texto completo. Los posibles valores van de "Enero" a "Diciembre".
g	Hora del día en formato de 12 horas sin ceros a la izquierda. Los posibles valores van de "1" a "12".
G	Hora del día en formato de 24 horas sin ceros a la izquierda. Los posibles valores van desde "0" a "23".
h	Hora del día en formato de 12 horas con ceros a la izquierda. Los posibles valores van de "01" a "12".
H	Hora del día en formato de 24 horas con ceros a la izquierda. Los posibles valores van desde "0" a "23".
i	Minutos del día en formato de 24 horas con ceros a la izquierda. Los posibles valores van desde "00" a "59".
I	Horario de verano representado por un valor booleano. Devolverá "1" si la fecha se incluye en el horario de verano y "0" de lo contrario.
j	Día del mes en forma de número sin ceros a la izquierda. Los posibles valores van desde "1" a "31".
l	Día de la semana en formato de texto completo. Los posibles valores van desde "Lunes" a "Domingo".

Código	Descripción
L	Año bisiesto, representado por un valor booleano. Devolverá "1" si la fecha se incluye en un año bisiesto y "0" en caso lo contrario.
m	Mes del año en forma de número de dos dígitos con ceros a la izquierda. Los posibles valores van desde "01" a "12".
M	Mes del año en formato de texto abreviado de tres caracteres. Los posibles valores van desde "Ene" a "Dic".
n	Mes del año en formato de número sin ceros a la izquierda. Los posibles valores van desde "1" a "12".
O	Diferencia entre la zona horaria actual y la hora según el meridiano de Greenwich. Por ejemplo, +1600.
r	Fecha y hora con formato de RFC822, por ejemplo, Mie, 9 Oct 2002 18:45:30 +1600. (Agregada en PHP 4.0.4.)
s	Segundos que pasan del minuto con ceros a la izquierda. Los posibles valores van desde "00" a "59".
S	Sufijo ordinal para fechas con formato de dos caracteres.
t	Número total de días del mes. Los valores posibles van desde "28" a "31".
T	Zona horaria del servidor en formato de tres caracteres, por ejemplo, "EST".
U	Número total de segundos desde el 1 de enero de 1970 hasta este momento; también conocido como marca de tiempo de UNIX.
w	Día de la semana en formato de un solo dígito. Los valores posibles van desde "0" (domingo) a "6" (sábado).
W	Número de semana dentro del año, compatible con la norma ISO-8601. (Agregada en PHP 4.1.0).
y	Año en formato de dos dígitos, por ejemplo, "00".
Y	Año en formato de 4 dígitos, por ejemplo, "2000".
z	Día del año en formato de número. Los valores posibles van de "0" a "365".
Z	Diferencia con respecto a zona horaria actual en segundos. Los valores posibles van de "-43200" hasta "43200".

Marcas de tiempo de Unix

El segundo parámetro de la función `date()` es una marca de tiempo de Unix.

La mayor parte de los sistemas Unix almacenan la hora actual y la fecha en forma de número entero de 32 bits que contiene los números de segundos desde la media noche del 1 de enero de 1970, GMT, lo que también se conoce como Unix Epoch. Si no conoce este sistema puede parecerle extraño al principio, pero se trata de un estándar.

Las marcas de tiempo de Unix son una forma compacta de almacenar una fecha y una hora. Este sistema no se ha visto afectado por el efecto del año 2000 como ocurre con otros formatos de fecha compacta o abreviada. Si su software se sigue utilizando en 2038, experimentará problemas similares. Como las marcas de fecha no tienen tamaño fijo, se vinculan al tamaño de una C de gran tamaño, que tiene al menos 32 bits. La solución más probable es que para el año 2038, su compilador utilice un tipo de mayor tamaño.

Aunque esté ejecutando PHP en un servidor de Windows, éste seguirá siendo el formato utilizado por la función `date()` y varias otras funciones de PHP.

Si desea convertir la fecha y la hora a una marca de tiempo de Unix, puede utilizar la función `mktime()`, cuya sintaxis es la siguiente:

```
int mktime (int hora, int minuto, int segundo, int mes,
      int día, int año [, int hor_ver])
```

Los parámetros resultan bastante claros con la excepción del último, *hor_ver*, que representa si la fecha se corresponde con el horario de verano o no. Puede establecer este parámetro en 1 para seleccionar el horario de verano o 0, en caso lo contrario (el valor predeterminado es 1). Este valor es opcional y se suele utilizar muy poco.

El problema principal que presenta esta función es el orden de los parámetros, que resulta poco intuitivo. Además este orden no se presta a dejar la hora sin establecer. Si no quiere establecer la hora, puede utilizar ceros en los parámetros *hora*, *minuto* y *segundos*. En cualquier caso, no es necesario definir los parámetros más a la derecha. Si deja los parámetros en blanco, se les asignarán los valores actuales. Por lo tanto, una llamada como

```
$timestamp = mktime();
```

devolverá la marca de tiempo de Unix correspondiente a la hora y la fecha. Por supuesto, también puede obtener el mismo resultado llamando a

```
$timestamp = date("U");
```

Puede pasar un año en formato de dos o cuatro dígitos a la función `mktime()`. Los valores de dos dígitos desde el 0 al 69 se interpretarán como los años 2000 a 2069 y los valores del 70 al 99 se interpretarán como los años 1970 a 1999.

Uso de la función getdate()

Otra función para la determinación de la hora que podría resultarle útil es la función `getdate()`.

Esta función consta de la siguiente sintaxis:

```
array getdate (int maca_de_tiempo)
```

Esta función toma una marca de tiempo como parámetro y devuelve una matriz asociativa que representa las partes de dicha fecha y hora como se muestra en la tabla 18.2.

Tabla 18.2. Pares de clave y valor de la matriz asociativa de la función getdate()

Clave	Valor
Seconds	Segundos, numérico
minutes	Minutos numérico
hours	Horas, numérico
mday	Día del mes, numérico
wday	Día de la semana, numérico
mon	Mes, numérico
year	Año, numérico
yday	Día del año, numérico
weekday	Día de la semana, formato de texto numérico
month	Mes, formato de texto completo

Validación de fechas

Puede utilizar la función `checkdate()` para comprobar si una fecha es válida. Esta función resulta especialmente útil para comprobar las fechas introducidas por el usuario. La función `checkdate()` tiene la siguiente sintaxis:

```
int checkdate (int mes, int día, int año)
```

Esta función comprueba si el año es un número entero válido situado entre 0 y 32767, si el mes es un entero entre 1 y 12 y si el día dado existe en dicho mes. La función tiene en cuenta los años bisiestos. Por ejemplo:

```
checkdate(9, 18, 1972);
```

devolverá true mientras que

```
checkdate(9, 31, 2000)
```

no lo hará.

Conversión entre formatos de fecha de PHP y MySQL

Las fechas y las horas en MySQL se recuperan de forma ligeramente diferente a como se esperaría. Las horas funcionan de forma relativamente normal, pero MySQL

espera a que se introduzca primero el año. Por ejemplo, el 29 de agosto de 2000, se introduciría como 2000-08-29. Las fechas recuperadas desde MySQL también adoptarán este orden de manera predeterminada.

Por lo tanto, para comunicar PHP y MySQL necesitaremos realizar alguna operación de conversión de fechas, la cual se puede realizar en cualquiera de los dos extremos.

Al introducir fechas dentro de MySQL desde PHP, puede colocarlas de forma sencilla en el formato correcto utilizando la función `date()` mostrada anteriormente. Procure utilizar las versiones de fecha y hora con ceros a la izquierda para no confundir a MySQL.

Si opta por realizar la conversión en MySQL, puede utilizar dos funciones útiles `DATE_FORMAT()` y `UNIX_TIMESTAMP()`.

La función `DATE_FORMAT()` funciona de forma similar a la de PHP pero utiliza códigos de formato diferentes. En EEUU, lo habitual es aplicar el formato MM-DD-AAAA en lugar de AAAA-MM-DD propio de MySQL. Para ello, puede escribir la consulta de la siguiente forma:

```
SELECT DATE_FORMAT(columna_fecha '%m %d %Y')
FROM   nombre_de_tabla;
```

El código de formato `%m` representa el mes en forma de número de dos dígitos; `%d`, representa los días en forma de número de dos dígitos; y `%Y` representa el año en forma de número de 4 dígitos. En la tabla 18.3 se recoge los códigos de formato MySQL más útiles.

Tabla 18.3. Código de formato para la función DATE_FORMAT() de MySQL

Código	Descripción
%M	Mes, texto completo
%W	Nombre del día de la semana, texto completo
%D	Día del mes, numérico, con sufijo de texto
%Y	Año, numérico, 4 dígitos
%y	Año, numérico, 2 dígitos
%a	Día de la semana, 3 caracteres
%d	Día del mes, numérico, con ceros a la izquierda
%e	Día del mes, numérico, sin ceros a la izquierda
%m	Mes, numérico, con ceros a la izquierda
%c	Mes, numérico, sin ceros a la izquierda
%b	Mes, texto, 3 caracteres
%j	Día del año, numérico

Código	Descripción
%H	Hora, reloj de 24 horas, con ceros a la izquierda
%k	Hora, reloj de 24 horas, sin ceros a la izquierda
%h o %I	Hora, reloj de 12 horas, con ceros a la izquierda
%l	Hora, reloj de 12 horas, sin ceros a la izquierda
%i	Minutos, numéricos, con ceros a la izquierda
%r	Hora, en formato de 12 horas (hh:mm:ss [AM|PM])
%T	Hora, en formato de 24 horas (hh:mm:ss)
%S o %s	Segundos, numérico, ceros a la izquierda
%p	AM o PM
%w	Día de la semana, numérico, de 0 (domingo) a 6 (sábado)

La función `UNIX_TIMESTAMP` funciona de forma similar, pero convierte una columna en una marca de tiempo de Unix. Por ejemplo,

```
SELECT UNIX_TIMESTAMP(columna_fecha)
FROM nombre_de_tabla;
```

devolverá la fecha con formato de marca de tiempo de Unix. En PHP puede hacer como le parezca.

Como regla general, utilice una marca de tiempo de Unix para los cálculos de la fecha y el formato de fecha estándar al almacenar o mostrar fechas. Resulta más sencillo realizar cálculos de fechas y comparaciones con marcas de tiempo de Unix.

Cálculos de fecha

La forma más sencilla de calcular el intervalo de tiempo que discurre entre dos fechas en PHP consiste en utilizar la diferencia entre marcas de tiempo de UNIX. Hemos utilizado este enfoque en la secuencia de comandos ilustrada en el listado 18.1.

Listado 18.1. calc_age.php. Esta secuencia de comandos calcula la edad de una persona a partir de su fecha de nacimiento

```
<?php
   // establezca la fecha para el cálculo
   $day = 18;
   $month = 9;
   $year = 1972;

   // recuerde que necesita el día de nacimiento en formato día, mes y año
```

```
    $bdayunix = mktime ('', '', '', $month, $day, $year);
    // obtenga la marca de tiempo de unix correspondiente a la fecha de
    //nacimiento
    $nowunix = time(); // obtenga la marca de tiempo de unix correspondiente
    //al día de hoy
    $ageunix = $nowunix - $bdayunix; // calcule la diferencia
    $age = floor($ageunix / (365 * 24 * 60 * 60)); // convierta los
    //segundos en años

    echo "Age is $age";
?>
```

En esta secuencia de comandos, hemos establecido la fecha para el cálculo de la edad. En una aplicación real, es probable que esta información proceda de un formulario HTML. En primer lugar, llamamos a la función `mktime()` para calcular la marca de hora correspondiente al cumpleaños y a la fecha actual:

```
$bdayunix = mktime ('', '', '', $month, $day, $year);
$nowunix = mktime(); // obtenga la marca de tiempo de unix
correspondiente al día de hoy
```

Una vez que tenemos estas fechas en el mismo formato, bastará con restarlas:

```
$ageunix = $nowunix - $bdayunix;
```

Ahora viene la parte complicada: convertir este resultado en una unidad de medida que resulte más sencilla para los humanos. Podemos volver a convertir los años dividiendo el resultado por el número de segundos de los que consta un año. Seguidamente redondeamos el resultado utilizando la función `floor()` ya que una persona no tendrá, por ejemplo 20 años, hasta que los cumpla:

```
$age = floor($ageunix / (365 * 24 * 60 * 60)); // convierta los segundos en años
```

Tenga en cuenta que este enfoque presenta el problema del límite que impone el rango de las marcas de tiempo de Unix (por regla general, enteros de 32 bits). Puede que este ejemplo no sea una aplicación ideal para utilizar marcas de tiempo ya que sólo se puede utilizar para personas nacidas a partir de 1970.

Uso de funciones de calendario

PHP consta de una serie de funciones que permite convertir fechas entre diferentes sistemas de calendario. Los calendarios principales con los que se suele trabajar son el calendario gregoriano, el calendario juliano y la cuenta de días juliana.

El calendario gregoriano es el más extendido actualmente en los países occidentales. El 15 de octubre de 1582 del calendario gregoriano equivale al 5 de octubre de 1582 en el calendario juliano. Antes de dicha fecha, se solía utilizar el calendario juliano. Cada país pasó al calendario gregoriano en una fecha distinta y algunos no lo hicieron hasta principios del siglo XX.

Aunque es posible que haya oído hablar de estos dos calendarios, puede que no sea así con el tercero, la cuenta de días juliana. Este calendario se parece mucho a la marca de tiempo de Unix. En este calendario se cuenta el número de días desde una fecha en torno al año 4000 antes de Cristo. En sí mismo, no resulta especialmente útil, pero se puede utilizar para realizar conversiones entre formatos. Para pasar de un formato a otro, primero se convierte la fecha a la cuenta de días juliana y después se pasa el resultado al calendario deseado.

Para utilizar estas funciones con Unix, necesita haber compilado la extensión del calendario en PHP. Ésas se incorporan en la instalación de Windows estándar.

Para hacerse una idea, considere las siguientes sintaxis de las funciones que se utilizan para convertir fechas del calendario gregoriano al calendario juliano:

```
int gregoriantojd (int mes, int día, int año)
string jdtojulian(int díajuliano)
```

Para convertir una fecha, necesitaríamos llamar a ambas funciones:

```
$jd = gregoriantojd (9, 18, 1582);
echo jdtojulian($jd);
```

Este código devuelve la fecha juliana en formato mm/dd/aaaa.

Existen variaciones de estas funciones para realizar conversiones entre el calendario gregoriano, juliano, francés y judío, a marcas de tiempo de Unix.

Lecturas adicionales

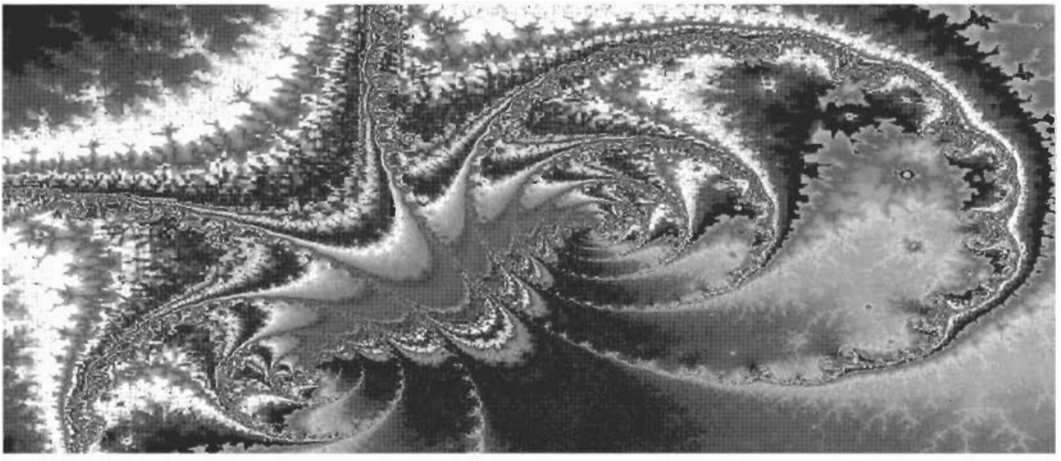
Si desea saber más sobre las funciones de fecha y hora de PHP y MySQL, puede consultar las secciones relevantes de sus manuales en

```
http://php.net/manual/en/ref.datetime.php
http://www.mysql.com/doc/en/Date_and_time_functions.html
```

Si va a realizar una conversión entre calendarios, consulte la página manual correspondiente a las funciones de calendario de PHP.

A continuación

Una de las funciones más útiles de PHP es su capacidad para crear imágenes al instante. En el siguiente capítulo se comentará cómo utilizar la función de bibliotecas de imagen para lograr algunos efectos útiles e interesantes.



19
Creación de imágenes

Una de las funciones más útiles de PHP es su capacidad para crear imágenes al instante. PHP incorpora varias funciones de información sobre imagenes y permite utilizar la biblioteca gd para crear nuevas imágenes o manipular las existentes.

En este capítulo veremos cómo utilizar las funciones de imagen para conseguir efectos útiles e interesantes. En concreto examinaremos los siguientes aspectos:

- Configuración de la compatibilidad para imágenes en PHP
- Comprensión de los formatos de imagen
- Creación de imágenes
- Uso de texto y fuentes para crear imágenes
- Dibujo de figuras y representación gráfica de datos

Vamos a utilizar dos ejemplos en los que generaremos botones de un sitio Web al instante y dibujaremos un diagrama de barras utilizando datos procedentes de una base de datos de MySQL.

Configuración de compatibilidad de imágenes en PHP

La compatibilidad para imágenes de PHP está disponible en la biblioteca gd en:

```
http://www.boutell.com/gd/
```

Desde la versión 4.3, PHP incorpora su propia versión de la biblioteca gd. Esta versión consta de funciones adicionales y suele estar más actualizada, por lo que es aconsejable utilizarla.

En algunas versiones anteriores de PHP se puede optar por escoger entre las bibliotecas gd y gd2. Es aconsejable inclinarse por la segunda ya que resulta más estable.

En Windows, las imágenes PNG y JPEG disponen de compatibilidad inherente.

Si utiliza Unix y quiere trabajar con imágenes en formato PNG, deberá instalar libpng y zlib desde los siguientes lugares (respectivamente):

```
http://www.libpng.org/pub/png/
http://www.gzip.org/zlib/
```

Necesitará configurar PHP con las siguientes opciones:

```
-with-png-dir=/path/to/libpng
-with-zlib-dir=/path/to/zlib
```

Si utiliza Unix y quiere trabajar con imágenes en formato JPEG, necesitará descargar jpeg-6b y volver a compilar gd con la compatibilidad jpeg incluida. Diríjase al siguiente sitio para realizar la descarga:

```
ftp://ftp.uu.net/graphics/jpeg/
```

Seguidamente, necesitará volver a configurar PHP con la opción

```
-with-jpeg-dir=/path/to/jpeg-6b
```

y volver a compilarlo.

Si quiere utilizar fuentes TrueType en sus imágenes, necesitará utilizar la biblioteca FreeType. Esta biblioteca se incluye en PHP 4.

Puede descargarla desde

```
http://www.freetype.org/
```

Si desea utilizar fuentes PostScript de tipo 1, deberá descargar la biblioteca t1lib disponible en

```
ftp://sunsite.unc.edu/pub/Linux/libs/graphics/
```

Tras ello necesitará ejecutar el programa de configuración de PHP con

```
-with-t1lib[=ruta/a/t1lib]
```

Formatos de imagen

La biblioteca gd admite los formatos JPEG, PNG y WBMP. Ha dejado de admitir el formato GIF. A continuación, repasaremos rápidamente cada uno de estos formatos.

JPEG

JPEG equivale a Joint Photographic Experts Group (Grupo conjunto de expertos en fotografía) y es el nombre de un conjunto de estándares. El formato de archivo al que se hace referencia al hablar de imágenes JPEG se llama JFIF y se corresponde con uno de los estándares del grupo mencionado.

Las imágenes JPEG se suelen utilizar para almacenar fotografías u otras imágenes con muchos colores o gradaciones de color. Este formato utiliza compresión con pérdida (es decir, que para reducir una fotografía en un archivo de menor tamaño se pierda parte de la calidad de la imagen). Como las imágenes JPEG deben conservar lo esencial de sus referentes analógicos, con las gradaciones de color, el ojo humano puede tolerar alguna pérdida de calidad. Este formato no resulta adecuado para dibujos lineales, texto o bloques sólidos de color.

Si lo desea puede leer más sobre JPEG/JFIF en el sitio oficial de JPEG:

```
http://www.jpeg.org/
```

PNG

PNG equivale a Portable Network Graphics (Gráficos portátiles de red). Este formato de archivo es el sustituto del formato GIF (del inglés Graphics Interchange Format, Formato de intercambio de gráficos) por razones que comentaremos seguidamente. El formato PNG incorpora compresión sin pérdida, por lo que resulta adecuado para imágenes que contengan texto, líneas rectas y bloques sencillos de color como encabezados y botones de sitios Web (los mismos propósitos a los que servía antes el formato GIF).

El formato PNG ofrece mejor compresión que el formato GIF así como transparencia variable, corrección gamma y dos entrelazados dimensionales. Sin embargo, no admite animaciones. Para ello, debe utilizar el formato extendido MNG, que está actualmente en fase de desarrollo.

Puede leer más sobre el formato PNG en su sitio oficial:

```
http://www.libpng.org/pub/png/
```

WBMP

WBMP equivale a Wireless Bitmap (Mapa de bits inalámbrico). Se trata de un formato de archivo diseñado específicamente para dispositivos inalámbricos.

GIF

GIF equivale a Formato de intercambio de gráficos. Se trata de un formato comprimido sin pérdida utilizado en la Web para almacenar imágenes que contengan texto, líneas rectas y bloques de un solo color.

Seguro que se está preguntado por qué gd no admite este formato.

La respuesta es que solía hacerlo, hasta la versión 1.3. Si desea instalar y utilizar funciones GIF en lugar de funciones PNG, puede descargar la versión 1.3 de gd de

```
http://www.linuxguruz.org/downloads/gd1.3.tar.gz
```

Tenga en cuenta que los creadores de gd no aconsejan utilizar esta versión y han dejado de incorporar compatibilidad para el formato GIF. Esta copia de la versión GIF es probable que deje de estar disponible en el futuro.

Existe una segunda razón para no admitir imágenes GIF. El formato GIF estándar utiliza una forma de compresión conocida como LZW (Lempel Ziv Welch) que está sujeta a una patente propiedad de UNISYS. Los proveedores de los programas que leen y escriben formatos GIF deben pagar por el uso de la licencia a UNISYS. Por ejemplo, Adobe paga una cuota de licencia por los productos como Photoshop que se utilizan para crear imágenes GIF. Los creadores de la biblioteca de código también deben pagar una cuota, así como los usuarios de dichas bibliotecas. Por lo tanto, si utiliza la versión GIF de la biblioteca gd en su sitio Web, deberá abonar las elevadas cuotas que aplica UNISYS.

Esta situación resulta poco afortunada porque el formato GIF llevaba mucho tiempo utilizándose antes de que UNISYS decidiera cobrar por su uso. De hecho, se convirtió en uno de los estándares de la Web. El paso dado por UNISYS no ha sentado nada bien en la comunidad de desarrollo Web. Si lo desea, puede leer más al respecto (y hacerse su propia opinión) en el sitio de UNISYS:

```
http://www.unisys.com/about__unisys/lzw/
```

y en el sitio opuesto, Burn All Gifs, en

```
http://burnallgifs.org/
```

No somos abogados y nada de lo que digamos aquí debe interpretarse como un consejo legal, pero en nuestra opinión resulta más sencillo utilizar el formato PNG, independientemente de las distintas posiciones.

La compatibilidad de los navegadores para el uso de imágenes con formato PNG está mejorando; sin embargo, la patente sobre LZW expira el 19 de junio de 2003, por lo que el resultado final está por ver.

Creación de imágenes

Los cuatro pasos básicos para crear una imagen en PHP son los siguientes:

1. Creación de un lienzo sobre el que trabajar.
2 Dibujar formas e imprimir texto en dicho lienzo.
3. Generar el gráfico final.
4. Liberar los recursos.

Comenzaremos por examinar una secuencia de comandos muy sencilla para la creación de imágenes. En el listado 19.1 se recoge esta secuencia de comandos.

Listado 19.1. simplegraph.php. Genera un gráfico compuesto de una sencilla línea con la secuencia de texto Sales

```
<?php
// cree la imagen
   $height = 200;
   $width = 200;
   $im = ImageCreate($width, $height);
   $white = ImageColorAllocate ($im, 255, 255, 255);
   $black = ImageColorAllocate ($im, 0, 0, 0);

// dibuje la imagen
   ImageFill($im, 0, 0, $black);
   ImageLine($im, 0, 0, $width, $height, $white);
   ImageString($im, 4, 50, 150, 'Sales', $white);

// genera la imagen
   Header ('Content-type: image/png');
   ImagePng ($im);

// libere recursos
   ImageDestroy($im);
?>
```

El resultado de ejecutar esta secuencia de comandos se ilustra en la figura 19.1.

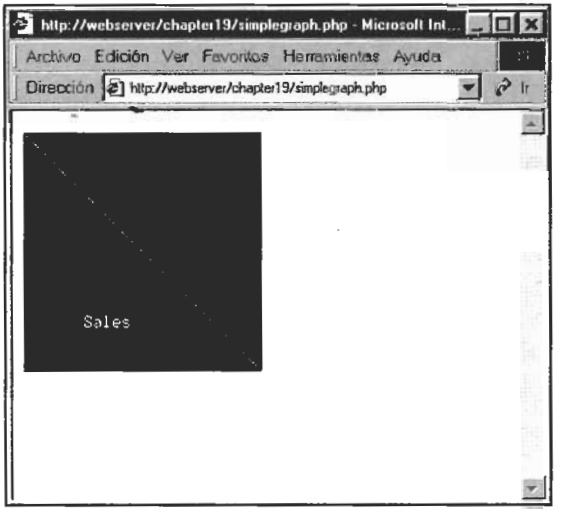


Figura 19.1. La secuencia de comandos dibuja un fondo de color negro y agrega una línea y una secuencia de texto a la imagen

Vamos a desgranar los pasos seguidos para crear esta imagen.

Creación de un lienzo

Para empezar a construir y cambiar una imagen en PHP, necesitará crear un identificador de imagen. Existen dos formas básicas de hacerlo. Una consiste en crear un lienzo en blanco, para lo cual se puede llamar a la función `ImageCreate()`, como hemos hecho en la secuencia de comandos de la siguiente forma:

```
$im = ImageCreate($width, $height);
```

Es necesario pasar dos parámetros a `ImageCreate()`. El primero es la anchura de la imagen y el segundo su altura. La función devolverá un identificador para la nueva imagen. (Su uso resulta parecido a los identificadores de archivo.)

También se puede leer un archivo de imagen existente y, a continuación, filtrarlo, modificar su tamaño o agregarlo. Para ello puede utilizar las funciones `ImageCreateFromPNG()`, `ImageCreateFromJPEG()` o `ImageCreateFromGIF()`, según el formato de archivo que esté leyendo. Estas funciones toman un nombre de archivo como parámetro, por ejemplo,

```
$im = ImageCreateFromPNG('baseimage.png');
```

En una sección posterior del capítulo se incluye un ejemplo en el que se utilizan imágenes para crear botones al instante.

Dibujo o impresión de texto en la imagen

La operación de dibujar o escribir texto en la imagen consta de dos fases.

En primer lugar, debe seleccionar los colores que se utilizarán en el dibujo. Como es probable que ya sepa, los colores que se muestran en un monitor se componen de diferentes cantidades de luz roja, verde y azul. Los formatos de imagen utilizan una paleta de colores que se compone de un subconjunto dado de todas las combinaciones posibles de estos colores. Para utilizar un color para dibujar una imagen, es necesario agregarlo a la paleta de la imagen. Este paso se debe realizar para todos los colores que se desee utilizar, incluidos el negro y el blanco.

Puede seleccionar los colores para la imagen llamando a la función `ImageColorAllocate()`.

Necesita pasar su identificador de imagen y los valores rojo, verde y azul del color que desee dibujar dentro de la función.

En el listado 19.1, se utilizan dos colores: el negro y el blanco. Los asignamos llamando a

```
$white = ImageColorAllocate ($im, 255, 255, 255);
$black = ImageColorAllocate ($im, 0, 0, 0);
```

La función devuelve un identificador de color que podemos utilizar para acceder al color posteriormente.

En segundo lugar, para dibujar la imagen, existen varias funciones disponibles, según lo que queramos dibujar: líneas, arcos, polígonos o texto.

Las funciones de dibujo requieren los siguientes parámetros:

- El identificador de imagen
- Las coordenadas de inicio y a veces las del final de lo que queremos dibujar
- El color en el que desea dibujar
- La información de fuente para el texto

En este caso, hemos utilizado dos funciones de dibujo. Vamos a examinar cada una de ellas por separado.

En primer lugar, pintamos un fondo negro sobre el que dibujar utilizando la función `ImageFill()`:

```
ImageFill($im, 0, 0, $black);
```

Esta función toma un identificador de imagen, las coordenadas iniciales del área que pintar (x e y) y el color de relleno como parámetros.

Nota

Observe que las coordenadas de la imagen se establecen a partir de la esquina superior izquierda, que es x=0 e y=0. La esquina inferior derecha de la imagen es x=$width, y=$height. Tenga cuidado porque este sistema es el opuesto al que se suele utilizar en los gráficos.

A continuación, hemos dibujado una línea que va de la esquina superior izquierda (`0, 0`) hasta la esquina inferir derecha (`$width, $height`) de la imagen.

Esta función toma el identificador de imagen, el punto de partida de x e y correspondiente a la línea, el punto final y el color como parámetros.

Por último, agregamos una leyenda al gráfico:

```
ImageString($im, 4, 50, 150, 'Sales', $white);
```

La función `ImageString()` toma varios parámetros ligeramente diferentes. Su sintaxis es la siguiente:

```
int imagestring (resource im, int fuente, int x, int y, string s, int col)
```

Toma como parámetros el identificador de la imagen, la fuente, las coordenadas x e y en las que empezar a escribir el texto, el texto que escribir y el color.

El valor de la fuente puede establecerse de 1 a 5, que representan un conjunto de fuentes incorporadas. Como alternativa, puede utilizar fuentes TrueType y PostScript Type 1. Cada uno de estos conjuntos de fuentes consta de sus funciones correspondientes.

En el siguiente ejemplo utilizaremos las funciones TrueType.

Una buena razón para utilizar uno de los conjuntos alternativos de funciones de fuentes es que el texto escrito por la función `ImageString()` así como por las funciones asociadas, como `ImageChar()` (escribe un carácter en la imagen), está sin suavizar. Las funciones de TrueType y PostScript producen texto suavizado.

Si no está seguro de las diferencias, examine la figura 19.3. En las zonas en las que aparecen curvas o líneas en ángulo el texto sin suavizar parece dentado. La curva y el ángulo se consiguen utilizando un efecto de "escalera". En la imagen suavizada, en las zonas de curvas o ángulos, se utilizan píxeles con colores que varían entre el utilizado en el texto y el del fondo con el objeto de suavizar la apariencia del texto.

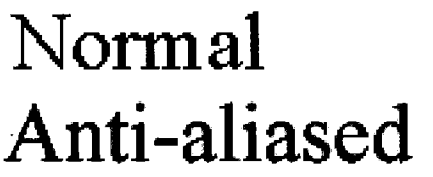


Figura 19.2. El texto normal parece dentado, especialmente cuando el tamaño de la fuente es grande. En el texto suavizado, las curvas y las esquinas de las letras pierden el carácter dentado

Generación del gráfico final

Puede dirigir la imagen final directamente al navegador para su representación o a un archivo. En este ejemplo, vamos a representar la imagen en el navegador. Esta operación consta de dos pasos. En primer lugar, necesitamos indicarle al navegador Web que vamos a representar una imagen en lugar de texto o código HTML. Para ello utilizamos la función `Header()` en la que especificamos el tipo MIME de la imagen:

```
Header ('Content-type: image/png');
```

Por regla general, al recuperar un archivo en el navegador, el tipo MIME es lo primero que envía el servidor Web. Para una página HTML o PHP, lo primero que se envía será

```
Content-type:  text/html
```

Esto indica al navegador cómo interpretar los datos que siguen.

En este caso, queremos decirle al navegador que vamos a enviar una imagen en lugar de la salida HTML habitual. Para ello, podemos utilizar la función `Header()`, que todavía no hemos visto.

Esta función envía cadenas de encabezado HTML sin procesar. También se suele utilizar para realizar redireccionamientos HTTP. En esta operación, se indi-

ca al navegador que cargue una página diferente en lugar de la seleccionada. Este recurso se suele utilizar al mover una página.

Por ejemplo:

```
Header ('Location: http://www.domain.com/new_home_page.html');
```

Al utilizar la función `Header()` tenga en cuenta que no se puede ejecutar si ya se ha enviado un encabezado HTTP para la página. PHP enviará un encabezado HTTP automáticamente en cuanto se devuelva algo al navegador. Por lo tanto, si ha devuelto alguna instrucción `echo` o incluso algún espacio en blanco antes de la etiqueta PHP de apertura, se enviarán los encabezados y PHP devolverá un mensaje de advertencia al intentar llamar a `Header()`. Sin embargo, puede enviar varios encabezados HTTP con varias llamadas a la función `Header()` en la misma secuencia de comandos, aunque todas ellas deben aparecer antes de que se envíe cualquier salida al navegador.

Tras enviar los datos del encabezado, devolvemos los datos de la imagen con una llamada a

```
ImagePng ($im);
```

Esta secuencia envía la salida al navegador en formato PNG. Si desea utilizar un formato diferente, puede llamar a `ImageJPEG()` (si está activada la compatibilidad para JPEG) o `ImageGIF()` (si tiene una versión antigua de gd).

También tendrá que enviar el encabezado correspondiente en primer lugar; es decir:

```
Header ('Content-type: image/jpeg');
```

o

```
Header ('Content-type: image/gif');
```

La segunda opción que puede utilizar, como alternativa a las anteriores, es escribir la imagen en un archivo en lugar de hacerlo en el navegador. Para ello, puede agregar un segundo parámetro opcional a `ImagePNG()` (o una función similar para el resto de los formatos compatibles):

```
ImagePNG($im, $filename);
```

Recuerde que se aplican todas las reglas habituales sobre la escritura en un archivo desde PHP (por ejemplo, haber configurado correctamente los permisos).

Liberación de recursos

Cuando haya terminado con una imagen, debe devolver los recursos que se han estado utilizando al servidor destruyendo su identificador. Para ello puede realizar una llamada a `ImageDestroy()`:

```
ImageDestroy($im);
```

Uso de imágenes generadas automáticamente en otras páginas

Como un encabezado sólo se puede enviar una vez y ésta es la única forma de indicar al navegador que estamos enviando datos de imagen, resulta un tanto complicado incrustar imágenes creadas al instante en una página normal. A continuación, se indican las tres posibles opciones:

1. Pude utilizar una página completa en forma de imagen, como hicimos en el ejemplo anterior.
2. Puede copiar la imagen en un archivo como se indicó anteriormente y utilizar la etiqueta `<img>` para hacer referencia a él.
3. Puede colocar la secuencia de comandos para la generación de imágenes en una etiqueta de imagen.

Ya hemos visto los primeros dos métodos. A continuación, repasaremos brevemente el tercero.

Para utilizar esta técnica, se incluye la imagen dentro del código HTML mediante una etiqueta de imagen situada de la siguiente forma:

```
<img src="simplegraph.php" height="200" width="200" alt="Sales going down" />
```

En lugar de colocar una imagen PNG, JPEG o GIF, insertamos la secuencia de comandos PHP que genera la imagen en la etiqueta SRC.

Ésta se recuperará y el resultado se agregará en línea, como se ilustra en la figura 19.3.



Figura 19.3. La imagen en línea generada dinámicamente parece igual que una imagen regular para un usuario final

Uso de texto y fuentes para crear imágenes

Vamos a examinar un ejemplo más complejo. Resulta útil poder crear botones u otras imágenes para un sitio Web de manera dinámica. Puede crear botones sencillos utilizando un rectángulo de color de fondo mediante las técnicas explicadas anteriormente.

En este ejemplo, vamos a generar botones utilizando una plantilla de botón vacía con efectos como bordes con relieve, que resultan mucho más sencillos de generar con Photoshop, GIMP u otras herramientas gráficas. Con la biblioteca de imágenes de PHP, podemos partir de una imagen básica y dibujar sobre ella.

Utilizaremos fuentes TrueType para poder recurrir a la función de texto suavizado. Las funciones de fuente TrueType tienen sus propios elementos especiales, que se comentarán posteriormente.

El proceso básico consiste en tomar una secuencia de texto y generar un botón colocando el texto encima. El texto se centrará horizontal y verticalmente en el botón, y se utilizará el tamaño más grande que resulte posible encajar en el botón.

Crearemos una interfaz para el generador de botones con fines de prueba y experimentación. En la figura 19.4 se ilustra esta interfaz. (No hemos incluido el código HMTL correspondiente a este formulario porque resulta muy sencillo, pero podrá encontrarlo en el CD en `design_button.html`.)



Figura 19.4. La interfaz permite a un usuario seleccionar el color del botón y escribir el texto deseado

Puede utilizar este tipo de interfaz para un programa que genere sitios Web automáticamente.

También puede llamar a la secuencia de comandos escrita de manera interna y generar todos los botones de un sitio Web al instante.

En la figura 19.5 se muestra el resultado típico de esta secuencia de comandos.

El botón es generado por una secuencia de comandos llamada `make_button.php`.

Esta secuencia de comandos se ilustra en el listado 19.2.

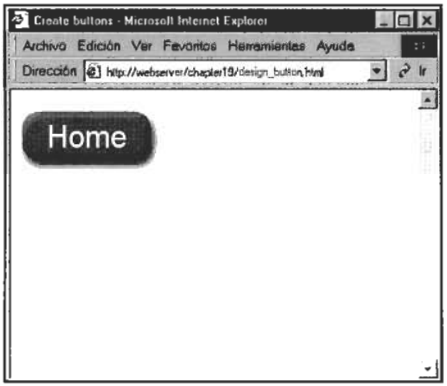


Figura 19.5. Un botón generado por la secuencia de comandos make_button.pho

Listado 19.2. make_button.php. Esta secuencia de comandos se puede llamar desde el formulario design_button.html o desde dentro de una etiqueta de imagen HTML

```
<?php
// compruebe que tenemos los datos de variable correctos
// las variables son button-text y color

$button_text = $HTTP_POST_VARS['button_text'];
$color = $HTTP_POST_VARS['color'];

if (empty($button_text) || empty($color))
{
   echo 'Could not create image - form not filled out correctly';
   exit;
}

// cree una imagen con el fondo correcto y compruebe el tamaño
$im = imagecreatefrompng ($color.'-button.png');

$width_image = ImageSX($im);
$height_image = ImageSY($im);

// Nuestras imágenes necesitan un margen de 18 píxeles desde el borde
//de la imagen
$width_image_wo_margins = $width_image - (2 * 18);
$height_image_wo_margins = $height_image - (2 * 18);

// Calcule si el tamaño de la fuente encaja y redúzcalo hasta que lo haga
// Parta del tamaño más grande que parezca encajar en los botones
$font_size = 33;

// Debería indica a GD2 dónde se alojan las fuentes
putenv('GDFONTPATH=C:\WINNT\Fonts');
$fontname = 'arial';

do
{
```

```
    $font_size--;

    // Busque el tamaño del texto en dicho tamaño de fuente
    $bbox=imagettfbbox ($font_size, 0, $fontname, $button_text);

    $right_text = $bbox[2];    // coordenada derecha
    $left_text = $bbox[0];    // coordenada izquierda
    $width_text = $right_text - $left_text;// ¿cuál es su achura?
    $height_text = abs($bbox[7] - $bbox[1]);  // ¿cuál es su altura?

}
while ( $font_size>8 &&
        ( $height_text>$height_image_wo_margins ||
          $width_text>$width_image_wo_margins )
        );

if ( $height_text>$height_image_wo_margins ||
       $width_text>$width_image_wo_margins )
{
    // no encaja ningún tamaño legible en el botón
    echo 'Text given will not fit on button.<br />';
}
else
{
    // Hemos encontrado un tamaño de fuente que encaja
    // Ahora debemos determinar dónde colocarlo

    $text_x = $width_image/2.0 - $width_text/2.0;
    $text_y = $height_image/2.0 - $height_text/2.0 ;

    if ($left_text < 0)
        $text_x += abs($left_text); // agregue factor para proyección
        //izquierda

    $above_line_text = abs($bbox[7]); // ¿altura con respecto a la
    //línea base?
    $text_y += $above_line_text;          // agregue factor de línea base

    $text_y -= 2; // factor de ajuste para la forma de nuestra plantilla

    $white = ImageColorAllocate ($im, 255, 255, 255);

    ImageTTFText ($im, $font_size, 0, $text_x, $text_y, $white, $fontname,
                  $button_text);

    Header ('Content-type: image/png');
    ImagePng ($im);
}

ImageDestroy ($im);
?>
```

Ésta es la secuencia de comandos más larga que hemos examinado hasta ahora. Vamos a analizarla sección por sección.

Comenzamos por una sencilla comprobación de errores y establecemos el lienzo sobre el que comenzaremos a trabajar.

Lienzo base

En el listado 19.2, en lugar de empezar desde cero, partiremos de una imagen existente para el botón. Disponemos de tres opciones con respecto al color del botón: rojo (`red-button.png`), verde (`green-button.png`) y azul (`blue-button.png`).

El color seleccionado por el usuario se almacena en la variable `$color` desde el formulario.

En primer lugar establecemos un nuevo identificador de imagen en función del botón pertinente:

```
$im = imagecreatefrompng ($color.'-button.png');
```

La función `ImageCreateFromPNG()` toma un nombre de archivo de una imagen PNG como parámetro y devuelve el nuevo identificador de imagen para la imagen que contiene una copia de dicho PNG. Tenga en cuenta que este hecho no modifica la imagen PNG base en absoluto. Podemos utilizar las funciones `ImageCreateFromJPEG()` y `ImageCreateFromGIF()` de la misma forma que se ha instalado la compatibilidad adecuada.

Nota

La llamada a `ImageCreateFromPNG()` sólo crea la imagen en memoria. Para guardar la imagen en un archivo o dirigirla al navegador para mostrarla, debemos llamar a la función `ImagePNG()`. No tardaremos en abordar este aspecto, pero todavía nos queda trabajo que realizar sobre nuestra imagen.

Cómo ajustar el texto sobre el botón

En la variable `$button_text` tenemos almacenado texto escrito por el usuario. Nuestra intención es imprimir dicho texto sobre el botón utilizando el cuerpo de fuente más grande que encaje en dicho espacio. Para ello, utilizamos el conocido método de ensayo y error mediante iteraciones.

En primer lugar establecemos algunas variables relevantes. Las dos primeras son la altura y la anchura de la imagen del botón:

```
$width_image  = ImageSX($im);
$height_image = ImageSY($im);
```

Las segundas representan el margen desde el borde del botón. Las imágenes para los botones que estamos utilizando tienen relieve, por lo que tendremos que prever espacio alrededor de los bordes del texto. Si va a utilizar imágenes diferen-

tes, el valor será distinto. En nuestro caso utilizaremos un margen de 18 píxeles a cada lado.

```
$width_image_wo_margins = $width_image - (2 * 18);
$height_image_wo_margins = $height_image - (2 * 18);
```

También necesitamos establecer el tamaño de fuente inicial. Comenzaremos por 32 (en realidad 33, que iremos reduciendo) porque éste es, aproximadamente, el cuerpo de texto más grande que encaja en el botón:

```
$font_size = 33;
```

Si utiliza gd2, deberá indicar dónde se alojan las fuentes. Para ello, debe establecer la variable de entorno `GDFONTPATH` de la siguiente forma:

```
putenv('GDFONTPATH=C:\WINNT\Fonts');
```

También establecemos el nombre de la fuente que queremos utilizar. Vamos a utilizar una fuente con funciones TrueType, que buscará en el archivo de fuente situado en la ubicación anterior y adjuntará el nombre del archivo con `.ttf` (TrueType Font).

```
$fontname = 'arial';
```

Tenga en cuenta que dependiendo del sistema operativo puede que tenga que agregar ".ttf" al final del nombre de la fuente.

Si no dispone de la fuente Arial (que vamos a utilizar aquí) en su sistema, puede sustituirla fácilmente por otra fuente TrueType.

A continuación, utilizamos un bucle para ir reduciendo el tamaño de la fuente hasta que el texto remitido encaje en el botón:

```
do
{
   $font_size--;

   // determine el tamaño del texto correspondiente a dicho cuerpo de texto
   $bbox=imagettfbbox ($font_size, 0, $fontname, $button_text);

   $right_text = $bbox[2]; // coordenada de la derecha
   $left_text = $bbox[0];    // coordenada de la izquierda
   $width_text = $right_text - $left_text; // ¿cuál es su anchura?
   $height_text = abs($bbox[7] - $bbox[1]); // ¿cuál es su altura?

}
while ( $font_size>8 &&
          ( $height_text>$height_image_wo_margins ||
            $width_text>$width_image_wo_margins )
        );
```

Esta secuencia de código comprueba el tamaño del texto examinando la caja del texto. Para ello, utilizamos la función `ImageGetTTFBBox()`, que es una de las funciones de las fuentes TrueType. Tras averiguar el tamaño del texto, lo imprimimos sobre el botón utilizando

una fuente TrueType (en este caso Arial, pero puede utilizar cualquier otra que desee) y la función `ImageTTFText()`.

La caja de una secuencia de texto es el recuadro más pequeño que se puede dibujar alrededor del texto. En la figura 19.6, se incluye un ejemplo de una caja de texto.



Figura 19.6. Las coordenadas de la caja de texto se establecen en función de la línea base. El origen de las coordenadas utilizado aquí es (0,0)

Para obtener las dimensiones de la caja, llamamos a

```
$bbox=imagettfbbox ($font_size, 0, $fontname, $button_text);
```

Esta llamada se interpreta de la siguiente forma: "para el tamaño de fuente $font_size, con el texto inclinado a un ángulo de 0 grados y utilizando la fuente Arial, indicar las dimensiones del texto en $button_text".

Tenga en cuenta que necesitamos pasar la ruta al archivo que contiene la fuente dentro de la función. En este caso, se trata del mismo directorio que la secuencia de comandos (el predeterminado), por lo que no se ha especificado una ruta más larga.

La función devuelve una matriz con las coordenadas correspondientes a las esquinas de la caja de texto.

En la tabla 19.1 se recogen los contenidos de la matriz.

Tabla 19.1. Contenidos de la matriz de la caja de texto

Índice de la matriz	Contenidos
0	Coordenada X, esquina inferior izquierda
1	Coordenada Y, esquina inferior izquierda
2	Coordenada X, esquina inferior derecha
3	Coordenada Y, esquina inferior derecha
4	Coordenada X, esquina superior derecha
5	Coordenada Y, esquina superior derecha
6	Coordenada X, esquina superior izquierda
7	Coordenada Y, esquina superior izquierda

Para recordar los contenidos de la matriz, basta con tener en cuenta que la numeración comienza en la esquina inferir izquierda de la caja de texto y va aumentando en el sentido de las agujas del reloj.

Es necesario tener en cuenta una característica especial relativa a los valores de la función `ImageTTFBBox()`: se trata de coordenadas especificadas desde un origen; sin embargo, a diferencia de las coordenadas utilizadas con imágenes, que se especifican en función de la esquina superior izquierda, en este caso se hace en función de una línea base.

Vuelva a examinar la figura 19.6. Como observará, se ha dibujado una línea que subraya el texto. Esta línea es lo que se conoce como línea base. Algunas letras quedan por debajo de esta línea base, como la "y". Éstas se conocen como descendentes.

La parte izquierda de la línea base se utiliza como origen de las dimensiones, es decir, las coordenadas X e Y serán 0. El valor de las coordenadas X por encima de la línea base será positivo y por debajo será negativo.

Además, el texto puede tener valores de coordenadas correspondientes a puntos situados fuera de la caja. Por ejemplo, el texto podría empezar en la coordenada X con valor -1.

Por lo tanto, debemos tener cuidado a la hora de realizar cálculos con estos números.

En nuestro caso, vamos a calcular la anchura y la altura del texto de la siguiente forma:

```
$right_text = $bbox[2];   // coordenada derecha
$left_text  = $bbox[0];       // coordenada izquierda
$width_text = $right_text - $left_text;   // ¿cuál es la anchura?
$height_text = abs($bbox[7] - $bbox[1]);  // ¿cuál es la altura?
```

Tras ello, probamos la condición del bucle:

```
} while ( $font_size>8 &&
          ( $height_text>$height_image_wo_margins ||
          $width_text>$width_image_wo_margins )
   );
```

En esta secuencia se prueban dos conjuntos de condiciones. El primero es que la fuente resulte legible (no es aconsejable utilizar fuentes con un tamaño inferior a los 8 puntos porque el botón resulta difícil de leer).

El segundo conjunto de condiciones comprueba si el texto encajará en el espacio que le hemos asignado.

A continuación, comprobamos si los cálculos del proceso de iteración obtienen un tamaño de fuente aceptable o no; en caso negativo, devolverán un error:

```
if ( $height_text>$height_image_wo_margins ||
     $width_text>$width_image_wo_margins )
{
   // ningún tamaño legible encaja en el botón
   echo 'Text given will not fit on button.<br />';
}
```

Colocación del texto

Si todo sale bien, el siguiente paso consiste en calcular el punto en el que colocar la parte inicial del texto.

Se trata del punto medio del espacio disponible.

```
$text_x = $width_image/2.0 - $width_text/2.0;
$text_y = $height_image/2.0 - $height_text/2.0 ;
```

Debido a las complicaciones con la línea base relativa al sistema de coordenadas, necesitamos agregar algunos factores de corrección:

```
if ($left_text < 0)
    $text_x += abs($left_text); // agregue factor para proyección izquierda

$above_line_text = abs($bbox[7]);// ¿espacio por encima de la línea base?
$text_y += $above_line_text;     // agregue factor para la línea base

$text_y -= 2;       // factor de ajuste para la forma de nuestra plantilla
```

Estos factores de corrección permiten realizar unos pequeños ajustes para corregir la altura de nuestra imagen.

Cómo escribir el texto en el botón

Tras ello, el resto es sencillo.

Establecemos el color del texto, que será blanco:

```
$white = ImageColorAllocate ($im, 255, 255, 255);
```

Podemos utilizar la función `ImageTTFText()` para dibujar el texto en el botón:

```
ImageTTFText ($im, $font_size, 0, $text_x, $text_y, $white, $fontname,
              $button_text);
```

Esta función toma bastantes parámetros.

Siguiendo su orden, se trata del identificador de la imagen, el tamaño de la fuente en puntos, el ángulo al que dibujar el texto, las coordenadas X e Y del texto, el color del texto, el archivo de la fuente y, por último, el texto en sí.

Nota

El archivo de fuente debe incluirse en el servidor.
No es necesario que el equipo del cliente incluya la fuente porque la verá en forma de imagen.

Para terminar

Por último, podemos generar el botón en el navegador:

```
Header ('Content-type: image/png');
ImagePng ($im);
```

Seguidamente, llega el momento de liberar los recursos y finalizar la secuencia de comandos:

```
ImageDestroy ($im);
```

Y eso es todo. Como resultado, debería aparecer un botón en la ventana del navegador similar al que se ve en la figura 19.5.

Dibujo de figuras y representación gráfica de datos

En la última aplicación, trabajamos con imágenes y el texto existentes. En el siguiente ejemplo veremos cómo realizar dibujos.

A continuación, desarrollaremos una encuesta en nuestro sitio Web para comprobar a quién votarían los usuarios en unas elecciones ficticias. Almacenaremos los resultados de la encuesta en una base de datos MySQL y dibujaremos un gráfico de barras con dichos resultados utilizando funciones de imagen.

Estas funciones también se utilizan para crear gráficos. Puede generar diagramas con los datos deseados: ventas, visitas a la Web, etc.

En este ejemplo, configuraremos una base de datos llamada `poll`. Esta base de datos contiene una tabla llamada `poll_results`, que incluye los nombres de los candidatos en la columna `candidate` y el número de votos recibidos en la columna `num_votes`. También hemos creado un usuario para esta base de datos llamado `poll` cuya contraseña es `poll`. La operación completa de configuración lleva unos cinco minutos si ejecuta la secuencia de comandos SQL ilustrada en el listado 19.3. Para ello puede direccionar la secuencia de comandos a través de un inicio de sesión de usuario raíz de la siguiente forma:

```
mysql -u root -p < pollsetup.sql
```

También puede utilizar el inicio de sesión de cualquier otro usuario que disponga de los privilegios MySQL apropiados.

Listado 19.3. pollsetup.sql. Creación de la base de datos Poll

```
create database poll;
use  poll;
create table poll_results (
```

```
    candidate varchar(30),
    num_votes int
);
insert into poll_results values
    ('John Smith', 0),
    ('Mary Jones', 0),
    ('Fred Bloggs', 0)
;
grant all privileges
on poll.*
to poll@localhost
identified by 'poll';
```

Esta base de datos contiene tres candidatos. Crearemos una interfaz de voto a través de una página llamada `vote.html`. En el listado 19.4 se incluye el código para esta página.

Listado 19.4. vote.html. Los usuarios pueden votar a través de esta página

```
<html>
<head>
  <title>Polling</title>
<head>
<body>
<h1>Pop Poll</h1>
<p>Who will you vote for in the election?</p>
<form method="post" action="show_poll.php">
<input type="radio" name="vote" value="John Smith">John Smith<br />
<input type="radio" name="vote" value="Mary Jones">Mary Jones<br />
<input type="radio" name="vote" value="Fred Bloggs">Fred Bloggs<br /><br />
<input type="submit"  value="Show results">
</form>
</body>
```

El resultado de esta página se muestra en la figura 19.7.

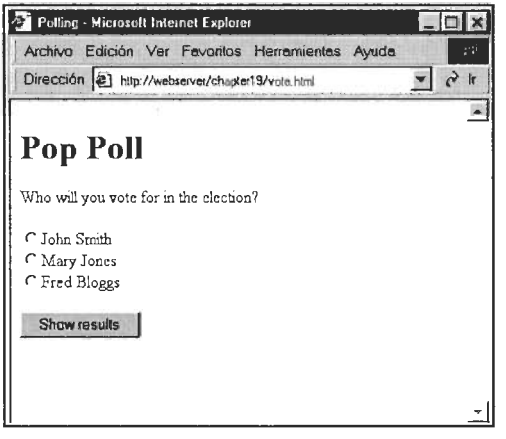


Figura 19.7. Los usuarios puede votar y, si hacen clic sobre el botón de envío, se mostrarán los resultados actuales de la encuesta

Cuando los usuarios hacen clic sobre el botón, agregaremos sus votos a la base de datos, obtendremos todos los votos de la base de datos y dibujaremos un diagrama de barras con los resultados actuales.

En la figura 19.8 se recoge un diagrama típico tras recoger varios votos.

La secuencia de comandos que genera esta imagen es bastante larga. Vamos a dividirla en cuatro partes que analizaremos por separado.

La mayor parte de la secuencia de comandos resulta conocida ya que hemos examinado bastantes ejemplos de MySQL parecidos. Hemos visto cómo pintar un lienzo de fondo en un color sólido y cómo imprimir las etiquetas de texto sobre él.

Las partes nuevas de esta secuencia de comandos son las encargadas de dibujar líneas y rectángulos. Nos centraremos en estas secciones. En el listado 19.5.1, se recoge la parte 1 (de las cuatro partes que componen la secuencia de comandos).

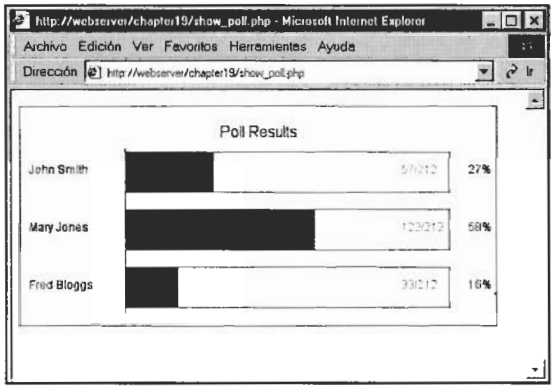


Figura 19.8. Los resultados de los votos se crean dibujando series de líneas, rectángulos y elementos de texto sobre un lienzo

Listado 19.5.1. show_poll.php. La parte 1 actualiza la base de datos de votos y recupera los nuevos resultados

```
<?php
/*****************************************
   Consulta para obtener información
*****************************************/

// obtenga el voto desde el formulario
$vote=$HTTP_POST_VARS['vote'];

// regístrese en la base de datos
if (!$db_conn = @mysql_connect('localhost', 'poll', 'poll'))
{
   echo 'Could not connect to db<br />';
   exit;
};
@mysql_select_db('poll');

if (!empty($vote)) // si el usuario rellena el formulario agregue su voto
```

```
{
   $vote = addslashes($vote);
   $query  =  "update  poll_results
               set num_votes = num_votes + 1
               where candidate = '$vote'";
   if(!($result = @mysql_query($query, $db_conn)))
   {
      echo  'Could  not  connect  to  db<br  />';
      exit;
   }
};

// obtenga los resultados actuales de la encuesta, hayan votado o no
$query  =  'select  *  from  poll_results';
if(!($result = @mysql_query($query, $db_conn)))
{
   echo  'Could  not  connect  to  db<br  />';
   exit;
}
$num_candidates  =  mysql_num_rows($result);

// calcule el número total de votos hasta el momento
$total_votes=0;
while  ($row  =  mysql_fetch_object  ($result))
{
      $total_votes +=  $row->num_votes;
}
mysql_data_seek($result, 0);// restablezca el puntero de resultado
```

La parte 1, ilustrada en el listado 19.5.1, establece la conexión a la base de datos MySQL, actualiza los votos en función del usuario introducido y recoge los nuevos votos. Tras obtener dicha información, podemos empezar con los cálculos para dibujar el gráfico. En el listado 19.5.2 se muestra la parte 2.

Listado 19.5.2. showpoll.phs. La parte 2 establece todas las variables para el dibujo

```
/*****************************************
   Cálculos iniciales para el gráfico
*****************************************/
// establezca las constantes
putenv('GDFONTPATH=C:\WINNT\Fonts');
$width=500;// anchura de la imagen en píxeles - ésta encajará en 640x480
$left_margin = 50; // espacio que dejar a la izquierda de la imagen
$right_margin= 50; // espacio que dejar a la derecha de la imagen
$bar_height = 40;
$bar_spacing = $bar_height/2;
$font = 'arial';
$title_size= 16; // punto
$main_size= 12; // punto
$small_size= 12; // punto
$text_indent = 10; // ubicación para las etiquetas de texto a la izquierda

// establezca el punto inicial desde el que dibujar
$x = $left_margin + 60;   // lugar en el que dibujar la línea base del
```

```
//gráfico
$y = 50;  // lugar en el que dibujar la línea base del gráfico
$bar_unit = ($width-($x+$right_margin)) / 100;    // un "punto" en el gráfico

// calcule la altura del gráfico - las barras, más los huecos, más cierta
//distancia de margen
$height = $num_candidates * ($bar_height + $bar_spacing) + 50;
```

La parte 2 establece algunas variables que se pueden utilizar para dibujar el gráfico.

El cálculo de los valores para este tipo de variables puede resultar tedioso, pero si se prevé con antelación el aspecto que desea que tenga la imagen al final el proceso de dibujo resultará mucho más sencillo. Para obtener los valores utilizados, dibujamos el efecto deseado en un papel y estimamos las proporciones necesarias. La variable `$width` es la anchura total del lienzo que utilizaremos. También establecemos los márgenes izquierdo y derecho (con `$left_margin` y `$right_margin`); el grosor y el espacio entre las barras (`$bar_height` y `$bar_spacing`); y la fuente, los tamaños de fuente y la ubicación de las leyendas (`$font`, `$title_size`, `$main_size`, `$small_size` y `$text_indent`).

Partiendo de estos valores, podemos realizar unos cuantos cálculos. Queremos dibujar una línea base desde la que partan todas las barras. Podemos determinar la ubicación de esta línea base utilizando el margen izquierdo más un espacio para las etiquetas de texto para establecer la coordenada X y un valor estimado obtenido de nuestro boceto en papel para la coordenada Y.

También podemos calcular dos valores importantes: en primer lugar, la distancia en el gráfico que representa una unidad:

```
$bar_unit = ($width-($x+$right_margin)) / 100;    // un "punto" en el gráfico
```

Se trata de la longitud máxima de las barras (desde la línea base hasta el margen derecho) dividido por 100 porque nuestro gráfico va a mostrar valores de porcentaje.

El segundo valor es la altura total que necesitamos para el lienzo:

```
$height = $num_candidates * ($bar_height + $bar_spacing) + 50;
```

Se trata básicamente de la altura de la barras multiplicada por el número de barras, más una cantidad extra de espacio para el título. En el listado 19.5.3 se recoge el código correspondiente a la parte 3.

Listado 19.5.3. La parte 3 crea el gráfico y lo deja listo para agregar datos

```
/*****************************************
   Crear la imagen de base
*****************************************/
// Cree un lienzo vacío
$im = imagecreate($width,$height);

// Asigne colores
```

```
$white=ImageColorAllocate($im,255,255,255);
$blue=ImageColorAllocate($im,0,64,128);
$black=ImageColorAllocate($im,0,0,0);
$pink = ImageColorAllocate($im,255,78,243);

$text_color  =  $black;
$percent_color  =  $black;
$bg_color  =  $white;
$line_color = $black;
$bar_color = $blue;
$number_color = $pink;

// Cree el lienzo en el que dibujar
ImageFilledRectangle($im,0,0,$width,$height,$bg_color);

// Dibuje líneas alrededor del lienzo
ImageRectangle($im,0,0,$width-1,$height-1,$line_color);

// Agregue el título
$title = 'Poll Results';
$title_dimensions = ImageTTFBBox($title_size, 0, $font, $title);
$title_length = $title_dimensions[2] - $title_dimensions[0];
$title_height  =  abs($title_dimensions[7]  -  $title_dimensions[1]);
$title_above_line  =  abs($title_dimensions[7]);
$title_x = ($width-$title_length)/2; // céntrelo en x
$title_y = ($y - $title_height)/2 + $title_above_line;
ImageTTFText($im,  $title_size,  0,  $title_x,  $title_y,
     $text_color, $font, $title);

// Dibuje una línea base ligeramente por encima de la primera barra
// hasta un poco por debajo de la última
ImageLine($im, $x, $y-5, $x, $height-15, $line_color);
```

En la parte 3 creamos la imagen de base, asignamos los colores y comenzamos a dibujar el gráfico.

Para rellenar el fondo del gráfico utilizamos:

```
ImageFilledRectangle($im,0,0,$width,$height,$bg_color);
```

La función `ImageFilledRectangle()`, como podría imaginar, dibuja un rectángulo relleno de color. El primer parámetro es, de la forma habitual, el identificador de la imagen. A continuación, debemos pasar las coordenadas X e Y del punto inicial y del punto final del rectángulo. Éstas se corresponden con la esquina superior izquierda y con la esquina inferior derecha, respectivamente. En este caso, rellenamos el lienzo completo con el color de fondo, que es el último parámetro y es blanco.

A continuación llamamos a

```
ImageRectangle($im,0,0,$width-1,$height-1,$line_color);
```

para dibujar un línea negra por el borde del lienzo. Esta función dibuja un rectángulo sin relleno. Los parámetros son los mismos. Fíjese en que hemos dibujado el rectángulo con las medidas `$width-1` y `$height-1`. Como resultado, se creará un lienzo con una altura y achura según estos valores desde (0,0). Si lo

dibujamos con los valores `$width` y `$height`, el rectángulo saldrá fuera del área del lienzo. Utilizamos la misma lógica y las mismas funciones que en la última secuencia de comandos y escribimos el título en el gráfico. Por último, dibujamos una línea base para las barras con

```
ImageLine($im, $x, $y-5, $x, $height-15, $line_color);
```

La función `ImageLine()` dibuja una línea en la imagen especificada (`$im`) desde un conjunto de coordenadas ($x, $y-5) a otra ($x, $height-15), en el color especificado por `$line_color`.

En este caso, dibujamos la línea base ligeramente por encima de dónde queremos dibujar la primera barra hasta un poco por encima de la parte inferior del lienzo. Ya estamos preparados para rellenar de datos el gráfico. En el listado 19.5.4 se muestra al parte 4.

Listado 19.5.4. showpoll.php. La parte 4 dibuja los datos reales sobre el gráfico y termina

```
/*****************************************
   Dibujar datos en el gráfico
*****************************************/
// Obtenga cada línea de la base de datos y dibuje las barras
//correspondientes
while ($row = mysql_fetch_object ($result))
{
   if ($total_votes > 0)
      $percent = intval(round(($row->num_votes/$total_votes)*100));
   else
      $percent = 0;

   // muestre el porcentaje de este valor
   ImageTTFText($im, $main_size, 0, $width-30, $y+($bar_height/2),
                $percent_color, $font, $percent.'%');
   if ($total_votes > 0)
      $right_value = intval(round(($row->num_votes/$total_votes)*100));
   else
      $right_value = 0;

   // longitud de la barra para este valor
   $bar_length = $x + ($right_value * $bar_unit);

   // dibuje la barra para este valor
   ImageFilledRectangle($im, $x, $y-2, $bar_length, $y+$bar_height,
$bar_color);

   // dibuje el título para este valor
   ImageTTFText($im, $main_size, 0, $text_indent, $y+($bar_height/2),
                $text_color, $font, $row->candidate);

   // dibuje la línea mostrando el 100%
   ImageRectangle($im, $bar_length+1, $y-2,
        ($x+(100*$bar_unit)), $y+$bar_height, $line_color);

   // muestre los números
```

```
    ImageTTFText($im, $small_size, 0, $x+(100*$bar_unit)-50,
$y+($bar_height/2),
        $number_color, $font, $row->num_votes.'/'.$total_votes);

    // pase a la siguiente barra
    $y=$y+($bar_height+$bar_spacing);
}

/*****************************************
    Mostrar imagen
*****************************************/
Header('Content-type: image/png');
ImagePng($im);


/*****************************************
    Liberar recursos
*****************************************/
ImageDestroy($im);
?>
```

La parte 4 recorre los candidatos desde la base de datos uno a uno, calcula los porcentajes de votos y dibuja las barras y las leyendas para cada uno.

Para agregar las leyendas hemos vuelto a utilizar la función `ImageTTFText()`. Dibujamos las barras como rectángulos rellenos de color utilizando `ImageFilledRectangle()`:

```
ImageFilledRectangle($im, $x, $y-2, $bar_length, $y+$bar_height, $bar_color);
```

Agregamos los contornos para indicar el 100% utilizando `ImageRectangle()`:

```
ImageRectangle($im, $bar_length+1, $y-2,
        ($x+(100*$bar_unit)), $y+$bar_height, $line_color);
```

Tras dibujar las barras, volvemos a generar la imagen con `ImagePNG()` y liberamos los recursos con `ImageDestroy()`.

Ésta secuencia de comandos es larga, pero resulta fácil de adaptar a las diferentes necesidades o a las encuestas autogeneradas a través de una interfaz. Un elemento importante que falta en esta secuencia de comandos es un mecanismo contra engaños ya que los usuarios no tardarán en descubrir que pueden votar tantas veces como deseen, lo que hará que los resultados no sirvan para mucho. Puede utilizar el mismo método para dibujar gráficos de líneas o incluso gráficos circulares, si se le dan bien las matemáticas.

Otras funciones de imagen

Además de las funciones de imagen que hemos utilizado en este capítulo, existen otras que permiten dibujar curvas (`ImageArc()`), polígonos (`ImagePolygon()`), así como variaciones sobre las utilizadas hasta ahora. Comience siem-

pre con un boceto de lo que desee dibujar y, a continuación, consulte el manual para buscar cualquier función que pueda necesitar.

Lecturas adicionales

En Internet encontrará una gran cantidad de material de lectura. Si tiene problemas con las funciones de imagen, a veces resulta aconsejable examinar la documentación de gd porque las funciones de PHP contienen esta biblioteca. Para ello, diríjase a

```
http://www.boutell.com/gd/
```

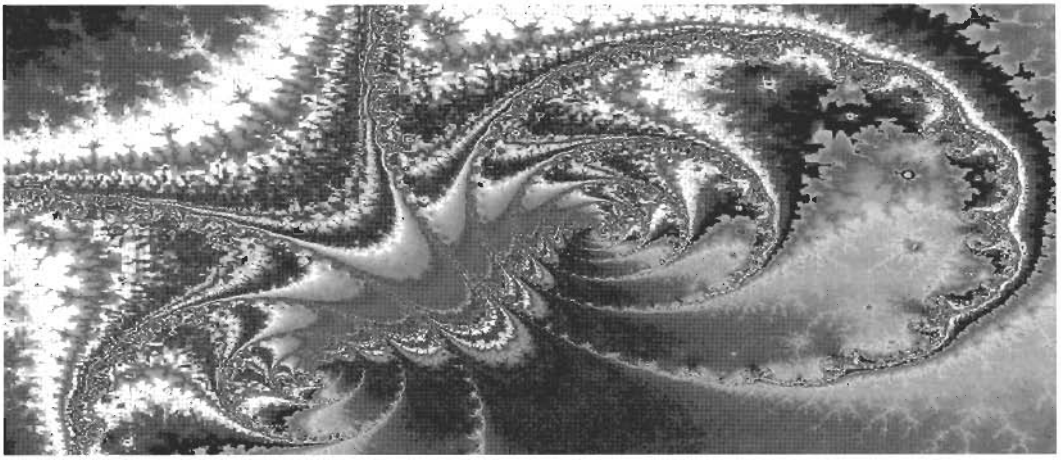
También existen algunos cursos prácticos excelentes sobre tipos dados de aplicaciones gráficas, en concreto en Zend y Devshed:

```
http://www.zend.com
http://devshed.com
```

La aplicación de diagrama de barras utilizada en este capítulo está inspirada en la secuencia de comandos de diagrama de barras dinámico escrita por Steve Maranda y disponible en Devshed.

A continuación

En el siguiente capítulo, analizaremos la práctica funcionalidad de control de sesión de PHP (novedad en PHP 4).



20
Uso del control de sesión en PHP

En este capítulo analizaremos la funcionalidad del control de sesión en PHP 4. Nos centraremos en los siguientes aspectos:

- Qué es el control de sesión
- Cookies
- Configuración de una sesión
- Variables de sesión
- Sesiones y autenticación

Qué es el control de sesión

Seguramente habrá escuchado que "HTTP es un protocolo sin estado". Esto significa que el protocolo no dispone de un método incorporado para conservar el estado entre dos transacciones. Cuando un usuario solicita una página y después otra, HTTP no cuenta con ninguna técnica para que podamos saber que ambas solicitudes provienen del mismo usuario.

El concepto del control de sesión consiste en poder realizar el seguimiento del usuario durante una sola sesión en un sitio Web.

Si podemos hacerlo, podremos conectar a un usuario y mostrar contenido en función de su nivel de autorización o de sus preferencias personales. Podemos realizar el

seguimiento del comportamiento del usuario. Podemos implementar un carro de la compra.

En anteriores versiones de PHP, el control de sesión se admitía a través de PHPLib, la biblioteca base de PHP, que sigue siendo una herramienta muy útil. Encontrará más información al respecto en `http://phplib.sourceforge.net/`.

En cuanto a la versión 4, PHP incluye funciones de control de sesión propias. Conceptualmente son similares a PHPLib pero ésta última ofrece funcionalidad adicional. Si las funciones nativas no se adaptan a sus necesidades, puede que le interese utilizar esta biblioteca.

Funcionalidad de sesión básica

En PHP, las sesiones se controlan por un solo Id. de sesión, un número criptográficamente aleatorio, generado por PHP y que se almacena en el lado del cliente mientras dura la sesión. Se puede almacenar en el ordenador del usuario en forma de cookie o pasarse por medio de URL.

El Id. de sesión actúa como una clave que le permite registrar determinadas variables como variables de sesión. Los contenidos de estas variables se almacenan en el servidor. El Id. de sesión es la única información visible en el lado del cliente. Si, al realizar una determinada conexión a su sitio, se puede ver el Id. de sesión por medio de una cookie o del URL, podrá acceder a las variables de sesión almacenadas en el servidor (aunque puede cambiarlo para utilizar una base de datos si pretende escribir su propia función, como veremos en el apartado sobre configuración del control de sesión). Probablemente habrá utilizado sitios Web que almacenen un Id. de sesión en el URL. Si en su URL hay una cadena de datos aparentemente aleatorios, es muy probable que se trate de alguna forma de control de sesión.

Las cookies son una solución diferente al problema de conservación del estado entre varias transacciones al tiempo que se mantiene un URL de aspecto limpio.

Qué es una cookie

Una *cookie* es un pequeño fragmento de información que las secuencias de comandos pueden almacenar en un equipo del lado del cliente. Puede definir una cookie en el equipo de un usuario si envía un encabezado HTTP que contenga datos con el siguiente formato:

```
Set-Cookie:  NOMBRE=VALOR;  [expires=FECHA;]  [path=RUTA;]
  [domain=NOMBRE_DOMINIO;]  [secure]
```

De esta forma se crea una cookie con el nombre *NOMBRE* y con el valor *VALOR*. El resto de parámetros son opcionales. El campo `expires` establece la fecha hasta la que será válida la cookie (si no se establece una fecha de caducidad, la cookie será permanente a menos que la eliminemos manualmente). `path` y `domain` se pueden utilizar de forma conjunta para especificar el o los URL para los que se utiliza la

cookie. La palabra clave `secure` significa que la cookie no se enviará a través de una conexión HTTP.

Cuando un navegador se conecta a un URL, primero busca las cookies almacenadas localmente. Si ninguna de ellas es relevante para el URL al que se ha conectado, se devuelven al servidor.

Configuración de cookies desde HTTP

Podemos configurar cookies manualmente en PHP por medio de la función `setcookie()`. Su sintaxis es la siguiente:

```
int setcookie (string nombre [, string valor [, int caducidad [,
string ruta
[, string dominio [, int secure]]]]])
```

Los parámetros se corresponden exactamente a los del encabezado Set-Cookie mencionado anteriormente. Si define una cookie como

```
setcookie ('mycookie', 'value');
```

cuando el usuario visite la siguiente página de su sitio (o vuelva a cargar la página actual), podremos acceder a la cookie a través de `$_COOKIE['mycookie']` o de `$HTTP_COOKIE_VARS["mycookie"]` (o, si ha activado `register_globals`, directamente como `$myccokie`).

Se puede eliminar una cookie si se invoca `setcookie()` de nuevo con el mismo nombre de cookie y una fecha de caducidad pasada. También se puede definir manualmente una cookie con ayuda de la función `Header()` y la sintaxis de cookie que mostramos anteriormente. Debe saber que es necesario enviar los encabezados de cookie *antes que ningún otro encabezado* o no funcionarán (se trata de una limitación de las cookies más que una limitación de PHP).

Uso de cookies con sesiones

Las cookies tienen algunos problemas asociados como por ejemplo que algunos navegadores no las aceptan o que algunos usuarios pueden desactivarlas en sus navegadores. Es una de las razones por las que las sesiones PHP utilizan un método cookie/URL dual (como veremos en breve). Al utilizar sesiones PHP, no tendremos que definir las cookies manualmente. Las funciones de sesión se encargan de ello por nosotros.

Podemos utilizar la función `session_get_cookie_params()` para ver los contenidos de una cookie definida por el control de sesión. Devuelve una matriz asociativa que contiene los elementos `lifetime`, `path` y `domain`.

También podemos utilizar

```
session_set_cookie_params($lifetime, $path, $domain [, $secure]);
```

para definir los parámetros de la cookie de sesión.

Si necesita más información sobre cookies, puede consultar la especificación sobre cookies en el sitio de Netscape, `http://home.netscape.com/newsref/std/cookie_spec.html`.

(Puede ignorar el hecho de que este documento se denomina "especificación preliminar", ya que ha sido así desde 1995).

Almacenamiento del Id. de sesión

De forma predeterminada, PHP utiliza cookies con sesiones. Si existe la posibilidad, se definirá una cookie para almacenar el Id. de sesión.

El otro método que se puede utilizar consiste en añadir el Id. de sesión al URL. Podemos indicar que se realice automáticamente si compilamos PHP con la opción `--enable-trans-sid`. Se realiza de forma predeterminada desde PHP 4.2.

También podemos incrustar manualmente el Id. de sesión en enlaces para que se pase de esta forma. El Id. de sesión se almacena en la constante SID. Para pasarlo manualmente, debe añadirlo al final de un enlace similar a un parámetro GET:

```
<a  href="link.php?<?=SID?>">
```

Normalmente resulta más sencillo compilar con `--enable-trans-sid`, siempre que sea posible.

Implementación de sesiones simples

Los pasos básicos del uso de sesiones son los siguientes:

- Inicio de sesión
- Registro de variables de sesión
- Uso de variables de sesión
- Anulación de variables registradas y eliminación de la sesión

Estos pasos no siempre tienen lugar al mismo tiempo en la misma secuencia de comandos y algunos de ellos pueden tener lugar en varias secuencias. A continuación analizaremos cada uno de los pasos individualmente.

Inicio de sesión

Antes de poder utilizar la funcionalidad de sesiones, será necesario iniciar una sesión. Hay tres formas de hacerlo.

La primera, y la más sencilla, consiste en iniciar una secuencia de comandos con una llamada a la función `session_start()`:

```
session_start();
```

Esta función comprueba si ya hay un Id. de sesión actual. En caso contrario, creará uno. Si ya existe el Id., básicamente carga las variables de sesión registradas para que podamos utilizarlas. Es aconsejable invocar `session_start()` al inicio de todas las secuencias de comandos que utilicen sesiones. La segunda técnica consiste en iniciar una sesión al intentar registrar una variable de sesión por medio de `session_register()` (como veremos en el siguiente apartado).

La tercera forma de iniciar una sesión consiste en configurar PHP para que lo haga automáticamente cuando alguien visite el sitio. Puede hacerlo por medio de la opción `session.auto_start` en el archivo `php.ini`, que comentaremos al analizar la configuración.

Registro de variables de sesión

El registro de variables de sesión ha cambiado últimamente en PHP. En PHP 4.1, las variables de sesión se almacenaban en la matriz superglobal `$_SESSION` y también en la antigua `$HTTP_SESSION_VARS`. Para poder crear una variable de sesión, basta con definir un elemento en una de estas matrices, como se indica a continuación:

```
$_SESSION['myvar'] = 5;
```

o

```
$HTTP_SESSION_VARS['myvar'] = 5;
```

Si utiliza una versión anterior de PHP o si ha activado `register_globals`, para poder realizar el seguimiento de una variable desde una secuencia de comandos a otra, puede registrarla con una llamada a `session_register()`. Por ejemplo, para registrar la variable `$myvar`, podría utilizar el siguiente código:

```
$myvar = 5;
session_register('myvar');
```

Fíjese en que necesita pasar a `session_register()` una cadena que contenga el nombre de la variable. No hace falta que esta cadena incluya el símbolo `$`.

De esta forma se registra la variable y se realiza el seguimiento de su valor. El seguimiento de la variable se realiza hasta que finalice la sesión o hasta que se anule su registro manualmente. Se puede registrar más de una variable al mismo tiempo si se utiliza una lista de nombres de variable separadas por comas, como por ejemplo:

```
session_register('myvar1', 'myvar2');
```

Si utiliza las matrices `$_SESSION` o `$HTTP_SESSION_VARS`, no pruebe ni utilice la función `session_register()`.

Uso de variables de sesión

Para poder introducir una variable de sesión en ámbito y utilizarla, primero debe iniciar una sesión. Tras ello podrá acceder a la variable de través de las matri-

ces $_SESSION o $HTTP_SESSION_VARS como por ejemplo $HTTP_SESSION_VARS['myvar']. Si ha activado register_globals, podrá acceder a ella a través del nombre corto, como por ejemplo $myvar.

Si ha activado register_globals, recuerde que una variable de sesión no se puede reemplazar con datos GET o POST, una buena opción de seguridad pero que debe recordar al diseñar el código.

Por otra parte, tendrá que prestar especial atención cuando compruebe si se han configurado las variables de sesión (por ejemplo a través de isset() o empty()). Recuerde que las variables las puede configurar el usuario a través de GET o POST. Puede comprobar una variable para ver si se trata de una variable de sesión registrada si invoca la función session_is_registered() de la siguiente forma:

```
$result = session_is_registered('myvar');
```

De esta forma comprueba si $myvar es una variable de sesión registrada y devuelve true o false. Si utiliza $_SESSION o $HTTP_POST_VARS NO debería utilizar la función session_is_registered(). Puede comprobar directamente si los elementos de la matriz están definidos utilizando por ejemplo:

```
if (isset($HTTP_SESSION_VARS['myvar'])) ...
```

Anulación del registro de variables y eliminación de la sesión

Cuando haya terminado con una variable de sesión puede anular su registro.

Si utiliza las matrices $_SESSION o $HTTP_SESSION_VARS, puede hacerlo directamente como se indica a continuación:

```
unset($HTTP_SESSION_VARS['myvar']);
```

Si ha activado register_globals, tendrá que borrar las variables de sesión por medio de la función session_unregister():

```
session_unregister("myvar");
```

Como en el caso anterior, esta función requiere el nombre de la variable que queremos anular en forma de cadena sin el símbolo $. Esta función sólo puede anular el registro de una variable por vez (al contrario que session_register()). Sin embargo, podemos utilizar session_unset() para anular el registro de todas las variables de sesión actuales.

No intente utilizar la función session_unregister() si está utilizando $_SESSION o $HTTP_SESSION_VARS directamente.

Cuando haya terminado con una sesión, primero debería anular el registro de todas las variables y, tras ello, invocar

```
session_destroy();
```

para borrar el Id. de sesión.

Ejemplo de sesión sencilla

Seguramente todo esto le parezca ligeramente abstracto, por lo que veremos un ejemplo. Implementaremos un conjunto de tres páginas.

En la primera página, iniciaremos una sesión y registraremos la variable `$HTTP_SESSION_VARS ['sess_var']`. El código para realizar esta operación se muestra en el listado 20.1

Listado 20.1. page1.php. Inicio de una sesión y registro de una variable

```
<?php
   session_start();

   $HTTP_SESSION_VARS['sess_var'] = "Hello world!";

   echo 'El contenido de $HTTP_SESSION_VARS[\'sess_var\'] es '
        .$HTTP_SESSION_VARS['sess_var'].'<br />';
?>
  <a href="page2.php">Next page</a>
```

Hemos registrado la variable y definido su valor. El resultado de esta secuencia de comandos se muestra en la figura 20.1.

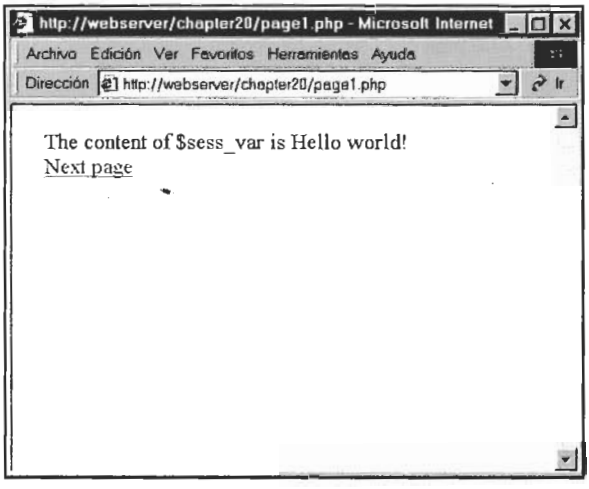


Figura 20.1. Valor inicial de la variable de sesión mostrada por page1.php

El valor *final* de la variable de la página es el que estará disponible en las páginas posteriores.

Al final de la secuencia de comandos, se *serializa* o congela la variable de sesión, hasta que se vuelva a cargar por medio de la siguiente llamada a `session_start()`.

Por esta razón comenzaremos la siguiente secuencia de comandos con la invocación de `session_start()`. Lo mostramos en el listado 20.2.

Listado 20.2. page2.php; Acceso a una variable de sesión y anulación de su registro

```
<?php
   session_start();

   echo 'El contenido de $HTTP_SESSION_VARS[\'sess_var\'] es '
        .$HTTP_SESSION_VARS['sess_var'].'<br />';

   unset($HTTP_SESSION_VARS['sess_var']);
?>
  <a href="page3.php">Next page</a>
```

Después de invocar `session_start()`, la variable `$HTTP_SESSION_VARS ['sess_var']` está disponible con el valor almacenado previamente, como puede comprobar en la figura 20.2.

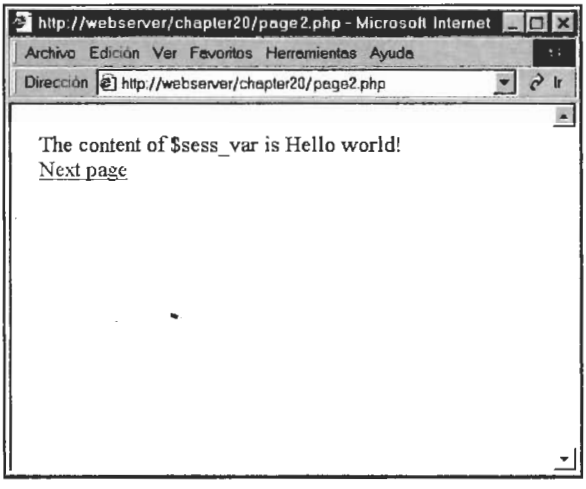


Figura 20.2. El valor de la variable de sesión se ha pasado junto el Id. de sesión a pag2.php

Una vez utilizada la variable, anulamos su definición. La sesión sigue existiendo pero la variable `$HTTP_SESSION_VARS ['sess_var']` ya no es una variable registrada. Por último, pasamos a la página `page3.php`, la última secuencia de comandos de nuestro ejemplo. El correspondiente código se incluye en el listado 20.3.

Listado 20.3. page3.php: Finalización de la sesión

```
<?php

   session_start();
```

```
    echo 'El contenido de $HTTP_SESSION_VARS[\'sess_var\'] es '
         .$HTTP_SESSION_VARS['sess_var'].'<br />';

    session_destroy();
?>
```

Como puede comprobar en la figura 20.3, ya no tenemos acceso al valor de `$HTTP_SESSION_VARS['sess_var']`.

En algunas versiones de PHP anteriores a la 4.3 se producía un error al intentar anular elementos de `$HTTP_SESSION_VARS` o `$_SESSION`. Si no puede anular elementos (es decir, si permanecen configurados), puede intentar utilizar `session_unregister()` para borrar estas variables.

El uso de `session_unregister()` no es recomendable, pero si quiere que su código funcione en todas las versiones de PHP4, es la única opción de la que dispone.

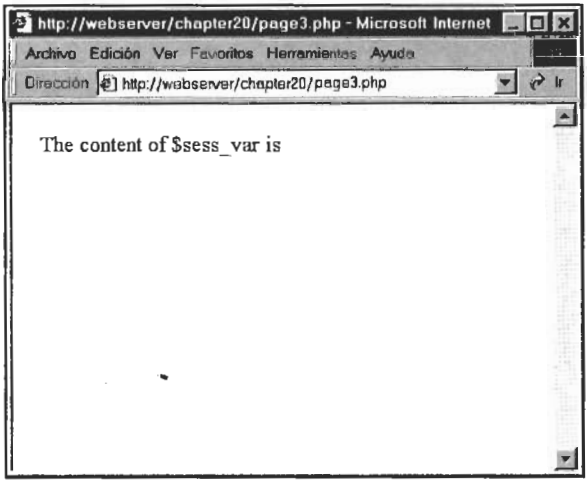


Figura 20.3. La variable de la que hemos anulado el registro ya no está disponible

Para finalizar invocamos `session_destroy()` para eliminar el Id. de la sesión.

Configuración del control de sesión

Existe un conjunto de opciones de configuración para sesiones que puede definir en su archivo `php.ini`.

Algunas de las más útiles, así como una descripción de las mismas, se incluyen en la tabla 20.1

Tabla 20.1. Opciones de configuración de sesión

Nombre de la opción	Predeterminada	Efecto
`session.auto_start`	0 (desactivada)	Inicia sesiones automáticamente.
`session.cache_expire`	180	Define la duración de las páginas de sesión en caché, en minutos.
`session.cookie_domain`	nada	Dominio que se configura en la cookie de sesión.
`session.cookie_lifetime`	0	Duración de la cookie de Id. de sesión en el equipo del usuario. El valor predeterminado, 0, indica que dura hasta que se cierre el navegador.
`session.cookie_path`	/	Ruta que se configura en la cookie de sesión.
`session.name`	PHPESSID	Nombre de la sesión que se utiliza como nombre de la cookie en el sistema del usuario.
`session.save_handler`	archivos	Define dónde se almacenan los datos de la sesión. Puede indicar que se trate de una base de datos, pero tendrá que escribir sus propias funciones.
`session.save_path`	/tmp	Ruta en la que se almacenan los datos de la sesión. De forma más general, el argumento pasado al almacén procesado y definido por `session.save_handler`.
`session.use_cookies`	1 (activado)	Configura sesiones para que utilicen cookies en el lado del cliente.

Implementación de autenticación con el control de sesión

Por último, veremos un ejemplo más elaborado sobre el uso del control de sesión.

Probablemente el uso más habitual del control de sesión consiste en realizar el seguimiento de los usuarios una vez autenticados a través de un mecanismo de inicio de sesión. En este ejemplo combinaremos la autenticación de una base de datos MySQL con el uso de sesiones para conseguir esta funcionalidad. De esta forma asentaremos las bases del proyecto que analizaremos en un capítulo posterior sobre

autenticación y personalización de usuarios, y que podremos volver a utilizar en otros proyectos. También utilizaremos de nuevo la base de datos de autenticación que definimos en un capítulo anterior para utilizar `mod_auth_msyql`.

El ejemplo cuenta con tres sencillas secuencias de comandos. La primera, `authmain.php`, incluye un formulario de registro y autenticación para miembros de nuestro sitio Web. La segunda, `members_only.php`, muestra información solamente a los miembros que se hayan conectado correctamente. La tercera, `logout.php`, permite anular el registro de un miembro.

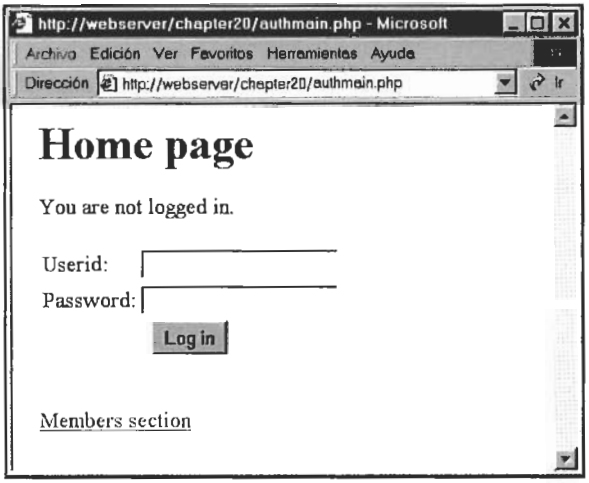


Figura 20.4. Como el usuario no se ha conectado, se muestra una página de inicio de sesión



Figura 20.5. Los usuarios no registrados no pueden acceder a los contenidos del sitio; recibirán este mensaje

Para comprender cómo funciona, fíjese en la figura 20.4, la página inicial que muestra `authmain.php`.

Esta página permite al usuario conectarse. Si intenta acceder a la sección de miembros sin registrarse primero, obtendrá el mensaje que se muestra en la figura 20.5.

Sin embargo, si el usuario se conecta primero (con username:testuser y password:test123, como definimos en un capítulo anterior) y tras ello intenta acceder a la página de miembros, obtendrá el resultado que mostramos en la figura 20.6.

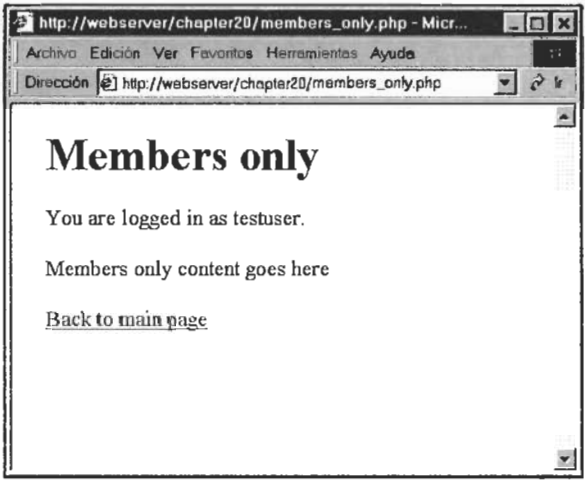


Figura 20.6. Después de conectarse, el usuario puede acceder a la zona de miembros

Veamos el código de esta aplicación. La mayor parte del mismo se concentra en `authmain.php`, que se muestra en el listado 20.4.

Lo analizaremos paso a paso.

Listado 20.4. authmain.php: Parte principal de la aplicación de autenticación

```
<?php
session_start();

if  (isset($HTTP_POST_VARS['userid'])  &&
isset($HTTP_POST_VARS['password']))
{
   // si el usuario acaba de intentar conectarse
   $userid  =  $HTTP_POST_VARS['userid'];
   $password  =  $HTTP_POST_VARS['password'];

   $db_conn  =  mysql_connect('localhost',  'webauth',  'webauth');
   mysql_select_db('auth',  $db_conn);
   $query  =  'select  *  from auth  '
            ."where name='$userid' "
            ." and pass=password('$password')";
   $result  =  mysql_query($query, $db_conn);
```

```
    if (mysql_num_rows($result) >0 )
    {
       // si están en la base de datos, registre el Id. de usuario
       $HTTP_SESSION_VARS['valid_user'] = $userid;
    }
}
?>
 <html>
 <body>
 <h1>Home page</h1>
 <?
    if (isset($HTTP_SESSION_VARS['valid_user']))
    {
       echo 'You are logged in as: '.$HTTP_SESSION_VARS['valid_user'].' <br />';
       echo '<a href="logout.php">Log out</a><br />';
    }
    else
    {
       if (isset($userid))
       {
          // si han intentado conectarse y no lo han conseguido
          echo 'Could not log you in';
       }
       else
       {
          // todavía no han intentado conectarse o se han desconectado
          echo 'You are not logged in.<br />';
       }

       // proporcione un formulario para que se conecten
       echo '<form method="post" action="authmain.php">';
       echo '<table>';
       echo '<tr><td>Userid:</td>';
       echo '<td><input type="text" name="userid"></td></tr>';
       echo '<tr><td>Password:</td>';
       echo '<td><input type="password" name="password"></td></tr>';
       echo '<tr><td colspan="2" align="center">';
       echo '<input type="submit" value="Log in"></td></tr>';
       echo '</table></form>';
    }
?>
<br>
<a href="members_only.php">Members section</a>
</body>
</html>
```

Esta secuencia de comandos incluye, por alguna razón, una compleja lógica, ya que muestra el formulario de inicio de sesión, también la acción para el mismo y contiene HTML para intentos de inicio de sesión satisfactorios y fallidos. Las actividades del formulario dependen de la variable de sesión `valid_user`. La idea básica es que si un usuario se conecta satisfactoriamente, registraremos una variable de sesión con el nombre `$HTTP_SESSION_VARS['valid_user']` que contenga el Id. del usuario.

Lo primero que hacemos en esta secuencia de comandos es invocar `session_start()`. De esta forma se carga la variable de sesión `valid_user` si se ha registrado.

En la primera pasada por la secuencia de comandos, no se aplica ninguna de las condiciones `if` y el usuario continúa hasta el final de la misma, donde le indicaremos que no se ha conectado y le ofreceremos un formulario para que lo haga:

```
echo '<form method="post" action="authmain.php">';
echo '<table>';
echo '<tr><td>Userid:</td>';
echo '<td><input type="text" name="userid"></td></tr>';
echo '<tr><td>Password:</td>';
echo '<td><input type="password" name="password"></td></tr>';
echo '<tr><td colspan="2" align="center">';
echo '<input type="submit" value="Log in"></td></tr>';
echo '</table></form>';
```

Cuando pulse el botón de envío del formulario, se vuelve a invocar esta secuencia de comandos y empezamos de nuevo desde el principio. En esta ocasión, tenemos un Id. de usuario y una contraseña que autenticar, almacenados como `$HTTP_POST_VARS['userid']` y `$HTTP_POST_VARS['password']`. Si estas variables están configuradas, pasamos al bloque de autenticación:

```
if (isset($HTTP_POST_VARS['userid']) && isset($HTTP_POST_VARS['password']))
{
   // si el usuario acaba de intentar conectarse
   $userid = $HTTP_POST_VARS['userid'];
   $password = $HTTP_POST_VARS['password'];

   $db_conn = mysql_connect('localhost', 'webauth', 'webauth');
   mysql_select_db('auth', $db_conn);
   $query = 'select * from auth '
            ."where name='$userid' "
            ." and pass=password('$password')";
   $result = mysql_query($query, $db_conn);
```

Nos conectamos a una base de datos MySQL y comprobamos el Id. de usuario y la contraseña. Si hay una combinación que coincida en la base de datos, creamos la variable `$HTTP_SESSION_VARS['valid_user']` que contiene el Id. de usuario de este usuario en concreto, por lo que a partir de ahora sabremos quién está conectado.

```
   if (mysql_num_rows($result) >0 )
   {
      // si se encuentran en la base de datos, registre el Id. de usuario
      $HTTP_SESSION_VARS['valid_user'] = $userid;
   }
}
```

Como ahora sabemos de quién se trata, no es necesario volver a mostrar el formulario de inicio de sesión.

En su lugar, le indicamos que ya sabemos quién es y le ofrecemos la opción de desconectarse:

```
if (isset($HTTP_SESSION_VARS['valid_user']))
{
   echo 'You are logged in as: '.$HTTP_SESSION_VARS['valid_user'].' <br />';
   echo '<a href="logout.php">Log out</a><br />';
}
```

Si intentamos conectar al usuario y, por alguna razón, no lo conseguimos, tendremos un Id. de usuario pero no una variable `HTTP_SESSION_VARS['valid_user']`, por lo que podremos ofrecerle un mensaje de error:

```
if (isset($userid))
{
   // si han intentado conectarse pero no lo han conseguido
   echo 'Could not log you in';
}
```

Eso es todo para la secuencia de comandos principal.

Seguidamente, analizaremos la página de miembros, cuyo código mostramos en el listado 20.5.

Listado 20.5. members_only.php. Código para la sección de miembros de nuestro sitio Web para comprobar la validez de los usuarios

```
<?php
   session_start();

   echo '<h1>Members only</h1>';

   // compruebe la variable de sesión

   if (isset($HTTP_SESSION_VARS['valid_user']))
   {
      echo '<p>You are logged in as '.$HTTP_SESSION_VARS['valid_user'].'</p>';
      echo '<p>Members only content goes here</p>';
   }
   else
   {
      echo '<p>You are not logged in.</p>';
      echo '<p>Only logged in members may see this page.</p>';
   }

   echo '<a href="authmain.php">Back to main page</a>';
?>
```

Este código es muy sencillo. Todo lo que hace es iniciar una sesión y comprobar si la sesión actual contiene un usuario registrado.

Para ello comprueba si se ha definido el valor de `$HTTP_SESSION_VARS['valid_user']`.

Si el usuario se ha conectado, le mostramos los contenidos de los miembros; en caso contrario le indicamos que no está autorizado.

Por último, la secuencia de comandos `logout.php` permite desconectar a un usuario del sistema.

El código correspondiente se incluye en el listado 20.6.

Listado 20.6. logout.php. Esta secuencia de comandos anula el registro de la variable de sesión y elimina la sesión

```
<?php
   session_start();

   $old_user = $HTTP_SESSION_VARS['valid_user'];  // almacene para
   //probar si "estaban" conectados
   unset($HTTP_SESSION_VARS['valid_user']);
   session_destroy();
?>
<html>
<body>
<h1>Log out</h1>
<?php
   if (!empty($old_user))
   {
      echo 'Logged out.<br />';
   }
   else
   {
      // si no estaban conectados pero accedieron de algún modo a esta página
      echo 'You were not logged in, and so have not been logged out.<br />';
   }
?>
<a href="authmain.php">Back to main page</a>
</body>
</html>
```

El código es muy sencillo pero lo analizaremos brevemente. Iniciamos una sesión, almacenamos el antiguo nombre de usuario del usuario, anulamos la variable `valid_user` y destruimos la sesión.

Tras ello, mostramos al usuario un mensaje que será diferente si se ha desconectado o no se ha conectado desde un principio.

Este sencillo conjunto de secuencias de comandos constituye la base de gran parte del trabajo que desarrollaremos en capítulos posteriores.

Lecturas adicionales

Las sesiones son una de las novedades de PHP4, pero PHPLib las ofrece desde hace tiempo.

Si necesita más información al respecto, las mejores fuentes son la página de PHPLib y la especificación sobre cookies.

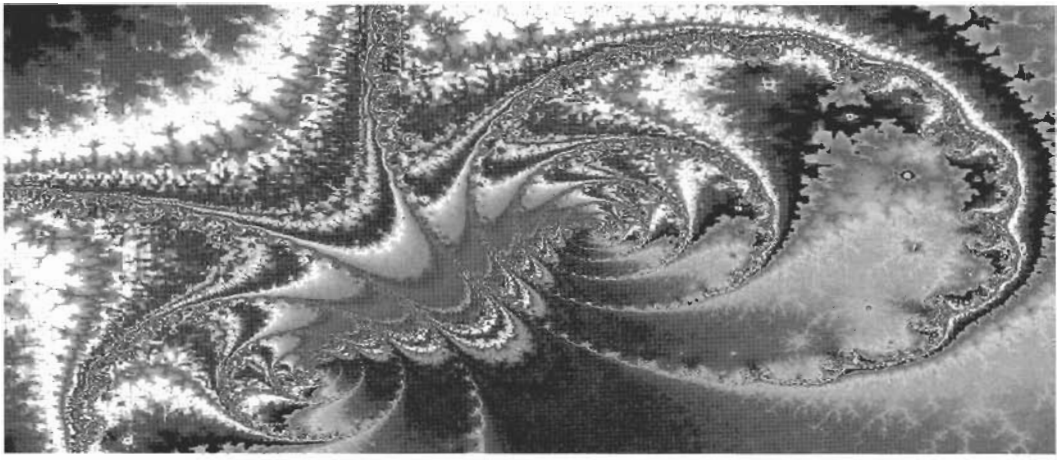
Ya mencionamos estos URL anteriormente en el capítulo, pero los volvemos a incluir para que los utilice de referencia:

```
http://phplib.sourceforge.net/
http://home.netscape.com/newsref/std/cookie_spec.html
```

A continuación

Prácticamente hemos terminado con esta parte del libro.

Antes de pasar a los proyectos, analizaremos brevemente algunos de los aspectos más útiles de PHP que no hemos mencionado hasta el momento.



21
Òtras características útiles

Algunas de las funciones y características útiles de PHP no encajan en una categoría concreta, razón por la que las describiremos en este capítulo.

Analizaremos los siguientes aspectos:

- Uso de comillas mágicas
- Evaluación de cadenas con `eval()`
- Finalización de ejecución: `die` y `exit`
- Serialización
- Obtención de información sobre el entorno PHP
- Modificación temporal del entorno de ejecución
- Carga de extensiones PHP
- Resalte de fuentes

Uso de comillas mágicas

Probablemente se haya fijado en que debe prestar especial atención al utilizar el símbolo de comillas (' y ") y las barras invertidas (\ ) dentro de una cadena. PHP se puede confundir si utilizamos una instrucción de cadena como la siguiente:

```
echo "color = "#FFFFFF"";
```

y generará un error de análisis. Para incluir comillas en una cadena, utilizaremos un tipo de comillas distinto al que se emplea para delimitar la cadena. Por ejemplo:

```
echo "color = '#FFFFFF'";
```

o

```
echo 'color = "#FFFFFF"';
```

son dos posibilidades válidas.

El mismo problema se produce con las entradas de usuario, así como con entradas o salidas de o hacia otros programas.

Si intentamos ejecutar una consulta como

```
insert into company values ('Bob's Auto Parts');
```

se producirá una confusión similar en el analizador de MySQL.

Ya hemos visto el uso de `addslashes()` y `stripslashes()` que escapan cualquier comilla simple, comilla doble, barra invertida y caracteres NUL.

PHP dispone de una opción muy útil que añade o elimina barras invertidas automática o mágicamente por nosotros. Con dos parámetros en nuestro archivo `.ini` podemos activar o desactivar las comillas mágicas para GET, POST, datos de cookies y otras fuentes.

El valor de la directiva `magic_quotes_gpc` controla si se utilizan las comillas mágicas en operaciones GET, POST y de cookies.

Al activar `magic_qoutes_gpc`, si alguien escribe `"Bob's Auto Parts"` en un formulario de nuestro sitio, nuestra secuencia de comandos recibirá `"Bob\'s Auto Parts"` porque a la comilla se le aplica conversión de escape.

La función `get_magic_quotes_gpc()` devuelve 1 ó 0, para indicarnos el valor actual de `magic_quotes_gpc`. Esto resulta de gran utilidad para probar si necesitamos aplicar `stripslashes()` a datos recibidos de un usuario.

El valor de `magic_quotes_runtime` controla si las funciones que obtienen datos de bases de datos y de archivos utilizan las comillas mágicas.

Para obtener el valor de `magic_quote_runtime`, utilice la función `get_magic_quotes_runtime()`, que devuelve 1 ó 0. Las comillas mágicas se pueden desactivar en una secuencia de comandos concreta por medio de la función `set_magic_quotes_runtime()`.

Evaluación de cadenas: eval()

La función `eval()` evalúa cadenas como si se tratara de código PHP.

Por ejemplo:

```
eval ( "echo 'Hello World';" );
```

adopta los contenidos de la cadena y los ejecuta. Esta línea genera el mismo resultado que

```
echo 'Hello World';
```

En determinados casos `eval()` puede resultar muy útil.

Puede que le interese almacenar bloques de código en una base de datos y recuperarlos y aplicarles `eval()` posteriormente.

Puede que quiera generar código en un bucle y, tras ello, utilizar `eval()` para ejecutarlo.

Puede también utilizar `eval()` para actualizar o corregir código existente. Si dispone de gran cantidad de secuencias de comandos en las que es necesario aplicar un cambio predecible, podría (aunque no sería muy eficaz) escribir una secuencia de comandos que cargue una secuencia de comandos antigua en una cadena, ejecute un `regexp` para realizar los cambios y tras ello utilice `eval()` para ejecutar la secuencia de comandos modificada.

Incluso podemos pensar que haya alguien de mucha confianza en alguna parte que quiera permitir la entrada y ejecución de código PHP en su navegador.

Finalización de ejecución: die y exit

Hasta ahora hemos utilizado la instrucción `exit` para detener la ejecución de una secuencia de comandos. Como recordará, aparece en su propia línea, como mostramos a continuación:

```
exit;
```

No devuelve nada. También podemos utilizar su alias `die()`.

Si necesitamos una finalización más útil, podemos pasar un parámetro a `exit()`, que podemos utilizar para mostrar un mensaje de error o para ejecutar una función antes de finalizar una secuencia de comandos. Seguramente le resulte familiar a los programadores de Perl.

Por ejemplo:

```
exit('Script ending now');
```

Normalmente se utiliza `or` con una instrucción que puede fallar, como por ejemplo al abrir un archivo o conectarse a una base de datos:

```
mysql_query($query) or die('Could not execute query');
```

En lugar de imprimir simplemente un mensaje de error, puede invocar una última función antes de que finalice la secuencia de comandos:

```
{
    echo 'MySQL error was: ';
```

```
    echo mysql_error();
}

mysql_query($query) or die(err_msg());
```

Puede resultar muy útil para indicarle al usuario la razón por la que ha fallado la secuencia de comandos. También podríamos enviarnos un correo electrónico a nosotros mismos para saber si se ha producido un error grave o añadir errores a un archivo de registro.

Serialización

La serialización es el proceso de convertir todo lo que se pueda almacenar en una variable PHP o en un objeto en un flujo de bytes para almacenarlo en una base de datos o pasarlo a través de un URL de una página a otra. Sin esto, resultaría muy complicado almacenar o pasar todos los contenidos de una matriz o de un objeto.

Desde la aparición del control de sesión, su utilidad se ha visto reducida. La serialización de datos se aplica principalmente al tipo de cosas para las que ahora utilizamos el control de sesión. De hecho, las funciones de control de sesión serializan variables de sesión para almacenarlas entre solicitudes PHP.

Sin embargo, puede que le interese almacenar un objeto o matriz PHP en un archivo o en un documento. Si es su intención, hay dos funciones que debe conocer: `serialize()` y `unserialize()`.

Puede invocar la función `serialize()` de esta forma:

```
$serial_object = serialize($my_object);
```

Si quiere saber lo que realmente hace la serialización, basta con fijarse en lo que devuelve la función. Convierte los contenidos de un objeto o de una matriz en una cadena. Por ejemplo, fijémonos en el resultado de la ejecución de `serialize()` en un sencillo objeto de empleado, definido e instanciado de esta forma:

```
class employee
{
   var $name;
   var $employee_id;
};

$this_emp = new employee;
$this_emp->name = 'Fred';
$this_emp->employee_id = 5324;
```

Si lo serializamos y lo duplicamos en un navegador, el resultado será

```
O:8:"employee":2:{s:4:"name";s:4:"Fred";s:11:"employee_id";i:5324;}
```

Es bastante sencillo ver la relación entre los datos del objeto original y los datos serializados.

Como los datos serializados son sólo texto, podemos escribirlos en una base de datos o donde queramos. Sepa que es necesario aplicar `addslashes()` a cualquier dato antes de escribirlo en una base de datos, como de costumbre. Apreciará la necesidad de esto si se fija en las comillas de la cadena serializada anteriormente.

Para recuperar el objeto, se invoca `unserialize()`:

```
$new_object = unserialize($serial_object);
```

Evidentemente, si ha invocado `addslashes()` antes de incluir el objeto en la base de datos, tendrá que invocar `stripslashes()` antes de deserializar la cadena.

Otro aspecto que conviene destacar al serializar clases o al utilizarlas como variables de sesión es que PHP necesita conocer la estructura de una clase antes de poder volver a instanciarla. Por ello, será necesario incluir el archivo de definición de clase antes de invocar `session_start()` o `unserialize()`.

Cómo obtener información sobre el entorno PHP

Existen distintas funciones que podemos utilizar para encontrar información sobre la configuración de PHP.

Cómo saber qué extensiones se han cargado

Es muy sencillo saber qué conjuntos de funciones están disponibles y qué funciones están disponibles en cada uno de dichos conjuntos. Basta con utilizar las funciones `get_loaded_extensions()` y `get_extension_funcs()`.

La función `get_loaded_extensions()` devuelve una matriz de todos los conjuntos de funciones disponibles actualmente para PHP. Por su parte, si proporcionamos el nombre de una determinado conjunto de funciones o una extensión, `get_extensions_funcs()` devuelve una matriz de las funciones de dicho conjunto.

El código del listado 21.1 enumera todas las funciones disponibles para su instalación PHP al utilizar estas dos funciones.

Listado 21.1. list_functions.php. Esta secuencia de comandos enumera todas las extensiones disponibles para PHP y en cada extensión incluye una lista de las funciones de dicha extensión

```
<?php
   echo 'Conjuntos de funciones admitidos en esta instalación:<br />';
   $extensions = get_loaded_extensions();
   foreach ($extensions as $each_ext)
   {
      echo "$each_ext <br />";
      echo '<ul>';
      $ext_funcs = get_extension_funcs($each_ext);
```

```
        foreach($ext_funcs as $func)
        {
            echo "<li> $func </li>";
        }
        echo '</ul>';
    }
?>
```

Fíjese en que la función `get_loaded_extensions()` no adopta ningún parámetro y que la función `get_extension_funcs()` adopta como único parámetro el nombre de la extensión.

Esta información puede resultar muy útil para saber si hemos instalado satisfactoriamente una extensión.

Identificación del propietario de la secuencia de comandos

Podemos descubrir el usuario propietario de la secuencia de comandos en ejecución con una llamada a la función `get_current_user()`, como se muestra a continuación:

```
echo get_current_user();
```

En ocasiones puede resultar muy útil para resolver problemas relacionados con los permisos.

Cómo saber cuándo se ha modificado una secuencia de comandos

Una operación muy habitual consiste en añadir una última fecha de modificación a cada una de las páginas de un sitio.

Para comprobar la última fecha de modificación de una secuencia de comandos basta con utilizar la función `getlastmod()` (el nombre de la función no incluye un guión bajo) de esta forma:

```
echo date('g:i a, j M Y',getlastmod());
```

La función devuelve una marca de tiempo de Unix que podemos añadir a `date()` como hemos hecho en el ejemplo para generar una fecha legible.

Carga dinámica de extensiones

Se pueden cargar bibliotecas de extensiones dinámicas durante el tiempo de ejecución, si no están compiladas, por medio de la función `dl()`.

Esta función espera como parámetro el nombre del archivo que contiene la biblioteca. En Unix, se trata de nombres de archivo que terminan en `.so`; en Windows, terminan en `.dll`.

Un ejemplo de una invocación de `dl()` sería el siguiente:

```
dl('php_ftp.dll');
```

De esta forma se carga dinámicamente la extensión FTP (en un equipo bajo Windows).

No es necesario especificar el directorio en el que se encuentra el archivo; basta con configurarlo en el archivo `php.ini`. Una directiva denominada `extension_dir` especificará el directorio en el que PHP buscará bibliotecas para cargarlas dinámicamente.

Si tiene problemas para cargar extensiones dinámicamente, debe comprobar si en su archivo `php.ini` se incluye la directiva `enable_dl`. Si está desactivada, no podrá cargar extensiones dinámicamente.

Puede que si el ordenador con el que trabaja no es suyo, esté desactivada por motivos de seguridad.

No podrá utilizar `dl()` si PHP se ejecuta en modo seguro.

Modificación temporal del entorno de ejecución

Puede ver las directivas configuradas en el archivo `php.ini` o cambiarlas mientras dure una determinada secuencia de comandos, lo que por ejemplo puede resultar muy útil junto con la directiva `max_execution_time` si sabemos que la secuencia de comandos tardará en ejecutarse.

Podemos acceder y modificar las directivas por medio de las funciones `ini_get()` e `ini_set()`. En el listado 21.2 se incluye una sencilla secuencia de comandos que utiliza estas funciones.

Listado 21.2. inter.php. Esta secuencia de comandos restablece variables del archivo php.ini

```
<?php
   $old_max_execution_time = ini_set('max_execution_time', 120);
   echo "old timeout is $old_max_execution_time <br />";

   $max_execution_time = ini_get('max_execution_time');
   echo "new timeout is $max_execution_time <br />";
?>
```

La función `ini_set()` adopta dos parámetros. El primero es el nombre de la directiva de configuración de `php.ini` que queremos cambiar y el segundo, el valor por el que la cambiaremos. Devuelve el valor anterior de la directiva.

En este caso, restablecemos el valor de duración máximo predeterminado de una secuencia de comandos, 30 segundos, para que se ejecute durante 120 segundos.

La función `ini_get()` simplemente comprueba el valor de una determinada directiva de configuración. El nombre de la directiva se debe pasar a la misma en forma de cadena. En este caso sólo lo utilizamos para comprobar que el valor ha cambiado realmente.

Cómo resaltar fuentes

PHP incorpora un resaltador de sintaxis similar a muchos IDE. En concreto, resulta muy útil para compartir código con terceros o presentarlo en una página Web para su análisis.

Las funciones `show_source()` y `highlight_file()` son idénticas (de hecho la primera es un alias de la segunda).

Ambas funciones aceptan como parámetro un nombre de archivo (archivo que debe ser un archivo PHP ya que, en caso contrario, el resultado que obtendremos no tendrá mucho sentido). Por ejemplo,

```
show_source('list_functions.php');
```

El archivo se duplicará en el navegador con un texto resaltado en varios colores en función de si se trata de un comentario, una cadena, una palabra clave o HTML. El resultado se imprime sobre un color de fondo. El contenido que no encaja en ninguna de estas categorías se imprime en el color predeterminado.

La función `highlight_string()` funciona del mismo modo pero adopta como parámetro una cadena que imprime en el navegador con un formato de sintaxis resaltado.

Podemos definir los colores para el resalte de la sintaxis en el archivo `php.ini`, en la sección que mostramos a continuación:

```
; Colors for Syntax Highlighting mode
highlight.string    =    #DD0000
highlight.comment   =    #FF8000
highlight.keyword   =    #007700
highlight.bg        =    #FFFFFF
highlight.default   =    #0000BB
highlight.html      =    #000000
```

Los colores se muestran en formato RGB HTML estándar.

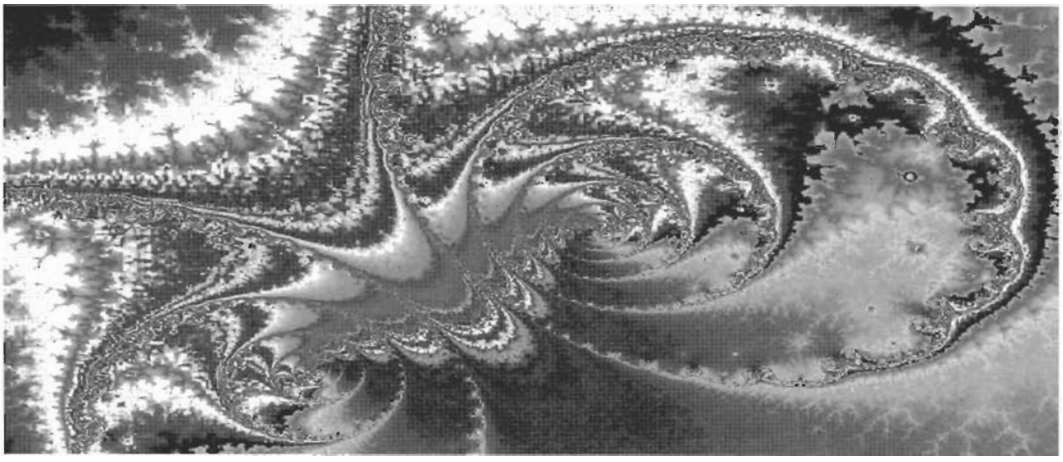
A continuación

En la siguiente parte analizaremos una serie de proyectos prácticos relativamente complicados que utilizan PHP y MySQL. Estos proyectos constituyen ejemplos muy útiles para tareas similares que tenga que desarrollar y demuestran el uso de PHP y MySQL en proyectos de mayor tamaño.

En el siguiente capítulo nos centraremos en algunos problemas a los que nos enfrentaremos al codificar grandes proyectos con PHP. Abordaremos principios de ingeniería de software como por ejemplo el diseño, la documentación y la gestión de cambios.

Parte V

Creación de proyectos PHP y MySQL prácticos



22
Uso de PHP y MySQL en grandes proyectos

En las anteriores partes de este libro hemos analizado distintos componentes y usos de PHP y MySQL. Aunque hemos intentado que todos los ejemplos fueran interesantes y relevantes, todos han sido muy sencillos, formados por una o dos secuencias de comandos de hasta 100 líneas de código aproximadamente.

Al generar aplicaciones Web reales, las cosas no suelen ser tan sencillas. Hace tiempo, se consideraba a un sitio Web como "interactivo" cuando incluía correo de formulario y poco más. Sin embargo, en la actualidad, los sitios Web se han convertido en aplicaciones Web, es decir, un fragmento de software convencional que se entrega por la Web.

Este cambio de enfoque significa un cambio de escala. Los sitios Web pasan de tener un puñado de secuencias de comandos a miles y miles de líneas de código. Los proyectos de este tamaño requieren la misma planificación y gestión que cualquier otro desarrollo de software.

Antes de pasar a analizar los proyectos en este apartado del libro, nos detendremos en algunas de las técnicas que podemos utilizar para gestionar proyectos Web de gran tamaño.

Se trata de un arte en expansión y resulta muy complicado dominarlo correctamente, lo que comprobará si analiza el mercado.

En este capítulo nos detendremos en los siguientes aspectos:

- Aplicación de ingeniería de software al desarrollo Web
- Planificación y ejecución de un proyecto de aplicación Web
- Reutilización de código

- Escritura de código mantenible
- Implementación de control de versión
- Selección de un entorno de desarrollo
- Documentación de un proyecto
- Prototipos
- Separación de lógica, contenido y presentación: PHP, HTML y CSS
- Optimización de código

Aplicación de ingeniería de software al desarrollo Web

Como seguramente ya sabrá, la ingeniería de software es la aplicación de un enfoque sistemático y cuantificable al desarrollo de software.

Es decir, se trata de la aplicación de los principios de ingeniería al diseño de software.

También es un enfoque que falta en muchos proyectos Web, principalmente debido a dos razones.

La primera es que el desarrollo Web se suele gestionar de la misma forma que el desarrollo de informes escritos. Es un ejercicio de estructura de documentos, diseño gráfico y producción, un paradigma orientado a documentos.

Es totalmente válido para sitios de pequeño o mediano tamaño pero al aumentar el número de contenidos dinámicos de los sitios Web hasta alcanzar el nivel en el que dichos sitios ofrecen servicios en lugar de documentos, este paradigma ya no sirve.

Hay mucha gente que ni siquiera piensa en utilizar prácticas de ingeniería de software en un proyecto Web.

La segunda razón por la que no se ha utilizado la ingeniería de software es que el desarrollo de aplicaciones Web es diferente al de aplicaciones normales.

Los plazos son mucho más reducidos y la presión por finalizar el sitio es constante. La ingeniería de software consiste en hacer cosas de forma ordenada y planificada así como en dedicar tiempo a la planificación.

Con los proyectos Web, normalmente se tiene la sensación de que no hay tiempo para planificar.

Si no planificamos un proyecto Web, sufriremos los mismos problemas que si no planificamos cualquier proyecto de software: aplicaciones defectuosas, fechas de entrega sin cumplir y código ilegible.

Por ello, el truco consiste en identificar las partes de la ingeniería de software que funcionan en esta nueva disciplina del desarrollo de aplicaciones Web y en rechazar las que no lo hacen.

Planificación y ejecución de un proyecto de aplicación Web

No existe una metodología o ciclo vital idóneo para un proyecto Web. No obstante, existen distintos aspectos que se pueden tener en cuenta a la hora de preparar un proyecto.

Los enumeramos todos y, en los siguientes apartados, describiremos algunos con mayor detalle.

Estas consideraciones tienen un orden concreto pero no es necesario que lo siga si no se ajusta a su proyecto.

El énfasis recae en estar alerta sobre los posibles errores y en seleccionar las técnicas que nos puedan servir.

- Antes de empezar, piense en lo que quiere generar. Piense en el objetivo final. Piense en quién va a utilizar su aplicación Web, es decir, en su público de destino. Muchos proyectos Web técnicamente perfectos no funcionan porque nadie ha comprobado si hay usuarios interesados en dicha aplicación.
- Pruebe su aplicación y divídala en componentes. ¿De qué partes o pasos se compone su aplicación? ¿Cómo funciona cada uno de estos componentes? ¿Cómo se combinan entre sí? El diseño de escenarios, guiones gráficos o incluso de casos prácticos puede ser muy útil para determinar estos aspectos.
- Una vez confeccionada la lista de componentes, compruebe cuáles existen ya. Si un módulo preconfigurado ya cuenta con dicha funcionalidad, utilícelo.

  No olvide buscar código existente dentro y fuera de su organización. Existen muchos componentes de código gratuitos, en especial en la comunidad de código abierto.
- Tome decisiones sobre aspectos del proceso, algo que se suele ignorar en los proyectos Web. Por aspectos del proceso nos referimos a aspectos como los estándares de codificación, estructuras de directorios, administración del control de versión, entorno de desarrollo, nivel y estándares de documentación, y asignación de tareas a los integrantes del equipo.
- Genere un prototipo, basado en toda la información anterior. Muéstrelo a los usuarios. Repítalo.
- Recuerde que, en todo esto, es muy importante separar el contenido y la lógica de la aplicación, como veremos posteriormente con mayor detalle.
- Realice todas las optimizaciones que considere oportunas.
- Mientras avance, pruebe concienzudamente la aplicación, como haría con cualquier otro proyecto de desarrollo de software.

Reutilización de código

Los programadores suelen cometer el error de volver a escribir código que ya existe. Una vez identificados los componentes necesarios para su aplicación o, a menor escala, las funciones que necesita, compruebe lo que está disponible antes de iniciar el desarrollo. Uno de los puntos fuertes de PHP como lenguaje es la extensa biblioteca de funciones que incorpora. Siempre debería comprobar si existe una función que haga lo que quiere conseguir. No le resultará difícil encontrar la que busca. Una buena forma de hacerlo consiste en examinar el manual por grupos de funciones.

En ocasiones los programadores vuelven a escribir funciones accidentalmente ya que no han comprobado si una de las funciones existentes en el manual ofrece la funcionalidad que necesitan. Debería marcar siempre el manual si se encuentra en línea o descargar la versión actual y examinarla localmente. Sepa, sin embargo, que el manual online se actualiza frecuentemente y que también tiene la ventaja de poder consultar el manual anotado. Éste es una magnífica fuente que contiene comentarios, sugerencias y ejemplos de código de otros usuarios que suelen resolver las dudas que pueda tener después de leer el manual básico. La versión en español del manual se encuentra en la página `http://www.php.net/manual/es/`.

Algunos programadores con experiencia en un lenguaje diferente pueden verse tentados a escribir funciones contenedoras para cambiar el nombre de las funciones PHP de manera que coincidan con las del lenguaje que conocen. Esta práctica no es nada aconsejable, ya que dificulta la lectura y el mantenimiento del código para otros. Si está aprendiendo un nuevo lenguaje, tendrá que aprender a utilizarlo correctamente. Además, la inclusión de un nivel de llamada de funciones de esta manera ralentizaría el código. Por todas estas razones, es un enfoque que debería evitar.

Si la funcionalidad que necesita no se encuentra en la biblioteca PHP principal, tiene dos opciones. Si necesita algo muy sencillo, puede optar por escribir su propia función u objeto. Sin embargo, si pretende generar una funcionalidad más compleja, como un carro de la compra, un sistema Web de correo electrónico o foros Web, no se sorprenda si alguien ya lo ha generado. Una de las ventajas de trabajar en la comunidad de código abierto es que el código para este tipo de componentes de aplicación está disponible gratuitamente. Si encuentra un componente similar al que necesita, aunque no sea exacto, puede utilizar el código fuente como punto de partida y modificar o generar el suyo propio.

Si opta por diseñar sus propias funciones o componentes, debería ofrecerlas a la comunidad PHP en cuanto termine. Es el principio que convierte a la comunidad de programadores PHP en algo tan útil y activo.

Cómo escribir código mantenible

El problema del mantenimiento es infravalorado en las aplicaciones Web, en especial por que las escribimos a toda prisa. Empezar el código y acabarlo rápidamente

suele ser más importante que planificarlo con antelación. Sin embargo, con una pequeña revisión previa nos ahorraremos gran parte del camino cuando sea necesario generar la siguiente iteración de la aplicación.

Estándares de codificación

La mayoría de las organizaciones IT disponen de estándares de codificación, directrices de estilo para seleccionar nombres de archivos y variables, directrices para comentar código, directrices para identificarlo, etc.

Debido al paradigma de código que aplicamos previamente al desarrollo Web, los estándares de codificación se han infravalorado. Si el código lo diseña personalmente o forma parte de un pequeño equipo, puede que no le dé la importancia que tiene. No lo haga por temor a que el equipo o el proyecto puedan crecer. No sólo obtendrá un producto sin sentido, sino también un montón de programadores que no saben ni qué hacer con el código existente.

Convenciones de nomenclatura

Los objetivos de la definición de una convención de nomenclatura son:

- Facilitar la lectura del código. Si define variables y nombres de funciones con sentido, podrá leer el código prácticamente como si se tratara de una frase normal o, al menos, pseudocódigo.
- Que los nombres de identificadores sean fáciles de recordar. Si sus identificadores tienen un formato coherente, resultará más sencillo recordar el nombre de una determinada función o variable.

Los nombres de variables deberían describir los datos que contienen. Si se trata de almacenar el apellido de alguien, llame a la variable `$apellido`. Tendrá que encontrar un equilibrio entre longitud y legibilidad. Por ejemplo, si almacena el nombre en `$n`, será más sencillo de escribir pero el código será más difícil de entender. Si almacena el nombre en `$apellido_del_usuario_actual` será más claro, pero tardará más en escribirlo (y es más proclive a errores ortográficos) y realmente no ayuda tanto. Tendrá que tomar una decisión en cuanto a las mayúsculas. En PHP, los nombres de variables distinguen entre mayúsculas y minúsculas, como hemos mencionado anteriormente. Tendrá que decidir si los nombres de las variables van en minúscula, en mayúscula o una mezcla de ambas, por ejemplo, las primeras letras de las palabras en mayúscula. Por lo general se utilizan siempre minúsculas, ya que resulta más sencillo de recordar.

También es aconsejable distinguir entre variables y constantes con mayúsculas. Una práctica habitual consiste en utilizar minúsculas con las variables (por ejemplo, `$result`) y mayúsculas con las constantes (por ejemplo, `PI`).

Una práctica desaconsejable que algunos programadores emplean es utilizar dos variables con el mismo nombre pero en minúsculas y mayúsculas, como por ejemplo `$name` y `$Name`. Espero que sepa por qué no es aconsejable.

También conviene evitar modelos complicados, como por ejemplo `$WaReZ`, ya que nadie podrá recordar cómo funciona.

Debería pensar en qué esquema utilizará con nombres de variables de varias palabras.

Por ejemplo:

```
$username
$user_name
$UserName
```

son esquemas que se suelen utilizar. Puede utilizar cualquiera, pero hágalo siempre de forma coherente. También puede establecer un límite máximo de dos a tres palabras en un nombre de variable.

Los nombres de funciones tienen la misma consideración, con algunos aspectos adicionales. Los nombres de funciones suelen estar orientados a verbos. Por ejemplo, funciones de PHP como `addslashes()` o `msyql_connect()` describen lo que van a hacer o los parámetros con los que se pasan. Esto mejora considerablemente la legibilidad del código.

Debe saber que estas dos funciones tienen un esquema de nomenclatura diferente para utilizar nombres de funciones de varias palabras. A este respecto, las funciones de PHP no son coherentes, probablemente por la gran cantidad de gente que las ha escrito, pero principalmente porque muchos nombres de funciones se han adoptado sin cambiar de distintos lenguajes y API.

También debe recordar que los nombres de función no distinguen entre mayúsculas y minúsculas en muchos módulos PHP, es decir, se añade el nombre de la función como prefijo al módulo PHP. Por ejemplo, todas las funciones MySQL empiezan por `mysql_` y todas las funciones IMAP con `imap_`. Si por ejemplo tiene un módulo de carro de la compra en su código, podría añadir el prefijo `carro_` a la función de dicho módulo.

En definitiva, no importa qué convenciones y estándares utilice cuando escriba el código, siempre que aplique directrices coherentes.

Comentarios de código

Todos los programas deberían incluir un cierto número de comentarios. Y se preguntará cuál este número. Normalmente debería añadir comentarios a cada uno de los siguientes elementos:

- Archivos, ya sean secuencias de comando completas o archivos incluidos. Cada archivo debe tener un comentario que indique lo que es el archivo, para qué es, quién lo ha escrito y cuándo se ha actualizado.
- Funciones. Los comentarios de una función deben especificar qué hace la función, qué entrada espera y qué devuelve.
- Clases. Los comentarios deben describir el objetivo de la clase. Los métodos de clases deben tener el mismo tipo y nivel de comentarios que cualquier otra función.

- Fragmentos de código dentro de una secuencia de comandos o de una función. Es muy útil escribir una secuencia de comandos que comience con una serie de comentarios y tras ello completar el código de cada sección. Una secuencia de comandos inicial sería como ésta:

```
<?
// valide datos introducidos
// envíe a la base de datos
// informe de los resultados
?>
```

  Resulta muy útil por que una vez completadas todas las secciones con llamadas de funciones u otros elementos, el código ya está comentado.

- Código complejo. Cuando le lleve mucho tiempo escribir algo o lo tenga que hacer de forma poco habitual, escriba un comentario que explique por qué lo ha hecho. De esta forma, la próxima vez que lea el código sabrá exactamente lo que tiene que hacer.

Otra norma general que debe seguir es comentar el código mientras avanza. Puede pensar en volver atrás y comentar el código una vez terminado el proyecto. Le garantizo que no sucederá, a menos que tenga unas fechas de entrega menos estrictas y mayor autodisciplina que nosotros.

Sangrados

Al igual que en cualquier lenguaje de programación, es aconsejable sangrar el código de forma coherente y con sentido. Es igual que redactar un resumen o una carta de negocios. El sangrado facilita la lectura del código y hace que se entienda con mayor rapidez.

Por lo general, cualquier bloque de programación dentro de una estructura de control debería sangrarse del código adyacente. El grado de sangrado debe ser perceptible (es decir, más de un espacio) pero no excesivo. Personalmente creo que no debería utilizar tabulaciones. Aunque son más fáciles de añadir, consumen gran cantidad de espacio en la pantalla de la mayoría de los usuarios. Suelo utilizar un nivel de sangrado de dos a tres espacios en todos los proyectos. La forma de disponer las llaves también es un problema. Las dos técnicas más habituales son las siguientes:

Técnica 1:

```
if (condición) {
   // haga algo
}
```

Técnica 2:

```
if (condición)
{
   // haga algo más
}
```

Puede utilizar cualquiera de las dos, pero la que seleccione debe utilizarla con coherencia a lo largo del proyecto para evitar confusiones.

División del código

El código monolítico de gran tamaño es horrible. Mucha gente incluye en una línea principal de código una enorme secuencia de comandos que se encarga de todo. Es mucho mejor dividir el código en funciones y/o clases y añadir los elementos relacionados en archivos. Por ejemplo puede incluir todas las funciones relacionadas con bases de datos en un archivo con el nombre `dbfunctions.php`.

Entre las razones para dividir el código en fragmentos de menor tamaño podemos destacar las siguientes:

- Resulta más sencillo leer y comprender el código.
- El código es más reutilizable y se minimizan las redundancias. Por ejemplo, el archivo `dbfunctions.php` anterior se podría reutilizar en todas las secuencias de comandos que quiera conectar a la base de datos. Si tuviera que cambiar su funcionamiento, sólo tendría que hacerlo en un punto.
- Facilita el trabajo en equipo. Si el código se divide en componentes, puede asignar la responsabilidad de dichos componentes a distintos integrantes del equipo. También evitará que un programador tenga que esperar a que otro termine con un archivo para proseguir con su trabajo.

Al iniciar un proyecto, debería pensar en cómo va a dividirlo en componentes. Esto implica establecer líneas entre áreas de funcionalidad, pero no se preocupe ya que puede cambiar después de comenzar un proyecto. También debe decidir qué componentes hay que generar en primer lugar, qué componentes dependen de otros y la fecha límite para desarrollarlos.

Incluso si todos los integrantes del equipo trabajan con todos los fragmentos de código, es aconsejable asignar la responsabilidad principal de cada componente a una persona en concreto. De esta forma, habrá un responsable por si algo sale mal con el componente.

También debería haber alguien que actúe como director, es decir, una persona que compruebe que todos los componentes están preparados y se combinan con los componentes restantes. Esta persona también se encarga del control de versión, que analizaremos más adelante. Puede tratarse del director del proyecto o se puede asignar como responsabilidad independiente.

Uso de una estructura de directorios estándar

Al iniciar un proyecto, es necesario pensar en cómo se reflejará la estructura de los componentes en la estructura de directorios del sitio Web. Al igual que no es aconsejable utilizar una única secuencia de comandos de gran tamaño que contenga toda la funcionalidad, tampoco es aconsejable tener un directorio gigante que contenga todo. Decida cómo va a dividirlo entre componentes, lógica, contenido y bibliotecas de código compartido. Documente su estructura y asegúrese de que todo el que trabaje en el proyecto dispone de una copia, lo que nos lleva al siguiente punto.

Cómo documentar y compartir funciones internas

Al desarrollar bibliotecas de funciones es aconsejable que estén disponibles para el resto de programadores del equipo. Habitualmente, cada uno de los programadores de un equipo escribe su propio conjunto de funciones de base de datos, de fecha o de depuración, lo que supone una pérdida de tiempo. Es mejor compartir las funciones y las clases. Recuerde que incluso si el código se almacena en una zona o en un directorio disponible para todo el equipo, no lo sabrán a menos que se lo indique. Debe desarrollar un sistema para documentar bibliotecas de funciones internas y para compartirlas con los programadores de su equipo.

Implementación del control de versión

El control de versión es el arte de la gestión de cambios simultáneos aplicada al desarrollo de software. Los sistemas de control de versión funcionan como un archivo central y proporcionan una interfaz controlada para acceder y compartir el código (y también la documentación).

Imagine que tiene que mejorar un código pero, por accidente, lo divide y no puede recuperar la forma que tenía, haga lo que haga. O imagine que su cliente decide que una versión anterior del sitio estaba mejor. O que tiene que utilizar una versión anterior por motivos legales.

Imagine otro caso en el que dos integrantes de su equipo de programación quieren trabajar con el mismo archivo. Puede que ambos hayan abierto y modificado el archivo al mismo tiempo, sobrescribiendo los cambios. Puede que tengan una copia con la que trabajen localmente y la modifiquen de distinta forma. Si ha pensado en que todo esto puede suceder, puede que haya un programador sentado sin hacer nada mientras espera a que otro termine de modificar un archivo.

Todos estos problemas se pueden solucionar con un sistema de control de versión.

Estos sistemas pueden monitorizar los cambios de cada archivo de forma que no sólo podamos ver el estado actual de un archivo, sino también su aspecto en un momento concreto. Esta opción nos permite devolver un fragmento de código dividido a una versión que sabemos que funciona. Podemos adjuntar un conjunto de instancias de archivos concretos como versión de lanzamiento, lo que significa que podemos proseguir con el desarrollo del código pero con acceso a una copia de la versión de lanzamiento actual en cualquier momento.

Los sistemas de control de versión también contribuyen a que varios programadores puedan trabajar conjuntamente en el código. Cada programador puede obtener una copia del código del archivo y, cuando realice algún cambio, se almacenan en el repositorio, o se confirman De esta forma, los sistemas de control de versión pueden saber quién ha realizado cada uno de los cambios en el sistema.

Estos sistemas suelen disponer de una función para administrar actualizaciones simultáneas, lo que significa que dos programadores pueden modificar el mismo archivo al mismo tiempo. Por ejemplo, imagine que John y Mary han sacado una

copia de la versión más reciente del proyecto. John concluye sus cambios en un determinado archivo y los confirma. Mary también realiza cambios en ese archivo e intenta confirmarlos.

Si no han hecho los cambios en la misma parte del archivo, el sistema de control de versión combinará las dos versiones del archivo.

Si los cambios entran en conflicto, Mary recibirá una notificación y se le mostrarán las dos versiones. De esta forma puede ajustar su versión del código para evitar los conflictos.

El sistema de control de versión utilizado por la mayoría de los programadores de Unix y/o de código abierto es CVS (Sistema de versiones concurrentes). CVS es código abierto, viene incorporado prácticamente en todos los sistemas Unix y también se puede ejecutar en DOS o Windows y Mac. Admite un modelo cliente-servidor para poder sacar o confirmar código desde cualquier equipo con una conexión a Internet, siempre que el servidor CVS se pueda ver en la red. Se utiliza para el desarrollo de PHP, Apache y Mozilla, entre otros proyectos principales, en parte al menos por esta razón.

Puede descargar CVS para su sistema desde la página de CVS en `http://www.cvshome.org/`.

Aunque el sistema CVS base es una herramienta de línea de comandos, existen diversos complementos que le proporcionan una interfaz más vistosa, incluyendo interfaces basadas en Java y Windows, a las que también se puede acceder desde la página de CVS.

Hay alternativas comerciales a CVS. Una de ellas es perforce, que se ejecuta en la mayoría de las plataformas más conocidas y es compatible con PHP. Aunque es comercial, se ofrecen licencias gratuitas para proyectos de código abierto en la dirección `http://www.perforce.com/`.

Selección de un entorno de desarrollo

Al hablar del control de versión hay que hacer referencia al tema de los entornos de desarrollo.

Realmente basta con un editor de textos y un navegador para realizar las pruebas, pero los programadores suelen ser más productivos en un entorno integrado o IDE.

Existen diversos proyectos gratuitos para generar un IDE PHP dedicado, incluyendo KPHPDevelop, para el entorno de escritorio KDE bajo Linux, que podrá encontrar en `http://kphpdev.sourceforge.net/`.

Sin embargo, actualmente los mejores IDE PHP son todos comerciales. Zend Studio de `zend.com`, Komodo de `activestate.com` y PHPEd de `nusphere.com` cuentan con IDE muy completos.

Todo ellos disponen de una descarga de prueba pero es necesario adquirir el producto para continuar utilizándolo. Komodo dispone de una licencia de uso no comercial muy económica.

Documentación de nuestros proyectos

Podemos producir distintos tipos de documentación para nuestros proyectos de programación incluyendo, entre otros, los siguientes:

- Documentación de diseño
- Documentación técnica / guía del programador
- Diccionario de datos (incluyendo documentación sobre clases)
- Guía del usuario (aunque la mayoría de las aplicaciones Web deberían ser autoexplicativas)

Nuestro objetivo no es enseñarle a escribir documentación técnica sino sugerirle que lo haga todo más fácil si automatiza parte del proceso.

En algunos lenguajes, existen diversas formas de generar automáticamente algunos de estos documentos, sobre la documentación técnica y los diccionarios de datos. Por ejemplo, `javadoc` genera un árbol de archivos HTML que contienen prototipos y una descripción de los miembros de clases para programas Java.

Algunas utilidades de este tipo disponibles para PHP son las siguientes:

- `phpdoc`, disponible en `http://www.phpdoc.de/`.
  Es un sistema utilizado por PEAR para documentar código. Debe saber que el término phpDoc se utiliza para describir varios proyectos de este tipo, de los cuales es uno.
- `PHPDocumentor`, disponible en `http://phpdocu.sourceforge.net`.
  Genera un resultado muy similar al de `javadoc` y parece que funciona correctamente. También parece que cuenta con un equipo de programación más activo que los otros dos.
- `phpautodoc`, disponible en `http://sourceforge.net/projects/phpautdoc/`.
  Como en el caso anterior, produce un resultado similar al de `javadoc`.

Si necesita más aplicaciones de este tipo, (y componentes PHP en general), consulte SourceForge, en la dirección `http://sourceforge.net`.

Lo utiliza básicamente la comunidad UNIX/Linux pero también hay numerosos proyectos para otras plataformas.

Prototipos

La creación de prototipos es un ciclo vital de desarrollo utilizado habitualmente para diseñar aplicaciones Web. Un prototipo es una herramienta muy útil para satisfacer los requisitos del cliente. Normalmente se trata de una versión simplificada parcialmente funcional de una aplicación que se puede utilizar para comentar

con el cliente y como base del sistema final. A menudo, las múltiples iteraciones de un prototipo generan la aplicación final. La ventaja de este enfoque es que nos permite trabajar junto al cliente o el usuario final para producir un sistema con el que estén conformes y que, de algún modo, les pertenezca.

Para poder crear de forma conjunta un prototipo con rapidez, necesitará determinados conocimientos y herramientas. Y aquí es donde mejor funciona un enfoque basado en componentes. Si tiene acceso a un conjunto de componentes ya existentes, disponibles tanto de forma interna como pública, podrá hacerlo con mayor rapidez. Otra herramienta de gran utilidad para el rápido desarrollo de prototipos son las plantillas, que veremos en el siguiente apartado.

Existen dos problemas principales a la hora de utilizar el enfoque de prototipos. Tendrá que conocerlos para poder evitarlos y así utilizar este enfoque con todo su potencial.

El primer problema es que, por alguna razón, a los programadores les resulta difícil desprenderse del código que han escrito. Los prototipos se escriben con rapidez y, con la ventaja del tiempo transcurrido, comprobará que no ha generado un prototipo de forma óptima. Las secciones extensas de código se pueden mejorar, pero si la estructura general no es correcta, entonces tiene un verdadero problema. El problema es que normalmente las aplicaciones Web se generan bajo una gran presión y en ocasiones no hay tiempo para corregirlas. De esta forma nos encontramos con un sistema pobremente diseñado que resulta difícil de mantener.

Todo esto se puede evitar con una cierta planificación como hemos mencionado anteriormente. No obstante, recuerde que en ocasiones resulta más sencillo realizar un boceto y empezar de nuevo que corregirlo. Aunque pueda parecerle que no tiene tiempo para ello, le ahorrará muchos sufrimientos posteriores.

El segundo problema de los prototipos es que un sistema se puede convertir en un prototipo eterno. Cada vez que piensa que ha terminado su cliente le sugiere nuevas mejoras o funcionalidades adicionales o actualizaciones del sitio. Todo esto puede provocar que no acabe el proyecto.

Para evitar este problema, diseñe un plan del proyecto con un número de iteraciones fijo así como una fecha después de la que no se permita añadir nueva funcionalidad sin planificar, programar y diseñar un presupuesto.

Separación de lógica y contenido

Probablemente esté familiarizado con el uso de HTML para describir una estructura de documentos Web así como de hojas de estilo en cascada (CSS) para describir su apariencia. Esta idea de separar la presentación del código se puede extender a la programación. Por lo general, los sitios resultan más sencillos de utilizar y de mantener a largo plazo si separamos la lógica del contenido de la presentación, en definitiva, si separamos PHP y HTML.

En proyectos sencillos con un número reducido de líneas de código, puede convertirse en un inconveniente más que en una ventaja. Cuando los proyectos crecen,

resulta fundamental encontrar la forma de separar lógica y contenido. Si no lo hace, su código será cada vez más difícil de mantener. Si decide aplicar un nuevo diseño a su sitio Web e incrusta gran cantidad de HTML en el código, el cambio del diseño se convertirá en una pesadilla.

Hay tres enfoques básicos a la hora de separar lógica y contenido, que describimos a continuación:

- Utilice archivos para almacenar distintas partes del contenido. Es un enfoque muy simple pero si el sitio es básicamente estático, funciona perfectamente. Este tipo de enfoque se explicó en un ejemplo de un capítulo anterior.
- Utilice un API de funciones o de clases con un conjunto de funciones miembro para conectar contenido dinámico a plantillas de páginas estáticas. También describimos este enfoque en un capítulo anterior.
- Utilice un sistema de plantillas. Permite analizar plantillas estáticas y utilizar expresiones regulares para reemplazar etiquetas de marcadores de posición con datos dinámicos. La principal ventaja es que si otra persona diseña nuestras plantillas, como por ejemplo un diseñador gráfico, no es necesario que domine el código PHP. Podremos utilizar las plantillas proporcionadas con mínimas modificaciones.
  Existen diversos sistemas de plantillas disponibles. Probablemente el más conocido sea Smarty, que puede encontrar en `http://smarty.php.net/`.

Optimización de código

Si proviene de un entorno no relacionado con la programación Web, la optimización le resultará de gran importancia. Al utilizar PHP, la mayor parte del tiempo de espera de un usuario se debe a la conexión y descarga de una aplicación Web. La optimización del código apenas tiene efecto sobre estos tiempos de espera.

Uso de optimización sencilla

No obstante puede realizar algunas sencillas optimizaciones que sí suponen una cierta mejoría. Muchas están relacionadas con aplicaciones que integran una base de datos como MySQL en el código PHP, como indicamos a continuación:

- Reduzca las conexiones a bases de datos. La conexión a una base de datos suele ser la parte más lenta de cualquier secuencia de comandos. Para evitarlo puede utilizar conexiones persistentes.
- Aumente la velocidad de la consultas de bases de datos. Reduzca el número de consultas realizadas y asegúrese de optimizarlas. Con una consulta compleja (y, por lo tanto, lenta), hay más de una forma de hacer las cosas. Ejecute sus

consultas desde la interfaz de línea de comandos de la base de datos y experimente con distintos enfoques para acelerar la operación. En MySQL, puede utilizar la instrucción `EXPLAIN` para ver dónde puede estar el error de la consulta (en un capítulo anterior se describió el uso de esta instrucción). Por lo general, conviene minimizar las uniones y maximizar el uso de índices.

- Minimice la generación de contenido estático desde PHP. Si todo el HTML que genera proviene de `echo` o `print()`, tardará mucho más (es uno de los argumentos para cambiar hacia la separación de lógica y contenido que mencionamos antes). También se aplica a la generación dinámica de botones de imagen; podemos utilizar PHP para generar los botones una vez y volver a utilizarlos cuando sea necesario. Si solamente genera páginas estáticas a partir de funciones o plantillas cada vez que se carga una página, puede pensar en ejecutar las funciones o en un utilizar las funciones una vez y en guardar el resultado.
- Utilice funciones de cadena en lugar de expresiones regulares siempre que sea posible. Son más rápidas.
- Utilice cadenas de comillas simples en lugar de cadenas de comillas dobles siempre que sea posible. PHP evalúa las cadenas de comillas dobles y busca variables que pueda sustituir. Las cadenas de comillas simples no se evalúan. Por otra parte, si aparece entre comillas simples, seguramente se trate de contenido estático. Repase lo que esté haciendo y compruebe si puede eliminar toda la cadena si la convierte a HTML estático.

Uso de productos Zend

Zend Technologies es propietaria del motor de programación de secuencia de comandos de PHP (código_abierto) que se ha utilizado desde PHP 4. Además del motor básico, también puede descargar Zend Optimizer. Se trata de un optimizador multipasada que optimiza el código por nosotros y que puede aumentar la velocidad de ejecución de las secuencias de comandos desde un 40% a un 100%. Para ejecutar el optimizador necesitará PHP 4.0.2 o una versión superior. Aunque se trata de código cerrado, se puede descargar gratuitamente desde el sitio de Zend, `http://www.zend.com`.

El funcionamiento de este complemento consiste en optimizar el código generado por la compilación en tiempo de ejecución de nuestra secuencia de comandos. Entre otros productos de Zend podemos destacar Zend Studio, Zend Accelerator, Zend Encoder así como acuerdos de compatibilidad comercial.

Comprobaciones

La revisión y la comprobación del código es otro aspecto básico de la ingeniería de software que se suele pasar por alto en programación Web. Es muy habitual

tratar de ejecutar el sistema con dos o tres pruebas y decir que funciona correctamente, un error muy común.

Compruebe que ha revisado y probado exhaustivamente distintos casos antes de preparar la producción del proyecto.

Le sugerimos dos enfoques que puede utilizar para reducir los errores de su código (nunca se pueden eliminar de forma conjunta, pero sí es posible eliminar o minimizar la mayoría de ellos).

En primer lugar, adopte una práctica de revisión de código. Es el proceso mediante el que otro programador o equipo de programadores revisan el código y sugieren mejoras. Mediante este tipo de análisis se sugiere lo siguiente:

- Errores que ha pasado por alto
- Comprobaciones sobre las que no ha pensado
- Optimización
- Mejoras de la seguridad
- Componentes existentes que puede utilizar para mejorar un fragmento de código
- Funcionalidad adicional

Incluso si trabaja por su cuenta, es aconsejable encontrar un "colega programador" que esté en la misma situación y revisar el código entre ambos.

La segunda sugerencia consiste en localizar personas que prueben nuestras aplicaciones Web y que representen a los usuarios finales del producto. La principal diferencia entre aplicaciones Web y aplicaciones de escritorio es que cualquiera puede utilizar una aplicación Web. No debe asumir que los usuarios estarán familiarizados con la informática. No es aconsejable proporcionales un extenso manual o una mínima tarjeta de referencia. Por el contrario, las aplicaciones Web se deben documentar y explicar por sí mismas. Debe pensar en cómo utilizarán los usuarios la aplicación. La capacidad de uso tiene una importancia fundamental.

Resulta muy complicado comprender los problemas a los que se enfrentan los usuarios noveles si somos experimentados programadores o usuarios de la Web. Una forma de solucionar este aspecto consiste en recurrir a gente que represente a usuarios convencionales.

En el pasado, sólo se publicaban aplicaciones Web en versiones beta para solucionar este problema. Cuando crea que ha corregido la mayoría de los errores, ofrezca la aplicación a un reducido grupo de usuarios de prueba y acepte un tráfico reducido en el sitio. Puede ofrecer servicios gratuitos a los 100 primeros usuarios a cambio de información sobre el sitio. Le garantizo que obtendrá alguna combinación de datos o de usos sobre la que no había pensado.

Si tiene que generar un sitio Web para una empresa cliente, ésta puede proporcionarle un grupo de usuarios noveles y hacer que sus empleados trabajen en el sitio (es una de las ventajas implícitas de fomentar la sensación de propiedad del producto por parte del cliente).

Lecturas adicionales

Existe gran cantidad de material sobre este área que podríamos describir; hemos hecho referencia a la ciencia de la ingeniería de software, sobre lo que se han escrito numerosos libros.

Una obra que explica la dicotomía entre sitio-Web-como-documento y sitio-Web-como-aplicación es *Web Site Engineering: Beyond Web Page Design* de Thomas A. Powell. Cualquier libro sobre ingeniería de software le servirá como referencia.

Si necesita más información sobre el control de versión puede visitar el sitio Web de CVS, `http://www.cvshom.org`.

No existen muchas publicaciones sobre control de versión (lo que resulta sorprendente debido a su importancia) pero puede probar con *Open Source Development with CVS* de Karl Franz Fogel o *CVS Pocket Reference* de Gregor N. Purdy.

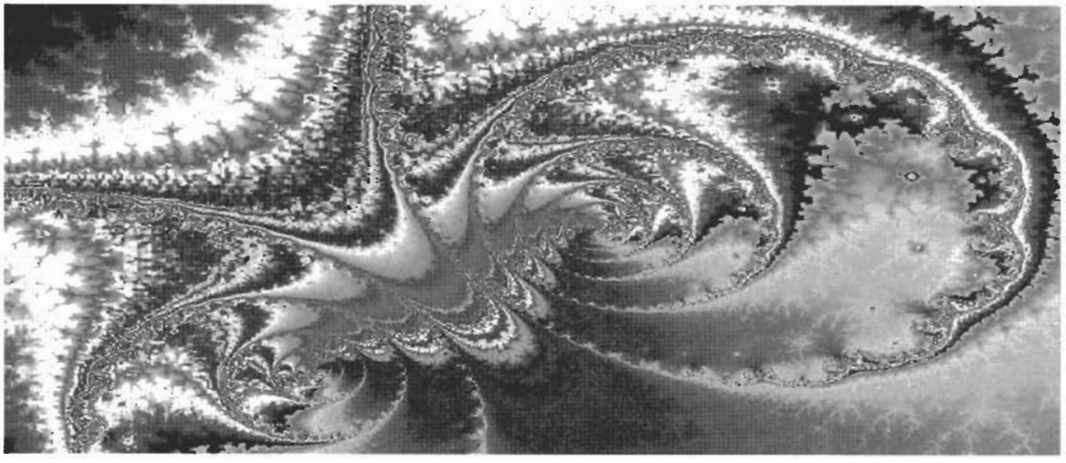
En SourceForge encontrará componentes PHP, IDE o sistemas de documentación, `http://sourceforge.net`.

Muchos de los temas analizados en este capítulo se describen en artículos del sitio de Zend, donde seguramente encontrará más información al respecto. También puede descargar el optimizador del sitio; `www.zend.com`.

Si este capítulo le ha parecido interesante, no dude en visitar Extreme Programming, un método de desarrollo de software dirigido a dominios en los que los requisitos cambian con frecuencia, como es el caso del desarrollo Web. El sitio Web de Extreme Programming es `http://www.extremeprogramming.org`.

A continuación

En el siguiente capítulo veremos distintos tipos de errores de programación, mensajes de error PHP así como técnicas para localizar y procesar errores.



23

Depuración

En este capítulo analizaremos la depuración de secuencias de comandos de PHP. Si ha realizado los ejemplos del libro o ha utilizado PHP antes, seguramente habrá desarrollado sus propias técnicas y conocimientos de depuración.

Al aumentar la complejidad de nuestros proyectos, la depuración también se complica.

Aunque mejoremos nuestros conocimientos, es muy probable que los errores impliquen un mayor número de archivos o de interacciones entre el código de varias personas.

En este capítulo abordaremos los siguientes temas:

- Tipos de errores de programación
  - Errores sintácticos
  - Errores de ejecución
  - Errores lógicos
- Mensajes de error
- Niveles de error
- Cómo desencadenar nuestros propios errores
- Resolución de errores
- Depuración remota

Errores de programación

Independientemente del lenguaje que utilice, existen tres tipos generales de errores de programación:

- Errores sintácticos
- Errores de ejecución
- Errores lógicos

Veremos cada uno de estos tipos antes de analizar las técnicas para detectar, procesar, evitar y solucionar errores.

Errores sintácticos

Los lenguajes disponen de una serie de normas denominada *sintaxis* que las instrucciones deben cumplir para que sean válidas. Esto se aplica tanto a los lenguajes naturales, como el español, como a los lenguajes de programación, como PHP. Si una instrucción no cumple las normas de un lenguaje, se dice que se ha producido un error sintáctico.

También se suelen denominar errores de analizador cuando se habla de lenguajes interpretados como PHP o errores de compilador, cuando se trata de lenguajes compilados como C o Java.

Si no cumplimos las reglas sintácticas de un idioma, por ejemplo del español, es probable que la gente entienda lo que queremos decir, algo que no suele pasar con los lenguajes de programación. Si una secuencia de comandos no cumple las reglas sintácticas de PHP, es decir, si contiene errores sintácticos, el analizador de PHP no podrá procesarla. Los humanos podemos obtener información de datos parciales o en conflicto. Los ordenadores no.

Entre otras muchas reglas, la sintaxis de PHP requiere que las instrucciones terminen en punto y coma, que las cadenas se incluyan entre comillas y que los parámetros pasados a funciones se separen con comas y se encierren entre paréntesis.

Si incumplimos estas normas, es improbable que nuestra secuencia de comandos de PHP funcione y es muy probable que genere un mensaje de error la primera vez que intentemos ejecutarla.

Una de las principales ventajas de PHP son los útiles mensajes de error que indican que algo no funciona correctamente. Un mensaje de error de PHP nos indica qué ha salido mal, en qué archivo se ha producido el error y en qué línea se encuentra.

Un mensaje de error se parece a lo siguiente:

```
Parse error: parse error in
/home/book/public_html/chapter23/error.php on line 2
```

Este error lo produjo esta secuencia de comandos:

```
<?php
   $date  =  date(m.d.y');
?>
```

Apreciará que hemos intentado pasar una cadena a la función `date()` pero que accidentalmente hemos olvidado la comilla de apertura que indica el inicio de la cadena. Los errores sintácticos sencillos como éste son los más fáciles de localizar. Podemos cometer un error similar, pero más difícil de localizar, si olvidamos terminar la cadena, como se muestra en este ejemplo:

```
<?php
   $date = date('m.d.y);
?>
```

Esta secuencia de comandos generará el siguiente mensaje de error:

```
Parse error: parse error in
/home/book/public_html/chapter23/error.php on line 4
```

Evidentemente, como nuestra secuencia de comandos sólo tiene tres líneas, nuestro error no se encuentra realmente en la cuarta. Los errores en los que abrimos algo pero olvidamos cerrarlo se muestran de esta forma. Este tipo de problemas aparecen al utilizar comillas simples y dobles, y también con las distintas formas de paréntesis.

La siguiente secuencia de comandos generará un error sintáctico similar:

```
<?php
   if (true)
   {
      echo 'aquí va el error';
?>
```

Resulta complicado localizar estos errores si se generan a partir de una combinación de varios archivos o si se producen en un archivo de gran tamaño. La aparición de `"parse error on line 1001"` en un archivo de 1.000 líneas puede ser suficiente para arruinarnos el día y nos indica que deberíamos intentar escribir código más modular.

Por lo general, los errores sintácticos son los más sencillos de localizar. Si comete un error sintáctico, PHP le mostrará un mensaje de error en el que indica dónde lo puede encontrar.

Errores de ejecución

Los errores de ejecución pueden resultar difíciles de detectar y corregir. Una secuencia de comandos contiene un error sintáctico o no lo contiene. Si lo tiene, el analizador lo detecta. Los errores de ejecución no sólo se producen a causa de los contenidos de la secuencia de comandos.

Pueden deberse a interacciones entre las secuencias de comandos y otros eventos o condiciones.

La siguiente instrucción:

```
include  ('nombredearchivo.php');
```

es un argumento de PHP perfectamente válido.

No contiene errores sintácticos.

Sin embargo, puede generar un error de ejecución.

Si ejecuta esta instrucción y no existe `nombredearchivo.php` o el usuario con el que se ejecuta la secuencia de comandos no tiene permiso de lectura, obtendremos un error como éste:

```
Fatal error: Failed opening required 'nombredearchivo.php'
(include_path='.:/usr/local/lib/php') in
/home/book/public_html/chapter23/error.php on line 1
```

Aunque no hay nada erróneo en nuestro código, como depende de un archivo que puede o no existir en distintos momentos cuando se ejecute el código, puede generar un error de ejecución.

Las tres siguientes instrucciones de PHP son válidas.

Desafortunadamente, combinadas, intentan realizar algo imposible: dividir por cero.

```
$i = 10;
$j = 0;
$k = $i/$k;
```

Este fragmento de código genera la siguiente advertencia:

```
Warning: Division by zero in
/home/book/public_html/chapter23/div0.php on line 3
```

Resulta muy sencillo de corregir. Nadie escribiría a propósito un código que intentara dividir por cero, pero si no se comprueban las entradas de usuario se puede producir este tipo de error.

Se trata de uno de los muchos errores de ejecución que podemos ver al probar nuestro código.

Entre las principales causas de los errores de ejecución destacamos las siguientes:

- Llamadas a funciones que no existen
- Lectura o escritura de archivos
- Interacción con MySQL u otras bases de datos
- Conexiones a servicios de red
- Comprobación de entrada de datos incorrecta

Veremos cada una de estas causas individualmente.

Llamadas a funciones que no existen

Es muy habitual invocar funciones que no existen. Las funciones incorporadas suelen utilizar nombres sin excesiva coherencia. Por ejemplo, `strip_tags()` tiene un guión bajo mientras que `stripslashes()` no lo tiene.

También es muy común invocar una función que no existe en la secuencia de comandos actual, pero sí en alguna otra parte. Si su código contiene una llamada a una función que no existe, como por ejemplo

```
nonexistent_function();
```

obtendrá un mensaje de error similar al siguiente:

```
Fatal error: Call to undefined function: nonexistent_function()
in /home/book/public_html/chapter23/error.php on line 1
```

Del mismo modo, si invoca una función que existe pero lo hace con un número incorrecto de parámetros, recibirá una advertencia.

La función `strstr()` requiere dos cadenas: un pajar en el que buscar y una aguja que encontrar. Si por el contrario la invocamos de este modo:

```
strstr();
```

Obtendremos la siguiente advertencia:

```
Warning: Wrong parameter count for strstr() in
/home/book/public_html/chapter23/error.php on line 1
```

Como PHP no permite la sobrecarga de funciones, esta línea siempre será incorrecta aunque no siempre veremos esta advertencia. La misma instrucción dentro de la siguiente línea de comandos también es incorrecta:

```
<?php
   if($var == 4)
   {
      strstr();
   }
?>
```

pero a excepción del raro caso de que la variable `$var` tenga el valor 4, la invocación de `strstr()` no se producirá, por lo que no se mostrará la advertencia.

La invocación incorrecta de funciones es algo muy habitual pero como los mensajes de error resultantes identifican la línea exacta y la llamada que origina el problema, se pueden corregir fácilmente. Solamente resultan difíciles de encontrar si el proceso de prueba es pobre y no comprueba todo el código ejecutado condicionalmente. Al realizar pruebas, uno de los objetivos consiste en ejecutar todas las líneas de código una vez. Otro es probar todas las condiciones y clases de entrada.

Lectura o escritura de archivos

Aunque cualquier cosa puede salir mal en algún momento de la vida útil de un programa, algunas son más probables que otras. Los errores de acceso a archivos se suelen producir tan a menudo que es necesario procesarlos correctamente. Los

discos duros fallan o se llenan y los errores humanos hacen que cambien los directorios de permisos.

Funciones como `fopen()`, que suelen fallar a menudo, normalmente devuelven un valor para indicar que se ha producido un error. En el caso de `fopen()`, el valor `false` indica la presencia de algún fallo.

En funciones que indican la presencia de un error, es necesario revisar con atención el valor devuelto por cada llamada.

Interacción con MySQL u otras bases de datos

La conexión a MySQL y el uso del mismo puede generar muchos errores. Por sí misma, la función `mysql_connect()` puede generar al menos los siguientes errores:

- `MySQL Connection Failed: Can't connect to MySQL server on 'hostname' (111)`
- `MySQL Connection Failed: Can't connect to local MySQL server through socket '/tmp/mysql.sock' (111)`
- `MySQL Connection Failed: Unknown MySQL Server Host 'hostname' (2)`
- `MySQL Connection Failed: Access denied for user: 'username@localhost' (Using password: YES)`

Como habrá imaginado, `mysql_connect()` devuelve el valor `false` cuando se produce un error. Esto significa que este tipo de errores se pueden detectar y corregir fácilmente.

Si no detiene la ejecución regular de su secuencia de comandos y procesa estos errores, la secuencia de comandos continuará en su intento de conexión a la base de datos. Si intenta ejecutar consultas y obtener resultados sin una conexión MySQL válida, los visitantes verán una pantalla repleta de mensajes de error, algo poco profesional.

Otras muchas funciones PHP relacionadas con MySQL como `mysql_pconnect()`, `mysql_select_db()` y `mysql_query()` también devuelven `false` para indicar que se ha producido un error.

Si se produce un error, puede acceder al texto del mensaje de error por medio de la función `msyql_error()` o a un código de error por medio de `msyql_errno()`. Si la última función MySQL no ha generado un error, `mysql_error()` devuelve una cadena vacía y `msqyl_errno()` devuelve `0`.

Por ejemplo, imagine que se ha conectado al servidor y ha seleccionado una base de datos.

El siguiente fragmento de código:

```
$result = mysql_query( 'select * from does_not_exist' );
echo mysql_errno();
echo '<br />';
```

```
echo  mysql_error();
```

puede dar como resultado:

```
1146
Table 'dbname.does_not_exist' doesn't exist
```

Fíjese en que el resultado de estas funciones hace referencia a la última función MySQL ejecutada (que no sea `mysql_error()` o `mysql_errno()`). Si quiere conocer el resultado de un comando, compruébelo antes de ejecutar otros.

Al igual que los errores de interacción de archivos, también se producen errores de interacción de bases de datos. Incluso después de completar el desarrollo y la prueba de un servicio, descubrirá que el demonio MySQL (`mysqld`) se ha colapsado o se ha quedado sin conexiones disponibles. Si su base de datos se ejecuta en otro equipo físico, dependerá de otro conjunto de componentes de hardware y de software que pueden fallar, es decir, otra red, otra conexión, otra tarjeta de red, otros enrutadores, etc., entre el servidor Web y el equipo de base de datos.

Tendrá que comprobar si las solicitudes de su base de datos se cumplen antes de intentar utilizar el resultado. No tiene sentido intentar ejecutar una consulta si no se puede establecer una conexión a la base de datos, ni tiene sentido extraer y procesar los resultados tras ejecutar una consulta fallida.

Debe saber que existe una diferencia entre una consulta fallida y una consulta que simplemente no puede devolver datos o afectar a ninguna fila.

Una consulta SQL que contenga errores sintácticos SQL o que haga referencia a bases de datos, tablas o columnas que sí existen puede fallar, por ejemplo la siguiente consulta:

```
select * from does_not_exist;
```

falla porque el nombre de la tabla no existe y genera un número y un mensaje de error que se pueden recuperar con `mysql_errno()` y `mysql_error()`.

Una consulta SQL sintácticamente válida y que haga referencia solamente a bases de datos, tablas y columnas que existan no fallará. Sin embargo, puede que no devuelva resultados si se realiza en una tabla vacía o busca datos que no existen. Imagine que se ha conectado correctamente a una base de datos y que tiene una tabla con el nombre `exists` y una columna con el nombre `column name`. La siguiente consulta:

```
select * from exists where column_name = 'not in database';
```

será satisfactoria pero no devolverá ningún resultado.

Antes de utilizar el resultado de la consulta tendrá que comprobar si produce algún error y si devuelve resultados.

Conexiones a redes de servicios

Aunque los dispositivos y otros programas de su sistema pueden fallar ocasionalmente, apenas lo harán a menos que la calidad no sea la correcta. Al utilizar una red

para conectarse a otros equipos y al software de dichos equipos, tendrá que aceptar que parte del sistema fallará. Al conectarse de un equipo a otro, depende de numerosos dispositivos y servicios que no están bajo su control.

Aunque seamos repetitivos, debe comprobar el valor devuelto por las funciones que intenten actuar con un servicio de red.

Una invocación de función como la siguiente

```
$sp = fsockopen ( 'localhost', 5000 );
```

no mostrará ningún mensaje de error si no puede conectarse al puerto 5000 del equipo `localhost`.

Si volvemos a escribir la invocación como

```
$sp = fsockopen ( 'localhost', 5000, &$errorno, &$errorstr );
if(!$sp)
   echo "ERROR: $errorno: $errorstr";
```

se comprobará el valor devuelto para ver si se ha producido un error y se mostrará un mensaje de error que puede ayudarle a solucionar el problema. En este caso, se generaría el siguiente resultado:

```
ERROR: 111: Connection refused
```

Los errores de ejecución son más difíciles de eliminar que los errores sintácticos ya que el analizador no puede detectar el error la primera vez que se ejecuta el código. Como los errores de ejecución se producen como respuesta a una combinación de eventos, su detección y resolución puede resultar complicada. El analizador no puede indicar automáticamente que una línea determinada generará un error. Las pruebas que realice deben tener en cuenta una de las situaciones que generen el error.

El procesamiento de errores de ejecución requiere una cierta planificación: comprobar los diferentes tipos de errores que pueden producirse y, tras ello, emprender la acción adecuada. También es necesario probar todas las clases de errores de ejecución que puedan producirse.

Esto no significa que tenga que simular todos y cada uno de los distintos errores que puedan producirse. Por ejemplo, MySQL puede proporcionar uno de aproximadamente 200 números y mensajes de error diferentes.

Tendrá que simular un error en todas las invocaciones de función que puedan generar errores así como un error para cada tipo que se procese por un bloque de código diferente.

Comprobación de entrada de datos incorrecta

A menudo asumimos la validez de los datos que introducen los usuarios. Si estos datos no coinciden con nuestras expectativas, pueden generar un error, bien de ejecución o bien lógico (que analizaremos en el siguiente apartado).

Un ejemplo clásico de error de ejecución se produce cuando se procesan datos introducidos por los usuarios y olvidamos aplicar `AddSlashes()` a los mismos.

Esto significa que si tenemos un usuario con el nombre O'Grady que contiene un apostrofe, obtendremos un error de la función de base de datos.

En el siguiente apartado, describiremos con más detalle estos errores provocados a causa de suposiciones sobre datos introducidos.

Errores lógicos

Los errores lógicos son el tipo de error más difícil de localizar y eliminar. Este tipo de error se produce cuando un código perfectamente válido hace lo que se le dice, pero no lo que el programador esperaba.

Los errores lógicos se pueden producir a causa de un simple error ortográfico, como por ejemplo:

```
for ( $i = 0; $i < 10; $i++ );
{
   echo 'doing something<br />';
}
```

Este fragmento de código es perfectamente válido. Cumple correctamente la sintaxis de PHP. No depende de servicios externos, por lo que es poco probable que falle durante el tiempo de ejecución. A menos que lo haya revisado minuciosamente, es probable que no haga lo que pensamos o lo que el programador espera.

A simple vista, parece que iterará diez veces el bucle `for`, repitiendo `"doing something"` cada vez.

La inclusión de un punto y coma al final de la primera línea significa que el bucle no afecta a las líneas siguientes. El bucle `for` iterará diez veces sin resultados y, tras ello, la instrucción `echo` se ejecutará una vez.

Como este código es una forma perfectamente válida pero ineficaz de escribir código para obtener este resultado, el analizador no se queja. Los ordenadores son muy inteligentes para algunas cosas pero carecen de sentido común o de inteligencia. Un ordenador hace exactamente lo que se le dice. Tendrá que asegurarse de que lo que le dice es exactamente lo que necesita.

Los errores lógicos no se deben a ningún fallo del código, sino a un fallo del programador que ha escrito el código que indica al ordenador lo que debe hacer. Como resultado, los errores no se pueden detectar automáticamente. No sabremos que se ha producido un error y no tendremos un número de línea en el que buscar el problema. Los errores lógicos sólo se detectan por medio de las pruebas adecuadas.

Un error lógico como el del ejemplo anterior se puede cometer con facilidad, pero también resulta sencillo de corregir ya que la primera vez que se ejecute el código obtendremos un resultado distinto al esperado. La mayoría de los errores lógicos son ligeramente más nocivos.

Los errores lógicos problemáticos suelen deberse a suposiciones erróneos de los programadores. En el capítulo anterior recomendamos la colaboración de otros programadores para revisar nuestro código y sugerir casos de prueba adicionales así como acudir a nuestro público de destino en lugar de a programadores para realizar las pruebas. La suposición de que los usuarios sólo introducen determinados tipos

de datos es muy habitual y también es muy posible pasarlo por alto si realizamos nuestras propias comprobaciones.

Imagine que disponemos de un cuadro de texto para pedidos en un sitio de comercio. ¿Ha supuesto que todos los usuarios han introducido únicamente valores positivos? Si un usuario introduce -10, ¿el software cargará en la tarjeta de crédito del usuario diez veces el precio del artículo?

Imagine que tiene un cuadro para introducir una cantidad en euros. ¿Los usuarios pueden introducir la cantidad con y sin el símbolo del euro? ¿Pueden introducir cifras con millares separadas por puntos? Algunos de estos aspectos se pueden comprobar en el lado del cliente (por ejemplo por medio de JavaScript) para aliviar la carga de nuestro servidor.

Si pasamos información a otra página, puede que haya pensado que en la cadena que estamos pasando existen caracteres con un significado especial en un URL como los espacios.

El número posible de errores lógicos es infinito. No hay una forma automática de comprobar todos ellos. La única solución consiste, en primer lugar, en intentar eliminar suposiciones que implícitamente hayamos codificado en la secuencia de comandos y, tras ello, probar con todos los tipos de entradas válidas y no válidas posibles, asegurándonos de que obtenemos el resultado anticipado de todos ellos.

Ayuda de depuración de variables

Al aumentar la complejidad de los proyectos, puede resultar muy útil disponer de algún código de ayuda para identificar el origen de los errores. En el listado 23.1 incluimos un fragmento de código que puede encontrar de utilidad. Este código duplica los contenidos de las variables pasadas a nuestra página.

Listado 23.1. dump_variables.php. Este código se puede incluir en páginas para volcar los contenidos de las variables para la depuración

```
<?php
   // estas líneas aplican el formato de comentarios HTML al resultado
   // e invocan dump_array repetidamente

   echo '\n<!- BEGIN VARIABLE DUMP ->\n\n';

   echo  '<!- BEGIN GET VARS ->\n';
   echo '<!- '.dump_array($HTTP_GET_VARS).' ->\n';

   echo '<!- BEGIN POST VARS ->\n';
   echo '<!- '.dump_array($HTTP_POST_VARS).' ->\n';

   echo '<!- BEGIN SESSION VARS ->\n';
   echo '<!- '.dump_array($HTTP_SESSION_VARS).' ->\n';

   echo '<!- BEGIN COOKIE VARS ->\n';
   echo  '<!-  '.dump_array($HTTP_COOKIE_VARS).'  ->\n';
```

```
    echo '\n<!- END VARIABLE DUMP ->\n';

// dump_array() adopta una matriz como parámetro
// Procesa una iteración por dicha matriz y crea una cadena
// para representar la matriz como conjunto

function dump_array($array)
{
    if(is_array($array))
    {
        $size = count($array);
        $string = '';
        if($size)
        {
            $count = 0;
            $string .= '{ ';
            // añada la clave y el valor de cada elemento a la matriz
            foreach($array as $var => $value)
            {
                $string .= "$var = $value";
                if($count++ < ($size-1))
                {
                    $string .= ', ';
                }
            }
            $string .= ' }';
        }
        return $string;
    }
    else
    {
        // si no es una matriz, devuélvalo
        return $array;
    }
}
?>
```

Este código procesará una iteración por cuatro matrices de variables que recibe la página. Si la página se ha invocado con variables GET, variables POST, cookies o tiene variables de sesión, dará como resultado todos estos elementos. Se generará una línea de HTML para cada tipo de variable. Las líneas se parecerán a la siguiente:

```
<!- { var1 = 'valor1', var2 = 'valor2', var3 = 'valor3' } ->
```

Tendremos que incluir el resultado dentro de un comentario HTML para que se pueda ver, pero que no interfiera con la forma en la que el navegador representa los elementos de página visibles. Se trata de una buena técnica para generar información de depuración. El resultado exacto dependerá de las variables pasadas a la página, pero si lo añadimos a uno de los ejemplos de autenticación de un capítulo anterior, incluye las siguientes líneas al HTML generado por la secuencia de comandos:

```
<!- BEGIN VARIABLE DUMP ->

<!- BEGIN GET VARS ->
```

```
<!--
<!-- BEGIN POST VARS -->
<!-- { userid = testuser, password = test123 } -->
<!-- BEGIN SESSION VARS -->
<!-- { valid_user = testuser } -->
<!-- BEGIN COOKIE VARS -->
<!-- { PHPSESSID = 2552dc82bb465af56d65e9309f75fd68 } -->

<!-- END VARIABLE DUMP -->
```

Apreciará que muestra las variables POST enviadas desde el formulario de inicio de sesión de la página anterior: `userid` y `password`. También muestra la variable de sesión que utilizamos para almacenar el nombre del usuario: `valid_user`. Como vimos en un capítulo anterior, PHP utiliza una cookie para vincular variables de sesión a determinados usuarios. Nuestra secuencia de comandos duplica el número pseudo aleatorio, `PHPSESSID`, que se almacena en dicha cookie para identificar a un determinado usuario.

Niveles de informes de errores

PHP nos permite especificar la claridad de los errores. Podemos modificar los tipos de eventos que generarán mensajes. De forma predeterminada, PHP informa de todos los errores que no sean avisos.

El nivel de informes de error se establece por medio de una serie de constantes predefinidas, como se indica en la tabla 23.1.

Tabla 23.1. Constantes de informes de errores

Valor	Nombre	Significado
1	`E_ERROR`	Informa de errores graves del tiempo de ejecución
2	`E_WARNING`	Informa de errores no graves del tiempo de ejecución
4	`E_PARSE`	Informa de errores de análisis
8	`E_NOTICE`	Informa de avisos, notificaciones de que algo que hemos hecho es un error
16	`E_CORE_ERROR`	Informa de fallos al iniciarse el motor de PHP
32	`E_CORE_WARNING`	Informa de fallos no graves durante el inicio del motor de PHP
64	`E_COMPILE_ERROR`	Informa de errores de compilación
128	`E_COMPILE_WARNING`	Informa de errores de compilación no graves

Valor	Nombre	Significado
256	E_USER_ERROR	Informa de errores desencadenados por el usuario
512	E_USER_WARNING	Informa de advertencias desencadenadas por el usuario
1024	E_USER_NOTICE	Informa de avisos desencadenados por el usuario
2048	E_ALL	Informa de todos los errores y advertencias

Cada constante representa un tipo de error que se puede mencionar o ignorar. Por ejemplo, si especificamos el nivel de error E_ERROR, solamente se informará de errores graves.

Estas constantes se pueden combinar por medio de aritmética binaria para generar distintos niveles de error.

El nivel de error predeterminado, el informe de todos los errores que no sean avisos, se especifica como

```
E_ALL & ~E_NOTICE
```

Esta expresión está formada por dos de las constantes predefinidas combinadas con operadores aritméticos en orden de bits.

El símbolo & es el operador AND en orden de bits y el símbolo ~ el operador NOT en orden de bits. Esta expresión se puede leer como E_ALL AND NOT E_NOTICE.

E_ALL es, por sí mismo, una combinación de todos los tipos de error restantes. Se podría sustituir por los otros niveles combinados por medio del operador OR en orden de bits (|).

```
E_ERROR | E_WARNING | E_PARSE | E_NOTICE | E_CORE_ERROR | E_CORE_WARNING |
E_COMPILE_ERROR |E_COMPILE_WARNING | E_USER_ERROR | E_USER_WARNING |
E_USER_NOTICE
```

Del mismo modo, el nivel de informe de error predeterminado se podría especificar por todos los errores a excepción de avisos combinados con OR.

```
E_ERROR | E_WARNING | E_PARSE | E_CORE_ERROR | E_CORE_WARNING | E_COMPILE_ERROR
|E_COMPILE_WARNING | E_USER_ERROR | E_USER_WARNING | E_USER_NOTICE
```

Modificación de los parámetros de informes de error

Podemos configurar los parámetros de informes de error globalmente en el archivo php.ini o por secuencia de comandos.

Para modificar el informe de errores en todas las secuencias de comando, puede cambiar estas cuatro líneas en el archivo `php.ini` predeterminado:

```
error_reporting  =     E_ALL & ~E_NOTICE
display_errors   =     On
log_errors       =     Off
track_errors     =     Off
```

Los parámetros globales predeterminados son para:

- Informar de todos los errores menos de avisos
- Mostrar mensajes de error como HTML en resultado estándar
- No registrar mensajes de error en disco
- No realizar el seguimiento de errores y almacenar el error en la variable

El cambio que seguramente realice es convertir el nivel de informe de errores a `E_ALL`. Esto provocará que se informe de numerosos avisos, de incidentes que pueden indicar un error o pueden deberse a que el programador se aproveche de la débil naturaleza de PHP y del hecho de que automáticamente inicializa las variables a `0`.

Durante la depuración, puede resultarle útil establecer un nivel `error_reporting` superior. En código de producción, si proporciona mensajes de errores propios, resulta más profesional desactivar `display_errors` y activar `log_errors`, a la vez que se mantiene el nivel `error_reporting` con un valor elevado. De esta forma podrá hacer referencia a errores detallados en los registros si aparece algún problema.

Al activar `track_errors` podemos controlar errores en nuestro propio código, en lugar de dejar que PHP nos ofrezca su funcionalidad predeterminada. Aunque PHP cuenta con mensajes de error muy útiles, su comportamiento predeterminado no es agradable cuando algo sale mal.

De forma predeterminada, cuando se produce un error grave, PHP ofrece el siguiente resultado:

```
<br>
<b>Tipo de error</b>:  mensaje de error en <b>ruta/archivo.php</b>
en línea <b>númeroLínea</b><br>
```

y detiene la ejecución de la secuencia de comandos. Si se trata de errores no graves, se muestra el mismo texto, pero se permite que prosiga la ejecución.

Este resultado HTML resalta el error, pero su aspecto es pobre. Es poco probable que el estilo del mensaje de error encaje con el del sitio. Puede que los usuarios de Netscape no vean ningún resultado si el contenido de la página se muestra dentro de una tabla. Esto se debe a que el HTML que abre pero no cierra elementos de tabla, como por ejemplo

```
<table>
 <tr><td>
```

```
 <br>
 <b>Tipo de error</b>:  mensaje de error en <b>ruta/archivo.php</b>
en línea <b>NúmeroLínea</b><br>
```

se representa en forma de pantalla en blanco en Netscape.

No es necesario conservar el comportamiento de procesamiento de errores predeterminado de PHP ni siquiera utilizar los mismos parámetros en todos los archivos. Para cambiar el nivel de informe de errores en la secuencia de comandos actual, podemos invocar la función `error_reporting()`.

Al pasar una constante de informe de error o una combinación de constantes, se establece el nivel de la misma forma en la que lo hace la directiva similar en `php.ini`. La función devuelve el nivel de informe de error anterior. Una forma habitual de utilizar esta función es la siguiente:

```
// desactive el informe de errores
$old_level = error_reporting(0);
// añada aquí el código que genera advertencias
// vuelva a activar el informe de errores
error_reporting($old_level);
```

Este fragmento de código desactiva los informes de error, lo que nos permite ejecutar código que pueda generar advertencias que no queremos ver.

No es aconsejable desactivar permanentemente los informes de error ya que dificulta la localización y corrección de los errores del código.

Cómo desencadenar nuestros propios errores

La función `trigger_error()` se puede utilizar para desencadenar nuestros propios errores. Los errores creados de esta forma se procesan de la misma forma que los errores PHP convencionales.

La función requiere un mensaje de error y, opcionalmente, se le puede asignar un tipo de error. El tipo de error debe ser `E_USER_ERROR`, `E_USER_WARNING` o `E_USER_NOTICE`. Si no se especifica uno, el predeterminado es `E_USER_NOTICE`.

La forma de utilizar `trigger_error()` es la siguiente:

```
trigger_error('This computer will self destruct in 15 seconds',
E_USER_WARNING);
```

(Esta función se añadió en la versión 4.0.1 de PHP.)

Procesamiento correcto de errores

Si tiene conocimientos de C++ o de Java, puede que eche de menos el procesamiento de excepciones cuando utilice PHP. Las excepciones permiten que las funciones indiquen que se ha producido un error y dejan la resolución del mismo a un

controlador de excepciones. Aunque PHP no cuenta con excepciones, en la versión 4.0.1 se presentó un mecanismo que se podía utilizar de forma similar.

Ya hemos visto cómo podemos desencadenar nuestros propios errores. También podemos utilizar nuestros propios controladores de error para capturar errores.

La función `set_error_handler()` nos permite proporcionar una función que se invocará cuando se produzcan errores en el nivel de usuario, advertencias y avisos. Podemos configurar `set_error_handler()` con el nombre de la función que queramos utilizar como controlador de errores.

Nuestra función de procesamiento de errores debe adoptar dos parámetros: un tipo de error y un mensaje de error. En función de estas dos variables, la función puede decidir cómo procesar el error. El tipo de error debe ser una de las constantes de tipo de error definidas. El mensaje de error es una cadena descriptiva.

Una llamada a `set_error_handler()` tiene este aspecto:

```
set_error_handler('myErrorHandler');
```

Después de indicar a PHP que utilice una función con el nombre `myErrorHandler()`, debemos proporcionar una función con dicho nombre, que debe tener el siguiente prototipo:

```
myErrorHandler(int tipo_error, string mensaje_error)
```

pero cuyo funcionamiento depende de nosotros.

Entre las acciones lógicas destacamos las siguientes:

- Mostrar el mensaje de error proporcionado
- Almacenar información en un archivo de registro
- Enviar el error por correo electrónico a una dirección
- Finalizar la secuencia de comandos con una llamada para salir

En el listado 23.2 se incluye una secuencia de comandos que declara un controlador de error, lo configura por medio de `set_error_handler()` y, tras ello, genera algunos errores.

Listado 23.2. handle.php. Esta secuencia de comandos declara un controlador de error personalizado y genera distintos errores

```
<?php
   // La función de control de error
   function myErrorHandler ($errno, $errstr, $errfile, $errline)
   {
      echo "<br /><table bgcolor='#cccccc'><tr><td>
              <p><b>ERROR:</b> $errstr</p>
              <p>Please try again, or contact us and tell us that
              the error occurred in line $errline of file '$errfile'</p>";
      if ($errno == E_USER_ERROR||$errno == E_ERROR)
      {
         echo '<p>This error was fatal, program ending</p>';
```

```
            echo '</td></tr></table>';
            //cierre los recursos abiertos, incluyendo el pie de página, etc
            exit;
        }
        echo '</td></tr></table>';
    }
    // Defina el controlador de error
    set_error_handler('myErrorHandler');

    //desencadene distintos niveles de error
    trigger_error('Trigger function called', E_USER_NOTICE);
    fopen('nofile', 'r');
    trigger_error('This computer is beige', E_USER_WARNING);
    include ('nofile');
    trigger_error('This computer will self destruct in 15 seconds',
E_USER_ERROR);
?>
```

El resultado de esta secuencia de comandos se muestra en la figura 23.1.

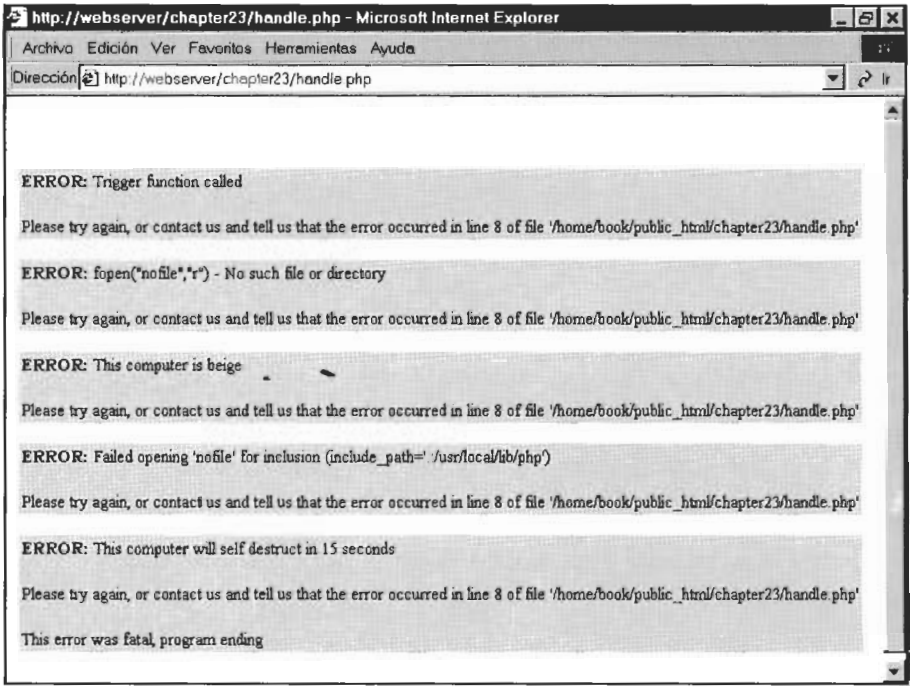


Figura 23.1. Si utilizamos nuestro propio controlador de errores podemos mostrar mensajes de error más amistosos que los de PHP

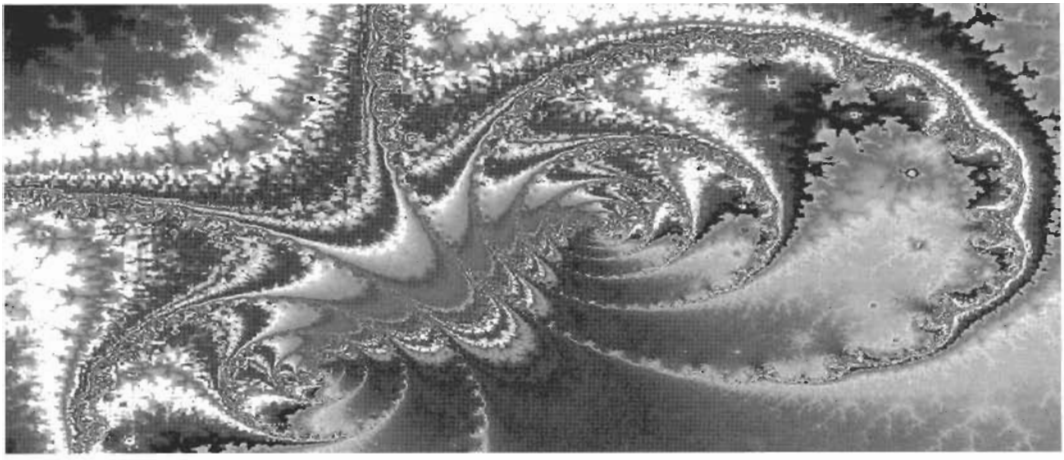
Este controlador de errores personalizado no hace nada más aparte del comportamiento predeterminado. Como hemos escrito este código personalmente, puede hacer cualquier cosa. Nos permite elegir lo que los usuarios verán cuando se pro-

duzca un error y cómo presentar esta información de forma que se ajuste al resto del sitio. Es más, nos permite decidir lo que sucede: que prosiga la secuencia de comandos, que el mensaje se registre o se muestre, o que se avise automáticamente a la asistencia técnica.

Debe saber que su controlador de errores no es responsable de procesar todos los tipos de errores. Algunos, como los errores de análisis y los errores de ejecución graves, desencadenan el comportamiento predeterminado. Si este aspecto le preocupa, debe revisar atentamente los parámetros antes de pasarlos a funciones que puedan generar errores graves y desencadenar su propio nivel de errores `E_USER_ERROR` si sus parámetros pueden causar algún fallo.

A continuación

En el siguiente capítulo sobre autenticación y personalización de usuarios, comenzaremos a generar nuestro primer proyecto. En éste, aprenderemos a reconocer usuarios que regresan a nuestro sitio y a modificar nuestros contenidos apropiadamente.



24

Cómo generar autenticación y personalización de usuarios

En este proyecto haremos que los usuarios se registren en nuestro sitio. Tras ello, podremos saber en qué están interesados y podremos mostrarles el contenido apropiado. Es lo que se denomina personalización de usuarios.

Este proyecto en concreto permitirá a los usuarios generar una serie de marcadores en la Web así como sugerir otros enlaces que les puedan interesar en función de su comportamiento anterior.

De forma más general, la personalización de usuarios se puede utilizar prácticamente en cualquier aplicación Web para mostrar a los usuarios el contenido que quieren en el formato que desean.

En este proyecto, y en los posteriores, comenzaremos con una serie de requisitos similares a los que se reciben de un cliente.

Convertiremos estos requisitos en un conjunto de componentes de la solución, generaremos un diseño para combinar dichos componentes y, tras ello, los implementaremos.

En este proyecto, implementaremos la siguiente funcionalidad:

- Conexión y autenticación de usuarios
- Administración de contraseñas
- Registro de las preferencias de los usuarios
- Personalización de contenidos
- Recomendación de contenidos en función de la información existente sobre un usuario

El problema

Nuestro objetivo es generar un prototipo para un sistema de creación de marcadores en línea, que denominaremos PHPBookmark y que es similar (pero con menor funcionalidad) al disponible en Backflip: `http://backflip.com`.

Nuestro sistema permitirá conectarse a los usuarios y almacenar sus marcadores personales así como obtener recomendaciones de otros sitios que puedan estar interesados en visitar en función de sus preferencias personales.

Estos requisitos de solución se agrupan en tres categorías principales.

En primer lugar, es necesario poder identificar a los usuarios individualmente. Necesitamos algún método de autenticación.

En segundo lugar, necesitamos poder almacenar marcadores de un determinado usuario. Los usuarios deben tener la opción de añadir y eliminar marcadores.

Por último, necesitamos poder recomendar a los usuarios sitios en los que puedan estar interesados, en función de lo que conozcamos de los usuarios.

Componentes de la solución

Una vez determinados los requisitos del sistema, podemos empezar a diseñar la solución y sus componentes. Veremos posibles soluciones para los tres requisitos principales que acabamos de enumerar.

Identificación y personalización de usuarios

Existen distintas alternativas para autenticar a los usuarios, como ya hemos visto antes en el libro. Como queremos vincular a un usuario a cierta información personalizada, almacenaremos la conexión y la contraseña del mismo en una base de datos MySQL y realizaremos la autenticación en función de la base de datos.

Si vamos a permitir que los usuarios se registren con un nombre de usuario y una contraseña, necesitaremos los siguientes componentes:

- Los usuarios deben poder registrarse con un nombre de usuario y una contraseña. Será necesario restringir de algún modo la longitud y el formato de estos datos. Tendremos que almacenar las contraseñas en un formato codificado por motivos de seguridad.
- Los usuarios deben poder conectarse con los detalles que hayan proporcionado durante el proceso de registro.
- Los usuarios deben poder desconectarse cuando hayan terminado de utilizar el sitio. No es un aspecto muy importante si utilizan el sitio desde su PC doméstico, pero sí, en cuanto a seguridad se refiere, si lo utilizan desde un PC compartido.

- El sitio debe poder comprobar si un usuario está conectado o no, y acceder a datos del usuario conectado.
- Los usuarios deben poder cambiar su contraseña como medida de seguridad.
- Los usuarios olvidarán sus contraseñas. Deben poder restablecerlas sin necesidad de que les ayudemos personalmente. Una forma habitual de hacerlo consiste en enviar al usuario la contraseña a la dirección de correo electrónico que haya indicado durante el registro. Esto significa que tendremos que almacenar su dirección de correo electrónico durante el registro. Como almacenamos las contraseñas codificadas y no podemos descodificar la contraseña original, tendremos que generar una nueva, configurarla y enviársela al usuario.

Escribiremos funciones para todas estas opciones. La mayoría de ellas se podrán reutilizar, o al menos con mínimos cambios, en otros proyectos.

Almacenamiento de marcadores

Para almacenar los marcadores de un usuario, necesitamos definir algún espacio en nuestra base de datos MySQL. También necesitaremos la siguiente funcionalidad:

- Los usuarios deben poder recuperar y ver sus marcadores.
- Los usuarios deben poder añadir nuevos marcadores. Debemos comprobar que se trate de URL válidos.
- Los usuarios deben poder eliminar marcadores.

Como dijimos antes, escribiremos funciones para cada una de estas opciones.

Recomendación de marcadores

Podemos adoptar diferentes enfoques para recomendar marcadores a un usuario. Podríamos recomendar el más popular o los más populares de un determinado tema. Para este proyecto, implementaremos un sistema de sugerencias "afines" que busque usuarios que tengan el mismo marcador que nuestro usuario conectado y sugieran sus marcadores a nuestro usuario. Para evitar la recomendación de marcadores personales, solamente recomendaremos marcadores almacenados por más de un usuario.

De nuevo, podemos escribir una función para implementar esta funcionalidad.

Presentación de la solución

Después de realizar algunos bocetos, hemos obtenido el diagrama de flujo de sistema que mostramos en la figura 24.1.

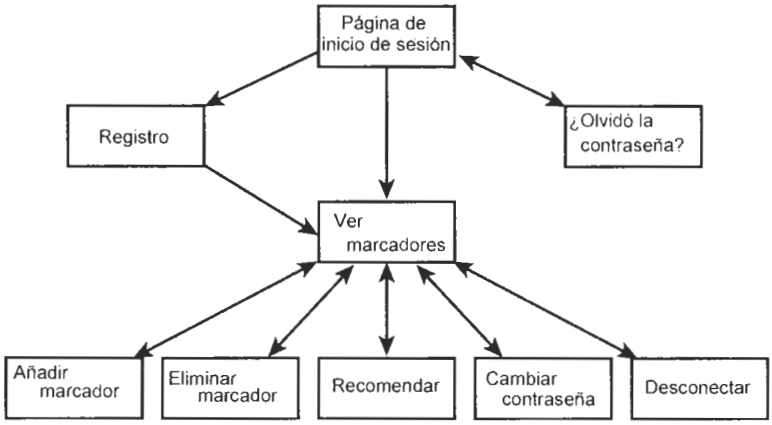


Figura 24.1. Este diagrama muestra las posibles rutas del sistema PHPBookmark

Generaremos un módulo para cada cuadro de este diagrama; algunos necesitarán una secuencia de comandos y otros dos. También definiremos bibliotecas de funciones para:

- Autenticación de usuarios
- Almacenamiento y recuperación de marcadores
- Validación de datos
- Conexiones a bases de datos
- Representación en el navegador. Incluiremos todo el código HTML en esta biblioteca de funciones, asegurándonos de que la presentación visual es coherente a lo largo del sitio (se trata del enfoque de API de funciones para separar lógica y contenido).

También será necesario diseñar una base de datos para el sistema.

Analizaremos la solución con más detalle, pero todo el código de esta aplicación se incluye en el archivo correspondiente a este capítulo del CD-ROM. En la tabla 24.1 se recoge un resumen de los archivos incluidos.

Tabla 24.1. Archivos de la aplicación PHPBookmark

Nombre del archivo	Descripción
`bookmarks.sql`	Instrucciones SQL para crear la base de datos PHP Bookmark
`login.php`	Página principal con el formulario de conexión al sistema
`register_form.php`	Formulario para que los usuarios se registren en el sistema

Nombre del archivo	Descripción
`register_new.php`	Secuencia de comandos para procesar nuevos registros
`forgot_form.php`	Formulario que deben completar los usuarios si han olvidado su contraseña
`forgot_passwd.php`	Secuencia de comandos para restaurar contraseñas olvidadas
`member.php`	Página principal de un usuario, con una vista de todos sus marcadores actuales
`add_bm_form.php`	Formulario para añadir nuevos marcadores
`add_bms.php`	Secuencia de comandos para añadir los nuevos marcadores a la base de datos
`delete_bms.php`	Secuencia de comandos para eliminar marcadores seleccionados de la lista del usuario
`recommend.php`	Secuencia de comandos para sugerir recomendaciones a un usuario, en función de usuarios con intereses similares
`change_passwd_form.php`	Formulario que deben completar los miembros para cambiar sus contraseñas
`change_passwd.php`	Secuencia de comandos para cambiar la contraseña de un usuario en la base de datos
`logout.php`	Secuencia de comandos para desconectar a un usuario de la aplicación
`bookmark_fns.php`	Colección de archivos de inclusión para la aplicación
`data_valid_fns.php`	Funciones para validar datos introducidos por el usuario
`db_fns.php`	Funciones para conectarse a la base de datos
`user_auth_fns.php`	Funciones para la autenticación de usuarios
`url_fns.php`	Funciones para añadir y eliminar marcadores , y para hacer recomendaciones
`output_fns.php`	Funciones para aplicar formato HTML a los resultados
`bookmark.gif`	Logotipo de PHPBookmark

En primer lugar implementaremos la base de datos MySQL para esta aplicación ya que la necesitaremos prácticamente con todas las opciones restantes para que funcionen.

Tras ello, analizaremos el código en el mismo orden en el que se ha escrito, empezando desde la página inicial, pasando por la autenticación de usuarios, el almacenamiento y recuperación de marcadores y, por último, las recomendaciones. Este orden es bastante lógico; basta con definir las dependencias y generar en primer lugar los elementos que necesitaremos para los módulos posteriores.

Nota

Para que el código de este proyecto funcione como está escrito, tendrá que activar las comillas mágicas. Si no lo ha hecho anteriormente, tendrá que añadir `addslashes()` a los datos que incluya en la base de datos y `stripslashes()` en los datos recibidos de la base de datos. Lo hemos utilizado como método abreviado.

Implementación de la base de datos

Para la base de datos PHPBookmark sólo necesitamos un esquema bastante sencillo. Tendremos que almacenar usuarios así como sus direcciones de correo electrónico y contraseñas. También será necesario almacenar el URL de un marcador. Un usuario puede tener varios marcadores y varios usuarios pueden registrar el mismo marcador. Por lo tanto, tendremos dos tablas, `user` y `bookmark`, como se indica en la figura 24.2.

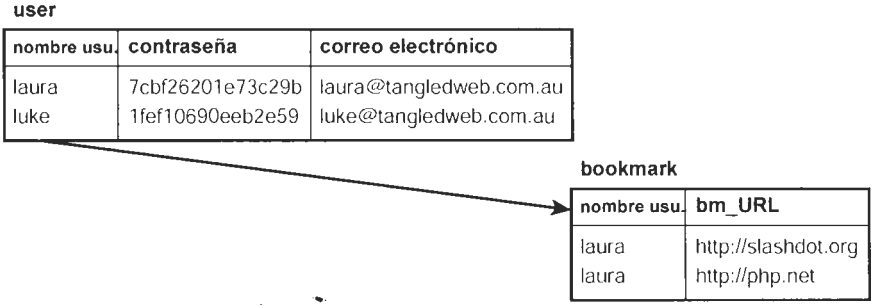


Figura 24.2. Esquema de la base de datos para el sistema PHPBookmark

La tabla `user` almacenará el nombre de usuario (que es la clave principal), la contraseña y la dirección de correo electrónico.

La tabla `bookmark` almacenará pares de nombre de usuario y marcador (`bm_URL`). El nombre de usuario de esta tabla hará referencia al nombre de usuario de la tabla `user`. El SQL para crear esta base de datos, y para crear un usuario para realizar la conexión a la base de datos desde la Web, se muestra en el listado 24.1. Debería modificarlo si piensa utilizarlo en su sistema y cambiar la contraseña del usuario por algo más seguro.

Listado 24.1. bookmarks.sql. Archivo SQL para configurar la base de datos de marcadores

```
create database bookmarks;
use bookmarks;

create table user (
```

```
    username  varchar(16)  primary  key,
    passwd  char(16)  not  null,
    email  varchar(100)  not  null
);

create  table  bookmark  (
    username varchar(16) not null,
    bm_URL  varchar(255)  not  null,
    index (username),
    index  (bm_URL)
);

grant select, insert, update, delete
on bookmarks.*
to bm_user@localhost identified by 'password';
```

Puede configurar esta base de datos en su sistema si ejecuta este conjunto de comandos como el usuario raíz de MySQL. Puede hacerlo con el siguiente comando en la línea de comandos de su sistema:

```
mysql  -u  root  -p  <  bookmarks.sql
```

Tras ello, le instará a que introduzca su contraseña.

Una vez configurada la base de datos, implementaremos el sitio básico.

Implementación del sitio básico

La primera página que generaremos será `login.php`, que permite a los usuarios conectarse al sistema.

El código de esta primera página se incluye en el listado 24.2.

Listado 24.2. login-php. Página inicial del sistema PHPBookmark

```
<?php
 require_once('bookmark_fns.php');
 do_html_header('');

 display_site_info();
 display_login_form();

 do_html_footer();
?>
```

Este código es muy sencillo. Básicamente invoca funciones del API de funciones que construiremos para esta aplicación. Veremos los detalles de estas funciones en breve.

Con tan sólo observar el archivo, comprobamos que se incluye otro (que contiene las funciones) y tras ello se invocan algunas funciones para representar un encabezado HTML, mostrar contenidos y representar un pie de página HTML.

El resultado de esta secuencia de comandos se muestra en la figura 24.3.

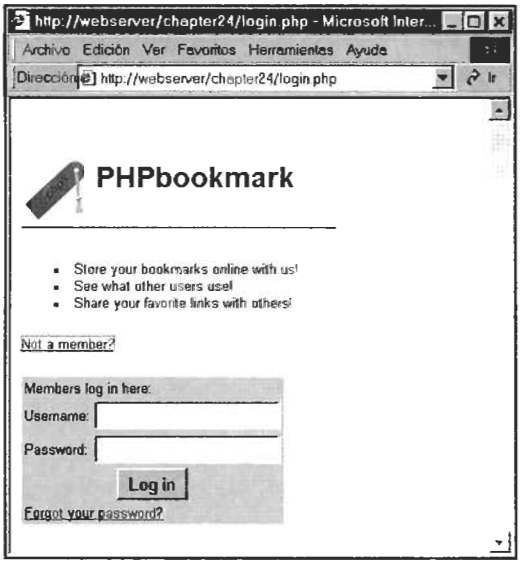


Figura 24.3. La página inicial del sistema PHPBookmark se genera por medio de las funciones de representación HTML de login.php

Todas las funciones del sistema se incluyen en el archivo `bookmark_fns.php`, que mostramos en el listado 24.3.

Listado 24.3. bookmark_fns.php. Archivo que incluye las funciones de la aplicación de marcadores

```
<?php
   // Podemos incluir este archivo en todos nuestros archivos
   // de esta forma, todos los archivos contendrán todas nuestras funciones
   require_once('data_valid_fns.php');
   require_once('db_fns.php');
   require_once('user_auth_fns.php');
   require_once('output_fns.php');
   require_once('url_fns.php');
?>
```

Como puede comprobar, el archivo es un contenedor de los otros cinco archivos que utilizaremos en la aplicación. Lo hemos estructurado de esta forma ya que las funciones se agrupan en categorías lógicas. Algunas de estas categorías pueden resultarle útiles para otros proyectos, por lo que hemos incluido cada grupo de funciones en un archivo distinto, para que sepa dónde encontrarlas cuando las necesite.

Hemos creado el archivo `bookmark_fns.php` porque utilizaremos la mayoría de los cinco archivos de funciones en nuestras secuencias de comandos. Resulta más sencillo incluir este archivo en cada secuencia de comandos que tener cinco instrucciones de inclusión de archivos.

Fíjese en que la construcción `require_once()` sólo existe en PHP desde la versión 4.0.1pl2.

Si está utilizando una versión anterior, tendrá que utilizar `require()` o `include()` y comprobar que los archivos no se cargan varias veces.

En este caso en concreto, utilizaremos funciones del archivo `output_fns.php`. Son funciones muy sencillas que dan como resultado HTML sencillo. Este archivo incluye las cuatro funciones que hemos utilizado en `login.php`, es decir, `do_html_header()`, `display_site_info()`, `display_login_form()` y `do_html_footer()` entre otras.

No analizaremos todas estas funciones detalladamente sino que las veremos en un ejemplo. El listado 24.4 incluye el código de `do_html_header()`.

Listado 24.4. Función do_hmtl_header() de output_fns.php. Esta función representa el encabezado estándar que aparecerá en todas las páginas de la aplicación

```
function do_html_header($title)
{
   // imprima un encabezado HTML
?>
   <html>
   <head>
      <title><?php echo $title;?></title>
      <style>
         body { font-family: Arial, Helvetica, sans-serif; font-size: 13px }
         li, td { font-family: Arial, Helvetica, sans-serif; font-size: 13px
}
         hr { color: #3333cc; width=300; text-align=left}
         a { color: #000000 }
      </style>
   </head>
   <body>
   <img src="bookmark.gif" alt="PHPbookmark logo" border="0"
         align="left" valign="bottom" height="55" width="57" />
   <h1> PHPbookmark</h1>
   <hr />
<?php
   if($title)
      do_html_heading($title);
}
```

Como puede apreciar, la única lógica de esta función consiste en añadir el correspondiente título y encabezado a la página. Las otras funciones que hemos utilizado en `login.php` son similares. La función `display_site_info()` añade texto general sobre el sitio; `display_login_form()` muestra el formulario de color gris de la figura 24.3 y `do_html_footer()` añade un pie de página HTML estándar a la página.

Las ventajas de aislar o eliminar HTML del flujo de lógica normal se analizaron en un capítulo anterior. En este caso utilizaremos el enfoque de API de funciones y, en el siguiente capítulo, el enfoque basado en plantillas, para ver el contraste.

Si observa la figura 24.3 comprobará que en esta página hay tres opciones: los usuarios pueden registrarse, conectarse si ya se han registrado o restablecer su contraseña si la han olvidado. Para implementar estos módulos, pasaremos al siguiente apartado, la autenticación de usuarios.

Implementación de la autenticación de usuarios

Existen cuatro elementos principales para el módulo de autenticación de usuarios: registro de usuarios, conexión y desconexión, modificación de contraseñas y restablecimiento de contraseñas. Veremos cada uno de ellos con mayor detalle.

Registro

Para registrar a un usuario, necesitamos obtener sus detalles por medio de un formulario e introducirlos en la base de datos. Cuando el usuario pulse sobre el enlace "Not a member?" de la página `login.php`, accederá a un formulario de registro generado por `register_form.php`. Esta secuencia de comandos se muestra en el listado 24.5.

Listado 24.5. register_form.php. Este formulario permite a los usuarios registrarse en PHPBookmarks

```
<?php
   require_once('bookmark_fns.php');
   do_html_header('User Registration');

   display_registration_form();

   do_html_footer();
?>
```

Como en el caso anterior, comprobará que es una página muy sencilla y que sólo invoca funciones de la biblioteca de resultados de `output_fns.php`. El resultado de esta secuencia de comandos se muestra en la figura 24.4.

Figura 24.4. El formulario de registro recupera los detalles necesarios para la base de datos. Los usuarios deben escribir su contraseña dos veces, por si se equivocan

El formulario de color gris de esta página se representa por medio de la función `display_registration_form()`, incluida en `output_fns.php`. Cuando el usuario pulsa el botón **Register**, accede a la secuencia de comandos `register_new.php`, secuencia que se recoge en el listado 24.6.

Listado 24.6. register_new.php. Esta secuencia de comandos valida los nuevos datos del usuario y los añade a la base de datos

```
<?php
    // incluya archivos de funciones para esta aplicación
    require_once('bookmark_fns.php');

   //cree nombres de variable cortos
   $email=$HTTP_POST_VARS['email'];
```

```
    $username=$HTTP_POST_VARS['username'];
    $passwd=$HTTP_POST_VARS['passwd'];
    $passwd2=$HTTP_POST_VARS['passwd2'];
     // inicie una sesión que puede necesitar más tarde
     // iníciela ahora ya que debe ir antes de los encabezados
      session_start();


     // compruebe que los formularios están completados
      if  (!filled_out($HTTP_POST_VARS))
     {
         do_html_header('Problem:');
         echo 'You have not filled the form out correctly - please go
back'
              .' and try again.';
         do_html_footer();
         exit;
     }

     // dirección de correo electrónica no válida
      if  (!valid_email($email))
     {
         do_html_header('Problem:');
         echo 'That is not a valid email address. Please go back '
              .' and try again.';
         do_html_footer();
         exit;
     }

     // las contraseñas no coinciden
     if  ($passwd != $passwd2)
     {
         do_html_heading('Problem:');
         echo 'The passwords you entered do not match - please go
back'
              .' and try again.';
         do_html_footer();
         exit;
     }

     // compruebe que la longitud de las contraseñas es correcta
     // el nombre de usuario puede estar incompleto, pero las contraseñas se
     // enredan si son demasiado largas.
      if  (strlen($passwd)<6  ||  strlen($passwd)  >16)
     {
         do_html_header('Problem:');
         echo 'Your password must be between 6 and 16 characters.'
              .'Please go back and try again.';
         do_html_footer();
         exit;
     }
     // intente registrar
      $reg_result  =  register($username,  $email,  $passwd);
     if  ($reg_result  ===  true)
     {
```

```
        // registre la variable de sesión
         $HTTP_SESSION_VARS['valid_user']  =  $username;


        // ofrezca un enlace a la página de miembros
        do_html_header('Registration successful');
        echo 'Your registration was successful.  Go to the members page '
             .'to start setting up your bookmarks!';
        do_html_url('member.php', 'Go to members page');
    }
    else
    {
        // en caso contrario, proporcione un enlace hacia atrás, indicando
que prueben de nuevo
        do_html_header('Problem:');
        echo $reg_result;
        do_html_footer();
        exit;
    }

    // finalice la página
    do_html_footer();

?>
```

Es la primera secuencia de comandos con cierta complejidad que hemos visto en esta aplicación.

La secuencia comienza incluyendo los archivos de funciones de la aplicación e inicia una sesión (cuando el usuario está registrado, crearemos su nombre de usuario como variable de sesión, como hicimos en un capítulo anterior).

Tras ello, validamos los datos introducidos por el usuario. Existen varias condiciones que debemos probar:

- Comprobar que el formulario se ha completado. Para probarlo, invocamos la función `filled_out()` de esta forma:

  ```
  if  (!filled_out($HTTP_POST_VARS))
  ```

  Es una de las funciones que hemos escrito personalmente. Se encuentra en la biblioteca de funciones del archivo `data_valid_fns.php`. La analizaremos más adelante.

- Comprobar que la dirección de correo electrónico proporcionada es válida. Lo probaremos de esta forma:

  ```
  if  (valid_email($email))
  ```

  De nuevo, es una de las funciones que hemos escrito, de la biblioteca `data_valid_fns.php`.

- Comprobar que las dos contraseñas sugeridas por el usuario son idénticas, como se indica a continuación:

  ```
  if  ($passwd  !=  $passwd2)
  ```

- Comprobar que la contraseña tiene la longitud correcta:

```
if (strlen($passwd)<6 || strlen($passwd) >16)
```

  En nuestro ejemplo, la contraseña debe tener al menos 6 caracteres para que resulte difícil descubrirla y menos de 16 para que se ajuste a la base de datos.

Las funciones de validación de datos que hemos utilizado, `filled_out()` y `valid_email()`, se muestran en los listados 24.7 y 24.8 respectivamente.

Listado 24.7. Función filled_out() de data_valid_fns.php. Esta función comprueba que el formulario se ha completado

```
function  filled_out($form_vars)
{
   // compruebe que cada variable tiene un valor
   foreach ($form_vars as $key => $value)
   {
       if (!isset($key) || ($value == ''))
            return false;
   }
   return true;
}
```

Listado 24.8. Función valid_email() de data_valid_fns.php. Esta función comprueba si la dirección de correo electrónico es válida

```
function valid_email($address)
{
   // compruebe que una dirección de correo es válida
   if (ereg('^[a-zA-Z0-9_\.\-]+@[a-zA-Z0-9\-]+\.[a-zA-Z0-9\-\.]+$',
$address))
      return true;
   else
      return false;
}
```

La función `filled_out()` espera recibir una matriz de variables que, por lo general, serán las matrices `$HTTP_POST_VARS` o `$HTTP_GET_VARS`. Comprueba si están completadas y devuelve `true` si lo están y `false` en caso contrario.

La función `valid_email()` utiliza las expresiones regulares que desarrollamos en uno de los primeros capítulos para validar direcciones de correo electrónico. Devuelve `true` si la dirección parece válida y `false` en caso contrario.

Una vez validados los datos introducidos, podemos intentar el registro del usuario. Si observa el listado 24.6, comprobará que se hace de esta forma:

```
$reg_result = register($username, $email, $passwd);
if ($reg_result === true)
{
   // registre la variable de sesión
   $HTTP_SESSION_VARS['valid_user']  =  $username;

   // ofrezca un enlace a la página de miembros
```

```
    do_html_header('Registration successful');
    echo 'Your registration was successful. Go to the members page '
         .'to start setting up your bookmarks!';
    do_html_url('member.php', 'Go to members page');
}
```

Como ve, invocamos la función `register()` con el nombre de usuario, dirección de correo electrónico y contraseña que hemos introducido. Si todo es correcto, registramos el nombre de usuario como variable de sesión y le proporcionamos al usuario un enlace a la página principal de miembros. En la figura 24.5 puede comprobar el resultado.

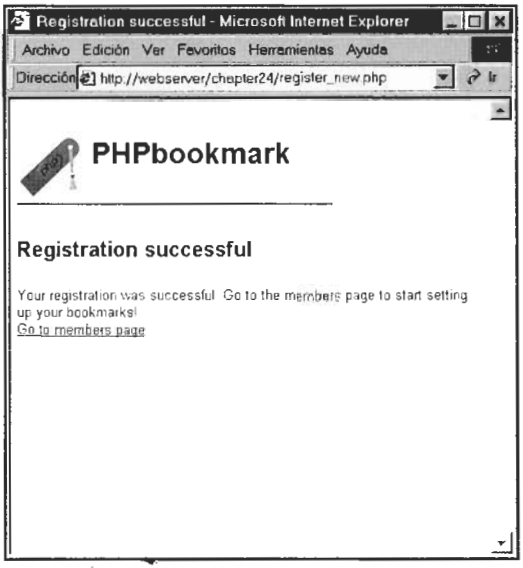


Figura 24.5. El registro ha sido satisfactorio; el usuario puede acceder a la página de miembros

La función `register()` se encuentra en la biblioteca incluida `user_auth_fns.php`.

Esta función se muestra en el listado 24.9.

Listado 24.9. Función register() de user_auth_fns.php. Esta función intenta añadir la información del nuevo usuario a la base de datos

```
function register($username, $email, $password)
// registre un nuevo usuario en la base de datos
// devuelva true o un mensaje de error
{
  // conéctese a la base de datos
    $conn = db_connect();
    if (!$conn)
      return 'Could not connect to database server - please try
```

```
later.';

   // compruebe si el nombre de usuario es exclusivo
   $result = mysql_query("select * from user where
username='$username'");
   if (!$result)
       return 'Could not execute query';
   if (mysql_num_rows($result)>0)
       return 'That username is taken - go back and choose another one.';

   // si es correcto, añádalo a la base de datos
   $result = mysql_query("insert into user values
                 '$username', password('$password'), '$email')");
   if (!$result)
      return 'Could not register you in database - please try again later.';

   return true;
}
```

No hay nada nuevo en esta función, simplemente se conecta a la base de datos que definimos anteriormente. Si el nombre de usuario seleccionado ya está utilizado o la base de datos no se puede actualizar, devolverá `false`. En caso contrario, actualizará la base de datos y devolverá `true`.

Debe saber que estamos realizando la conexión a la base de datos con una de las funciones que hemos escrito, `db_connect()`. Esta función simplemente proporciona una ubicación que contiene el nombre de usuario y la contraseña para conectarse a la base de datos. De esta forma, si cambiamos la contraseña de la base de datos, bastará con cambiar un archivo en nuestra aplicación. En el listado 24.10 podemos ver esta función.

Listado 24.10. Función db_connect() de db_fns.php. Esta función se conecta a la base de datos MySQL

```
function db_connect()
{
    $result = mysql_pconnect('localhost', 'bm_user', 'password');
    if (!$result)
        return false;
    if (!mysql_select_db('bookmarks'))
        return false;

    return $result;
}
```

Cuando los usuarios se registran, pueden conectarse y desconectarse desde las páginas destinadas para ello, que generaremos a continuación.

Conexión

Si los usuarios introducen sus datos en el formulario de `login.php` (figura 24.3) y lo envían, accederán a la secuencia de comandos `member.php`. Esta secuencia los

conecta si han utilizado este formulario. También muestra todos los marcadores relevantes para los usuarios que están conectados. Es la parte central de toda la aplicación.

En el listado 24.11 se incluye esta secuencia de comandos.

Listado 24.11. member.php. Esta secuencia de comandos es la parte principal de la aplicación

```
<?php

// incluya archivos para esta aplicación
require_once('bookmark_fns.php');
session_start();

//cree nombres de variables cortos
$username = $HTTP_POST_VARS['username'];
$passwd = $HTTP_POST_VARS['passwd'];

if ($username && $passwd)
// acaban de intentar conectarse
{
      if (login($username, $passwd))
      {
         // si están en la base de datos, registre el Id. de usuario
         $HTTP_SESSION_VARS['valid_user'] = $username;
      }
      else
      {
         // conexión no satisfactoria
         do_html_header('Problem:');
         echo 'You could not be logged in.
               You must be logged in to view this page.';
         do_html_url('login.php', 'Login');
         do_html_footer();
         exit;
      }
}

do_html_header('Home');
check_valid_user();
// obtenga los marcadores que haya guardado este usuario
if ($url_array = get_user_urls($HTTP_SESSION_VARS['valid_user']))
   display_user_urls($url_array);

// ofrezca un menú de opciones
display_user_menu();

do_html_footer();
?>
```

Seguramente reconozca la lógica de esta secuencia, ya que estamos utilizando algunas de las ideas de capítulos anteriores.

En primer lugar, comprobamos si el usuario proviene de la página principal, es decir, si acaba de completar el formulario, e intentamos conectarlo de esta forma:

```
if ($username && $passwd)
// acaban de intentar conectarse
{
    if (login($username, $passwd))
    {
        // si están en la base de datos, registre el Id. de usuario
        $HTTP_SESSION_VARS['valid_user'] = $username;
    }
```

Comprobará que intentamos conectarlo por medio de una función llamada `login()`, que hemos definido en la biblioteca `user_auth_fns.php` y cuyo código analizaremos en breve.

Si el usuario se conecta satisfactoriamente, registramos su sesión como hicimos antes y almacenamos el nombre de usuario en la variable de sesión `valid_user`.

Si todo funciona correctamente, le mostramos la página de miembros:

```
do_html_header('Home');
check_valid_user();
// obtenga los marcadores que haya guardado este usuario
if ($url_array = get_user_urls($HTTP_SESSION_VARS['valid_user']));
    display_user_urls($url_array);

// ofrezca un menú de opciones
display_user_menu();

do_html_footer();
```

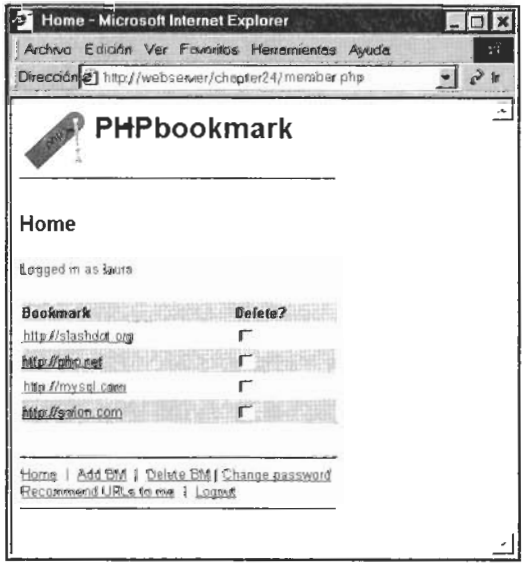


Figura 24.6. La secuencia de comandos member.php comprueba que el usuario está conectado, recupera y muestra sus marcadores y le ofrece un menú de opciones

Esta página se ha creado con las funciones de resultados. Apreciará que hemos empleado algunas funciones nuevas como son `check_valid_user()`, de `user_auth_fns.php`; `get_user_urls()`, de `url_fns.php` y `display_user_urls()`, de `output_fns.php`. La función `check_valid_user()` comprueba si el usuario actual tiene una sesión registrada. Va dirigida a usuarios que no se acaban de conectar, sino que están en el medio de una sesión. La función `get_user_urls()` obtiene los marcadores de un usuario de la base de datos y `display_user_urls()` representa los marcadores en una tabla en el navegador. Analizaremos `check_valid_user()` en breve y las otras dos restantes en el apartado sobre almacenamiento y recuperación de marcadores.

La secuencia de comandos `member.php` finaliza la página y muestra el menú con la función `display_user_menu()`.

En la figura 24.6 se muestra un ejemplo del resultado que muestra `member.php`.

A continuación describiremos las funciones `login()` y `check_valid_user()` con mayor detalle. La función `login()` se muestra en el listado 24.12.

Listado 24.12. Función login() de user_auth_fns.php. Esta función comprueba los datos de un usuario en la base de datos

```
function login($username, $password)
// compruebe el nombre de usuario y la contraseña en la base de datos
// si es sí, devuelva true
// en caso contrario devuelva false
{
   // conéctese a la base de datos
   $conn = db_connect();
   if (!$conn)
      return false;

   // compruebe si el nombre de usuario es exclusivo
   $result = mysql_query("select * from user
             where username='$username'
             and passwd = password('$password')");
   if (!$result)
       return false;

   if (mysql_num_rows($result)>0)
       return true;
   else
        return false;
}
```

Como puede apreciar, esta función se conecta a la base de datos y comprueba que existe un usuario con la combinación de nombre de usuario y contraseña proporcionada. Devuelve `true` si lo hay o `false` si no lo hay o si no puede comprobar las credenciales del usuario.

La función `check_valid_user()` no se conecta otra vez a la base de datos, sino que simplemente comprueba que el usuario tiene una sesión registrada, es decir, que ya se ha conectado.

En el listado 24.13 podemos ver esta función.

Listado 24.13. Función check_valid_user() de user_auth_fns.php. Esta función comprueba si el usuario tiene una sesión válida

```
function check_valid_user()
// compruebe si alguien se ha conectado y notifíqueselo en caso contrario
{
   global $HTTP_SESSION_VARS;
   if (isset($HTTP_SESSION_VARS['valid_user']))
   {
        echo 'Logged in as '.$HTTP_SESSION_VARS['valid_user'].'.';
        echo '<br / >';
   }
   else
   {
        // no están conectados
        do_html_heading('Problem:');
        echo 'You are not logged in.<br />';
        do_html_url('login.php', 'Login');
        do_html_footer();
        exit;
   }
}
```

Si el usuario no está conectado, la función le indicará que debe conectarse para ver esta página y le proporcionará un enlace a la página de conexión.

Desconexión

Probablemente se habrá fijado en el enlace "Logout" que aparece en el menú de la figura 24.6. Se trata de un enlace a la secuencia de comandos `logout.php`. El código de esta secuencia se muestra en el listado 24.14.

Listado 24.14. logout.php. Esta secuencia de comandos finaliza la sesión de un usuario

```
<?php

// incluya archivos de funciones para esta aplicación
require_once('bookmark_fns.php');
session_start();
$old_user  =  $HTTP_SESSION_VARS['valid_user'];
//  almacene  para  probar  si  *estaban*  conectados
unset($HTTP_SESSION_VARS);
$result_dest = session_destroy();

// inicie resultado html
do_html_header('Logging Out');

if (!empty($old_user))
{
   if  ($result_dest)
   {
     //  si  estaban  conectados  y  ahora  están  desconectados
```

```
        echo 'Logged out.<br />';
        do_html_url('login.php', 'Login');
    }
    else
    {
      // estaban conectados y no se podían desconectar
        echo 'Could not log you out.<br />';
    }
}
else
{
    // si no estaban conectados pero han accedido de algún modo a esta
    // página
    echo 'You were not logged in, and so have not been logged out.<br />';
    do_html_url('login.php', 'Login');
}

do_html_footer();

?>
```

De nuevo, este código le resultará familiar, ya que está basado en el código que diseñamos en un capítulo anterior.

Modificación de contraseñas

Si un usuario utiliza la opción de menú "Change Password", accederá al formulario que mostramos en la figura 24.7.

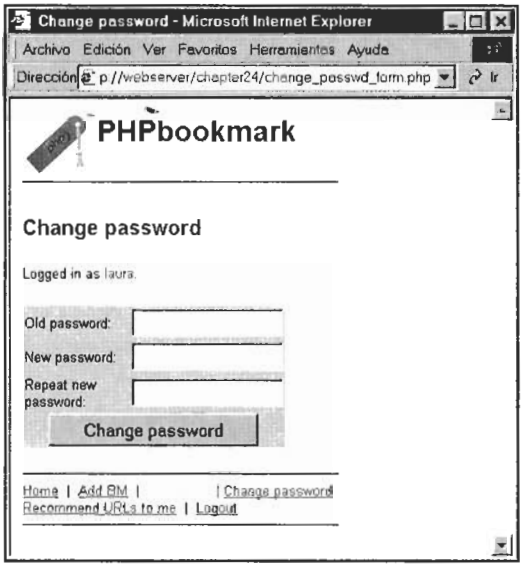


Figura 24.7. La secuencia de comandos change_passwd_form.php proporciona un formulario desde el que los usuarios pueden cambiar sus contraseñas

Este formulario se genera a partir de la secuencia de comandos `change_passwd_form.php`. Se trata de una sencilla secuencia que utiliza las funciones de la biblioteca de resultados, por lo que no hemos incluido su código.

Al enviar este formulario, se desencadena la secuencia de comandos `change_passwd.php`, que mostramos en el listado 24.15.

Listado 24.15. change_passwd.php. Esta secuencia de comandos intenta cambiar la contraseña de un usuario

```
<?php
   require_once('bookmark_fns.php');
   session_start();
   do_html_header('Changing password');

   // cree nombres de variable cortos
   $old_passwd = $HTTP_POST_VARS['old_passwd'];
   $new_passwd  =  $HTTP_POST_VARS['new_passwd'];
   $new_passwd2  =  $HTTP_POST_VARS['new_passwd2'];


 check_valid_user();
 if (!filled_out($HTTP_POST_VARS))
 {
    echo 'You have not filled out the form completely.
            Please try again.';
    display_user_menu();
    do_html_footer();
    exit;
 }
 else
 {
      if ($new_passwd!=$new_passwd2)
          echo 'Passwords entered were not the same.  Not changed.';
      else  if  (strlen($new_passwd)>16  ||  strlen($new_passwd)<6)
          echo 'New password must be between 6 and 16 characters.Try
again.';
      else
      {
            // intente actualización
            if  (change_password($HTTP_SESSION_VARS['valid_user'],
               $old_passwd,
                    $new_passwd))
                echo 'Password changed.';
            else
                echo 'Password could not be changed.';
      }

 }
    display_user_menu();
    do_html_footer();
?>
```

Esta secuencia de comandos comprueba si el usuario está conectado (por medio de `check_valid_user()`), que ha completado el formulario de contraseñas (por

medio de `filled_out()`) y que las nuevas contraseñas son idénticas y tienen la longitud correcta. Nada nuevo. Si todo funciona correctamente, invoca la función `change_passwd()` de esta forma:

```
if (change_password($HTTP_SESSION_VARS['valid_user'], $old_passwd,
$new_passwd))
   echo 'Password changed.';
else
   echo 'Password could not be changed.';
```

Esta función es de la biblioteca `user_auth_fns.php` y su código lo incluimos en el listado 24.16.

Listado 24.16. Función change_passwd() de user_auth_fns.php. Esta función intenta actualizar una contraseña de usuario en la base de datos

```
function change_password($username, $old_password, $new_password)
// cambie la contraseña de username/old_password por new_password
// devuelva true o false
{
   // si la contraseña antigua es correcta
   // cambie su contraseña por new_password y devuelva true
   // en caso contrario, devuelva false
   if (login($username, $old_password))
   {
      if (!($conn = db_connect()))
         return false;
      $result = mysql_query( "update user
      set passwd = password('$new_password')
      where username = '$username'");
      if (!$result)
         return false; // no ha cambiado
      else
         return true;  // ha cambiado satisfactoriamente
   }
   else
      return false; // la contraseña antigua era incorrecta
}
```

La función comprueba que la antigua contraseña proporcionada es correcta, con ayuda de la función `login()` que ya hemos descrito anteriormente. Si es correcta, la función realiza la conexión a la base de datos y actualiza la contraseña con el nuevo valor.

Restablecimiento de contraseñas olvidadas

Además de modificar contraseñas, tendremos que hacer algo cuando el usuario olvide su contraseña, algo muy habitual. Habrá comprobado que en la página inicial, `login.php`, incluimos el enlace "Forgotten your password?" para los usuarios con este problema. Mediante este enlace los usuarios acceden a la secuencia de comandos `forgot_form_php`, que utiliza las funciones de resultado para mostrar un formulario como el de la figura 24.8.

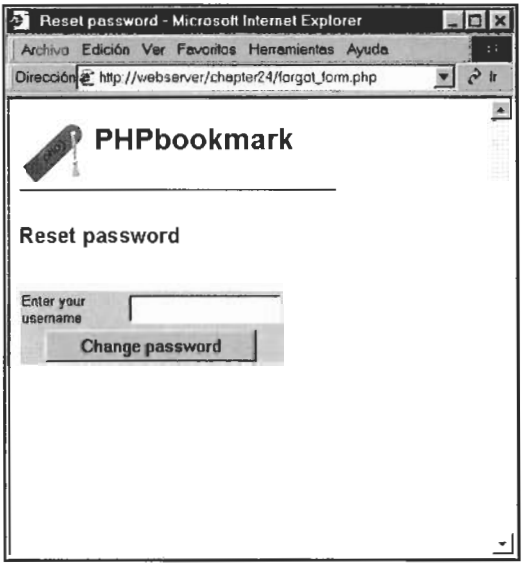


Figura 24.8. La secuencia de comandos forgot_form.php proporciona un formulario con el que los usuarios pueden solicitar el restablecimiento y el envío de sus contraseñas

Esta secuencia de comandos es muy sencilla; simplemente utiliza las funciones de resultado, por lo que no la describiremos detalladamente.

Al enviar el formulario, invoca la secuencia de comandos `forgot_passwd.php`, que es más interesante.

En el listado 24.17 podemos ver esta secuencia.

Listado 24.17. forgot_passwd.php Esta secuencia de comandos restablece la contraseña de un usuario por valor aleatorio y le envía la nueva

```
<?php
   require_once("bookmark_fns.php");
   do_html_header("Resetting password");

   //creación de un nombre de variable corto
   $username = $HTTP_POST_VARS["username"];

   if ($password=reset_password($username))
   {
      if (notify_password($username, $password))
         echo "Your new password has been sent to your email address.";
      else
         echo "Your password could not be mailed to you."
              ." Try pressing refresh.";
   }
   else
      echo "Your password could not be reset - please try again later.";
```

```
    do_html_url('login.php', 'Login');

   do_html_footer();
?>
```

Como puede apreciar, esta secuencia de comandos utiliza dos funciones principales para realizar esta tarea: `reset_password()` y `notify_password()`. Veamos cada una de ellas. La función `reset_password()` genera una contraseña aleatoria para el usuario y la añade a la base de datos. Su código se incluye en el listado 24.18.

Listado 24.18. Función reset_password() de user_auth_fns.php. Esta secuencia de comandos restablece la contraseña de un usuario con un valor aleatorio y le envía la nueva

```
function reset_password($username)
// establezca la contraseña del nombre de usuario con un valor aleatorio
// devuelva la nueva contraseña o false en caso de fallo
{
   // obtenga una palabra aleatoria del diccionario con una longitud de 6 a
   // 13 caracteres
   $new_password = get_random_word(6, 13);

   if($new_password==false)
      return false;
   // añádale un número comprendido entre 0 y 999
   // para mejorar ligeramente la contraseña
   srand ((double) microtime() * 1000000);
   $rand_number = rand(0, 999);
   $new_password .= $rand_number;

   // defina con esto la contraseña del usuario en la base de datos o
devuelva false
   if (!($conn = db_connect()))
         return false;
   $result = mysql_query( "update user
   set passwd = password('$new_password')
   where username = '$username'");
   if (!$result)
      return false; // no ha cambiado
   else
      return $new_password; // ha cambiado correctamente
}
```

La función genera una contraseña aleatoria. Para ello obtiene una palabra aleatoria de un diccionario, utiliza la función `get_random_word()` y le añade como sufijo un número aleatorio comprendido entre 0 y 999. La función `get_random_word()` se encuentra también en la biblioteca `user_auth_fns.php` y su código se incluye en el listado 24.19.

Listado 24.19. Función get_random_word() de user_auth_fns.php. Esta función obtiene una palabra aleatoria de un diccionario y la utiliza para generar contraseñas

```
function  get_random_word($min_length,  $max_length)
// obtenga del diccionario una palabra aleatoria comprendida entre las dos
```

```
// longitudes y devuélvala
{
    // genere una palabra aleatoria
    $word = '';
    // no olvide cambiar esta ruta para que se ajuste a su sistema
    $dictionary = '/usr/dict/words';  // el diccionario ispell
    $fp = fopen($dictionary, 'r');
    if(!$fp)
       return false;
    $size = filesize($dictionary);

    // vaya a un punto aleatorio del diccionario
    srand ((double) microtime() * 1000000);
    $rand_location = rand(0, $size);
    fseek($fp, $rand_location);

    // obtenga la siguiente palabra de la longitud correcta en el archivo
    while (strlen($word)< $min_length || strlen($word)>$max_length
            || strstr($word, "'"))
    {
        if (feof($fp))
            fseek($fp, 0);// si está al final, vaya hasta el principio
        $word = fgets($fp, 80);  // evite la primera palabra ya que podría ser
                                 // parcial
        $word = fgets($fp, 80);  // la contraseña potencial
    };
    $word=trim($word); // recorte el \n final de fgets
    return $word;
}
```

Para que funcione, la función necesita un diccionario. Si utiliza un sistema Unix, el revisor ortográfico incorporado, ispell, contiene un diccionario de palabras, que normalmente se encuentra en `/usr/dict/words` como en este caso, o en `/usr/share/dict/words`. Si no lo encuentra en ninguna de estas ubicaciones, en la mayoría de los sistemas lo podrá localizar si escribe

```
locate dict/words
```

Si utiliza algún otro sistema o no quiere instalar ispell, no se preocupe. Puede descargar listas de palabras como las utilizadas por ispell de la siguiente dirección: `http://wordlist.sourceforge.net/`.

Este sitio también cuenta con diccionarios en muchos otros idiomas, por lo que si quiere utilizar una palabra aleatoria en noruego o esperanto, por ejemplo, puede descargar cualquiera de estos diccionarios. El formato de los archivos muestra cada palabra en una línea y las líneas separadas entre sí.

Para obtener una palabra aleatoria de este archivo, seleccionamos una ubicación aleatoria comprendida entre 0 y el tamaño del archivo, y leemos desde ese punto del archivo.

Si leemos desde la ubicación aleatoria hasta la siguiente línea, probablemente obtendremos una palabra parcial, por lo que ignoramos la línea en la que hemos abierto el archivo y tomamos la siguiente palabra invocando dos veces `fgets()`.

La función tiene dos partes muy interesantes.

Por un lado, si llegamos al final del archivo mientras buscamos una palabra, volvemos de nuevo al comienzo:

```
if (feof($fp))
   fseek($fp, 0); // si está al final, vaya hasta el principio
```

Por otra parte, podemos buscar una palabra de una longitud determinada; comprobamos cada palabra que obtenemos del diccionario y, si no está comprendida entre `$min_length` y `$max_lenght`, seguimos buscando.

Al mismo tiempo, también podemos eliminar palabras con apóstrofes (comillas simples). Podríamos suprimirlos al utilizar la palabra, pero resulta más sencillo obtener la palabra exacta.

Volvamos a `reset_password()`. Una vez generada la nueva contraseña, actualizamos la base de datos para indicarlo y devolvemos la nueva contraseña a la secuencia de comandos principal.

Esto se pasa a `notify_password()`, que la enviará por correo electrónico al usuario.

La función `notify_password()` se describe en el listado 24.20.

Listado 24.20. Función notify_password() de user_auth_fns.php. Esta función envía la contraseña restablecida por correo electrónico al usuario

```
function notify_password($username, $password)
// informe al usuario que se ha modificado su contraseña
{
      if (!($conn = db_connect()))
         return false;
      $result = mysql_query("select email from user
                where username='$username'");
      if (!$result)
      {
         return false; // no ha cambiado
      }
      else if (mysql_num_rows($result)==0)
      {
         return false; // el nombre de usuario no está en la base de datos
      }
      else
      {
         $email = mysql_result($result, 0, 'email');
         $from = "From: support@phpbookmark \r\n";
         $mesg = "Your PHPBookmark password has been changed to $password
\r\n"
         ."Please change it next time you log in. \r\n";


         if (mail($email, 'PHPBookmark login information', $mesg, $from))
            return true;
         else
            return  false;
      }
}
```

En esta función, con un nombre de usuario y una contraseña, basta con buscar la dirección de correo electrónico de dicho usuario en la base de datos y utilizar la función `mail()` de PHP para enviarla. Sería más seguro proporcionar a los usuarios una verdadera contraseña aleatoria, generada a partir de una combinación de letras minúsculas y mayúsculas, números y signos de puntuación, en lugar de nuestra palabra y número aleatorio. Sin embargo, una contraseña como `'zigazg487'` resulta más sencilla de leer y de escribir que una verdaderamente aleatoria. A menudo a los usuarios les resulta difícil saber si un carácter de una cadena aleatoria es 0 u O (cero u O mayúscula) o 1 o l (el uno o la ele minúscula). En nuestro sistema, el archivo de diccionario contiene aproximadamente 45.000 palabras. Si un pirata informático supiera cómo creamos las contraseñas y conociera el nombre de un usuario, tendría que probar con una media de 22.500.000 contraseñas para adivinar una. Este nivel de seguridad parece adecuado para este tipo de aplicación incluso si nuestros usuarios rechazan el consejo enviado por correo electrónico para que cambien su contraseña.

Implementación del almacenamiento y recuperación de marcadores

A continuación veremos cómo se almacenan, recuperan y eliminan los marcadores de un usuario.

Cómo añadir marcadores

Los usuarios pueden añadir marcadores si pulsan el enlace **Add BM** del menú de usuario. De esta forma acceden al formulario mostrado en la figura 24.9. Como en casos anteriores, esta secuencia de comandos utiliza únicamente las funciones de resultados, por lo que no la analizaremos en este apartado. Al enviar el formulario, invoca la secuencia de comandos `add_bms.php`, que mostramos en el listado 24.21.

Listado 24.21. add_bms.php. Esta secuencia de comandos añade nuevos marcadores a la página personal de un usuario

```
<?php
require_once('bookmark_fns.php');
session_start();

  //cree un nombre de variable corto
  $new_url = $HTTP_POST_VARS['new_url'];

  do_html_header('Adding bookmarks');
  check_valid_user();
  if (!filled_out($HTTP_POST_VARS))
  {
    echo 'You have not filled out the form completely.
          Please try again.';
```

```
        display_user_menu();
        do_html_footer();
        exit;
    }
    else
    {
        // compruebe el formato del URL
        if (strstr($new_url, 'http://')===false)
            $new_url = 'http://'.$new_url;

        // compruebe si el URL es válido
        if (@fopen($new_url, 'r'))
        {
            // intente añadir un marcador
            if (add_bm($new_url))
                echo 'Bookmark added.';
            else
                echo 'Could not add bookmark.';
        }
        else
            echo 'Not a valid URL.';
    }

    // obtenga los marcadores almacenados por este usuario
    if ($url_array = get_user_urls($HTTP_SESSION_VARS['valid_user']));
        display_user_urls($url_array);

    display_user_menu();
    do_html_footer();
?>
```

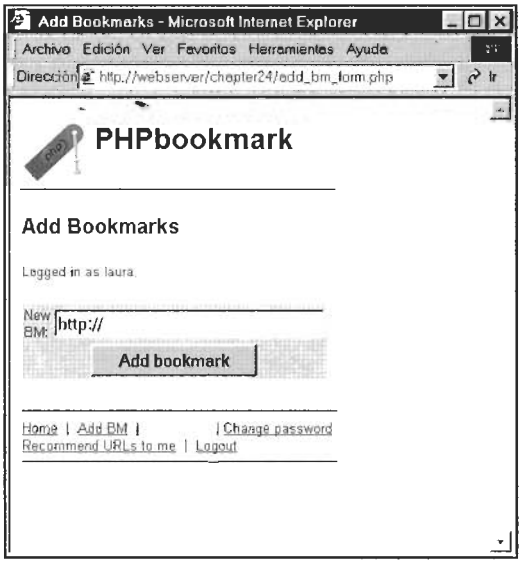


Figura 24.9. La secuencia de comandos add_bm_form.php proporciona un formulario desde el que los usuarios pueden añadir marcadores a sus páginas de marcadores

De nuevo, esta secuencia de comandos utiliza el modelo de validación, entradas de base de datos y resultados.

Para validar, primero comprobamos si el usuario ha completado el formulario por medio de `filled_out()`.

Tras ellos, realizamos dos comprobaciones de URL. Primero, por medio de `strstr()`, comprobamos si el URL comienza por `http://`. Si no lo hace, lo añadimos al inicio del URL. Tras ello, podemos comprobar que el URL existe realmente. Como recordará de capítulos anteriores, podemos utilizar `fopen()` para abrir un URL que empiece por `http://`. Si podemos abrir este archivo, asumimos que el URL es válido e invocamos la función `add_bm()` para añadirlo a la base de datos.

Fíjese en que la función `fopen()` sólo podrá abrir archivos si el servidor tiene acceso directo a Internet. Si necesita acceder a otros servidores HTTP a través de un servidor proxy, `fopen()` no funcionará.

Esta función y el resto de funciones relacionadas con marcadores se encuentran en la biblioteca de funciones `url_fns.php`. En el listado 24.22 incluimos el código de la función `add_bm()`.

Listado 24.22. Función add_bm() de url_fns.php. Esta función añade nuevos marcadores a la base de datos

```
function  add_bm($new_url)
{
   // Añada el nuevo marcador a la base de datos

   echo "Attempting to add ".htmlspecialchars($new_url).'<br />';
   global $HTTP_SESSION_VARS;
   $valid_user  =  $HTTP_SESSION_VARS['valid_user'];

   if (!($conn = db_connect()))
      return false;

   // compruebe que no es un marcador repetido
   $result = mysql_query("select * from bookmark
             where username='$valid_user'
             and bm_URL='$new_url'");
   if  ($result  &&  (mysql_num_rows($result)>0))
      return false;

   // incluya el nuevo marcador
   if (!mysql_query( "insert into bookmark values
               ('$valid_user', '$new_url')"))
      return false;

   return  true;
}
```

Esta función es bastante sencilla. Comprueba que un usuario no tenga ya este marcador enumerado en la base de datos (aunque es improbable que los usuarios introduzcan un marcador dos veces, es muy posible que quieran actualizar la página). Si el marcador es nuevo, se introducirá en la base de datos.

Si se fija en `add_bm.php`, comprobará que lo último que hace es invocar `get_user_urls()` y `display_user_urls()`, lo mismo que `member.php`. A continuación analizaremos todas estas funciones.

Cómo mostrar marcadores

En la secuencia de comandos `member.php` y en la función `add_bm()` utilizamos las funciones `get_user_urls()` y `display_user_urls()`. Estas funciones obtienen los marcadores del usuario de la base de datos y los muestran, respectivamente.

La función `get_user_urls()` se encuentra en la biblioteca `url_fns.php` y la función `display_user_fns.php` en la biblioteca `output_fns.php`.

En el listado 24.23 encontrará la función `get_user_urls()`.

Listado 24.23. Función get_user_urls() de url_fns.php. Esta función recupera de la base de datos los marcadores de un usuario

```
function  get_user_urls($username)
{
   //extraiga de la base de datos todos los URL que haya almacenado este
   // usuario
   if (!($conn = db_connect()))
      return false;
   $result = mysql_query( "select bm_URL
               from bookmark
                 where username = '$username'");
   if (!$result)
      return false;

   //cree una matriz de los URL
   $url_array = array();
   for ($count = 1; $row = mysql_fetch_row ($result); ++$count)
   {
      $url_array[$count] = addslashes($row[0]);
   }
   return $url_array;
}
```

Analicemos brevemente esta función. Adopta un nombre de usuario como parámetro y recupera los marcadores de dicho usuario de la base de datos. Devuelve una matriz de dichos URL o el valor `false` si los marcadores no se pueden recuperar.

La matriz de `get_user_urls()` se puede pasar a `display_user_urls()`. Se trata, de nuevo, de una sencilla función de resultado HTML para imprimir los URL del usuario en formato de tabla, por lo que no la describiremos. En la figura 24.6 puede comprobar el aspecto del resultado. La función se encarga de añadir los URL a un formulario.

Junto a cada URL aparece una casilla de verificación que permite seleccionar los marcadores para su eliminación, como veremos a continuación.

Eliminación de marcadores

Cuando un usuario selecciona un marcador para su eliminación y pulsa la opción **Delete BM** del menú, se envía el formulario que contiene los URL. Cada una de las casillas de verificación se genera a partir del siguiente código de la función `display_user_urls()`:

```
echo "<td><input type=\"checkbox\" name=\"del_me[]\"
        value=\"$url\"></td>";
```

El nombre de cada entrada es `del_me[]`. Esto significa que, en la secuencia de comandos PHP activada por este formulario, tendremos acceso a una matriz denominada `$del_me` que contiene todos los marcadores que se van a eliminar.

Al pulsar sobre la opción **Delete BM** se activa la secuencia de comandos `delete_bms.php`.

Esta secuencia de comandos se muestra en el listado 24.24.

Listado 24.24. delete_bms.php. Esta secuencia de comandos elimina marcadores de la base de datos

```
<?php
   require_once('bookmark_fns.php');
   session_start();

   //cree nombres de variables cortos
   $del_me = $HTTP_POST_VARS['del_me'];
   $valid_user = $HTTP_SESSION_VARS['valid_user'];

   do_html_header('Deleting bookmarks');
   check_valid_user();
   if (!filled_out($HTTP_POST_VARS))
   {
      echo 'You have not chosen any bookmarks to delete.
            Please try again.';
      display_user_menu();
      do_html_footer();
      exit;
   }
   else
   {
      if (count($del_me) >0)
      {
         foreach($del_me as $url)
         {
            if  (delete_bm($valid_user,  $url))
               echo 'Deleted '.htmlspecialchars($url).'.<br />';
            else
               echo 'Could not delete '.htmlspecialchars($url).'.<br />';
         }
      }
      else
         echo 'No bookmarks selected for deletion';
```

```
    }

    // obtenga los marcadores que haya guardado este usuario
    if ($url_array = get_user_urls($valid_user));
       display_user_urls($url_array);

    display_user_menu();
    do_html_footer();
?>
```

La secuencia comienza con la ejecución de las validaciones habituales. Cuando sabemos que el usuario ha seleccionado algunos marcadores para su eliminación, los borramos con el siguiente bucle:

```
foreach($del_me  as  $url)
{
    if (delete_bm($valid_user, $url))
       echo 'Deleted '.htmlspecialchars($url).'.<br />';
    else
       echo 'Could not delete '.htmlspecialchars($url).'.<br />';
}
```

Como puede comprobar, la función `delete_bm()` se encarga de eliminar los marcadores de la base de datos.

Esta función se muestra en el listado 24.25.

Listado 24.25. Función delete_bm() de url_fns.php. Esta función elimina un solo marcador de la base de datos

```
function  delete_bm($user,  $url)
{
    // borre un URL de la base de datos
    if (!($conn = db_connect()))
       return false;

    // borre el marcador
    if (!mysql_query( "delete from bookmark
          where username='$user' and bm_url='$url'"))
       return false;
    return  true;
}
```

Como puede apreciar, se trata de nuevo de una sencilla función. Intenta eliminar el marcador de un determinado usuario de la base de datos.

Fíjese en que queremos eliminar un par nombre de usuario-marcador concreto.

Puede que otros usuarios tengan este URL marcado.

En la figura 24.10 puede ver un ejemplo de ejecución de la secuencia de comandos de eliminación en nuestro sistema.

Al igual que en la secuencia de comandos `add_bms.php`, una vez realizados los cambios en la base de datos, mostramos la nueva lista de marcadores por medio de `get_user_urls()` y `display_user_urls()`.

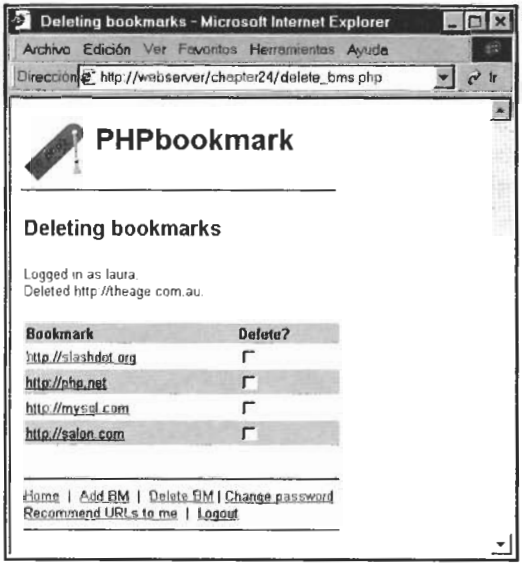


Figura 24.10. La secuencia de comandos de eliminación notifica al usuario los marcadores borrados y, tras ello, muestra los marcadores restantes

Implementación de recomendaciones

Por último, llegamos a la secuencia de comandos del enlace de recomendaciones, `recommend.php`.

Existen distintos enfoques a la hora de realizar recomendaciones. Hemos optado por lo que denominamos recomendación de "afinidades". Es decir, buscaremos otros usuarios que tengan al menos un marcador similar al de nuestro usuario. Los marcadores restantes de otros usuarios también le pueden interesar.

La forma más sencilla de implementarlo como consulta SQL consiste en utilizar una subconsulta. En primer lugar, obtenemos en una subconsulta la lista de usuarios similares y, tras ello, analizamos sus marcadores en una consulta externa.

Sin embargo, como recordará, MySQL no es compatible con subconsultas. Tendremos que realizar dos consultas diferentes e incluir el resultado de la primera en la segunda. Para ello, podemos definir una tabla temporal con los resultados de la primera consulta o procesar los resultados de la primera consulta con ayuda de PHP.

Hemos optado por la segunda opción, aunque ambos enfoques tienen sus ventajas. El uso de una tabla temporal es ligeramente más rápido pero el procesamiento en PHP facilita las tareas de comprobación y modificación del código.

En primer lugar ejecutamos la siguiente consulta:

```
select distinct(b2.username)
from bookmark b1, bookmark b2
where b1.username='$valid_user'
```

```
and  b1.username  !=  b2.username
and b1.bm_URL = b2.bm_URL
```

Esta consulta utiliza alias para unir el marcador de la base de datos a sí mismo, un concepto extraño pero en ocasiones útil. Imagine que existen dos tablas de marcadores, una con el nombre `b1` y otra llamada `b2`. En `b1`, trabajamos con el usuario actual y sus marcadores. En la otra tabla, analizamos los marcadores de todos los usuarios restantes. Buscamos otros usuarios (`b2.username`) que tengan un URL similar al del usuario actual (`b1.bm_URL = b2.bm_URL`) y que no sean el usuario actual (`b1.username != b2.username`).

Esta consulta genera una lista de usuarios afines al nuestro. Por medio de esta lista, podemos buscar sus marcadores restantes con ayuda de la siguiente consulta:

```
select bm_URL
from  bookmark
where  username  in  $sim_users
and bm_URL not in $user_urls
group  by  bm_URL
having  count(bm_URL)>$popularity
```

La variable `$sim_users` contiene la lista de usuarios afines. La variable `$user_urls` contiene la lista de los marcadores del usuario actual; si `b1` ya tiene un marcador, no tiene sentido recomendárselo. Por último, añadimos un filtro por medio de la variable `$popularity` ya que no queremos recomendar ningún URL que sea demasiado personal, por lo que únicamente sugeriremos direcciones URL que un determinado número de los otros usuarios de la lista de afines hayan marcado.

Si pensamos que nuestro sistema tendrá un elevado número de usuarios, podríamos ajustar `$popularity` para que sólo sugiera direcciones URL que hayan marcado un elevado número de usuarios. Los URL marcados por una gran cantidad de usuarios seguramente sean de gran calidad y sean más atractivos que una página Web media.

La secuencia de comandos completa para realizar recomendaciones se describe en los listados 24.26 y 24.27. La secuencia de comandos principal se denomina `recommend.php` (listado 24.26). Invoca la función de recomendación `recommend_urls()` de `url_fns.php` (listado 24.27).

Listado 24.26. recommend.php. Esta secuencia de comandos sugiere algunos marcadores que le pueden interesar a un usuario

```
<?php
   require_once('bookmark_fns.php');
   session_start();
   do_html_header('Recommending URLs');
   check_valid_user();
   $urls = recommend_urls($HTTP_SESSION_VARS['valid_user']);
   display_recommended_urls($urls);

   display_user_menu();
   do_html_footer();
?>
```

Listado 24.27. Función recommend_urls() de url_fns.php. Esta secuencia de comandos realiza las recomendaciones

```
function recommend_urls($valid_user, $popularity = 1)
{
   // Ofrecemos a los usuarios recomendaciones seminteligentes
   // Si tienen un URL en común con otros, puede que estén interesados en
   // otros URL que les parezcan atractivos
   if (!($conn = db_connect()))
      return false;

   // busque otros usuarios
   // con un URL similar al suyo

   if (!($result = mysql_query("
            select distinct(b2.username)
            from bookmark b1, bookmark b2
            where b1.username='$valid_user'
            and b1.username != b2.username
            and b1.bm_URL = b2.bm_URL
             ")))
       return false;
   if (mysql_num_rows($result)==0)
      return false;

   // cree un conjunto de usuarios con URL en común
   // para utilizarlos en la cláusula IN
   $row = mysql_fetch_object($result);
   $sim_users = "('".($row->username)."'";
   while ($row = mysql_fetch_object($result))
   {
        $sim_users .= ", '".($row->username)."'";
   }
   $sim_users .= ')';

   // cree una lista de URL de usuarios
   // para evitar la duplicación de los que ya conocemos
   if (!($result = mysql_query("
            select bm_URL
            from bookmark
            where username='$valid_user'")))
      return false;

   // cree un conjunto de URL de usuarios para utilizarlos en una cláusula
   // IN
   $row = mysql_fetch_object($result);
   $user_urls = "('".($row->bm_URL)."'";
   while ($row = mysql_fetch_object($result))
   {
        $user_urls .= ", '".($row->bm_URL)."'";
   }
   $user_urls .= ')';

   // como sencillo método para excluir páginas privadas de usuarios y
   // aumentar la posibilidad de recomendar URL atractivos
```

```
    // especificamos un nivel de popularidad mínimo
    // si $popularity = 1, más de un usuario debe tener
    // un URL antes de que podamos recomendarlo

    // determinar el número máximo de URL posibles
    if (!($result = mysql_query("
            select bm_URL
            from bookmark
            where username in $sim_users
            and bm_URL not in $user_urls
            group by bm_URL
            having count(bm_URL)>$popularity
            ")))
        return false;

    if (!($num_urls=mysql_num_rows($result)))
       return false;

    $urls = array();
    // genere una matriz de URL relevantes
    for ($count=0; $row = mysql_fetch_object($result); $count++)
    {
        $urls[$count] = $row->bm_URL;
    }

    return $urls;
}
```

En la figura 24.11 puede ver un ejemplo del resultado generado por `recommend.php`.

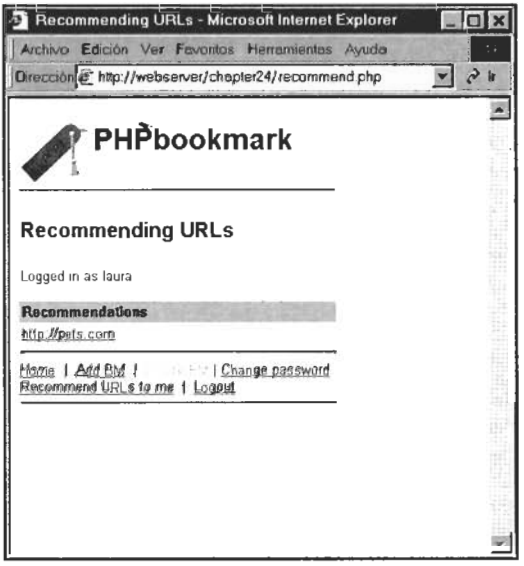


Figura 24.11. La secuencia de comandos ha recomendado que a este usuario le puede interesar pets.com. Otros dos usuarios de la base de datos interesados en slashdot.com lo tienen marcado

Conclusión y posibles ampliaciones

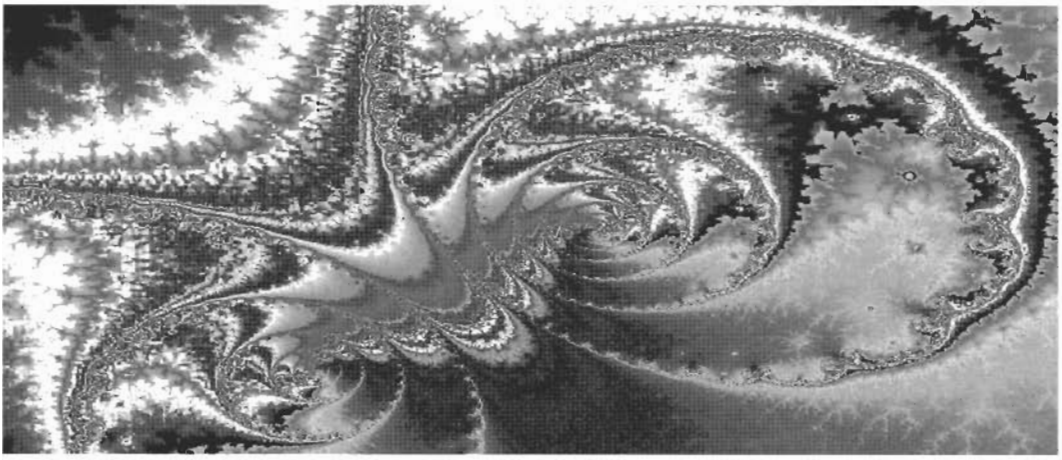
Hasta ahora hemos mostrado la funcionalidad básica de la aplicación PHPBookmark. También existen distintas extensiones posibles, entre las que destacamos las siguientes:

- Una clasificación de marcadores por tipo
- Un enlace "Agregar a mis marcadores" para las recomendaciones
- Recomendaciones basadas en los URL más populares de la base de datos o en un determinado tema
- Una interfaz administrativa para definir y administrar usuarios y temas
- Técnicas para que los marcadores recomendados sean más inteligentes y tengan mayor rapidez
- Comprobación de errores adicional de las entradas del usuario

Experimente. Es la mejor forma de aprender.

A continuación

En el siguiente proyecto crearemos un carro de la compra que permitirá a los usuarios navegar por nuestro sitio, realizar compras antes de confirmar sus pedidos y realizar un pago electrónico.



25
Construcción de un carro de la compra

En este capítulo aprenderemos a diseñar un carro de la compra básico, que añadiremos a la base de datos Book-O-Rama que implementamos en capítulos anteriores.

También exploraremos otra opción: la configuración y el uso de un carro de la compra PHP de código abierto existente.

En caso de que no lo haya escuchado antes, el término *carro de la compra* (a veces también se denomina *cesta de la compra*) se utiliza para describir un mecanismo de compra en línea específico.

Al navegar por nuestro catálogo en línea, puede añadir artículos a su carro de la compra.

Cuando termine, se pasa por la caja de la tienda, es decir, se adquieren los artículos del carro.

Para poder implementar el carro de la compra, tendremos que implementar la siguiente funcionalidad:

- Una base de datos de los productos que queremos comercializar en línea
- Un catálogo en línea de productos, enumerados por categoría
- Un carro de la compra para almacenar los artículos que el usuario quiere adquirir
- Una secuencia de comandos de caja que procese los detalles del pago y del envío
- Una interfaz de administración

El problema

Probablemente recuerde la base de datos Book-O-Rama que desarrollamos en capítulos anteriores. En este proyecto, implementaremos la tienda en línea de Book-O-Rama y haremos que funcione. Los requisitos de este sistema son los siguientes:

- Necesitamos una forma de conectar la base de datos al navegador del usuario. Los usuarios deben poder examinar los artículos por categoría.
- Los usuarios deben poder seleccionar artículos del catálogo para su posterior adquisición. Tendremos que realizar el seguimiento de los artículos seleccionados.
- Cuando hayan finalizado sus compras, tendremos que sumar su pedido, procesar los detalles de entrega y realizar el pago.
- También tendremos que diseñar una interfaz administrativa para el sitio de Book-O-Rama para que el administrador pueda añadir o modificar libros y las categorías del sitio.

Componentes de la solución

A continuación analizaremos las soluciones de cada uno de los requisitos expuestos anteriormente.

Generación de un catálogo en línea

Ya disponemos de una base de datos para el catálogo Book-O-Rama. Sin embargo, para esta aplicación seguramente necesitaremos modificar y añadir algunos elementos, como por ejemplo añadir categorías de libros, como indicamos en los requisitos.

También necesitamos incluir en la base de datos existente cierta información sobre envíos, direcciones, detalles de pagos, etc.

Ya sabemos cómo diseñar un interfaz en una base de datos MySQL por medio de PHP, por lo que esta parte de la solución será muy sencilla.

Cómo realizar el seguimiento de las compras de un usuario mientras compra

Disponemos de dos formas de realizar el seguimiento de las compras de un usuario mientras éste compra. Por un lado, podemos añadir sus selecciones a la base de datos y, por otro, utilizar una variable de sesión.

El uso de una variable de sesión para realizar el seguimiento de las selecciones de una página a otra resulta más sencillo de escribir ya que no tendremos que consultar constantemente la base de datos para obtener dicha información. También evitaremos acumular en la base de datos una gran cantidad de datos inservibles de usuarios que simplemente han estado navegando y después se lo han pensado mejor.

Por ello, necesitaremos diseñar una variable de sesión o un conjunto de variables para almacenar las selecciones de un usuario. Cuando éste termine de realizar sus compras y efectúe el pago, añadiremos esta información a la base de datos como registro de la transacción.

También podemos utilizar estos datos para mostrar un resumen del estado actual del carro en una esquina de la página, para que el usuario sepa en cualquier momento cuánto puede gastar.

Pagos

En este proyecto, sumaremos el pedido del usuario y anotaremos los detalles de entrega. No procesaremos los pagos. Existe una gran cantidad de sistemas de pago disponibles, cada uno con una implementación diferente. Escribiremos una función ficticia que se pueda reemplazar con una interfaz en el sistema seleccionado.

Los sistemas de pago se suelen vender en áreas geográficas más específicas que este libro. Por lo general, el funcionamiento de las distintas interfaces de procesamiento en tiempo real es muy similar. Tendrá que contratar una cuenta con una entidad bancaria en función de las tarjetas que acepte. Su proveedor de sistemas de pago especificará los parámetros que debe pasar a su sistema.

El sistema de pagos transmitirá sus datos al banco y devolverá un código correcto o uno de los distintos tipos de códigos de error. A cambio de pasar nuestros datos, el sistema le cargará una cuota fija o anual, así como una comisión en función del número o del valor de las transacciones. Algunos proveedores cobran incluso por transacciones anuladas.

El sistema de pagos que seleccione necesitará información del usuario (por ejemplo, un número de tarjeta de crédito), información que nos identifique (para especificar en qué cuenta se va a realizar el cargo) y la suma total de la transacción.

Podemos determinar el total de un pedido a partir de la variable de sesión del carro del usuario. Registraremos los detalles del pedido final en la base de datos y, al mismo tiempo, nos desharemos de la variable de sesión.

Interfaz de administración

Además de todo esto, diseñaremos una interfaz de administración que nos permitirá añadir, eliminar y modificar libros y categorías en la base de datos.

Una modificación muy habitual que podemos realizar es cambiar el precio de un producto (por ejemplo, para ofertas especiales o rebajas). Esto significa que al almacenar el pedido de un cliente, también tendremos que almacenar el precio que haya pagado por un artículo. Nuestra contabilidad sería desastrosa si los únicos registros

que tuviéramos fueran los artículos pedidos por cada cliente y el precio actual de cada uno de ellos.

Esto también significa que si el cliente devuelve o cambia un artículo, se le abone la cantidad correspondiente.

En este ejemplo no diseñaremos una interfaz de seguimiento de todos los pedidos. Puede añadir una que se ajuste a sus necesidades a este sistema básico.

Presentación de la solución

A continuación combinaremos todas las piezas.

El sistema cuenta con dos vistas básicas: la vista del usuario y la del administrador. Después de determinar la funcionalidad necesaria, definimos dos diseños de flujo del sistema, uno para cada vista, que mostramos en las figuras 25.1 y 25.2 respectivamente.

En la figura 25.1 se muestran los enlaces principales entre las secuencias de comandos de la parte del usuario del sitio. En primer lugar, el cliente accede a la página principal, en la que se enumeran todas las categorías de libros del sitio. Desde aquí, puede acceder a una categoría determinada y, tras ello, a los detalles concretos del libro.

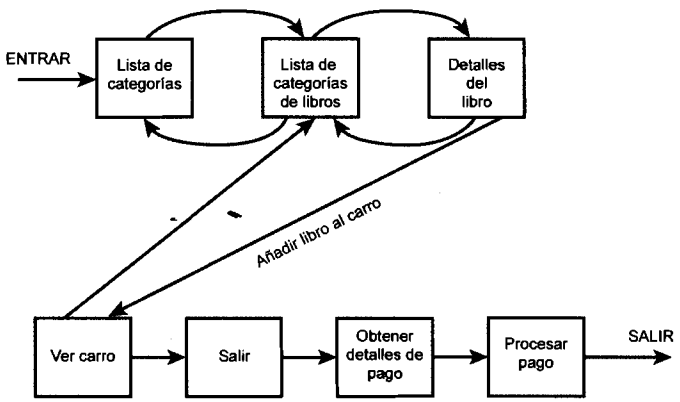


Figura 25.1. La vista de usuario del sistema Book-O-Rama permite a los usuarios examinar los libros por categoría, ver detalles de los libros, añadirlos al carro y comprarlos

A cada usuario se le proporciona un enlace para añadir un libro concreto a su carro. Desde el carro, podrá salir de la tienda en línea.

En la figura 25.2 puede ver la interfaz de administración. Tiene más secuencias de comandos, pero no gran cantidad de código nuevo.

Estas secuencias de comandos permiten al administrador registrar y añadir categorías y libros.

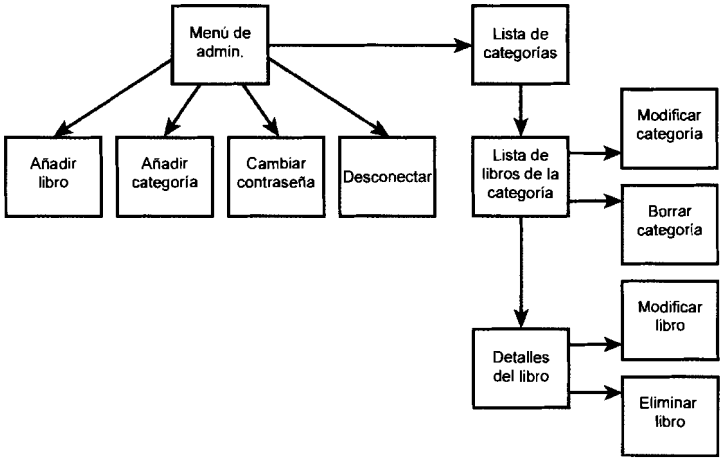


Figura 25.2. La vista de administración del sistema Book-O-Rama permite añadir, modificar y eliminar libros y categorías

La forma más sencilla de implementar la modificación y eliminación de libros y categorías consiste en mostrar al administrador una versión ligeramente distinta de la interfaz de usuario en el sitio.

El administrador puede seguir navegando por libros y categorías pero en lugar de tener acceso al carro de la compra, podrá acceder a un determinado libro o categoría y modificar o eliminar dicho libro o categoría. Si conseguimos que ambas secuencias de comandos funcionen tanto con usuarios como con administradores, nos ahorremos tiempo y esfuerzo.

Los tres módulos de código principales para esta aplicación son los siguientes:

- Catálogo
- Carro de la compra y procesamiento de pedidos (los hemos agrupado porque están íntimamente relacionados)
- Administración

Como en el proyecto anterior, tendremos que generar y utilizar una serie de bibliotecas de funciones. En este proyecto, utilizaremos un API de funciones similar al anterior. Intentaremos agrupar las partes del código que generen HTML en una misma biblioteca para cumplir el principio de separación entre lógica y contenido y, lo más importante, para que nuestro código resulte más fácil de leer y de mantener.

También tendremos que realizar algunos cambios en la base de datos Book-O-Rama.

Hemos cambiado el nombre de la base de datos por `book_sc` para distinguirla de la que creamos en capítulos anteriores.

Todo el código de este proyecto se incluye en el CD-ROM. En la tabla 25.1 encontrará un resumen de los archivos de la aplicación.

Tabla 25.1. Archivos de la aplicación Carro de la compra

Nombre	Módulo	Descripción
index.php	Catálogo	Página principal del sitio para los usuarios. Muestra una lista de las categorías del sistema.
show_cat.php	Catálogo	Muestra al usuario todos los libros de una determinada categoría.
show_book.php	Catálogo	Muestra al usuario los detalles de un libro en concreto
show_cart.php	Carro de la compra	Muestra al usuario los contenidos de su carro. También se utiliza para añadir artículos al carro.
checkout.php	Carro de la compra	Presenta al usuario los detalles del pedido. Obtiene datos de envío.
purchase.php	Carro de la compra	Obtiene detalles de pago del usuario.
process.php	Carro de la compra	Procesa los detalles de pago y añade el pedido a la base de datos.
login.php	Administración	Permite al administrador conectarse para realizar cambios.
logout.php	Administración	Desconecta al usuario administrador.
admin.php	Administración	Menú principal de administración.
change_password_form.php	Administración	Formulario para que el administrador pueda cambiar su contraseña.
change_password.php	Administración	Cambia la contraseña del administrador.
insert_category_form.php	Administración	Formulario para que el administrador pueda añadir una nueva categoría a la base de datos.
insert_category.php	Administración	Añade la nueva categoría a la base de datos.
insert_book_form.php	Administración	Formulario para que el administrador pueda añadir un nuevo libro al sistema.

Nombre	Módulo	Descripción
`insert_book.php`	Administración	Añade el nuevo libro a la base de datos.
`edit_category_form.php`	Administración	Formulario para que el administrador pueda modificar una categoría.
`edit_category.php`	Administración base de datos.	Actualiza las categorías en la
`edit_book_form.php`	Administración	Formulario para que el administrador modifique los detalles de un libro.
`edit_book.php`	Administración	Actualiza el libro en la base de datos.
`delete_category.php`	Administración	Elimina una categoría de la base de datos.
`delete_book.php`	Administración	Elimina un libro de la base de datos.
`book_sc_fns.php`	Funciones	Colección de archivos de inclusión de esta aplicación.
`admin_fns.php`	Funciones	Colección de funciones utilizadas por las secuencias de comandos administrativas.
`book_fns.php`	Funciones	Colección de funciones para almacenar y recuperar datos sobre libros.
`order_fns.php`	Funciones	Colección de funciones para almacenar y recuperar datos sobre pedidos.
`output_fns.php`	Funciones	Colección de funciones para representar HTML.
`data_valid_fns.php`	Funciones	Colección de funciones para validar entradas de datos.
`db_fns.php`	Funciones	Colección de funciones para conectarse a la base de datos `book_sc`.
`user_auth_fns.php`	Funciones	Colección de funciones para autenticar usuarios administrativos.
`book_sc.sql`	SQL	SQL para definir la base de datos`book_sc`.
`populate.sql`	SQL	SQL para añadir datos de ejemplo a la base de datos `book_sc`.

A continuación veremos la implementación de cada uno de estos módulos.

Nota

Esta aplicación tiene una gran cantidad de código. Gran parte del mismo implementa funcionalidad que ya hemos visto (en especial en el capítulo anterior) como por ejemplo el almacenamiento y recuperación de datos de la base de datos, y la autenticación del usuario administrativo. Nos detendremos brevemente en este código, ya que nos centraremos en las funciones del carro de la compra.

Para que el código de este proyecto funcione como está escrito, tendrá que activar las comillas mágicas. Si no lo ha hecho antes, tendrá que añadir `addslashes()` a los datos destinados a la base de datos MySQL y `stripslahes()` a los datos que provengan de la misma. Lo hemos utilizado como método abreviado útil.

Puede activar las comillas mágicas en cada directorio de un archivo `.htaccess` por medio de la directiva `php_value magic_quotes_gpc on` (para PHP 4) o `php3_magic_quotes_gpc on` (para PHP 3).

Implementación de la base de datos

Como hemos mencionado anteriormente, hemos realizado algunos cambios en la base de datos Book-O-Rama que presentamos en un capítulo anterior.

El SQL para crear la base de datos `book_sc` se muestra en el listado 25.1.

Listado 25.1. book_sc.sql. SQL para crear la base de datos book_sc

```
create database book_sc;

use book_sc;

create table customers
(
   customerid int unsigned not null auto_increment primary key,
   name char(40) not null,
   address char(40) not null,
   city char(20) not null,
   state  char(20),
   zip char(10),
   country char(20) not null
);

create table orders
(
```

```
    orderid int unsigned not null auto_increment primary key,
    customerid int unsigned not null,
    amount float(6,2),
    date date not null,
    order_status  char(10),
    ship_name char(40) not null,
    ship_address char(40) not null,
    ship_city char(20) not null,
    ship_state char(20),
    ship_zip char(10),
    ship_country char(20) not null
);

create  table  books
(
     isbn char(13) not null primary key,
     author char(30),
     title char(60),
     catid int unsigned,
     price float(4,2) not null,
     description varchar(255)
);

create table categories
(
    catid int unsigned not null auto_increment primary key,
    catname char(40) not null
);

create table order_items
(
    orderid int unsigned not null,
    isbn char(13) not null,
    item_price float(4,2) not null,
    quantity tinyint unsigned not null,
    primary key (orderid, isbn)
);

create  table  admin
(
    username char(16) not null primary key,
    password char(16) not null
);

grant select, insert, update, delete
on book_sc.*
to book_sc@localhost identified by 'password';
```

Aunque la interfaz Book-O-Rama original era totalmente válida, ahora disponemos de algunos requisitos adicionales que tendrán que estar disponibles en línea.

A continuación se enumeran los cambios que hemos realizado en la base de datos original:

- Hemos añadido más campos de dirección para los clientes. Ahora es más importante ya que queremos crear una aplicación más realista.

- Hemos añadido una dirección de envío al pedido. Puede que la dirección de contacto de un cliente no sea la misma que la dirección de envío, sobre todo si está utilizando el sitio para comprar un regalo.
- Hemos añadido una tabla de categorías y un Id. de categorías a la tabla de libros. Al clasificar los libros en categorías resultará más sencillo navegar por el sitio.
- Hemos añadido `item_price` a la tabla `order_items` para indicar que el precio de un artículo puede cambiar. Queremos saber lo que costaba cuando lo adquirió el cliente.
- Hemos añadido una tabla `admin` para almacenar los detalles de conexión y la contraseña del administrador.
- Hemos eliminado la tabla `reviews` (puede añadirla como extensión al proyecto).
  En su lugar, cada libro dispone de un campo de descripción en el que se incluye una breve información sobre el mismo.

Para configurar esta base de datos en nuestro sistema, es necesario ejecutar la secuencia de comandos `book_sc.sql` a través de MySQL como usuario raíz, de esta forma:

```
mysql -u root -p < book_sc.sql
```

(Tendrá que indicar su contraseña raíz.)

Antes de nada, tendrá que cambiar la contraseña del usuario `book_sc` por otra distinta.

Si cambia la contraseña en `book_sc` también tendrá que cambiarla en `db_fns.php` (como veremos en breve).

Hemos incluido también un archivo de datos de ejemplo con el nombre `populate.sql`.

Puede añadir los datos a la base de datos si lo ejecuta desde MySQL de la misma forma.

Implementación del catálogo en línea

Esta aplicación cuenta con tres secuencias de comandos de catálogo: la página principal, la página de categorías y la página de detalles de libros.

La página principal del sitio se genera por medio de la secuencia de comandos `index.php`, cuyo resultado se muestra en la figura 25.3.

Comprobará que, además de la lista de categorías del sitio, hay un enlace al carro de la compra en la esquina superior derecha así como información resumida de los contenidos del mismo. Esto aparecerá en todas las páginas mientras el usuario navega y realiza sus compras.

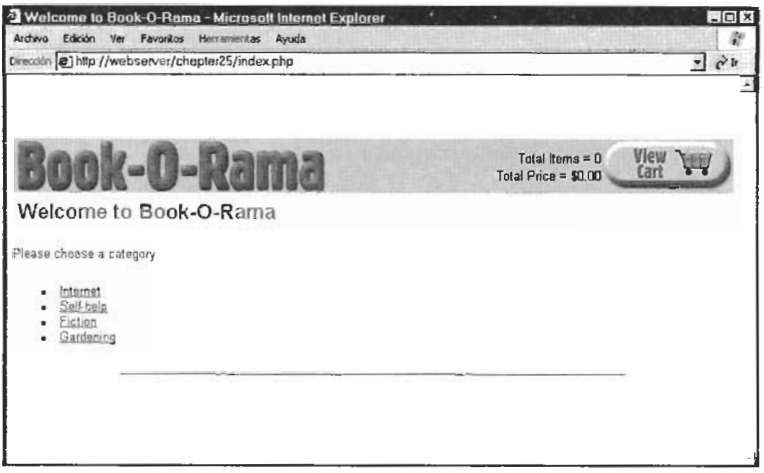


Figura 25.3. La página inicial del sitio enumera las categorías de libros en venta

Si el usuario pulsa sobre una de las categorías, accederá a la página de categorías, generada por la secuencia de comandos `show_cat.php`.

La página de categorías de la sección de libros sobre Internet se muestra en la figura 25.4.

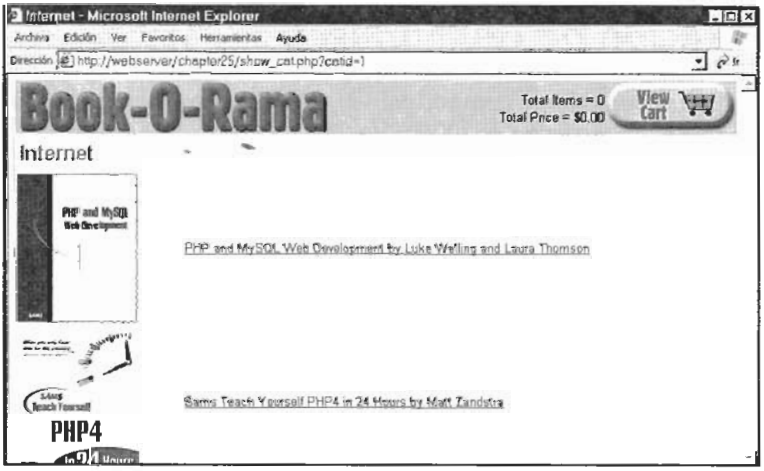


Figura 25.4. Cada uno de los libros de una categoría incluye una fotografía

Todos los libros de la categoría Internet se enumeran en forma de enlace.

Si el usuario pulsa sobre uno de estos enlaces, accederá a la página de detalles del libro.

En la figura 25.5 puede ver una página de detalles de un libro.

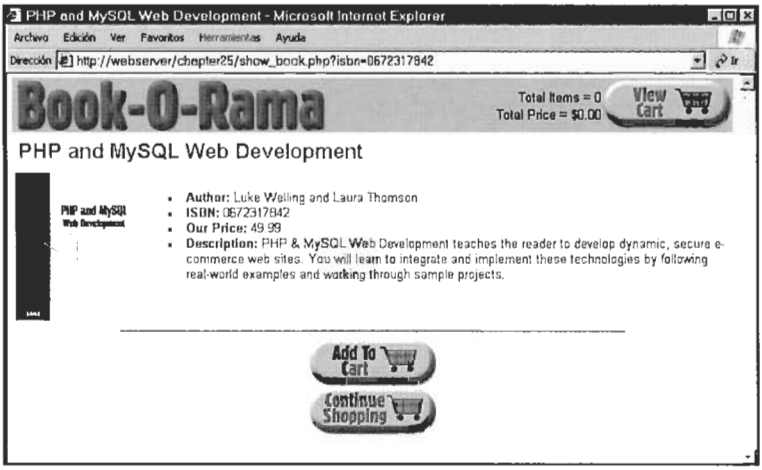


Figura 25.5. Cada libro cuenta con una página de detalles en la que se muestra información adicional así como una extensa descripción

En esta página, así como el enlace View Cart, vemos un enlace Add to Cart que permite al usuario seleccionar un artículo. Lo describiremos más adelante cuando veamos cómo se genera el carro de la compra.

A continuación analizaremos detalladamente cada una de estas secuencias de comandos.

Enumeración de categorías

La primera secuencia de comandos, `index.php`, enumera todas las categorías de la base de datos. Se muestra en el listado 25.2.

Listado 25.2. index.php. Secuencia de comandos para generar la página inicial del sitio

```
<?php
   include ('book_sc_fns.php');
   // El carro de la compra necesita sesiones; inicie una
   session_start();
   do_html_header('Welcome to Book-O-Rama');

   echo '<p>Please choose a category:</p>';

   // obtenga categorías de la base de datos
   $cat_array = get_categories();

   // muéstrelas como enlaces a las páginas del catálogo
   display_categories($cat_array);

   // si se conecta como administrador, muestre enlaces para añadir,
   // eliminar y modificar categorías
   if(isset($HTTP_SESSION_VARS['admin_user']))
```

```
    {
      display_button('admin.php', 'admin-menu', 'Admin Menu');
    }

    do_html_footer();
?>
```

En primer lugar, la secuencia añade `book_sc.php`, el archivo que incluye todas la bibliotecas de funciones de esta aplicación.

Tras ello, es necesario iniciar una sesión, para que la funcionalidad del carro de la compra sea efectiva. Todas las páginas del sitio utilizarán esta sesión.

Hay varias llamadas a funciones de representación HTML como `do_html_header()` y `do_html_footer()` (ambas incluidas en `output_fns.php`).

También tenemos un código que comprueba si el usuario se ha conectado como administrador y que le ofrece distintas opciones si lo ha hecho, como veremos en el apartado de funciones de administración.

La parte más importante de esta secuencia de comandos es

```
// obtenga categorías de la base de datos
$cat_array = get_categories();

// muéstrelas como enlaces a las páginas del catálogo
display_categories($cat_array);
```

Las funciones `get_categories()` y `display_categories()` se encuentran en las bibliotecas de funciones `book_fns.php` y `output_fns.php`, respectivamente. La función `get_categories()` devuelve una matriz de las categorías del sistema que, tras ello, se pasa a `display_categories()`. En el listado 25.3 podemos ver el código de `get_categories()`.

Listado 25.3. Función get_categories() de book_fns.php. Esta función recupera una lista de categorías de la base de datos

```
function get_categories()
{
    // consulte la base de datos para obtener una lista de categorías
    $conn = db_connect();
    $query = 'select catid, catname
            from categories';
    $result = @mysql_query($query);
    if (!$result)
      return false;
    $num_cats = @mysql_num_rows($result);
    if ($num_cats ==0)
       return false;
    $result = db_result_to_array($result);
    return $result;
}
```

Como puede comprobar, esta función se conecta a la base de datos y recupera una lista de todos los Id. y nombres de categoría. Hemos escrito y utilizado una

función denominada `db_result_to_array()` que se encuentra en `db_fns.php`. Esta función se muestra en el listado 25.4.

Adopta un identificador de resultados MySQL y devuelve una matriz de errores indexada numéricamente en la que cada fila es una matriz asociativa.

Listado 25.4. Función db_result_to_array() de db_fns.php. Función que convierte un identificador de resultados MySQL en una matriz de resultados

```
function db_result_to_array($result)
{
   $res_array = array();

   for ($count=0; $row = @mysql_fetch_array($result); $count++)
      $res_array[$count] = $row;

   return $res_array;
}
```

En nuestro caso, devolveremos esta matriz de nuevo a `index.php` donde la pasaremos a la función `display_categories()` de `output_fns.php`. Esta función muestra cada categoría en forma de enlace a una página que contiene los libros de dicha categoría. En el listado 25.5 podemos ver el código de la función.

Listado 25.5. Función display_categories() de output_fns.php. Función que muestra una matriz de categorías como lista de enlaces a dichas categorías

```
function  display_categories($cat_array)
{
   if (!is_array($cat_array))
   {
      echo 'No categories currently available<br />';
      return;
   }
   echo '<ul>';
   foreach ($cat_array as $row)
   {
      $url  =  'show_cat.php?catid='.($row['catid']);
      $title = $row['catname'];
      echo '<li>';
      do_html_url($url, $title);
      echo '</li>
   }
   echo '</ul>';
   echo '<hr />';
}
```

Esta función convierte cada una de las categorías de la base de datos en enlaces. Todos los enlaces llevan a la siguiente secuencia de comandos, `show_cat.php`, pero cada uno tiene un parámetro distinto, el Id. de la categoría o `catid` (se trata de un número exclusivo generado por MySQL y que se utiliza para identificar la categoría).

Este parámetro a la siguiente secuencia de comandos determinará en qué categoría nos encontramos.

Enumeración de libros en una categoría

El proceso para enumerar libros de una categoría es muy similar.

La secuencia de comandos encargada de ello es `show_cat.php`, que podemos ver en el listado 25.6.

Listado 25.6. show_cat.php. Esta secuencia de comandos muestra los libros de una determinada categoría

```
<?php
   include ('book_sc_fns.php');
   // El carro de la compra necesita sesiones; inicie una
   session_start();

   $catid  =  $HTTP_GET_VARS['catid'];
   $name = get_category_name($catid);

   do_html_header($name);

   // obtenga la información del libro de la base de datos
   $book_array = get_books($catid);

   display_books($book_array);


   // si se conecta como administrador, muestre enlaces para añadir y
   // eliminar libros
   if(isset($HTTP_SESSION_VARS['admin_user']))
   {
      display_button('index.php', 'continue', 'Continue Shopping');
      display_button('admin.php', 'admin-menu', 'Admin Menu');
      display_button("edit_category_form.php?catid=$catid",
       'edit-category', 'Edit Category');
   }
   else
      display_button('index.php',  'continue-shopping',  'Continue
Shopping');

   do_html_footer();
?>
```

Esta secuencia de comandos tiene una estructura muy similar a la de la página de índice, a excepción de que recupera libros en lugar de categorías.

Como de costumbre empezamos con `session_start()` y, tras ello, convertimos el Id. de categoría que hemos obtenido en un nombre de categoría por medio de la función `get_category_name()`, como se indica a continuación:

```
$name = get_category_name($catid);
```

Esta función busca el nombre de la categoría en la base de datos. Su código se muestra en el listado 25.7.

Listado 25.7. Función get_category_name() de book_fns.php. Esta función convierte un Id. de categoría en un nombre de categoría

```
function get_category_name($catid)
{
   // consulte la base de datos para obtener el nombre de un Id. de
   // categoría
   $conn = db_connect();
   $query = "select catname
      from categories
      where catid = $catid";
   $result = @mysql_query($query);
   if (!$result)
      return false;
   $num_cats = @mysql_num_rows($result);
   if ($num_cats ==0)
      return false;
   $result = mysql_result($result, 0, 'catname');
   return $result;
}
```

Una vez recuperado el nombre de la categoría, podemos representar un encabezado HTML y proceder a recuperar y enumerar los libros de la base de datos que se corresponden con la categoría seleccionada, como se muestra a continuación:

```
$book_array = get_books($catid);
display_books($book_array);
```

Las funciones `get_books()` y `display_books()` son muy similares a las funciones `get_categories()` y `display_categories()`, por lo que no las describiremos. La única diferencia es que estamos recuperando información de la tabla de libros en lugar de la de categorías. La función `display_books()` proporciona un enlace a todos los libros de una categoría por medio de la secuencia de comandos `show_book.php`. De nuevo, a cada enlace se le añade un parámetro como sufijo. En esta ocasión es el ISBN del libro en cuestión.

En la parte inferior de la secuencia de comandos `show_cat.php`, verá que hay un código para representar funciones adicionales si se conecta un administrador, como veremos en el apartado sobre funciones administrativas.

Cómo mostrar detalles sobre un libro

La secuencia de comandos `show_book.php` adopta como parámetro un ISBN y recupera y muestra los detalles del libro en cuestión.

El código de esta secuencia de comandos se incluye en el listado 25.8.

Listado 25.8. show_book.php. Esta secuencia de comandos muestra los detalles de un determinado libro

```
<?php
   include ('book_sc_fns.php');
```

```
    // El carro de la compra necesita sesiones; inicie una
    session_start();

    $isbn = $HTTP_GET_VARS['isbn'];

    // obtenga este libro de la base de datos
    $book = get_book_details($isbn);
    do_html_header($book['title']);
    display_book_details($book);

    // defina un URL para el "botón de continuación"
    $target  =  'index.php';
    if($book['catid'])
    {
       $target = 'show_cat.php?catid='.$book['catid'];
    }
    // si se conecta como administrador, muestre enlaces para modificar
    // libros
    if( check_admin_user() )
    {
       display_button("edit_book_form.php?isbn=$isbn", 'edit-item', 'Edit
Item');
       display_button('admin.php', 'admin-menu', 'Admin Menu');
       display_button($target, 'continue', 'Continue');
    }
    else
    {
       display_button("show_cart.php?new=$isbn", 'add-to-cart', 'Add '
             .$book['title'].' To My Shopping Cart');
       display_button($target, 'continue-shopping', 'Continue Shopping');
    }

    do_html_footer();
?>
```

Esta secuencia de comandos realiza el mismo tipo de acciones que hemos descrito en páginas anteriores.

En primer lugar iniciamos la sesión y, tras ello, utilizamos

```
$book  =  get_book_details($isbn);
```

para obtener de la base de datos la información del libro y

```
display_book_details($book);
```

para representar los datos en HTML.

Conviene mencionar que `display_book_details()` busca un archivo de imagen para el libro como `images/$isbn.jpg`. Si este archivo no existe, no se muestra ninguna imagen.

El resto de la secuencia de comandos configura la navegación. Un usuario normal tendrá las opciones **Continue Shopping**, lo que le devolverá a la página de categorías y **Add to Cart**, que añadirá el libro al carro de la compra. Si el usuario se ha conectado como administrador, dispondrá de opciones diferentes, que analizaremos más adelante.

Con esto se completan los elementos básicos del sistema de catálogo. A continuación describiremos el código correspondiente a la funcionalidad del carro de la compra.

Implementación del carro de la compra

La funcionalidad del carro de la compra se centra en la variable de sesión `cart`. Se trata de una matriz asociativa que utiliza ISBN como claves y cantidades como valores. Por ejemplo, si añadimos una sola copia de este libro al carro de la compra, la matriz contendrá

```
0672317842=> 1
```

Es decir, una copia del libro con el ISBN 0672317842. Cuando añadimos artículos al carro, se añaden a esta matriz.

Cuando revisamos el carro, utilizamos la matriz `cart` para buscar los detalles completos de los artículos en la base de datos.

También utilizamos otras dos variables de sesión para controlar la pantalla del encabezado que muestra Total Items y Total Price. Estas variables son `items` y `total_price` respectivamente.

Uso de la secuencia de comandos show_cart.php

En primer lugar veremos cómo se implementa el código del carro de la compra por medio de la secuencia de comandos `show_cart.php`.

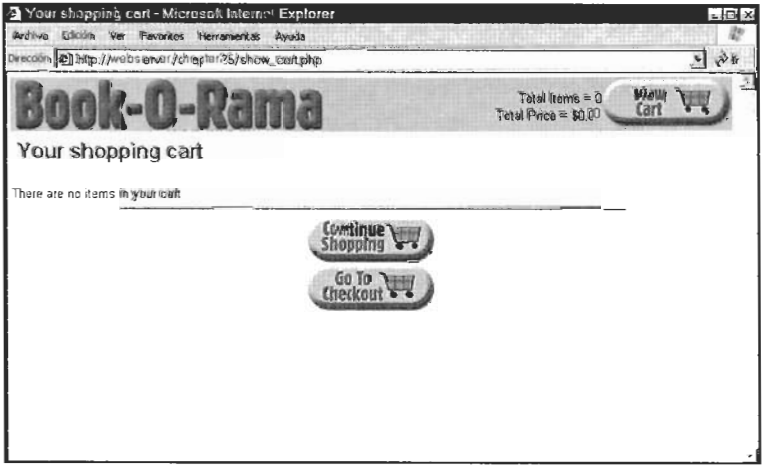


Figura 25.6. La secuencia de comandos show_cart.php sin parámetros muestra únicamente el contenido de nuestro carro

Se trata de la secuencia de comandos que muestra la página a la que accedemos si pulsamos sobre cualquier enlace View Cart o Add to Cart.

Si invocamos `show_cart.php` sin parámetros, podremos ver los contenidos de la misma.

Si la invocamos con un ISBN como parámetro, se añadirá al carro el artículo que tenga dicho ISBN.

Para comprenderlo, fíjese en la figura 25.6.

En este caso, hemos pulsado sobre el enlace View Cart con el carro vacío, es decir, no hemos seleccionado ningún artículo para su compra.

En la figura 25.7 podemos ver el carro después de seleccionar dos libros. En este caso, hemos accedido a esta página tras pulsar sobre el enlace Add to Cart de la página `show_book.php` del libro *PHP and MySQL Web Development*. Si se fija en la barra de URL, verá que en esta ocasión hemos invocado la secuencia de comandos con un parámetro.

El parámetro es `new` y tiene el valor `0672317842`, es decir, el ISBN del libro que acabamos de añadir al carro.

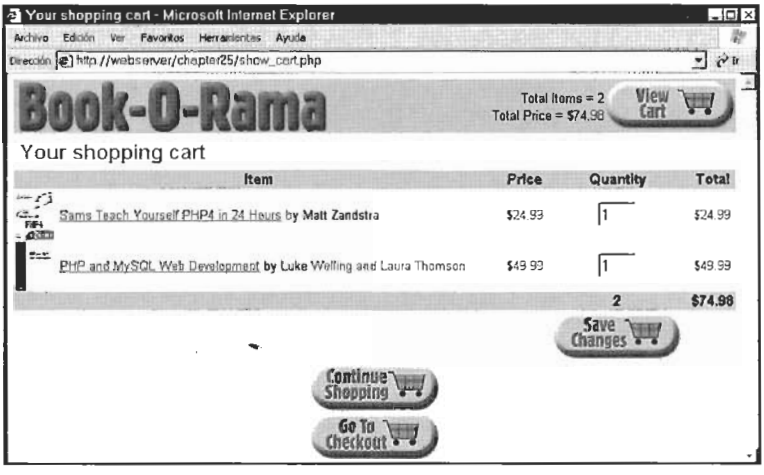


Figura 25.7. La secuencia show_cart.php con el nuevo parámetro añade un nuevo artículo al carro

Comprobará que desde esta página tenemos dos opciones.

Hay un botón **Save Changes** que podemos utilizar para cambiar la cantidad de artículos del carro. Para ello, podemos modificar las cantidades directamente y pulsar **Save Changes**.

De hecho se trata de un botón de envío que nos lleva de nuevo a la secuencia de comandos `show_cart.php` para actualizar el carro.

Además, hay un botón **Go to Checkout** que el usuario puede pulsar cuando quiera salir, como veremos más adelante.

Por el momento, nos detendremos en el código de la secuencia de comandos `show_cart.php`, que se muestra en el listado 25.9.

Listado 25.9. show_cart.php. Esta secuencia de comandos controla el carro de la compra

```
<?php
   include ('book_sc_fns.php');
   // El carro de la compra necesita sesiones; inicie una
   session_start();

   @ $new = $HTTP_GET_VARS['new'];

   if($new)
   {
      //nuevo elemento seleccionado
      if(!isset($HTTP_SESSION_VARS['cart']))
      {
         $HTTP_SESSION_VARS['cart'] = array();
         $HTTP_SESSION_VARS['items'] = 0;
         $HTTP_SESSION_VARS['total_price']  ='0.00';
      }
      if(isset($HTTP_SESSION_VARS['cart'][$new]))
         $HTTP_SESSION_VARS['cart'][$new]++;
      else
         $HTTP_SESSION_VARS['cart'][$new] = 1;
      $HTTP_SESSION_VARS['total_price'] =
                  calculate_price($HTTP_SESSION_VARS['cart']);
      $HTTP_SESSION_VARS['items']  =
calculate_items($HTTP_SESSION_VARS['cart']);

   }
   if(isset($HTTP_POST_VARS['save']))
   {
      foreach ($HTTP_SESSION_VARS['cart'] as $isbn => $qty)
      {
         if($HTTP_POST_VARS[$isbn]=='0')
            unset($HTTP_SESSION_VARS['cart'][$isbn]);
         else
            $HTTP_SESSION_VARS['cart'][$isbn] =
            $HTTP_POST_VARS[$isbn];
      }
      $HTTP_SESSION_VARS['total_price']  =
               calculate_price($HTTP_SESSION_VARS['cart']);
      $HTTP_SESSION_VARS['items'] =
calculate_items($HTTP_SESSION_VARS['cart']);
   }

   do_html_header('Your shopping cart');


if($HTTP_SESSION_VARS['cart']&&array_count_values($HTTP_SESSION_VARS['cart']))
      display_cart($HTTP_SESSION_VARS['cart']);
   else
   {
      echo '<p>There are no items in your cart</p>';
      echo '<hr />';
   }
```

```
    $target = 'index.php';

    // si acaba de añadir un elemento al carro, continúe comparando en dicha
    // categoría
    if($new)
    {
       $details = get_book_details($new);
       if($details['catid'])
          $target = 'show_cat.php?catid='.$details['catid'];
    }
    display_button($target, 'continue-shopping', 'Continue Shopping');

    // utilice esto si ha configurado SSL
    // $path = $HTTP_SERVER_VARS['PHP_SELF'];
    // $server = $HTTP_SERVER_VARS['SERVER_NAME'];
    // $path = str_replace('show_cart.php', '', $path);
    // display_button('https://'.$server.$path.'checkout.php',
            'go-to-checkout', 'Go To Checkout');

    // si no ha configurado SSL, utilice el siguiente código
    display_button('checkout.php', 'go-to-checkout', 'Go To
Checkout');


    do_html_footer();
?>
```

En esta secuencia de comandos hay tres partes principales: la representación del carro, la inclusión de los artículos en el carro y la aplicación de los cambios. Describiremos cada una de estas partes en los siguientes apartados.

Cómo ver el carro

Independientemente de la página de la que vengamos, podremos mostrar el contenido del carro. En el caso básico, cuando un usuario acaba de pulsar **View Cart**, la única parte del código que se ejecutará es la que mostramos a continuación:

```
if($HTTP_SESSION_VARS['cart']&&array_count_values($HTTP_SESSION_VARS['cart']))
    display_cart($HTTP_SESSION_VARS['cart']);
else
{
    echo '<p>There are no items in your cart</p>';
    echo '<hr />';
}
```

Como puede comprobar, si tenemos un cuadro con algún tipo de contenido, invocaremos la función `display_cart()`. Si el carro está vacío, mostraremos el correspondiente mensaje al usuario.

La función `display_cart()` simplemente imprime los contenidos del carro en un formato HTML legible, como puede comprobar en las figuras 25.6 y 25.7.

El código de esta función se incluye en `output_fns.php` que mostramos en el listado 25.10.

Aunque es una función de representación, es bastante compleja, razón por la que la incluimos a continuación.

Listado 25.10. Función display_cart() de output_fns.php. Esta función aplica un formato a los contenidos del carro de la compra y los imprime

```
function display_cart($cart, $change = true, $images = 1)
{
   // muestre los artículos del carro de la compra
   // opcionalmente permita cambios (true o false)
   // opcionalmente incluya imágenes (1 - sí, 0 - no)

   global $HTTP_SESSION_VARS;

   echo '<table border="0" width="100%" cellspacing="0">
           <form action="show_cart.php" method="post">
           <tr><th colspan="'. (1+$images) .'" bgcolor="#cccccc">Item</th>
           <th bgcolor="#cccccc">Price</th><th bgcolor="#cccccc">Quantity</
th>
           <th  bgcolor="#cccccc">Total</th></tr>';

   //muestre cada elemento como fila de una tabla
   foreach ($cart as $isbn => $qty)
   {
      $book = get_book_details($isbn);
      echo '<tr>';
      if($images  ==true)
      {
         echo '<td align="left">';
         if (file_exists("images/$isbn.jpg"))
         {
             $size = GetImageSize('images/'.$isbn.'.jpg');
             if($size[0]>0 && $size[1]>0)
             {
                echo '<img src="images/'.$isbn.'.jpg" border=0 ';
                echo 'width = '. $size[0]/3 .' height = ' .$size[1]/3 . '>';
             }
         }
         else
             echo ' ';
         echo  '</td>';
      }
      echo '<td align="left">';
      echo '<a href="show_book.php?isbn='.$isbn.'">'.$book['title'].
           '</a> by '.$book['author'];
      echo '</td><td align="center">$'.number_format($book['price'], 2);
      echo '</td><td align="center">';
      // si permitimos cambios, las cantidades se muestran en cuadros de
      // texto
      if ($change == true)
         echo '<input type="text" name="$isbn" value="$qty" size="3">';
      else
         echo $qty;
      echo '</td><td align="center">$'.number_format($book['price']*$qty,2).
           '</td></tr>\n';
```

```
    }
    // muestre el total de filas
    echo '<tr>
                <th colspan="'. (2+$images) .'"
bgcolor="#cccccc"> </td>
                <th align = center bgcolor="#cccccc">
                    '.$HTTP_SESSION_VARS['items'].'
                </th>
                <th align = center bgcolor="#cccccc">
                    $'..number_format($HTTP_SESSION_VARS['total_price'], 2).
                '</th>
            </tr>';
    // muestre el botón para guardar cambios
    if($change == true)
    {
        echo '<tr>
                    <td colspan="'. (2+$images) .'"> </td>
                    <td align="center">
                        <input type="hidden" name="save"
                        value="true">
                        <input type="image" src="images/save-
                        changes.gif"
                            border="0" alt="Save
                            Changes">
                    </td>
                    <td> </td>
                </tr>';
    }
    echo '</form></table>';
}
```

El flujo básico de esta función se describe a continuación:

1. Realiza un bucle por cada uno de los artículos del carro y pasa el ISBN de cada uno a `get_book_details()` para que podamos resumir los detalles de cada libro.
2. Proporciona una imagen para cada libro, si existe alguna. En este caso utilizamos las etiquetas HTML de altura y anchura de imagen para reducir el tamaño de la misma. Esto significa que las imágenes aparecerán ligeramente distorsionadas pero lo suficientemente reducidas para poder utilizarlas (siempre puede cambiar el tamaño con ayuda de la biblioteca gd que describimos en un capítulo anterior o generar manualmente imágenes de distinto tamaño para cada producto).
3. Cada entrada del carro dispone de un enlace al correspondiente libro, es decir, a `show_book.php` con el ISBN como parámetro.
4. Si invocamos la función con el parámetro `change` configurado como `true` (sin configurar adopta `true` de forma predeterminada), muestra los cuadros con las cantidades en forma de formulario con el botón **Save Changes** en la parte final (al volver a utilizar esta función después de salir, el usuario no debe poder cambiar su pedido).

Realmente esta función no es excesivamente complicada, pero requiere una gran cantidad de trabajo, por lo que debería repasarla con atención.

Cómo añadir artículos al carro

Si un usuario ha accedido a la página `show_cart.php` tras pulsar el botón **Add to Cart**, tendremos que realizar algunas operaciones antes de poder mostrarle los contenidos de su carro. En concreto, será necesario añadir el correspondiente artículo al carro, como indicamos a continuación.

En primer lugar, si el usuario todavía no ha añadido artículos al carro, no tendrá un carro, por lo que será necesario crear uno:

```
if(!isset($HTTP_SESSION_VARS['cart']))
{
   $HTTP_SESSION_VARS['cart']  =  array();
   $HTTP_SESSION_VARS['items']  =  0;
   $HTTP_SESSION_VARS['total_price'] ='0.00';
}
```

Inicialmente, el carro está vacío.

Una vez creado el carro, podemos añadir artículos al mismo:

```
if(isset($HTTP_SESSION_VARS['cart'][$new]))
   $HTTP_SESSION_VARS['cart'][$new]++;
else
   $HTTP_SESSION_VARS['cart'][$new] = 1;
```

En este caso, comprobamos si el artículo ya se encuentra en el carro. Si ya lo está, incrementamos en uno la cantidad de dicho artículo en el carro. En caso contrario, añadimos el nuevo artículo al carro.

Tras ello, será necesario calcular el precio y el número total de los artículos del carro. Para ello utilizaremos las funciones `calculate_price()` y `calculate_items()` de esta forma:

```
$HTTP_SESSION_VARS['total_price'] =
calculate_price($HTTP_SESSION_VARS['cart']);
$HTTP_SESSION_VARS['items'] = calculate_items($HTTP_SESSION_VARS['cart']);
```

Estas funciones se encuentran en la biblioteca de funciones `book_fns.php`. El código de cada una de ellas se muestra en los listados 25.11 y 25.12 respectivamente.

Listado 25.11. Función calculate_price() de book_fns.php. Esta función calcula y devuelve el precio total de los contenidos del carro de la compra

```
function calculate_price($cart)
{
   // calcule el precie total de todos los artículos del carro
   $price = 0.0;
   if(is_array($cart))
   {
```

```
    $conn = db_connect();
    foreach($cart as $isbn => $qty)
    {
       $query = "select price from books where isbn='$isbn'";
       $result = mysql_query($query);
       if ($result)
       {
          $item_price = mysql_result($result, 0, 'price');
          $price +=$item_price*$qty;
       }
    }
  }
  return $price;
}
```

Como puede comprobar, el funcionamiento de `calculate_price()` consiste en buscar el precio de cada artículo en la base de datos. Es un poco lento, por lo que para evitar repetir esta operación más de lo necesario, almacenaremos el precio (así como el número total de artículos) como variables de sesión y solamente repetiremos los cálculos cuando el carro cambie.

Listado 25.12. Función calculate_items() de book_fns.php. Esta función calcula y devuelve el número total de artículos del carro de la compra

```
function  calculate_items($cart)
{
  // sume todos los artículos del carro
  $items = 0;
  if(is_array($cart))
  {
    foreach($cart as $isbn => $qty)
    {
      $items  +=  $qty;
    }
  }
  return $items;
}
```

La función `calculate_items()` es más sencilla; simplemente suma las cantidades de cada artículo del carro para obtener el número total.

Cómo guardar el carro actualizado

Si hemos llegado a la secuencia de comandos `show_cart.php` tras pulsar el botón **Save Changes**, el proceso es ligeramente distinto. En este caso, hemos accedido a través del envío de un formulario. Si se fija en el código, comprobará que el botón **Save Changes** es el botón de envío de dicho formulario. Este formulario contiene la variable oculta `save`. Si configuramos esta variable, sabremos que hemos accedido a esta secuencia de comandos a través del botón **Save Changes**. Esto significa que probablemente el usuario ha modificado los valores de cantidad del carro y que tendremos que actualizarlos.

Si observa los cuadros de texto de la parte del formulario **Save Changes** de la secuencia de comandos, comprobará que reciben el nombre del ISBN del artículo que representan, como se indica a continuación:

```
echo '<input type="text" name="$isbn" value="$qty" size="3">';
```

Fíjese ahora en la parte de la secuencia de comandos que guarda los cambios:

```
if(isset($HTTP_POST_VARS['save']))
{
   foreach ($HTTP_SESSION_VARS['cart'] as $isbn => $qty)
   {
      if($HTTP_POST_VARS[$isbn]=='0')
         unset($HTTP_SESSION_VARS['cart'][$isbn]);
      else
         $HTTP_SESSION_VARS['cart'][$isbn]  =  $HTTP_POST_VARS[$isbn];
   }
   $HTTP_SESSION_VARS['total_price'] =
calculate_price($HTTP_SESSION_VARS['cart']);
   $HTTP_SESSION_VARS['items'] =
calculate_items($HTTP_SESSION_VARS['cart']);
}
```

Comprobará que estamos procesando el carro de la compra y que, en cada `isbn` del carro, comprobamos la variable POST con ese nombre. Se trata de los campos del formulario **Save Changes**.

Si alguno de los campos se configura con el valor cero, eliminamos dicho artículo del carro por medio de `unset()`. En caso contrario, actualizamos el carro para que coincida con los campos del formulario, de esta forma:

```
if($HTTP_POST_VARS[$isbn]=='0')
   unset($HTTP_SESSION_VARS['cart'][$isbn]);
else
   $HTTP_SESSION_VARS['cart'][$isbn] = $HTTP_POST_VARS[$isbn];
```

Después de realizar las actualizaciones, utilizamos `calculate_price()` y `calculate_items()` de nuevo para determinar los nuevos valores de las variables `total_price` e `items`.

Impresión de un resumen en la barra de encabezado

Habrá comprobado que en la barra de encabezado de todas las páginas del sitio se muestra un resumen del contenido del carro. Para obtener este resumen, se imprimen los valores de las variables de sesión `total_price` e `item`, operación que se lleva a cabo en la función `do_hmtl_header()`. Estas variables se registran cuando el usuario visita por primera vez la página `show_cart.php`. También necesitaremos cierta lógica para los casos en los que el usuario no haya visitado dicha página. Esta lógica también se encuentra en la función `do_html_ header()`:

```
if(!$HTTP_SESSION_VARS['items']) $HTTP_SESSION_VARS['items'] = '0';
```

```
if(!$HTTP_SESSION_VARS['total_price']) $HTTP_SESSION_VARS['total_price'] =
'0.00';
```

Cómo salir

Cuando el usuario pulsa el botón **Go to Checkout** desde el carro de la compra, se activa la secuencia de comandos `checkout.php`. A esta página, así como a las que se encuentran por detrás de la misma, se debe acceder a través de SSL, pero la aplicación de ejemplo no le obliga a que lo haga (en capítulos anteriores encontrará más información sobre SSL). La página de salida se muestra en la figura 25.8.

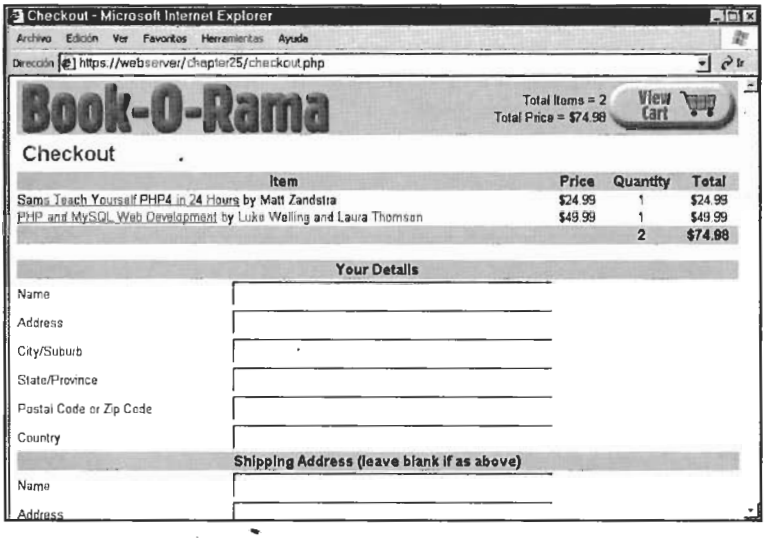


Figura 25.8. La secuencia de comandos checkout.php obtiene los detalles del cliente

Esta secuencia de comandos requiere que el cliente introduzca su dirección (y la dirección de envío en caso de que sea diferente). Es bastante sencilla, como puede comprobar en el código del listado 25.13.

Listado 25.13. checkout.php. Esta secuencia de comandos obtiene los detalles del cliente

```
<?php
   //incluya nuestro conjunto de funciones
   include ('book_sc_fns.php');

   // El carro de la compra necesita sesiones; inicio una
   session_start();

   do_html_header('Checkout');

if($HTTP_SESSION_VARS['cart']&&array_count_values($HTTP_SESSION_VARS['cart']))
```

```
    {
      display_cart($HTTP_SESSION_VARS['cart'], false, 0);
      display_checkout_form();
    }
    else
      echo '<p>There are no items in your cart</p>';

    display_button('show_cart.php', 'continue-shopping', 'Continue
Shopping');

    do_html_footer();
?>
```

No hay nada extraño en esta secuencia de comandos. Si el carro está vacío, se lo indica al cliente; en caso contrario, muestra el formulario de la figura 25.8.

Si el usuario quiere continuar y pulsa el botón **Purchase** situado en la parte inferior del formulario, accederá a la secuencia de comandos `purchase.php`, cuyo resultado se muestra en la figura 25.9.

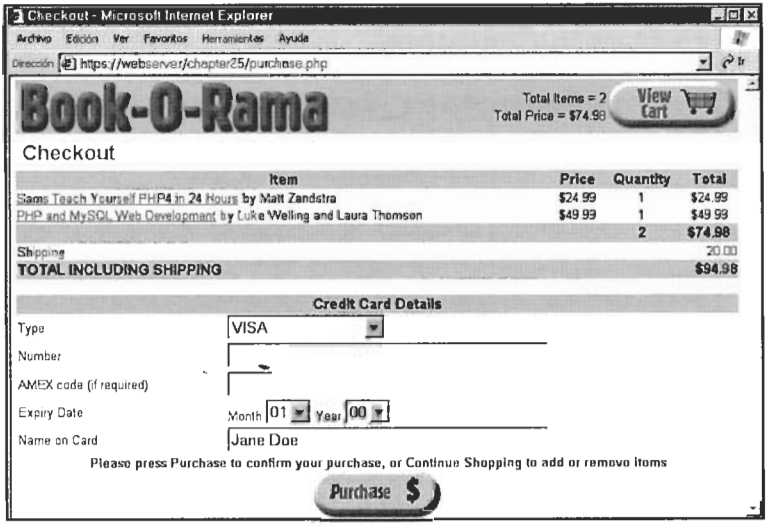


Figura 25.9. La secuencia de comandos purchase.php calcula el envío y el total del pedido, y obtiene los detalles de pago del cliente

El código de esta secuencia de comandos es ligeramente más complejo que el de `checkout.php`. Lo mostramos en el listado 25.14.

Listado 25.14. purchase.php. Esta secuencia de comandos muestra los detalles del pedido en la base de datos y obtiene los detalles de pago

```
<?php

  include ('book_sc_fns.php');
  // El carro de la compra necesita sesiones; inicie una
```

```
    session_start();

    do_html_header("Checkout");

    //cree nombres de variables cortos
    $name = $HTTP_POST_VARS['name'];
    $address = $HTTP_POST_VARS['address'];
    $city = $HTTP_POST_VARS['city'];
    $zip = $HTTP_POST_VARS['zip'];
    $country = $HTTP_POST_VARS['country'];

    // si se completa
    if($HTTP_SESSION_VARS['cart']&&$name&&$address&&$city&&$zip&&$country)
    {
       // se puede añadir a la base de datos
       if( insert_order($HTTP_POST_VARS)!=false )
       {
          //muestre el carro, sin permitir cambios y sin imágenes
          display_cart($HTTP_SESSION_VARS['cart'], false, 0);

          display_shipping(calculate_shipping_cost());

          //obtenga los datos de la tarjeta de crédito
          display_card_form($name);

          display_button('show_cart.php', 'continue-shopping', 'Continue
Shopping');
       }
       else
       {
          echo 'Could not store data, please try again.';
          display_button('checkout.php', 'back', 'Back');
       }
    }
    else
    {
       echo 'You did not fill in all the fields, please try again.<hr />';
       display_button('checkout.php', 'back', 'Back');
    }

    do_html_footer();
?>
```

La lógica es muy sencilla. Comprobamos que el usuario haya completado el formulario e introducido los detalles en la base de datos por medio de la función `insert_order()`. Se trata de una función muy sencilla que añade los detalles del cliente a la base de datos. Su código se incluye en el listado 25.15.

Listado 25.15. Función insert_order() de order_fns.php. Esta función añade todos los detalles del pedido del cliente a la base de datos

```
function insert_order($order_details)
{
```

```
    global $HTTP_SESSION_VARS;

    //extraiga order_details como variables
    extract($order_details);


    //defina la dirección de envío igual que la dirección normal
    if(!$ship_name&&!$ship_address&&!$ship_city&&
        !$ship_state&&!$ship_zip&&!$ship_country)
    {
       $ship_name = $name;
       $ship_address = $address;
       $ship_city = $city;
       $ship_state = $state;
       $ship_zip  =  $zip;
       $ship_country = $country;
    }

    $conn  =  db_connect();

    //añada  la  dirección  del  cliente
    $query  =  "select  customerid  from  customers  where
            name  =  '$name'  and  address  =  '$address'
            and city = '$city' and state = '$state'
            and zip = '$zip' and country = '$country'";
    $result = mysql_query($query);
    if(mysql_numrows($result)>0)
    {
       $customer_id = mysql_result($result, 0, 'customerid');
    }
    else
    {
       $query = "insert into customers values
                  ('',
'$name','$address','$city','$state','$zip','$country')";
       $result  =  mysql_query($query);
       if  (!$result)
           return false;
    }
    $query = "select customerid from customers where
            name = '$name' and address = '$address'
            and city = '$city' and state = '$state'
            and zip = '$zip' and country = '$country'";
    $result = mysql_query($query);
    if(mysql_numrows($result)>0)
       $customerid = mysql_result($result, 0, 'customerid');
    else
       return false;
    $date = date('Y-m-d');
    $query = "insert into orders values
            ('',  $customerid,  ".$HTTP_SESSION_VARS['total_price'].
             ", '$date', 'PARTIAL', '$ship_name',
             '$ship_address','$ship_city','$ship_state','$ship_zip',
             '$ship_country')";
    $result  =  mysql_query($query);
    if  (!$result)
```

```
        return false;

    $query = "select orderid from orders where
            customerid = $customerid and
            amount > ".$HTTP_SESSION_VARS['total_price']."-.001 and
            amount < ".$HTTP_SESSION_VARS['total_price']."+.001 and
            date = '$date' and
            order_status = 'PARTIAL' and
            ship_name = '$ship_name' and
            ship_address = '$ship_address' and
            ship_city = '$ship_city' and
            ship_state = '$ship_state' and
            ship_zip = '$ship_zip' and
            ship_country = '$ship_country'";
    $result = mysql_query($query);
    if(mysql_numrows($result)>0)
        $orderid = mysql_result($result, 0, 'orderid');
    else
        return false;


    // añada todos los libros
    foreach($HTTP_SESSION_VARS['cart'] as $isbn => $quantity)
    {
        $detail = get_book_details($isbn);
        $query = "delete from order_items where
                orderid = '$orderid' and isbn = '$isbn'";
        $result = mysql_query($query);
        $query = "insert into order_items values
                ('$orderid', '$isbn', ".$detail['price'].", $quantity)";
        $result = mysql_query($query);
        if(!$result)
            return false;
    }

    return $orderid;
}
```

La función es extensa ya que nos permite añadir los detalles del cliente, los detalles del pedido y los detalles de todos los libros que quiere comprar.

Tras ello cargamos los gastos de envío a la dirección del cliente y le indicamos a cuánto ascienden por medio de la siguiente línea de código:

```
display_shipping(calculate_shipping_cost());
```

El código que utilizamos para `calculate_shipping_cost()` siempre devuelve 20 dólares.

Cuando diseñe un sitio de compras, tendrá que seleccionar un método de entrega, determinar los gastos en función de los diferentes destinos y calcular el total correspondiente.

Seguidamente, mostramos un formulario al usuario en el que debe introducir los datos de su tarjeta de crédito, por medio de la función `display_card_form()` de la biblioteca `output_fns.php`.

Implementación del pago

Cuando el usuario pulsa el botón **Purchase**, procesaremos sus detalles de pago por medio de la secuencia de comandos `process.php`. En la figura 25.10 puede ver el resultado de un pago satisfactorio.

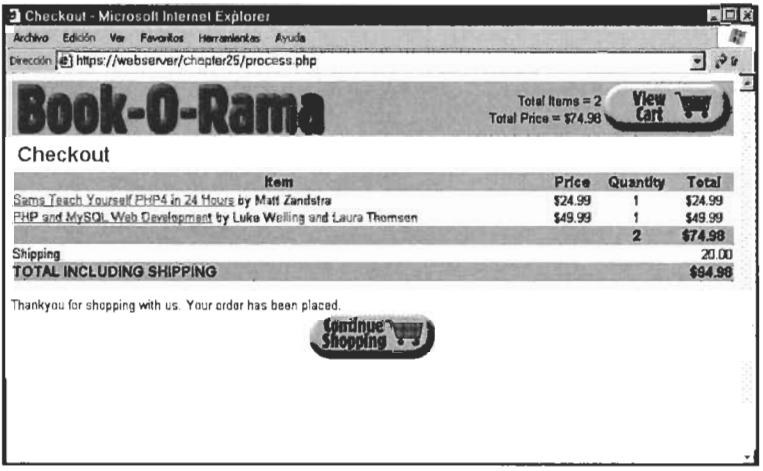


Figura 25.10. Esta transacción es correcta, por lo que se enviarán los artículos

En el listado 25.16 se incluye el código de `process.php`.

Listado 25.16. process.php. Secuencia de comandos que procesa el pago del cliente y le muestra el resultado

```
<?php
  include ('book_sc_fns.php');
  // El carro de la compra necesita sesiones; inicie una
  session_start();

  do_html_header('Checkout');

  $card_type = $HTTP_POST_VARS["card_type"];
  $card_number = $HTTP_POST_VARS['card_number'];
  $card_month = $HTTP_POST_VARS['card_month'];
  $card_year = $HTTP_POST_VARS['card_year"];
  $card_name = $HTTP_POST_VARS['card_name"];

  if($HTTP_SESSION_VARS['cart']&&$card_type&&$card_number&&
      $card_month&&$card_year&&$card_name )
  {
     //muestre el carro, sin permitir cambios y sin imágenes
```

```
      display_cart($HTTP_SESSION_VARS['cart'], false, 0);

      display_shipping(calculate_shipping_cost());

      if(process_card($HTTP_POST_VARS))
      {
         //vacíe el carro de la compra
         session_destroy();
         echo 'Thankyou for shopping with us. Your order has been placed.';
         display_button('index.php', 'continue-shopping', 'Continue
Shopping');
      }
      else
      {
      echo 'Could not process your card. ';
      echo 'Please contact the card issuer or try again.';
         display_button('purchase.php', 'back', 'Back');
      }
   }
   else
   {
      echo 'You did not fill in all the fields, please try again.<hr />';
      display_button('purchase.php', 'back', 'Back');
   }

   do_html_footer();
?>
```

La parte principal de esta secuencia de comandos se encuentra en estas líneas:

```
if(process_card($HTTP_POST_VARS))
{
   //vacíe el carro de la compra
   session_destroy();
   echo 'Thankyou for shopping with us. Your order has been placed.';
   display_button('index.php', 'continue-shopping', 'Continue
Shopping');
}
```

Procesamos la tarjeta del usuario y si todo es correcto, destruimos su sesión.

La función de procesamiento de la tarjeta, como la hemos escrito, simplemente devuelve `true`. Si tuviéramos que implementarlo realmente, necesitaríamos algún tipo de validación (comprobar que la fecha de caducidad es válida así como el número de la tarjeta de crédito) y, tras ello, procesar el pago.

Al diseñar un sitio real, tendrá que tomar una decisión sobre qué mecanismo de compensación de transacciones va a utilizar. Dispone de diferentes opciones:

- Puede contratar un servicio de compensación de transacciones de un proveedor. En función de la zona en la que viva, encontrará gran cantidad de alternativas. Algunas le ofrecen compensación en tiempo real y otras no. Si tiene pensado ofrecer un servicio en línea, le resultará imprescindible; si

solamente tiene pensado enviar productos, será menos importante. De cualquier forma, estos proveedores evitan tener que almacenar números de tarjetas de crédito.

- Puede enviarse a sí mismo un número de tarjeta de crédito a través de correo electrónico codificado, por ejemplo, con ayuda de PGP o GPG, como vimos en un capítulo anterior. Cuando reciba y descodifique el correo, podrá procesar las transacciones manualmente.
- Puede almacenar los números de las tarjetas de crédito en la base de datos. Esta opción no es recomendable a menos que sepa realmente lo que hace con la seguridad del sistema. En capítulos anteriores encontrará más información al respecto.

Esto es todo lo que necesitamos para los módulos del carro de la compra y de pago.

Implementación de una interfaz de administración

La interfaz de administración que hemos implementado es muy sencilla. Simplemente hemos diseñado una interfaz Web a la base de datos con un sistema de autenticación cliente. Es muy similar al código utilizado en uno de los capítulos anteriores. Lo hemos incluido para que la descripción resulte más completa pero no nos detendremos en su análisis.

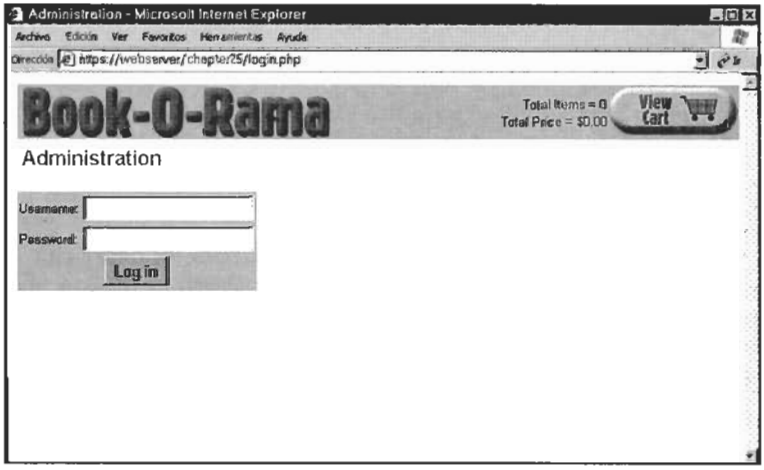


Figura 25.11. El usuario debe pasar por la página de conexión para acceder a las funciones de administración

La interfaz de administración requiere que el usuario se conecte por medio del archivo `login.php`, que le lleva al menú de administración, `admin.php`. Puede ver

la página de conexión en la figura 25.11 (hemos omitido el archivo `login.php` por motivos de brevedad, aunque es prácticamente idéntico al del capítulo anterior; no obstante, se incluye en el CD-ROM). El menú de administración se muestra en la figura 25.12.

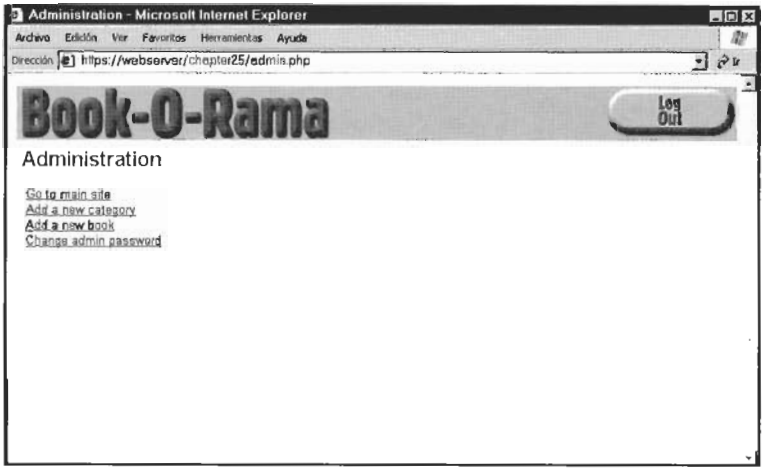


Figura 25.12. El menú de administración permite acceder a las funciones de administración

En el listado 25.17 se recoge el código del menú de administración.

Listado 25.17. admin.php. Esta secuencia de comandos autentica al administrador y le permite acceder a las funciones de administración

```
<?php

// incluya los archivos de inclusión de esta aplicación
require_once('book_sc_fns.php');
session_start();


if ($HTTP_POST_VARS["username"] && $HTTP_POST_VARS["passwd"])
// acaban de intentar conectarse
{

    $username = $HTTP_POST_VARS["username"];
    $passwd = $HTTP_POST_VARS["passwd"];

    if (login($username, $passwd))
    {
      // si están en la base de datos, registre el id. de usuario
      $HTTP_SESSION_VARS["admin_user"] = $username;
    }
    else
    {
      // conexión fallida
```

```
            do_html_header('Problem:');
            echo 'You could not be logged in.
                  You must be logged in to view this page.<br />';
            do_html_url('login.php', 'Login');
            do_html_footer();
            exit;
        }
}

do_html_header('Administration');
if (check_admin_user())
   display_admin_menu();
else
   echo 'You are not authorized to enter the administration area.';

do_html_footer();

?>
```

Probablemente este código le resulte familiar, ya que es parecido a una secuencia de comandos del capítulo anterior. Cuando el administrador llega hasta este punto, puede cambiar su contraseña o desconectarse. Como el código es idéntico al del capítulo anterior, no lo describiremos en este apartado.

Una vez conectado, identificamos al usuario administrador por medio de la variable de sesión `admin_user` y con la función `check_admin_user()`. Esta función, así como todas las utilizadas por las secuencias de comandos administrativas, se encuentra en la biblioteca de funciones `admin_fns.php`.

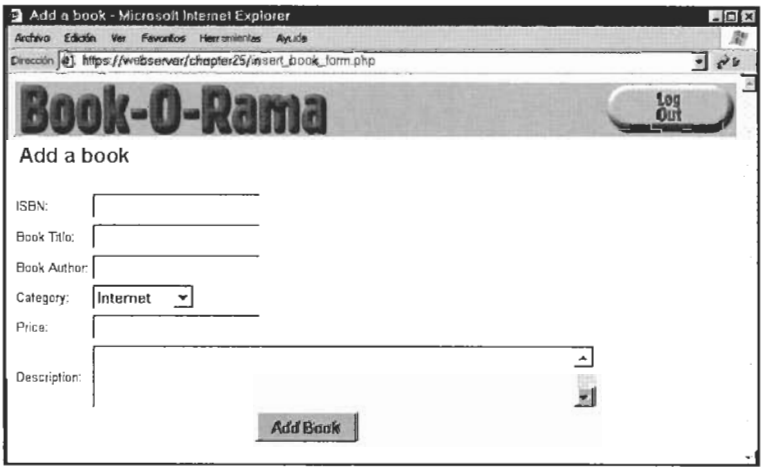


Figura 25.13. Este formulario permite al administrador añadir nuevos libros al catálogo en línea

Si el administrador opta por añadir una nueva categoría o un libro, puede hacerlo desde `insert_category_form.php` o `insert_book_form.php`, respectiva-

mente. Cada una de estas secuencias de comandos presenta al administrador un formulario que debe rellenar. Cada uno se procesa por la correspondiente secuencia de comandos (`insert_category_form.php` e `insert_book_form.php`), que verifica que el formulario se ha rellenado y añade los nuevos datos a la base de datos. Solamente nos detendremos en la secuencia de comandos para libros, ya que ambas son muy similares.

El resultado de `insert_book_form.php` se muestra en la figura 25.13.

Habrá apreciado que el campo **Category** para los libros es un elemento `SELECT` de HTML. Las opciones de `SELECT` provienen de una llamada a la función `get_categories()` que describimos anteriormente.

Al pulsar el botón **Add Book**, se activa la secuencia de comandos `insert_book.php`, cuyo código mostramos en el listado 25.18.

Listado 25.18. insert_book.php. Esta secuencia de comandos valida los datos de los nuevos libros y los añade a la base de datos

```
<?php

// incluya los archivos de inclusión de esta aplicación
require_once('book_sc_fns.php');
session_start();

do_html_header('Adding a book');
if (check_admin_user())
{
   if (filled_out($HTTP_POST_VARS))
   {
      $isbn = $HTTP_POST_VARS['isbn'];
      $title = $HTTP_POST_VARS['title'];
      $author = $HTTP_POST_VARS['author'];
      $catid = $HTTP_POST_VARS['catid'];
      $price = $HTTP_POST_VARS['price'];
      $description = $HTTP_POST_VARS['description'];

      if(insert_book($isbn, $title, $author, $catid, $price, $description))
         echo "Book '".stripslashes($title)."' was added to the database.<br />";
      else
         echo "Book '".stripslashes($title).
               "' could not be added to the database.<br />";
   }
   else
      echo 'You have not filled out the form. Please try again.';
   do_html_url('admin.php', 'Back to administration menu');
}
else
   echo 'You are not authorised to view this page.';

do_html_footer();

?>
```

Comprobará que esta secuencia de comandos invoca la función `insert_book()`. Esta función, así como las funciones restantes utilizadas por las secuencias de comandos administrativas, se almacena en la biblioteca de funciones `admin_fns.php`.

Además de poder añadir nuevas categorías y libros, el usuario administrativo puede modificar y eliminar estos elementos. Hemos implementado estas opciones reutilizando la mayor cantidad de código posible. Cuando el administrador pulsa sobre el enlace Go to Main Site en el menú de administración, accede al índice de categorías de `index.php` y puede navegar por el sitio igual que un usuario convencional, utilizando las mismas secuencias de comandos.

Sin embargo, existe una diferencia con la navegación de administración. Los administradores verán distintas opciones ya que disponen de la variable de sesión registrada `admin_user`. Por ejemplo, si se fija en la página `show_book.php` anterior, verá que hay opciones de menú diferentes. Fíjese en la figura 25.14.

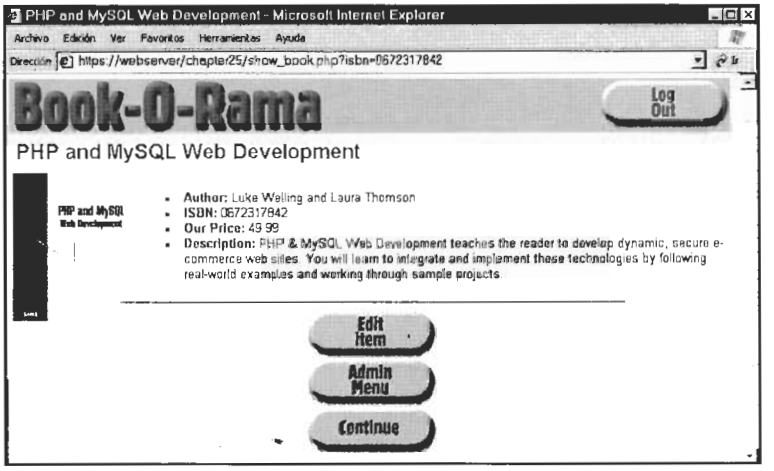


Figura 25.14. La secuencia de comandos show_book.php genera resultados diferentes para los usuarios administrativos.

En esta página, el administrador tiene acceso a dos nuevas opciones: **Edit Item** y **Admin Menu**. También comprobará que en la esquina superior derecha ya no se incluye el carro de la compra sino un botón **Log Out**.

Todo este código se concentra en el siguiente fragmento, tomado del listado 25.8:

```
if( check_admin_user() )
{
  display_button("edit_book_form.php?isbn=$isbn", 'edit-item', 'Edit
Item');
  display_button('admin.php', 'admin-menu', 'Admin Menu');
  display_button($target, 'continue', 'Continue');
}
```

Si se fija de nuevo en la secuencia de comandos `show_cat.php`, comprobará que también cuenta con tres opciones.

Si el administrador pulsa el botón **Edit Item**, accederá a la secuencia de comandos `edit_book_form.php`, cuyo resultado se muestra en la figura 25.15.

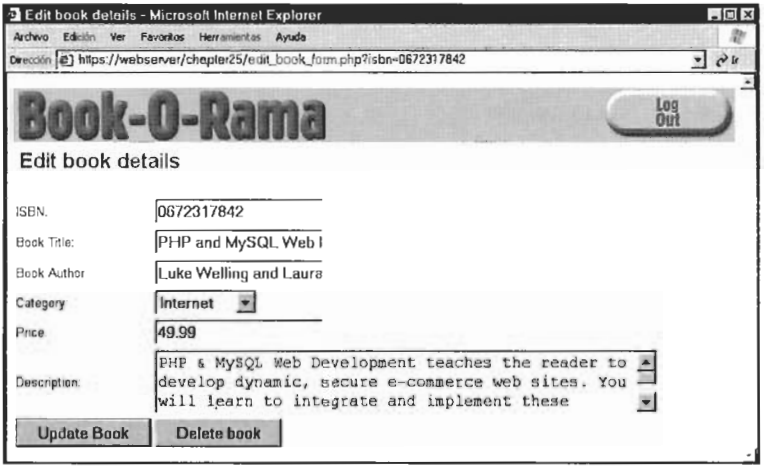


Figura 25.15. La secuencia de comandos edit_book_form.php permite al administrador modificar los detalles de un libro o eliminarlo

Se trata, de hecho, del mismo formulario que utilizamos anteriormente para obtener los detalles de los libros. En dicho formulario hemos incluido una opción para pasar y mostrar los datos de los libros existentes. Hicimos lo mismo con el formulario de categorías. Lo entenderá mejor con ayuda del listado 25.19.

Listado 25.19. Función display_book_form() de admin_fns.php. Este formulario funciona tanto como formulario de inserción como de modificación

```
function display_book_form($book = '')
// Esto muestra el formulario de libros.
// Es muy similar al formulario de categorías.
// Se puede utilizar para añadir o modificar libros.
// Para añadir libros, no pase ningún parámetro. Configurará $edit
// con el valor false, y el formulario irá hasta insert_book.php.
// Para actualizar libros, pase una matriz que contenga el libro.  El
// formulario se mostrará con los datos antiguos y apuntará a
// update_book.php.
// También añade un botón "Delete book".
{

   // si se ha pasado un libro existente, prosiga en "modo edición"
   $edit = is_array($book);

   // la mayor parte del formulario está en HTML simple con algunos
   // fragmentos de PHP opcionales
?>
   <form method="post"
         action="<?php echo
```

```
$edit?'edit_book.php':'insert_book.php';?>">
   <table border="0">
   <tr>
      <td>ISBN:</td>
      <td><input  type="text"  name="isbn"
            value="<?php  echo  $edit?$book['isbn']:'';  ?>"></td>
   </tr>
   <tr>
      <td>Book  Title:</td>
      <td><input  type="text"  name="title"
            value="<?php  echo  $edit?$book['title']:'';  ?>"></td>
   </tr>
   <tr>
      <td>Book  Author:</td>
      <td><input  type="text"  name="author"
            value="<?php  echo  $edit?$book['author']:'';  ?>"></td>
    </tr>
    <tr>
        <td>Category:</td>
        <td><select  name="catid">
        <?php
             // la lista de posibles categorías se obtiene de la base de
             // datos
             $cat_array=get_categories();
             foreach  ($cat_array  as  $thiscat)
             {
                echo '<option value="';
                echo  $thiscat['catid'];
                echo  '"';
                // si es un libro existente, añádalo a la categoría
                //  actual
                if  ($edit  &&  $thiscat['catid']  ==  $book['catid'])
                      echo  ' selected';
                echo  '>';
                echo  $thiscat['catname'];
                echo  "\n";
             }
             ?>
             </select>
          </td>
   </tr>
   <tr>
      <td>Price:</td>
      <td><input type="text" name="price"
            value="<?php echo $edit?$book['price']:''; ?>"></td>
   </tr>
   <tr>
       <td>Description:</td>
       <td><textarea rows="3" cols="50"
            name="description">
            <?php echo $edit?$book['description']:''; ?>
            </textarea></td>
      </tr>
      <tr>
         <td  <?php  if  (!$edit)  echo  'colspan="2"';  ?>  align="center">
              <?php
```

```
            if ($edit)
            // necesitamos el ISBN antiguo para localizar el libro en la
            // base de datos
            // si el ISBN se actualiza
            echo  '<input  type="hidden"  name="oldisbn"
               value="'.$book['isbn'].'">';
         ?>
         <input type="submit"
               value="<?php echo $edit?'Update':'Add'; ?> Book">
         </form></td>
         <?php
            if ($edit)
            {
               echo  '<td>';
               echo '<form method="post"
                  action="delete_book.php">';
               echo '<input type="hidden" name="isbn"
                  value="'.$book['isbn'].'">';
               echo '<input type="submit"
                  value="Delete book">';
               echo '</form></td>';
               }
            ?>
         </td>
      </tr>
   </table>
   </form>
<?php
}
```

Si pasamos una matriz que contenga los datos de los libros, el formulario se representará en modo de edición y completará los campos con los datos existentes:

```
<input type="text" name="price"
     value="<?php  echo  $edit?$book['price']:'';  ?>">
```

Tenemos incluso un botón de envío diferente. De hecho, para el formulario de edición tenemos dos: uno para actualizar el libro y otro para eliminarlo. Con éstos se invocan las secuencias de comandos `edit_book.php` y `delete_book.php`, que actualizan convenientemente la base de datos.

Las versiones de categorías de estas secuencias de comandos funcionan de forma similar a excepción de un aspecto. Cuando un administrador intenta eliminar una categoría, no la podrá eliminar si contiene algún libro (lo que se puede comprobar por medio de una consulta a la base de datos). De esta forma se evitan todos los problemas que podamos tener con anomalías de eliminación, como mencionamos en uno de los primeros capítulos.

En este caso, si se elimina una categoría que incluya libros, éstos quedarían huérfanos. No podremos saber en qué categoría se encuentran y no podremos desplazarnos hasta ellos.

Con esto hemos terminado el repaso a la interfaz de usuario. Si necesita más información, puede consultar el código incluido en el CD-ROM.

Ampliación del proyecto

Hemos construido un sistema de carro de la compra bastante sencillo. Sin embargo, podemos realizar numerosas mejoras al mismo:

- En una tienda en línea real, necesitaríamos algún tipo de sistema seguimiento y recepción de pedidos. Por el momento, no hay forma de saber qué pedidos se han realizado.
- Los clientes deben poder comprobar el progreso de sus pedidos sin tener que ponerse en contacto con nosotros. Es importante que el cliente no tenga que conectarse para navegar. Sin embargo, si proporcionamos a los clientes existentes una forma de autenticarse, podrán ver pedidos realizados previamente y nos permitirá combinar comportamientos para crear un perfil.
- Por ahora, es necesario enviar por FTP las imágenes de los libros al directorio de imágenes y asignarles el nombre correcto. Podríamos cargar archivos a la página de inserción de libros para que resultara más sencillo.
- Podríamos añadir inicio de sesión de usuarios, personalización y recomendaciones de libros, críticas en línea, programas afiliados, comprobación de las existencias, etc. Las posibilidades son infinitas.

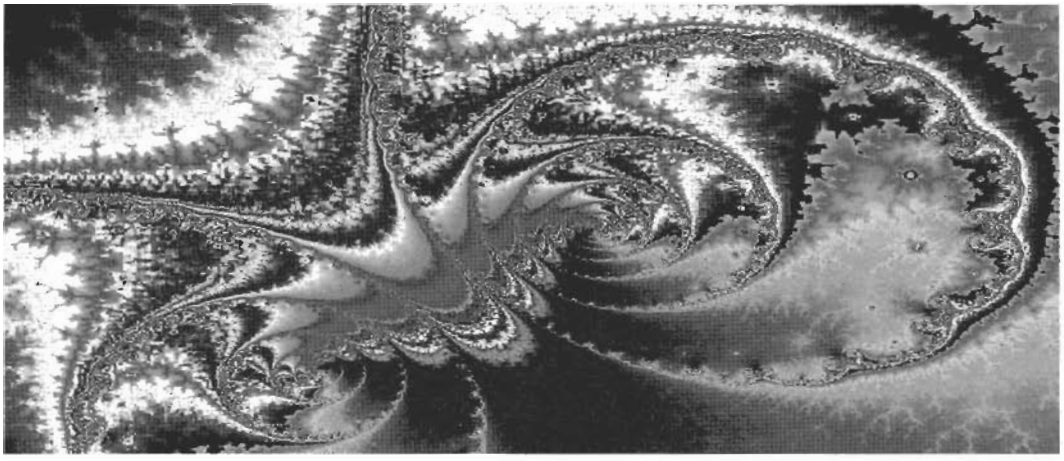
Uso de un sistema existente

Si quiere un carro de la compra totalmente equipado y operativo, puede optar por utilizar un sistema de carro de la compra existente. Un carro de código abierto muy conocido que se implementa en PHP es FishCartSQL, que puede encontrar en `http://www.fishcart.org/`.

Dispone de un gran número de opciones avanzadas como el seguimiento de clientes, ventas cronometradas, varios idiomas, procesamiento de tarjetas de crédito y compatibilidad con varias tiendas en línea en un servidor. Evidentemente, al utilizar un sistema ya existente, encontrará que no dispone de algunos elementos que necesita y viceversa. La ventaja de un producto de código abierto es que siempre se puedan cambiar los aspectos que no nos convenzan.

A continuación

En el siguiente capítulo aprenderemos a generar un sistema de gestión de contenidos en línea, muy indicado para administrar activos digitales, lo que puede resultar de gran utilidad si contamos con un sitio basado en contenidos.



26
Creación de un sistema de administración de contenidos

En este capítulo crearemos un sistema de administración de contenidos que nos permitirá almacenar, indexar y buscar textos y contenidos multimedia.

Los sistemas de administración de contenidos resultan extremadamente útiles en sitios Web en los que el contenido del mismo lo mantiene más de un autor, en los que el mantenimiento lo llevan a cabo empleados que no son técnicos o en los que el contenido y el diseño gráfico es competencia de distintos departamentos o diferentes personas.

Generaremos una aplicación que permita a los usuarios autorizados gestionar los activos digitales de una organización. Nos centraremos en los siguientes aspectos:

- Presentación de páginas Web por medio de diferentes plantillas
- Creación de un motor de búsqueda que indexe documentos en función de metadatos

El problema

Imagine que el equipo de desarrollo Web de SuperFastOnlineNews está formado por un excelente diseñador gráfico y por un determinado número de galardonados escritores. El sitio cuenta con páginas de noticias, de deportes y de información meteorológica que se actualizan regularmente. La página principal muestra los últimos titulares de las tres páginas de categorías anteriores.

En SuperFastOnlineNews, los diseñadores garantizan el aspecto perfecto de los contenidos del sitio Web. Es lo que mejor saben hacer. Por su parte, los escritores se encargan de redactar los mejores artículos pero no saben dibujar bien ni generar sitios Web. Tendremos que permitir que cada uno se centre en lo que mejor sabe hacer y que todos compartan sus resultados para de esta forma ofrecer el servicio superrápido de noticias que anuncia el sitio.

Requisitos de la solución

Nuestro objetivo será diseñar un sistema que:

- Aumente la productividad siempre que los escritores se centren en escribir y los diseñadores en diseñar
- Permita al editor revisar los artículos y decidir cuáles se van a publicar
- Cuente con un aspecto visual y operativo coherente en todo el sitio, obtenido por medio de plantillas de páginas
- Permita que los escritores puedan acceder únicamente a las zonas del sitio asignadas para ellos
- Permita que se pueda cambiar el aspecto visual y operativo de una determinada sección o de todo el sitio
- Evite que se cambie el contenido en directo

Modificación del contenido

En primer lugar, tendremos que pensar en cómo añadiremos el contenido al sistema y en cómo lo almacenaremos y modificaremos.

Introducción de contenido en el sistema

Tendremos que decidir qué artículos y componentes de diseño se enviarán. Se pueden utilizar tres métodos.

FTP

Los escritores y diseñadores pueden tener acceso a través de FTP a zonas del servidor Web en las que puedan cargar archivos desde su equipo local. Necesitaremos un estándar de notación fija para los archivos cargados (para identificar qué imágenes corresponden a cada artículo) o un sistema basado en la Web para hacerlo de forma separada a la carga desde FTP.

En este caso, el uso de FTP también supone problemas con los permisos. Debido a la flexibilidad que requiere este ejemplo, no utilizaremos FTP para permitir a los usuarios la carga de archivos.

Método de carga de archivos

Como mencionamos en un capítulo anterior, el protocolo HTTP cuenta con un método para cargar archivos por medio del navegador Web. PHP lo puede procesar de forma muy sencilla.

El método de carga de archivos también nos permite almacenar texto en una base de datos en lugar de hacerlo en forma de archivo. Para ello, podríamos leer en el archivo temporal y almacenar sus contenidos en la base de datos, en lugar de copiarlos en otro punto del sistema de archivos. En este proyecto no utilizaremos la carga de archivos con los artículos.

Más adelante describiremos la ventaja de la base de datos con respecto al sistema de archivos.

Cambios en línea

Podemos permitir que los usuarios creen y modifiquen documentos sin necesidad de utilizar FTP o la carga de archivos.

Por el contrario, podemos ofrecer a los colaboradores un cuadro de entrada de texto en pantalla en el que puedan modificar el contenido de sus artículos.

Este método es muy sencillo, pero eficaz. El navegador Web no cuenta con ninguna función de edición de texto aparte de las opciones de cortar y pegar del sistema operativo.

Sin embargo, cuando es necesario realizar algún cambio, por ejemplo, corregir un error ortográfico, la operación de seleccionar el contenido y corregirlo es muy rápida.

Al igual que la carga de archivos, los datos del formulario se pueden escribir en un archivo o se pueden almacenar en una base de datos.

Bases de datos frente a almacenamiento en archivos

Una de las decisiones más importantes que debemos tomar en las fases iniciales es determinar cómo se almacenará el contenido una vez cargado al sistema.

Como almacenaremos metadatos junto al texto del artículo, hemos optado por añadir las partes de texto del contenido a la base de datos. Aunque MySQL puede almacenar datos multimedia, optamos por almacenar imágenes cargadas en el sistema de archivos. Como mencionamos en capítulos anteriores, el uso de datos BLOB en una base datos MySQL puede reducir el rendimiento.

Simplemente almacenaremos el nombre de archivo de la imagen en la base de datos. Al utilizar el sistema de archivos, la etiqueta `<IMG SRC>` puede hacer referencia directamente al archivo de imagen, como de costumbre.

Si almacenamos una gran cantidad de datos, conviene optimizar el almacén de datos. Al igual que una base de datos requiere los índices adecuados para ser eficaz,

el sistema de archivos se beneficia considerablemente si se utiliza una estructura de directorios correcta.

En la figura 26.1 puede ver un ejemplo.

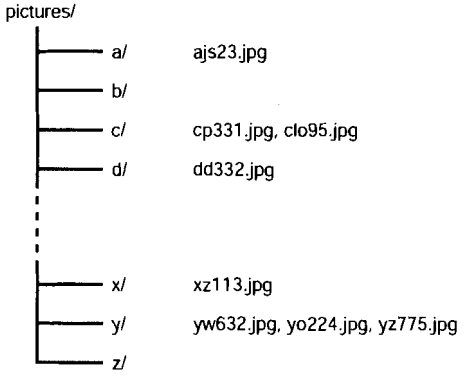


Figura 26.1. Esta estructura de directorios se ha estructurado para cargas de archivos

En este caso, el sistema de archivos se divide en directorios que representan el primer carácter de cada nombre de archivo. De esta forma, se dividen 10.000 archivos en 26 directorios, cada uno con aproximadamente 400 archivos. El acceso será mucho más rápido que si tuviéramos los 10.000 archivos en un solo directorio. Conviene destacar que el nombre de archivo utilizado se genera desde la secuencia de comandos de procesamiento de carga para garantizar su exclusividad.

En el ejemplo de la figura 26.1 se asume que todos los nombres de archivo empiezan por una letra minúscula.

Estructura de documentos

Los artículos de ejemplo que utilizaremos son breves, de uno o dos párrafos con una sola imagen, diseñados para usuarios con prisa. Se trata de documentos estructurados ya que contienen un titular y uno o dos párrafos de texto con una imagen.

Cuanto mayor sea la estructura del documento, con mayor facilidad se podrá dividir para almacenarlo en una base de datos. La ventaja es que todos los documentos se pueden presentar de forma estructurada y coherente.

Veamos nuestro ejemplo de artículos de noticias. Almacenaremos el titular en un campo aparte del texto del artículo y, por naturaleza, la imagen es un componente independiente del documento.

Con el titular como elemento independiente, podemos definir un tipo y un estilo estándar con el que lo representaremos y podremos separarlo del resto del artículo para crear nuestra página de titulares principal.

Otro enfoque que podríamos utilizar en documentos de gran tamaño consiste en aplicar una relación uno a varios entre los distintos párrafos, es decir, almacenar

cada párrafo en una fila de la base de datos y vincular cada uno a un Id. del documento principal. Este tipo de estructura dinámica de documentos nos permitirá presentar una página de contenidos para cada documento y mostrar cada sección independientemente, o mostrar todo el documento de una vez.

Uso de metadatos

Ya hemos decidido que cada registro de artículo incluirá un titular, el texto del artículo y una imagen. Sin embargo, podemos almacenar otros datos en el mismo registro.

De forma automática, nuestro sistema añadirá valores para el que haya creado el artículo y para la última fecha en la que se haya modificado. Estos valores se pueden representar en la parte inferior del artículo para añadir una fecha y una marca de tiempo sin que el autor tenga que preocuparse de incluir esta información.

También puede resultar útil añadir datos que no se muestren, lo que se denomina *metadatos*. Un buen ejemplo sería almacenar palabras clave que se podrían utilizar en el índice del motor de búsqueda.

En lugar de tener que buscar por todo el texto de todos los artículos, el motor de búsqueda se centrará en los metadatos de palabra clave de cada artículo para determinar la importancia a partir de dichos datos. Esto permite que el administrador del sitio disponga de control total sobre qué palabras y frases de búsqueda coinciden con qué documentos.

En nuestro ejemplo, permitiremos que se pueda asociar cualquier número de palabras clave a un artículo y asignaremos a cada palabra clave un valor que indique la relevancia de la misma en una escala del 1 al 10.

De esta forma, podemos diseñar un algoritmo de motor de búsqueda que clasifique las coincidencias en los artículos en función de esta relevancia, en lugar de un complejo algoritmo que tenga que interpretar el idioma y tomar decisiones en función de sus limitados conocimientos y las reglas fijas que lo controlan.

Esto no significa que tengamos que almacenar los metadatos en la base de datos. Nada le impide utilizar la etiqueta `<META>` en HTML o incluso recurrir a XML para generar los documentos. Sin embargo, siempre que sea posible, merece la pena aprovechar las ventajas de la base de datos en la que ya se encuentran los documentos.

Formato del resultado

A la hora de representar una página, nuestro sitio de noticias utiliza un sencillo pero estructurado formato. Cada página contiene un número de artículos, todos con el mismo formato. En primer lugar, se muestra el titular con letras grandes, seguido por la fotografía, situada más abajo y a la izquierda y, por último, el texto del artículo a la derecha. Toda la página se incluye en una plantilla de página estándar para garantizar el estilo y la coherencia de todo el sitio.

En la figura 26.2 podemos ver la estructura lógica de páginas que vamos a utilizar.

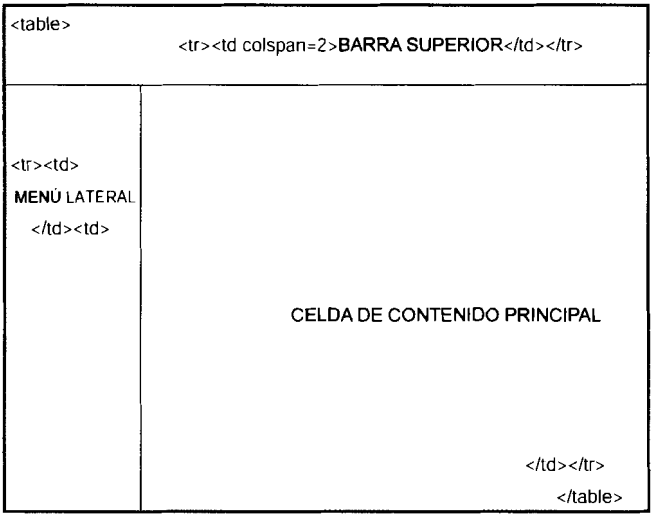


Figura 26.2. Estructura lógica de páginas

Este tipo de diseño es muy habitual. Seguramente, en las páginas que visita a diario, habrá comprobado que cuentan con una barra de menú a la izquierda o con enlaces rápidos en la parte superior y el resto de la navegación es estática independientemente de la página en la que se encuentre.

La implementación de una estructura de plantilla como ésta a partir de un diseño de página es muy sencilla. En el código HTML, encontrará la etiqueta `<TD>` en la que comienza la celda de contenido principal. Desde aquí, dividimos el HTML en dos archivos: la primera parte se convierte en el archivo de encabezado y la segunda, en el del pie de página. Siempre que se represente una página, primero mostraremos el encabezado, luego la página y por último el pie.

La implementación de un sitio con una plantilla de encabezado y de pie de página permite que estos archivos de plantilla se puedan cambiar con facilidad si se actualiza el diseño del sitio.

PHP cuenta con dos opciones de configuración que, en este caso, resultan de gran utilidad. Las directivas `auto_prepend_file` y `auto_append_file` se pueden configurar en cada directorio para especificar qué archivos se incluyen antes o después de procesar cualquier secuencia de comandos.

No obstante, tiene sus limitaciones. Si tiene alguna secuencia de comandos que no genere resultados y que envíe un encabezado de redireccionamiento, como por ejemplo

```
<?  header('Location:  destination.php');  ?>
```

los archivos de encabezado y pie se mostrarán y el redireccionamiento fallará ya que el resultado ya se ha enviado al navegador Web. Esto también resulta problemá-

tico con las cookies y las funciones de administración de sesión que incorpora PHP 4, ya que un encabezado de cookie no funciona correctamente después de enviar cualquier resultado al navegador Web.

Manipulación de imágenes

Los escritores que colaboran con sus artículos probablemente incluirán sus propias imágenes que los complementen. Nuestro objetivo es la coherencia pero, ¿qué sucede si un escritor carga una imagen de gran tamaño y calidad y otro una miniatura?

Suponiendo que las imágenes en cuestión sean fotografías, podemos insistir en que utilicen únicamente imágenes JPEG y aprovecharnos de las ventajas de las funciones de manipulación de imágenes de PHP. En un capítulo anterior se describe este aspecto con mayor detalle.

Hemos creado una sencilla secuencia de comandos con el nombre `resize_image.php` que cambia el tamaño de la imagen sobre la marcha para que se pueda mostrar con una etiqueta `<img src>`. En el listado 26.1 se incluye esta secuencia de comandos.

Listado 26.1. resize_image.php. Cambia el tamaño de una imagen JPEG sobre la marcha

```
<?php

$image  =  $HTTP_GET_VARS['image'];

if (!$max_width)
   $max_width = 80;
if (!$max_height)
   $max_height = 60;

$size = GetImageSize($image);
$width = $size[0];
$height = $size[1];

$x_ratio = $max_width / $width;
$y_ratio = $max_height / $height;

if ( ($width <= $max_width) && ($height <= $max_height) ) {
   $tn_width = $width;
   $tn_height = $height;
}
else if (($x_ratio * $height) < $max_height) {
   $tn_height = ceil($x_ratio * $height);
   $tn_width = $max_width;
}
else {
   $tn_width = ceil($y_ratio * $width);
   $tn_height  =  $max_height;
```

```
}

$src  =  ImageCreateFromJpeg($image);
$dst  =  ImageCreate($tn_width,$tn_height);
ImageCopyResized($dst, $src, 0, 0, 0, 0,
      $tn_width,$tn_height,$width,$height);
header('Content-type: image/jpeg');
ImageJpeg($dst, null, -1);
ImageDestroy($src);
ImageDestroy($dst);

?>
```

La secuencia de comandos adopta tres parámetros: el nombre de archivo de la imagen que se va a representar, la anchura máxima y la altura máxima en píxeles. Esto no quiere decir que si el tamaño máximo especificado es de 200x200, la imagen se escalará a 200x200. Por el contrario, se reducirá proporcionalmente la escala para que la parte de mayor tamaño, anchura o altura, sea de 200 píxeles y la otra dimensión sea de 200 píxeles o inferior. Por ejemplo, una imagen de 400x300 se reducirá a 200x150. De esta forma se conserva la proporción de la imagen.

El cambio de tamaño sobre la marcha en el servidor es una opción mucho más indicada que la especificación de los atributos `HEIGHT` y `WIDTH` en la etiqueta `<img>`. Puede que una imagen a gran tamaño y de alta resolución enviada por un escritor ocupe varios megabytes pero, al reducir la escala, se quede en menos de 100k. Por ello, no es necesario descargar el archivo de gran tamaño y pedir al navegador que cambie su tamaño.

En un capítulo anterior se describen detalladamente las funciones de manipulación de imágenes. En este caso utilizamos la función `ImageCopyResized()` para obtener, sobre la marcha, una escala de la imagen del tamaño correcto.

La clave de la operación de cambio de tamaño consiste en calcular los tamaños de altura y anchura. Es necesario identificar la proporción entre las dimensiones reales y las máximas. `$max_width` y `$max_height` se pueden pasar a la cadena de consulta; en caso contrario, se utilizarán los valores predeterminados especificados en la parte superior del listado.

```
$x_ratio = $max_width / $width;
$y_ratio = $max_height / $height;
```

Si la imagen ya es inferior a los tamaños máximos especificados, no se modifica ni el ancho ni la altura. En caso contrario, se puede utilizar tanto el índice X como el índice Y para escalar ambas dimensiones para que la imagen reducida no parezca desproporcionada, como indicamos a continuación:

```
if ( ($width <= $max_width) && ($height <= $max_height) ) {
   $tn_width = $width;
   $tn_height = $height;
}
else if (($x_ratio * $height) < $max_height) {
   $tn_height = ceil($x_ratio * $height);
   $tn_width  =  $max_width;
```

```
}
else {
    $tn_width = ceil($y_ratio * $width);
    $tn_height = $max_height;
}
```

Diseño y presentación de la solución

En la tabla 26.1 se recoge un resumen de los archivos de esta aplicación.

Tabla 26.1. Archivos de la aplicación de administración de contenido

Nombre	Tipo	Descripción
`create_database.sql`	SQL	SQL para definir la base de datos de contenidos y datos de ejemplo
`include_fns.php`	Funciones	Colección de archivos de inclusión de esta aplicación
`db_fns.php`	Funciones	Colección de funciones para conectarse a la base de datos de contenidos
`select_fns.php`	Funciones	Colección de funciones para ayudar a crear listas `SELECT`
`user_auth_fns.php`	Funciones	Colección de funciones para autenticar usuarios
`header.php`	Plantilla	Se muestra en la parte superior de todas las páginas de contenidos
`footer.php`	Plantilla	Se muestra en la parte inferior de todas las páginas de contenidos
`logo.gif`	Imagen	Archivo de logotipo mostrado en `header.php`
`headlines.php`	Aplicación	Muestra el titular más reciente de todas las páginas del sitio
`page.php`	Aplicación	Enumera los titulares y el texto de los artículos de una determinada página
`resize_image.php`	Aplicación	Cambia el tamaño de una imagen sobre la marcha para `headlines.php`
`search_form.php`	Aplicación	Formulario para introducir palabras clave para buscar por los contenidos del sitio

Nombre	Tipo	Descripción
search.php	Aplicación	Muestra titulares de contenido que coincidan con los criterios de las palabras clave
login.php	Aplicación	Autentica la contraseña de un usuario y lo conecta al sistema
logout.php	Aplicación	Desconecta al usuario del sistema
stories.php	Aplicación	Enumera los artículos que ha escrito el usuario conectado con una opción para añadir, modificar o eliminar artículos
story.php	Aplicación	Pantalla de detalles de artículos para modificar o añadir un nuevo artículo
story_submit.php	Aplicación	Añade nuevos artículos o aplica los cambios a partir de los datos introducidos en story.php
delete_story.php	Aplicación	Procesa la solicitud de eliminación de un artículo desde stories.php
keywords.php	Aplicación	Enumera las palabras clave de un artículo con una opción para añadir o eliminar palabras clave
keyword_add.php	Aplicación	Procesa la solicitud de añadir palabras clave de keywords.php
keyword_delete.php	Aplicación	Procesa la solicitud de eliminar palabras clave de keywords.php
publish.php	Aplicación	Lista de artículos del editor en el que se muestran los publicados con una opción para cambiar el estado de todos ellos
publish_story.php	Aplicación	Procesa una solicitud de publicación de publish.php
unpublish_story.php	Aplicación	Procesa una solicitud de no publicación de publish.php

Diseño de la base de datos

En el listado 26.2 se incluyen las consultas SQL para crear la base de datos a partir del sistema de contenidos. Este listado forma parte del archivo create_database.

`sql`. El archivo incluido en el CD también contiene consultas para añadir usuarios y artículos de ejemplo a la base de datos.

Listado 26.2. Fragmento de create_database.sql. Archivo SQL para definir la base de datos de contenidos

```
drop database if exists content;

create  database  content;

use  content;

drop table if exists writers;

create table writers (
   username     varchar(16) primary key,
   password     varchar(16) not null,
   full_name    text
);

drop table if exists stories;

create table stories (
   id           int primary key auto_increment,
   writer       varchar(16) not null,          # foreign key writers.username
   page         varchar(16) not null,          # foreign key pages.code
   headline     text,
   story_text   text,
   picture      text,
   created      int,
   modified     int,
   published    int
);

drop table if exists pages;

create table pages (
   code         varchar(16) primary key,
   description text
);

drop table if exists writer_permissions;

create table writer_permissions (
   writer       varchar(16) not null,          # foreign key writers.username
   page         varchar(16) not null           # foreign key pages.code
);

drop table if exists keywords;

create table keywords (
   story       int not null,                   # foreign key stories.id
   keyword     varchar(32) not null,
   weight      int not null
```

```
);

grant select, insert, update, delete
on content.*
to content@localhost identified by 'password';
```

En la tabla `writers` tendremos que almacenar cierta información sobre cada uno de los escritores, incluyendo un nombre de inicio de sesión y una contraseña. Almacenaremos sus nombres completos para añadirlos en cada artículo y para saludarlos cuando se conecten.

La tabla `pages` contiene el encabezado de cada página en el que se muestran los artículos. La tabla `writer_permissions` implementa una relación varios a varios que indica a qué páginas puede un escritor enviar artículos.

La tabla `stories` contiene campos independientes para `headline`, `story_text` y `picture`, como indicamos anteriormente. Los campos `created`, `modified` y `published` son campos enteros que almacenarán el valor de la marca de tiempo de Unix cuando sea necesario. Para crear la base de datos, ejecute el siguiente comando:

```
mysql -u root < create_database.sql
```

Implementación

Una vez que tenemos la base de datos y una función para cambiar el tamaño de las imágenes, pasaremos a crear la parte principal del sistema.

Interfaz de usuario

En primer lugar nos detendremos en `headlines.php` (listado 26.3) que será la primera página que vea cualquier visitante del sitio. Queremos que vea los titulares del último artículo de cada página.

Listado 26.3. headlines.php. Muestra el último titular de cada página

```
<?php

include('include_fns.php');
include('header.php');

$conn = db_connect();


$pages_sql = 'select * from pages order by code';
$pages_result = mysql_query($pages_sql, $conn);

while ($pages = mysql_fetch_array($pages_result)) {

   $story_sql = "select * from stories
        where page = '".$pages['code']."'
```

```
        and published is not null
        order by published desc";
    $story_result = mysql_query($story_sql, $conn);
    if (mysql_num_rows($story_result)) {
       $story = mysql_fetch_array($story_result);
       print '<table border="0" width="400">';
       print '<tr>';
       print '<td rowspan="2" width="100">';
       if ($story[picture])
          print "<img src=\"resize_image.php?image=$story[picture]\" />";
       print '</td>';
       print '<td>';
       print '<h3>'.$pages['description'].'</h3>';
       print $story['headline'];
       print '</td>';
       print '</tr>';
       print '<tr><td align="right">';
       print '<a href="page.php?page='.$pages['code'].'">';
       print '<font size="1">Read more '.$pages['code'].' ...</font>';
       print '</a>';
       print '</table>';
    }
}
```



Figura 26.3. Mostramos los titulares de cada página del sitio

Esta secuencia de comandos, al igual que el resto de secuencias públicas, incluye `header.php` en la parte inicial y `footer.php` al final. Cualquier resultado generado por esta secuencia de comandos se muestra dentro de la celda de contenido principal de la página. La parte más compleja la realizan dos consultas de base de datos. En primer lugar,

```
select * from pages order by code
```

recuperamos la lista de páginas de la base de datos. Seguidamente, se ejecutan los contenidos del bucle

```
"select * from stories
where page = '".$pages['code']."'
and published is not null
order by published desc"
```

para localizar los artículos de dicha página en orden inverso a la fecha de publicación.

En la figura 26.3 podemos ver el resultado de `headline.php` tras utilizar los datos de aplicación de ejemplo. Junto a cada titular, se genera un enlace de la siguiente forma:

```
<a href="page.php?page=1"><font size="1">Read more news...</font></a>
```

Esto se realiza con el bucle anterior de forma que el valor de la cadena de consulta `page` y el nombre de la página se impriman junto al correspondiente titular. Al pulsar sobre este enlace, el visitante accede a `page.php`, la lista completa de artículos de una determinada página. El código de `page.php` se incluye en el listado 26.4.

Listado 26.4. page.php. Muestra todos los artículos publicados en una página

```
<?php

if (!isset($HTTP_GET_VARS['page']))
{
   header('Location: headlines.php');
   exit;
}

$page = $HTTP_GET_VARS['page'];

include('include_fns.php');
include('header.php');

$conn = db_connect();


$sql = "select * from stories
        where page = '$page'
        and published is not null
        order by published desc";
$result = mysql_query($sql, $conn);

while ($story = mysql_fetch_array($result))
{
   print '<h2>'.$story['headline'].'</h2>';
   if ($story['picture']) {
      $size = getImageSize($story['picture']);
      $width  = $size[0];
      $height = $size[1];
      print "<img src=\"".$story['picture']."\" height=\"$height\"
width=\"$width\" align=\"left\" />";

   }
```

```
    print $story['story_text'];
    $w = get_writer_record($story['writer']);
    print '<br /><font size="1">';
    print $w['full_name'].', ';
    print date('M d, H:i', $story['modified']);
    print '</font>';
}


include('footer.php');
?>
```

Fíjese en que `page.php` requiere un valor para `page` para formular correctamente la primera consulta. En caso de que se invoque `page.php` directamente sin la cadena de consulta, la primera condición

```
if (!isset($HTTP_GET_VARS['page']))
{
    header('Location: headlines.php');
    exit;
}
```

enviará al visitante de nuevo a la página de titulares para que la omisión de `page` no genere un error.

La primera consulta

```
select * from stories
where page = '$page'
and published is not null
order by published desc
```

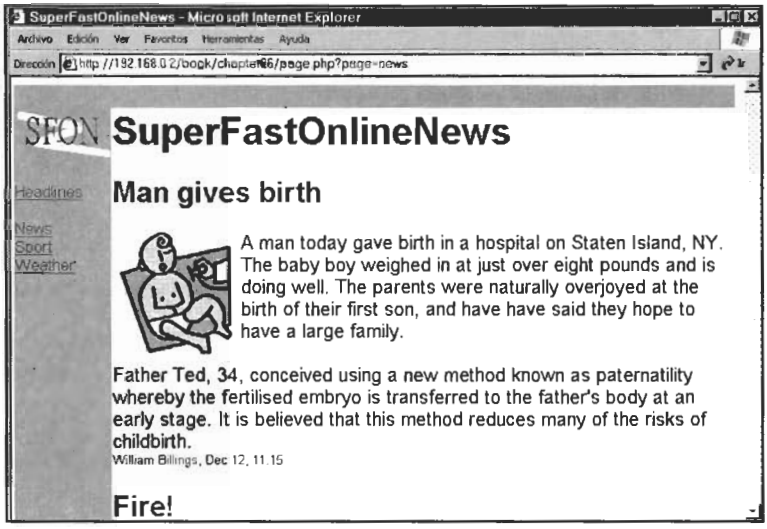


Figura 26.4. En la página de noticias se muestran todos los artículos publicados

busca todos los artículos publicados en la página especificada, con el último artículo publicado en primer lugar.

Dentro de cada bucle, se imprime la imagen cargada y el texto del artículo junto a la pantalla, seguido por el nombre del escritor y la fecha de la última modificación.

En la figura 26.4 podemos ver `page.php` en funcionamiento. Muestra todos los elementos de la página de noticias de la aplicación de ejemplo.

Sistema

A continuación veremos cómo se pueden añadir artículos al sistema. Los escritores empiezan en `stories.php`. Esta secuencia de comandos, una vez autenticado el escritor, muestra una lista de los artículos que haya escrito y también la fecha de publicación, opciones para añadir un nuevo artículo, para modificar o eliminar un artículo existente o para definir las palabras clave de búsqueda. En la figura 26.5 se incluye un ejemplo.

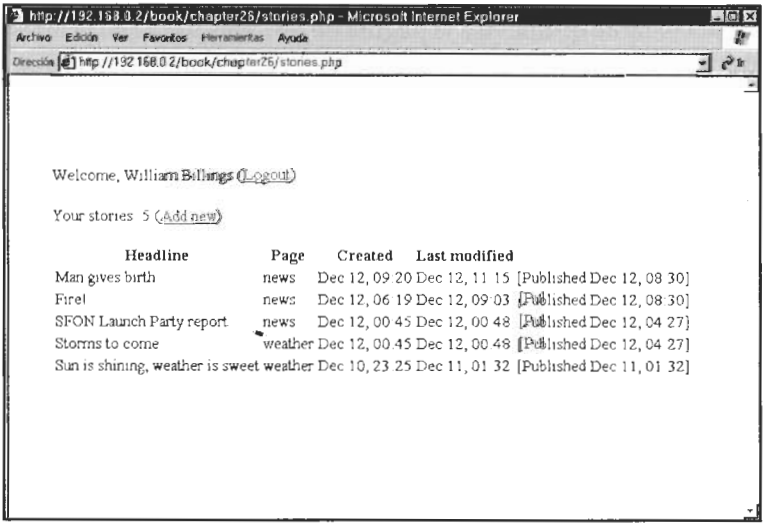


Figura 26.5. Página de administración de artículos para los escritores

A estas pantallas no se les aplica formato dentro de los archivos de encabezado y pie, pero se puede hacer si lo desea. Como solamente los escritores y el editor pueden utilizar estas secuencias de comandos, en nuestro sistema hemos optado por aplicar únicamente el formato necesario para crear un sistema que se pueda utilizar. El código de `stories.php` se incluye en el listado 26.5.

Listado 26.5. stories.php. Interfaz de administración de artículos para los escritores

```
<?php
```

```
include('include_fns.php');
session_start();

if  (!check_auth_user())  {
?>
 <form action="login.php" method="post">
 <table border="0">
  <tr>
    <td>Username</td>
    <td><input size="16" name="username"></td>
 </tr>
 <tr>
    <td>Password</td>
    <td><input size="16" type="password" name="password"></td>
 </tr>
 </table>
 <input type="submit" value="Log in'>
 </form>
  <?php
}
else {
   $conn = db_connect();

   $w = get_writer_record($HTTP_SESSION_VARS['auth_user']);

   print 'Welcome, '.$w['full_name'];
   print ' (<a href="logout.php">Logout</a>)';
   print '<p>';

   $sql  =  'select  *  from  stories  where  writer  =  \''.
        '$HTTP_SESSION_VARS['auth_user'].'\''.order by created desc';
   $result = mysql_query($sql, $conn);

   print 'Your stories: ';
   print mysql_num_rows($result);
   print ' (<a href="story.php">Add new</a>)';
   print '</p><br /><br />';

   if (mysql_num_rows($result)) {
      print '<table>';
      print '<tr><th>Headline</th><th>Page</th>';
      print '<th>Created</th><th>Last modified</th></tr>';
      while ($qry = mysql_fetch_array($result)) {
         print '<tr>';
         print '<td>';
         print $qry['headline'];
         print '</td>';
         print '<td>';
         print $qry['page'];
         print '</td>';
         print '<td>';
         print date('M d, H:i', $qry['created']);
         print '</td>';
         print '<td>';
         print date('M d, H:i', $qry['modified']);
         print '</td>';
```

```
            print '<td>';
            if  ($qry['published'])
               print '[Published '.date('M d, H:i', $qry['published']).']';
            else {
               print '[<a href="story.php?story='.$qry['id'].'">edit</a>] ';
               print '[<a href="delete_story.php?story='.$qry['id'].'">delete</a>] ';
               print '[<a href="keywords.php?story='.$qry['id'].'">keywords</a>]';
            }
            print '</td>';
            print '</tr>';
         }
         print '</table>';
      }

}
?>
```

El primer paso consiste en comprobar si el usuario se ha autenticado y, en caso contrario, mostrar únicamente un formulario de inicio de sesión.

La variable `auth_user` se configura una vez conectado el escritor. En este caso la autenticación no es realmente segura y, de hecho, debería procurar que la autenticación de los escritores fuera la correcta. En un capítulo anterior encontrará más información al respecto.

El formulario de inicio de sesión se envía a `login.php` que comprueba el nombre de usuario y la contraseña en función de los valores de la base de datos. Si el inicio de sesión es correcto, el usuario vuelve a la página de la que proviene, por medio del valor `HTPP_REFERER`. Esto significa que la secuencia de comandos de inicio de sesión se puede invocar desde cualquier página de invocación del sistema.

Tras ello, saludamos al escritor por su nombre y le ofrecemos la oportunidad de desconectarse. Este enlace siempre aparece en la parte superior de `stories.php` para que pueda desconectarse cuando haya terminado.

```
$w = get_writer_record($HTTP_SESSION_VARS['auth_user']);

print 'Welcome, '.$w['full_name'];
print ' (<a href="logout.php">Logout</a>)';
```

La función `get_writer_record()` se define en `db_fns.php` y devuelve una matriz de todos los campos de la tabla `writer` del nombre de usuario pasado.

La secuencia de comandos `logout.php` simplemente anula el valor de `auth_user`.

El siguiente SQL busca todos los artículos del escritor, empezando con el último que se haya añadido:

```
$sql = 'select * from stories where writer = \''
      '.$HTTP_SESSION_VARS['auth_user'].'\'' .order by created desc';
```

Estamos almacenando una marca de tiempo creada, modificada y publicada en cada registro de artículo.

Al añadir un nuevo artículo, las marcas de tiempo creada y modificada se configuran con la hora del sistema.

Todos los cambios posteriores únicamente actualizan el campo modificado.

Toda esta información se muestra en la pantalla de artículos, primero con

```
print date('M d, H:i', $qry['created']);
```

seguidamente con

```
print date('M d, H:i', $qry['modified']);
```

y por último con

```
if ($qry['published'])
   print '(Published '.date('M d, H:i', $qry['published']).')';
else {
   print '[<a href="story.php?story='.$qry['id'].'">edit</a>] ';
   print '[<a href="delete_story.php?story='.$qry['id'].'">delete</a>] ';
   print '[<a href="keywords.php?story='.$qry['id'].'">keywords</a>]';
}
```

De esta forma se muestra la fecha de publicación si corresponde; en caso contrario, muestra enlaces para modificar o eliminar dicho artículo y para definir palabras clave de búsqueda.

La secuencia de comandos para añadir un nuevo artículo o para modificar uno ya existente es `story.php`.

En la figura 26.6 puede comprobar cómo se modifica uno de los artículos de la base de datos de ejemplo.

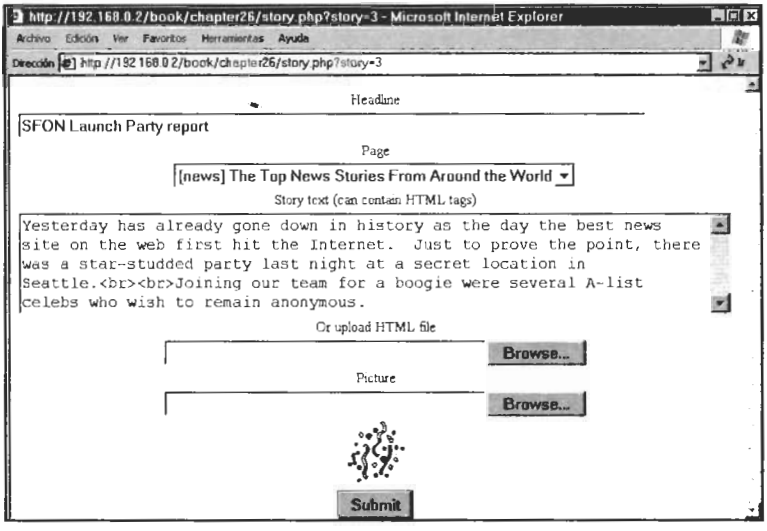


Figura 26.6. Modificación de un artículo

El código completo de `story.php` se recoge en el listado 26.6.

Listado 26.6. story.php. Se utiliza para crear o modificar un artículo

```
<?php

include ('include_fns.php');

if (isset($HTTP_GET_VARS['story']))
   $s = get_story_record($HTTP_GET_VARS['story']);

?>

 <form action="story_submit.php" method="post" enctype="multipart/form-
data">
 <input type="hidden" name="story" value="<?php print
$HTTP_GET_VARS['story'];?>">
 <input type="hidden" name="destination"
value="<?php print $HTTP_SERVER_VARS['HTTP_REFERER'];?>">
 <table>

 <tr>
   <td align="center">Headline<td>
 </tr>
 <tr>
   <td><input size="80" name="headline"
           value="<?php print $s['headline'];?>"></td>
  </tr>

 <tr>
   <td align="center">Page</td>
 </tr>
 <tr>
   <td align="center"><?php print query_select('page',
           "select p.code, p.description
           from pages p, writer_permissions wp, stories s
           where p.code = wp.page
           and wp.writer = s.writer
           and  s.id  =".$HTTP_GET_VARS['story'],  $s['page']);?></td>
 </tr>

 <tr>
   <td align="center">Story text (can contain HTML tags)</td>
 </tr>
 <tr>
   <td><textarea cols="80" rows="7" name="story_text"
           wrap="virtual"><?php print $s['story_text'];?></textarea>
   </td>
 </tr>

 <tr>
   <td align="center">Or upload HTML file</td>
 </tr>
 <tr>
   <td align="center"><input type="file" name="html" size="40"></td>
 </tr>

 <tr>
```

```
    <td align="center">Picture</td>
  </tr>
  <tr>
    <td align="center"><input type="file" name="picture" size="40"></td>
  </tr>

<?php
if ($s[picture]) {
   $size   = getImageSize($s['picture']);
   $width  = $size[0];
   $height = $size[1];
?>
  <tr>
    <td align="center">
      <img src="<?php print $s['picture'];?>"
           width="<?php print $width;?>" height="<?php print $height;?>">
    </td>
  </tr>
  <?php } ?>


  <tr>
    <td align="center"><input type="submit" value="Submit"></td>
  </tr>

   </table>
  </form>
```

Se puede utilizar la misma secuencia de comandos para añadir o para modificar. La acción depende de si `story` se configura al invocar la secuencia de comandos.

```
if (isset($HTTP_GET_VARS['story']))
  $s = get_story_record($HTTP_GET_VARS['story']);
```

La función `get_story_record()` se define en `db_fns.php` y devuelve una matriz de todos los campos de la tabla de artículos para el Id. de artículo especificado.

Si no se pasa ningún Id., `$story` será cero y `$s` no contendrá los elementos de la matriz.

```
  <input size="80" name="headline"
           value="<?php print $s['headline'];?>">
```

Si no se configura `story`, el código anterior no generará ningún valor a partir de la instrucción PHP, por lo que el cuadro de titular aparecerá vacío. Si se configura `story`, incluirá el texto del titular del artículo modificado.

```
print query_select('page',
           "select p.code, p.description
           from pages p, writer_permissions wp
           where p.code = wp.page
           and wp.writer = '".$HTTP_SESSION_VARS['auth_user']."'",
           $s['page']);
```

La función `query_select()` se define en `select_fns.php` y devuelve el código HTML para generar una lista `SELECT` a partir de una consulta SQL dada. El primer parámetro es el atributo `NAME` de `SELECT`.

La consulta SQL del segundo parámetro selecciona dos columnas: la primera es la parte `VALUE` de cada opción y la segunda aparece detrás de la etiqueta `OPTION` y corresponde al texto que se muestra en la lista. El tercer parámetro es opcional. Añade un atributo `SELECTED` a la opción cuyo valor coincide con el valor especificado.

```
<input type="hidden" name="story" value="<?php print $HTTP_GET_VARS['story'];?>">
```

De esta forma se define una variable de marcador de posición y se establece el nuevo valor del artículo a partir del `story` pasado. Al enviar el formulario, `story_submits` comprueba si hay un valor para `story` y genera una instrucción `UPDATE` o `INSERT` de SQL según corresponda.

En el listado 26.7 encontrará el código de `story_submit.php`.

Listado 26.7. story_submit.php. Se utiliza para añadir o actualizar un artículo en la base de datos

```
<?php

// story_submit.php
// añada / modifique el registro del artículo

include('include_fns.php');

$conn = db_connect();

$headline = $HTTP_POST_VARS['headline'];
$page = $HTTP_POST_VARS['page'];
$time = time();

if ( (isset($HTTP_POST_FILES['html']['name']) &&
 (dirname($HTTP_POST_FILES['html']['type']) == 'text')
        && is_uploaded_file($HTTP_POST_FILES['html']['tmp_name'])))
{
   $fp = fopen($HTTP_POST_FILES['html']['tmp_name'], 'r');
   $story_text = addslashes(fread($fp,
filesize($HTTP_POST_FILES['html']['tmp_name'])));
   fclose($fp);
}
else
   $story_text = $HTTP_POST_VARS['story_text'];



if (isset($HTTP_POST_VARS['story']) && $HTTP_POST_VARS['story']!='')
{   // Es una actualización
   $story = $HTTP_POST_VARS['story'];

   $sql = "update stories
```

```
            set headline = '$headline',
               story_text = '$story_text',
               page = '$page',
               modified = $time
            where id = $story";
}
else {          // Es un artículo nuevo
   $sql = "insert into stories
               (headline, story_text, page, writer, created, modified)
               values
               ('$headline', '$story_text', '$page', '"
               .$HTTP_SESSION_VARS['auth_user']."', $time, $time)";
}

$result = mysql_query($sql, $conn);

if (!$result) {
   print "There was a database error when executing <pre>$sql</pre>";
   print mysql_error();
   exit;
}


if ( (isset($HTTP_POST_FILES['picture']['name']) &&
        is_uploaded_file($HTTP_POST_FILES['picture']['tmp_name'])))
{

   if (!isset($story))
      $story = mysql_insert_id();

   $type = basename($HTTP_POST_FILES['picture']['type']);

   switch ($type) {
    case 'jpeg':
    case 'pjpeg': $filename = "pictures/$story.jpg";
                  move_uploaded_file($HTTP_POST_FILES['picture']['tmp_name'],
                    $filename);
                  $sql =  "update stories
                          set picture = '$filename'
                          where id = $story";
                  $result = mysql_query($sql, $conn);
                  break;
     default:      print 'Invalid picture format: '.
                          $HTTP_POST_FILES['picture']['type'];
   }
}

header('Location: '.$HTTP_POST_VARS['destination']);

?>
```

El enlace para eliminar artículos invoca `delete_story.php`, que funciona como una sencilla instrucción `DELETE` y devuelve al escritor a la página de invocación.

El código de `delete_story.php` se recoge en el listado 26.8.

Listado 26.8. delete_story.php. Se utiliza para borrar un artículo de la base de datos

```
<?php

// delete_story.php

include('include_fns.php');

$conn = db_connect();


$story = $HTTP_GET_VARS['story'];
$sql = "delete from stories where id = $story";
$result = mysql_query($sql, $conn);

header('Location: '.$HTTP_SERVER_VARS['HTTP_REFERER']);
?>
```

Búsquedas

Al pulsar sobre el enlace de palabras clave de la lista de artículos se accede a un nuevo formulario en el que se pueden introducir palabras clave en función del artículo. No hay límite en cuanto al número de palabras clave que se pueden introducir. A cada palabra se le asigna un valor ponderado, con un valor más alto para indicar que tiene mayor relevancia.

En la figura 26.7 se muestra la pantalla utilizada para definir palabras clave en función de un determinado artículo.

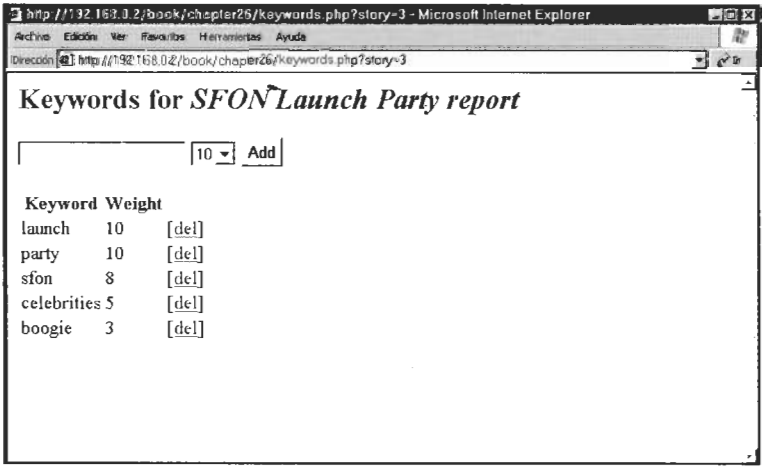


Figura 26.7. Definición de palabras clave para un artículo

La secuencia de comandos `keywords.php` es bastante sencilla, por lo que no la analizaremos con mucho detalle. Se incluye en el CD-ROM.

Esta secuencia de comandos desencadena las secuencias `keyword_add.php` y `keyword_delete.php`, también muy sencillas, razón por la que no las incluimos.

La secuencia de comandos `keyword_add.php` utiliza la siguiente consulta para añadir nuevas palabras clave a la base de datos:

```
insert into keywords (story, keyword, weight)
values  ($story,  '$keyword',  $weight)
```

Del mismo modo, `keyword_delete.php` utiliza la siguiente consulta para borrar una palabra clave:

```
delete from keywords where story = $story and keyword = '$keyword'
```

Lo interesante es la forma en la que se utilizan los valores para calcular una cifra porcentual relevante al realizar la búsqueda.

El formulario de búsqueda de `search_form.php` contiene un solo campo para palabras clave y se envía a `search.php`, que consulta la base de datos de artículos para encontrar contenidos que coincidan. El código de `search.php` se muestra en el listado 26.9.

Listado 26.9. search.php. Busca artículos que coincidan y calcula un valor de porcentaje de coincidencia

```
<?php

include('include_fns.php');
include('header.php');

$conn = db_connect();

if ($HTTP_POST_VARS['keyword']) {
   $k = split(' ', $HTTP_POST_VARS['keyword']);
   $num_keywords = count($k);
   for ($i=0; $i<$num_keywords; $i++) {
      if  ($i)
         $k_string .= "or k.keyword = '".$k[$i]."' ";
      else
         $k_string .= "k.keyword = '".$k[$i]."' ";
   }
   $and .= "and ($k_string) ";


$sql = "select s.id,
            s.headline,
            10 * sum(k.weight) / $num_keywords como puntuación
         from stories s, keywords k
         where s.id = k.story
         $and
         group by s.id, s.headline
         order by score desc, s.id desc";


$result = mysql_query($sql, $conn);
```

```
}
print '<h2>Search results</h2>';

if ($result  && mysql_num_rows($result)) {
   print '<table>';
   while  ($qry  =  mysql_fetch_array($result))  {
      print '<tr><td>';
      print $qry['headline'];
      print '</td><td>';
      print floor($qry['score']).'%';
      print '</td></tr>';
   }
   print '</table>';
}
else {
   print 'No matching stories found';
}
include('footer.php');
?>
```

En primer lugar, la cadena de palabra clave pasada a la secuencia de comandos se divide en palabras de búsqueda individuales. En este ejemplo no utilizamos técnicas de búsqueda avanzadas, como permitir el uso de las palabras clave `OR` o `AND` o la agrupación de palabras en una frase.

```
if ($HTTP_POST_VARS['keyword']) {
   $k = split(' ', $HTTP_POST_VARS['keyword']);
   $num_keywords = count($k);
   for ($i=0; $i<$num_keywords; $i++) {
      if ($i)
         $k_string .= "or k.keyword = '".$k[$i]."' ";
      else
         $k_string .= "k.keyword = '".$k[$i]."' ";
   }
   $and .= "and ($k_string) ";
```

Este código utiliza la función `split()` de PHP para crear una matriz que contiene todas las palabras de la cadena `keyword` separadas por un carácter de espacio. Si sólo se especifica una palabra, devuelve una única matriz de elementos y se ejecuta una vez el bucle correspondiente. En última instancia, la condición almacenada en `$and` tendrá este aspecto

```
and (k.keyword = 'keyword1' or k.keyword = 'keyword2' or k.keyword =
'keyword3')
```

La consulta de búsqueda generada a partir del código anterior sería

```
select s.id,
        s.headline,
        10  *  sum(k.weight)  /  $num_keywords  como  puntuación
from stories s, keywords k
where s.id = k.story
and (k.keyword = 'keyword1'
   or k.keyword = 'keyword2'
   or k.keyword = 'keyword3')
```

```
group by s.id, s.headline
order by score desc, s.id desc
```

El cálculo del valor es la suma de los valores ponderados de todas las palabras que coinciden dividido por el número de palabras clave buscadas y, tras ello, multiplicado por diez.

Esto favorece las búsquedas en las que todas las palabras clave introducidas coinciden con las palabras clave de la base de datos.

Como los valores ponderados oscilan entre 1 y 10, el valor máximo de la puntuación será 100.

Una búsqueda de tres palabras clave solamente coincidirá al 100 por ciento en un artículo si se encuentran las tres en dicho artículo y cada una tiene un valor ponderado del 10 por ciento.

Pantalla del editor

La única parte del sistema que no hemos descrito es cómo se publica un artículo una vez escrito. La secuencia de comandos publish.php (listado 26.10) se encarga de publicar un artículo.

Listado 26.10. publish.php. Enumera todos los artículos para que el editor pueda seleccionar cuáles va a mostrar en directo en el sitio

```
<?php

include('include_fns.php');

$conn = db_connect();

$sql = 'select * from stories order by modified desc';
$result = mysql_query($sql, $conn);

print '<h2>Editor admin</h2>';
print '<table>';
print '<tr><th>Headline</th><th>Last modified</th></tr>';
while ($story = mysql_fetch_array($result)) {
   print '<tr><td>';
   print $story['headline'];
   print '</td><td>';
   print date('M d, H:i', $story['modified']);
   print '</td><td>';
   if ($story[published]) {
      print '[<a
href="unpublish_story.php?story='.$story['id'].'">unpublish</a>] ';
   }
   else {
      print '[<a href="publish_story.php?story='.$story['id'].'">publish</
a>] ';
      print '[<a href="delete_story.php?story='.$story['id'].'">delete</a>] ';
   }
```

```
    print '[<a href="story.php?story='.$story['id'].'">edit</a>] ';

    print '</td></tr>';
}
print '</table>';
?>
```

Esta secuencia de comandos solamente estará disponible para los que tengan autorización para publicar artículos en directo en el sitio. En nuestra aplicación de ejemplo, el encargado será el editor del sitio pero, por motivos de simplicidad, esta secuencia de comandos no cuenta con control de acceso. No obstante, en una situación real, se protegerá.

Es muy similar a `stories.php` a excepción de que al editor se le asigna una pantalla en la que se muestran los artículos de todos los escritores, no sólo los suyos. La instrucción `if` garantiza que en cada artículo se presentan las opciones adecuadas. Se puede anular la publicación de los artículos publicados y los artículos no publicados se pueden publicar o eliminar.

Estos tres enlaces se envían a `unpublish_story.php`, `publish_story.php` y `delete_story.php`, respectivamente. La secuencia de comandos `publish_story.php` utiliza la siguiente consulta SQL:

```
update stories set published = $now
        where id = $story
```

De esta forma el artículo se marca como publicado y se autoriza que se pueda ver públicamente.

Del mismo modo, `unpublish_story.php` utiliza la siguiente consulta para marcar un artículo como no publicado e impide que se muestre al público:

```
update stories set published = null
        where id = $story
```

El enlace de modificación aparece independientemente de si se ha publicado o no el artículo, para que el editor siempre pueda realizar cambios. Esto es diferente en el nivel de acceso de los escritores, ya que únicamente pueden modificar un artículo antes de que se publique.

Ampliación del proyecto

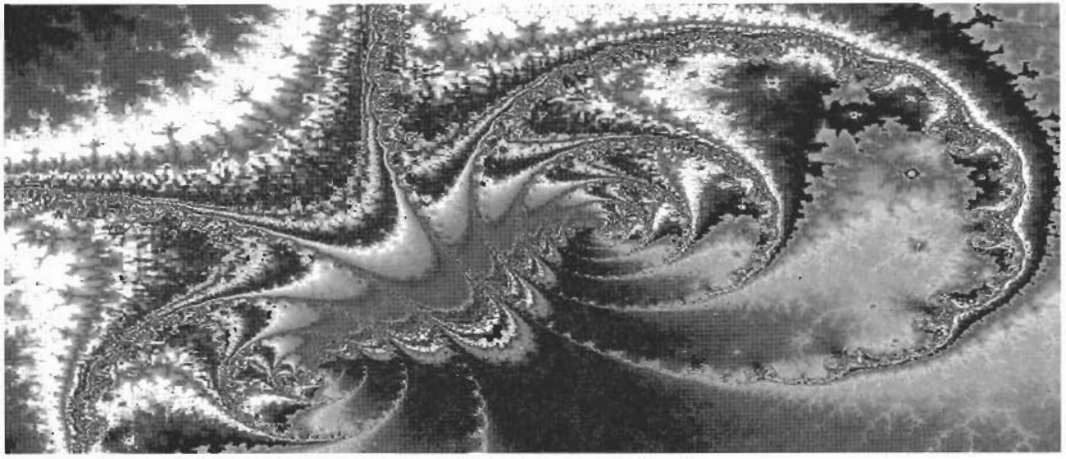
Existen diversas formas de ampliar este proyecto para que el sistema de administración de contenidos sea más completo:

- Se puede permitir el trabajo conjunto de grupos de usuarios en los artículos (colaboración).
- Se puede implementar un diseño de páginas más flexible para que los editores puedan ubicar texto e imágenes en la página.

- Se puede crear una biblioteca de imágenes para no repetir las que se utilicen con mayor frecuencia. También se puede asignar palabras clave y texto de los artículos a las imágenes.
- Se puede añadir funcionalidad de revisión ortográfica al editor de contenidos. Por ejemplo, se puede implementar la revisión con ayuda de la biblioteca aspell.

A continuación

En el siguiente proyecto crearemos una interfaz basada en la Web que nos permitirá comprobar y enviar correo electrónico desde la Web por medio de IMAP.



27
Creación de un servicio de correo electrónico basado en la Web

En la actualidad, aumenta a diario el número de sitios que ofrecen correo electrónico basado en la Web a sus usuarios. En este capítulo explicaremos cómo implementar una interfaz Web en un servidor de correo existente por medio de la biblioteca IMAP de PHP. La podemos utilizar para comprobar nuestro buzón a través de una página Web o incluso ampliarla para que admita gran cantidad de usuarios de correo electrónico masivo basado en la Web, al estilo de Hotmail. En este proyecto crearemos un cliente de correo electrónico, Warm Mail, que permitirá a los usuarios

- Conectarse a sus cuentas en servidores de correo POP3 o IMAP
- Leer correo
- Enviar correo
- Responder a mensajes de correo
- Reenviar mensajes de correo
- Borrar correo de sus cuentas

El problema

Para que un usuario pueda leer su correo, necesitamos algún modo de realizar la conexión a su servidor de correo que, normalmente, no será el mismo equipo que el servidor Web.

Necesitamos alguna forma de interactuar con su buzón, para ver qué mensajes ha recibido y para procesarlos individualmente.

Los servidores de correo admiten dos protocolos principales para leer los buzones de un usuario: POP3 e IMAP. Si podemos, admitiremos los dos. POP3 equivale a Protocolo de oficina de correos versión 3, mientras que IMAP significa Protocolo de acceso a mensajes de Internet.

La principal diferencia entre ambos es que POP3 está dirigido a, y lo utilizan principalmente, usuarios que se conectan brevemente a una red para descargar y borrar su correo desde un servidor. IMAP se utiliza en línea, para interactuar con correo que se almacena permanentemente en un servidor remoto. IMAP dispone de algunas funciones avanzadas que no utilizaremos.

Si está interesado en las diferencias entre estos protocolos, puede consultar sus correspondientes RFC (la RFC 1939 para POP versión 3 y la RFC 2060 para IMAP, versión 4 rev1). También encontrará un excelente artículo sobre el tema en la dirección `http://www.imap.org/papers/imap.vs.pop.brief.html`.

Ninguno de estos protocolos se diseñó para enviar correo; para ello, se utiliza el protocolo SMTP, que ya hemos utilizado anteriormente en PHP a través de la función `mail()`. Este protocolo se describe en la RFC 821.

Componentes de la solución

PHP cuenta con una excelente compatibilidad con IMAP y POP3, pero proporcionada a través de la biblioteca de funciones IMAP. Para poder utilizar el código que presentaremos en este capítulo, tendrá que instalar la biblioteca IMAP. Para saber si ya la tiene instalada, compruebe el resultado de la función `phpinfo()`. En caso contrario, tendrá que descargar la extensión. Puede obtener la última versión a través de FTP de `ftp://ftp.cac.washington.edu/imap/c-client.tar.Z`.

En UNIX, descargue el código fuente y compílelo para su sistema operativo. Tras ello, copie `rfc822.h`, `mail.h` y `linkage.h` en `/usr/local/include` o en otro directorio de su ruta de inclusión, ejecute la secuencia de comandos de configuración de PHP, incluyendo la directiva `–with-imap` en todos los parámetros que utilice, y recompile PHP.

Existe documentación sobre la compilación de la versión de Windows, pero es mucho más compleja que la compilación para UNIX. Para utilizar la extensión IMAP con Windows, descargue la versión en formato ZIP de PHP en lugar del Asistente de instalación. El archivo ZIP contiene numerosas extensiones, incluyendo IMAP. Abra su archivo `php.ini` y elimine el comentario de la línea:

```
extension=php_imap.dll
```

Un aspecto que conviene destacar es que aunque estas funciones reciben el nombre de IMAP, también funcionan con POP3 y NNTP. Las utilizaremos con IMAP y POP3 pero la aplicación Warm Mail también se puede ampliar para utilizar NNTP y emplearlo como lector de noticias además de como cliente de correo.

Esta biblioteca incluye una gran cantidad de funciones, pero para poder implementar la funcionalidad en esta aplicación, solamente utilizaremos algunas de ellas.

Las describiremos según las vayamos utilizando, pero debe saber que existen muchas más. Consulte la documentación si sus necesidades son distintas a las nuestras o si quiere dotar de opciones adicionales a la aplicación.

Podemos crear una aplicación de correo muy útil con tan sólo parte de las funciones incorporadas. Esto significa que solamente tendrá que recurrir a parte de la documentación.

Las funciones IMAP que utilizaremos en este capítulo son:

- `imap_open()`
- `imap_close()`
- `imap_headers()`
- `imap_header()`
- `imap_fetchheader()`
- `imap_body()`
- `imap_delete()`
- `imap_expunge()`

Para que un usuario pueda leer su correo, tendremos que obtener los detalles de su servidor y de su cuenta. En lugar de obtener continuamente esta información, definiremos una base de datos de nombres de usuario y contraseñas de usuarios en la que podamos almacenar los datos del usuario.

A menudo, los usuarios tienen más de una cuenta de correo electrónico (por ejemplo, una para casa y otra para el trabajo) por lo que tendremos que permitir que se conecten a cualquiera de ellas.

Por esta razón, es necesario permitirles que dispongan de varios conjuntos de información sobre cuentas en la base de datos.

Los usuarios podrán leer, contestar, reenviar y borrar correos electrónicos existentes así como enviar otros nuevos. Podemos realizar todas las operaciones de lectura por medio de IMAP o POP3 y todas las de escritura por medio de SMTP con `mail()`.

Veamos cómo se combinan todas las piezas.

Presentación de la solución

El flujo general de este sistema basado en la Web no es muy diferente al de otros clientes de correo electrónico.

En la figura 27.1 se incluye un diagrama que muestra el flujo y los módulos del sistema.

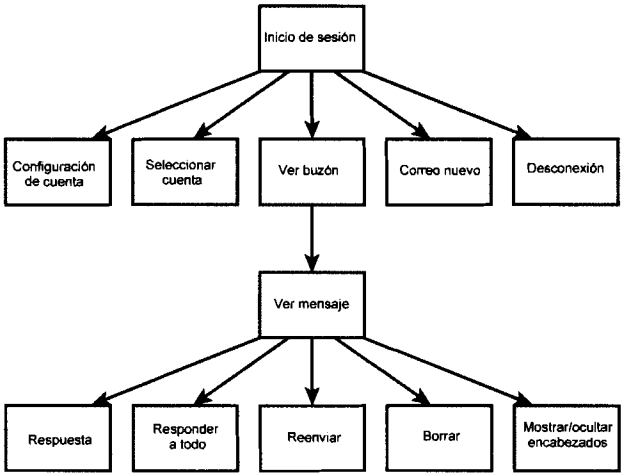


Figura 27.1. La interfaz de Warm Mail ofrece al usuario funcionalidad en el nivel de buzón y de mensajes

Como puede comprobar, primero será necesario que el usuario se conecte y, tras ello, le ofreceremos una serie de opciones. Podrá configurar una nueva cuenta de correo o seleccionar una de las ya existentes. También podrá ver su correo entrante (responder a él, reenviarlo o borrarlo) y enviar nuevos correos.

También le permitiremos ver encabezados detallados de un determinado mensaje. Mediante estos encabezados se puede saber mucho sobre un mensaje. Se puede saber de qué equipo proviene el correo, una opción muy útil para detectar correo basura. Se puede saber qué equipo lo ha reenviado y a qué hora llegó a cada host, lo que resulta muy útil para determinar el culpable de mensajes retrasados. Si la aplicación añade información opcional a los encabezados, incluso puede saber qué cliente de correo electrónico ha utilizado el emisor.

En este proyecto hemos utilizado una arquitectura de aplicación ligeramente diferente. En lugar de utilizar un conjunto de secuencias de comandos, una para cada módulo, tenemos una secuencia más extensa, `index.php`, que funciona como el bucle de eventos de un programa dirigido por GUI. Cada acción que realicemos en el sitio al pulsar un botón nos llevará de nuevo a `index.php`, pero con un parámetro diferente.

En función del parámetro, se invocarán distintas funciones para mostrar el correspondiente resultado al usuario. Como de costumbre, las funciones se encuentran en bibliotecas de funciones.

La arquitectura es adecuada para pequeñas aplicaciones como ésta. Se adecua a aplicaciones dirigidas principalmente por eventos en las que las acciones del usuario desencadenan la funcionalidad. El uso de un solo controlador de eventos no es aconsejable en arquitecturas de gran tamaño o en proyectos realizados por un equipo.

En la tabla 27.1 encontrará un resumen de los archivos utilizados en el proyecto Warm Mail.

Tabla 27.1. Archivos de la aplicación Warm Mail

Nombre	Tipo	Descripción
`index.php`	Aplicación	La secuencia de comandos principal que ejecuta toda la aplicación
`include_fns.php`	Funciones	Colección de archivos para esta aplicación
`data_valid_fns.php`	Funciones	Colección de funciones para validar los datos introducidos
`db_fns.php`	Funciones	Colección de funciones para realizar la conexión a la base de datos `mail`
`mail_fns.php`	Funciones	Colección de funciones relacionadas con correo electrónico para abrir buzones, leer correo, etc.
`output_fns.php`	Funciones	Colección de funciones para representar HTML
`user_auth_fns.php`	Funciones	Colección de funciones para autenticar usuarios
`create_database.sql`	SQL	SQL para definir la base de datos `book_sc` y configurar a un usuario

A continuación analizaremos la aplicación

Configuración de la base de datos

La base de datos de Warm Mail es muy sencilla ya que realmente no almacenaremos ningún correo electrónico en la misma.

Necesitaremos almacenar usuarios del sistema. Para cada usuario, tendremos que almacenar los siguientes campos:

- `username`. Su nombre de usuario preferido para Warm Mail
- `password`. Su contraseña preferida para Warm Mail
- `address`. Su dirección de correo electrónico preferida, que aparecerá en el campo `From` de los correos electrónicos enviados desde el sistema
- `displayname`. El nombre "legible" que aparecerá en los correos que envíen a otros usuarios.

También será necesario almacenar todas las cuentas que los usuarios quieran comprobar con el sistema. En cada cuenta, almacenaremos la siguiente información:

- `username`. El usuario Warm Mail al que pertenece esta cuenta.
- `server`. El equipo en el que se encuentra la cuenta, como por ejemplo localhost o mail.tangled.com.au.
- `port`. El puerto al que se realiza la conexión al utilizar esta cuenta. Normalmente será el 110 para servidores POP3 y el 143 para servidores IMAP.
- `type`. El protocolo utilizado para realizar la conexión a este servidor, ya sea `'POP3'` o `'IMAP'`.
- `remoteuser`. El nombre de usuario para conectarse al servidor de correo.
- `remotepassword`. La contraseña para conectarse al servidor de correo.
- `accountid`. Una clave exclusiva para identificar cuentas.

Al ejecutar el código SQL del listado 27.1 se configura la base de datos para esta aplicación.

Listado 27.1. create_database.sql. SQL para crear la base de datos Mail

```
create database mail;

use mail;

create table users
(
   username char(16) not null primary key,
   password char(16) not null,
   address char(100) not null,
   displayname char(100) not null
);

create table accounts
(
   username char(16) not null,
   server char(100) not null,
   port int not null,
   type char(4) not null,
   remoteuser char(50) not null,
   remotepassword char(50) not null,
   accountid int unsigned not null auto_increment primary key
);

grant select, insert, update, delete
on mail.*
to  mail@localhost  identified  by  'password';
```

Recuerde que puede ejecutar este SQL si escribe

```
mysql  -u  root  -p  <  create_database.sql
```

tendrá que introducir su contraseña de usuario raíz. También debería cambiar la contraseña del usuario de correo en `create_database.sql` y en `db_fns.php` antes de ejecutarlo. En el CD-ROM se incluye un archivo SQL con el nombre `populate.sql`. En esta aplicación, no vamos a crear un proceso de registro o de administración de usuarios. Puede añadir uno personalmente si quiere utilizar este software a mayor escala, pero si sólo lo quiere para uso personal, basta con recurrir a la base de datos. La secuencia de comandos `populate.sql` proporciona un modelo para hacerlo, por lo que sólo tiene que introducir sus detalles y ejecutarla para configurarse como usuario.

Arquitectura de secuencias de comandos

Como hemos mencionado anteriormente, esta aplicación utiliza una secuencia de comandos para controlarlo todo. Esta secuencia se denomina `index.php` y se incluye en el listado 27.2. Es bastante extensa, pero la analizaremos sección a sección.

Listado 27.2. index.php. Parte principal del sistema Warm Mail

```
<?php
// Este archivo es el cuerpo principal de la aplicación Warm Mail.
// Funciona básicamente como un equipo de estado y muestra a los
// usuarios el
// resultado de la acción que hayan seleccionado.

//*****************************************************************************
// Fase 1: preprocesamiento
// Realice todo el procesamiento necesario antes de enviar
// el encabezado
// de página y determine qué detalles mostrará en los encabezados de página
//*****************************************************************************

   include ('include_fns.php');
   session_start();

   //cree nombres de variable cortos
   $username = $HTTP_POST_VARS['username'];
   $passwd = $HTTP_POST_VARS['passwd'];
   if(isset($HTTP_POST_VARS['action']))
      $action = $HTTP_POST_VARS['action'];
      else
      $action = $HTTP_GET_VARS['action'];
   if(isset($HTTP_POST_VARS['account']))
      $account = $HTTP_POST_VARS['account'];
   else
      $account = $HTTP_GET_VARS['account'];

   $messageid = $HTTP_GET_VARS['messageid'];
```

```
$to =  $HTTP_POST_VARS['to'];
$cc =  $HTTP_POST_VARS['cc'];
$subject =  $HTTP_POST_VARS['subject'];
$message =  $HTTP_POST_VARS['message'];

$buttons = array();

//adjunte esta cadena si hay algo antes del encabezado que tenga
// resultados
$status = '';

// primero debe procesar las solicitudes de conexión y desconexión
if($username||$password)
{
   if(login($username, $passwd))
   {
      $status .= '<p>Logged in successfully.</p><br /><br /><br /><br />
         <br /><br />';
      $HTTP_SESSION_VARS['auth_user'] = $username;
      if(number_of_accounts($HTTP_SESSION_VARS['auth_user'])==1)
      {
         $accounts = get_account_list($HTTP_SESSION_VARS['auth_user']);
         $HTTP_SESSION_VARS['selected_account'] = $accounts[0];
      }
   }
   else
   {
      $status .= '<p>Sorry, we could not log you in with that
         username and password.</p><br /><br /><br /><br />
         <br /><br />';
   }
}
if($action == 'log-out')
{
   session_destroy();
   unset($action);
   unset($HTTP_SESSION_VARS);
}

// debe procesar la selección, eliminación o almacenamiento de cuentas
// antes de dibujar el encabezado
switch ( $action )
{
   case 'delete-account' :
   {
      delete_account($HTTP_SESSION_VARS['auth_user'], $account);
      break;
   }
   case 'store-settings' :
   {
      store_account_settings($HTTP_SESSION_VARS['auth_user'],
      $HTTP_POST_VARS);
      break;
   }
   case 'select-account' :
   {
```

```
        // si han seleccionado una cuenta válida, almacénela como variable
        // de sesión
        if($account&&account_exists($HTTP_SESSION_VARS['auth_user'],
        $account))
        {
           $HTTP_SESSION_VARS['selected_account'] = $account;
        }
     }
  }
  // defina los botones que aparecerán en la barra de herramientas
  $buttons[0]  =  'view-mailbox';
  $buttons[1] = 'new-message';
  $buttons[2] = 'account-setup';
  //solamente ofrezca un botón de desconexión si están conectados
  if(check_auth_user())
  {
     $buttons[4] = 'log-out';
  }

//
*****************************************************************************
// Fase 2: encabezados
// Envíe los encabezados HTML y la barra de menú correspondiente a la
// acción actual
//
*****************************************************************************
  if($action)
  {
     // muestre el encabezado con el nombre de la aplicación y la
     // descripción de la página o
     // acción
     do_html_header($HTTP_SESSION_VARS['auth_user'], "Warm Mail - ".
           format_action($action),
           $HTTP_SESSION_VARS['selected_account']);
  }
  else
  {
     // muestre el encabezado sólo con el nombre de la aplicación
     do_html_header($HTTP_SESSION_VARS['auth_user'], "Warm Mail",
     $HTTP_SESSION_VARS['selected_account']);
  }

  display_toolbar($buttons);

//
*****************************************************************************
// Fase 3: cuerpo
// En función de la acción, muestre el correspondiente contenido del cuerpo
//
*****************************************************************************
  // muestre cualquier texto generado por las funciones invocadas antes
  // del encabezado
  echo $status;

  if(!check_auth_user())
  {
```

```
   echo '<p>You need to log in';
   if($action&&$action!='log-out')
      echo ' to go to '.format_action($action);
   echo '.</p><br /><br />';
   display_login_form($action);
}
else
{
   switch ( $action )
   {
      // si hemos optado por configurar una cuenta, o acabamos de añadir
      // o
      // eliminar una cuenta, muestre la página de configuración de
      // cuentas
      case 'store-settings' :
      case 'account-setup' :
      case 'delete-account' :
      {
         display_account_setup($HTTP_SESSION_VARS['auth_user']);
         break;
      }
      case 'send-message' :
      {
         if(send_message($to, $cc, $subject, $message))
            echo '<p>Message sent.</p><br /><br /><br /><br /><br
            /><br />';
         else
            echo '<p>Could not send message.</p><br /><br /><br /><br
            />
               <br /><br />';
         break;
      }
         case 'delete' :
         {
            delete_message($HTTP_SESSION_VARS['auth_user'],
               $HTTP_SESSION_VARS['selected_account'],
               $messageid);
            // fíjese que no hay 'salto' - continuaremos con el siguiente
            // caso
      }
      case 'select-account' :
      case 'view-mailbox' :
      {
         // si se selecciona mailbox o view mailbox, muestre el buzón
         display_list($HTTP_SESSION_VARS['auth_user'],
            $HTTP_SESSION_VARS['selected_account']);
         break;
      }
      case 'show-headers' :
      case  'hide-headers'  :
      case 'view-message' :
      {
         // si acabamos de seleccionar un mensaje de la lista, o estamos
         // analizando
         // un mensaje y optamos por ocultar o ver los encabezados,
         // cargue un
```

```
    // mensaje
    $fullheaders = ($action=='show-headers');
    display_message($HTTP_SESSION_VARS['auth_user'],
          $HTTP_SESSION_VARS['selected_account'],
          $messageid, $fullheaders);
    break;
}
case 'reply-all' :
{
    //establezca cc como la línea cc antigua
    if(!$imap)
    $imap = open_mailbox($HTTP_SESSION_VARS['auth_user'],
          $HTTP_SESSION_VARS['selected_account']);
    if($imap)
    {
       $header = imap_header($imap, $messageid);
       if($header->reply_toaddress)
          $to = $header->reply_toaddress;
       else
          $to = $header->fromaddress;
       $cc = $header->ccaddress;
       $subject = 'Re: '.$header->subject;
       $body = add_quoting(stripslashes(imap_body($imap,
       $messageid)));
       imap_close($imap);

    display_new_message_form($HTTP_SESSION_VARS['auth_user'],
    $to, $cc, $subject, $body);
    }
    break;
}
case 'reply' :
{
    // establezca la dirección como reply-to o from del mensaje
    // actual
    if(!$imap)
       $imap =
       open_mailbox($HTTP_SESSION_VARS['auth_user'],
          $HTTP_SESSION_VARS['selected_account']);
    if($imap)
    {
       $header = imap_header($imap, $messageid);
       if($header->reply_toaddress)
          $to = $header->reply_toaddress;
       else
          $to = $header->fromaddress;
       $subject = 'Re: '.$header->subject;
       $body = add_quoting(stripslashes(imap_body($imap,
       $messageid)));
       imap_close($imap);

    display_new_message_form($HTTP_SESSION_VARS['auth_user'],
                $to, $cc, $subject, $body);
    }

    break;
```

```
        }
        case 'forward' :
        {
           //establezca el mensaje como cuerpo del mensaje actual
           if(!$imap)
              $imap =
              open_mailbox($HTTP_SESSION_VARS['auth_user'],
                 $HTTP_SESSION_VARS['selected_account']);
           if($imap)
           {
              $header = imap_header($imap, $messageid);
              $body = add_quoting(stripslashes(imap_body($imap,
              $messageid)));
              $subject = 'Fwd: '.$header->subject;
              imap_close($imap);

              display_new_message_form
              ($HTTP_SESSION_VARS['auth_user'],
                 $to, $cc, $subject, $body);
           }
           break;
        }
        case 'new-message' :
        {
           display_new_message_form($HTTP_SESSION_VARS['auth_user'],
                       $to, $cc, $subject, $body);
           break;
        }
     }
  }
//
*****************************************************************************
// Fase 4: pie
//
*****************************************************************************
  do_html_footer();
?>
```

Esta secuencia de comandos utiliza un enfoque de control de eventos. Contiene la lógica para determinar qué función hay que invocar para cada evento. En este caso, los eventos se desencadenan cuando el usuario pulsa alguno de los botones del sitio, cada uno de los cuales selecciona una acción. La mayoría de los botones se generan por medio de la función `display_button()` pero, si se trata de un botón de envío, se utiliza la función `display_form_button()`. Ambas funciones se encuentran en `output_fns.php`. Todas acceden a URL con la siguiente forma

```
index.php?action=log-out
```

El valor de la variable `action` cuando se invoca `index.php` determina qué controlador de evento se activa.

Las cuatro principales secciones de la secuencia de comandos son las siguientes:

1. Es necesario realizar algún procesamiento antes de enviar el encabezado de página al navegador, como por ejemplo iniciar la sesión, ejecutar cualquier

procesamiento previo para la acción seleccionada por el usuario y decidir el aspecto de los encabezados.

2. Procesamos y enviamos los correspondientes encabezados y la barra de menú para la acción seleccionada por el usuario.
3. Seleccionamos el cuerpo de la secuencia de comandos que se va a ejecutar, en función de la acción seleccionada. Las distintas acciones desencadenan diferentes llamadas de funciones.
4. Enviamos los pies de página.

Si observa el código de la secuencia de comandos, comprobará que estas cuatro secciones están marcadas con comentarios. Para comprender cómo funciona este sitio, describiremos el sitio acción por acción.

Conexión y desconexión

Cuando un usuario carga la página `index.php`, accede al resultado que mostramos en la figura 27.2.

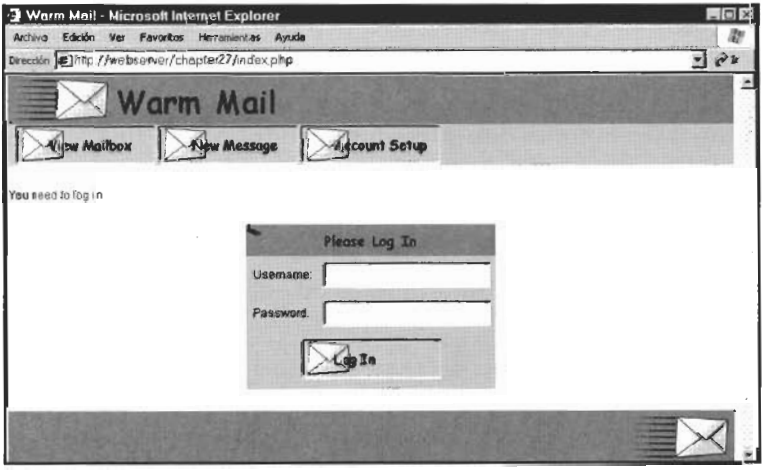


Figura 27.2. La pantalla de inicio de sesión de Warm Mail le pedirá su nombre de usuario y su contraseña

Éste es el comportamiento predeterminado de la aplicación. Como todavía no hemos seleccionado ninguna `$action` y no hemos introducido detalles de conexión, ejecutaremos las siguientes partes del código.

En la fase de preprocesamiento, en primer lugar ejecutamos el siguiente código:

```
include 'include_fns.php';
session_start();
```

Estas líneas inician la sesión que se utilizará para realizar el seguimiento de las variables de sesión `$auth_user` y `$selected_account`, que veremos más adelante.

Al igual que en el resto de aplicaciones, crearemos nombres de variables cortos. Lo hemos hecho en todas las secuencias de comandos relacionadas desde el primer capítulo, por lo que no merece la pena mencionarlo, excepto para la variable `action`. En función de qué parte de la aplicación provenga, puede tratarse de una variable `GET` o `POST`. Probamos la existencia de `$HTTP_POST_VARS['acción']` y, si no está definida, utilizamos la versión `GET`. El código más importante se concentra en estas cuatro líneas:

```
if(isset($HTTP_POST_VARS['acción']))
   $action  =  $HTTP_POST_VARS['acción'];
else
   $action  =  $HTTP_GET_VARS['acción'];
```

Tendremos que hacer lo mismo con la variable `account`, a la que normalmente se accede a través de `GET`, pero con la que se utiliza `POST` para eliminar una cuenta.

Si utiliza PHP 4.1 o una versión superior, puede evitar estos problemas si accede a las variables de formulario por medio de `$_REQUEST` (no la hemos utilizado aquí por cuestiones de compatibilidad).

Para ahorrar trabajo a la hora de personalizar la interfaz de usuario, los botones que aparecen en la barra de herramientas los controla una matriz. Declaramos una matriz vacía

```
$buttons = array();
```

y definimos los botones que queremos incluir en la página:

```
$buttons[0] = 'view-mailbox';
$buttons[1] = 'new-message';
$buttons[2] = 'account-setup';
```

En la fase del encabezado, imprimimos un encabezado normal de color vainilla:

```
do_html_header($HTTP_SESSION_VARS['auth_user'], 'Warm Mail',
                $HTTP_SESSION_VARS['selected_account']);
...
display_toolbar($buttons);
```

Este código imprime la barra de título y de encabezado y, tras ello, la barra de herramientas de botones que se puede ver en la figura 27.2. Estas funciones se encuentran en la biblioteca de funciones `output_fns.php`, pero como su efecto se puede comprobar en la imagen, no las analizaremos aquí.

Seguidamente llegamos al cuerpo del código:

```
if(!check_auth_user())
{
   echo '<p>You need to log in';
   if($action&&$action!='log-out')
   echo  '  to  go  to  '.format_action($action);
```

```
    echo '.</p><br /><br />';
    display_login_form($action);
}
```

La función `check_auth_user()` proviene de la biblioteca `user_auth_fns.php`. Hemos utilizado un código similar en algunos de los proyectos anteriores, para comprobar si el usuario está conectado. Si no lo está, como sucede en este caso, le mostramos un formulario de inicio de sesión, que puede ver en la figura 27.2. Este formulario se dibuja en la función `display_login_form()` de `output_fns.php`.

Si el usuario completa el formulario correctamente y pulsa el botón **Log In**, verá el resultado que se muestra en la figura 27.3.

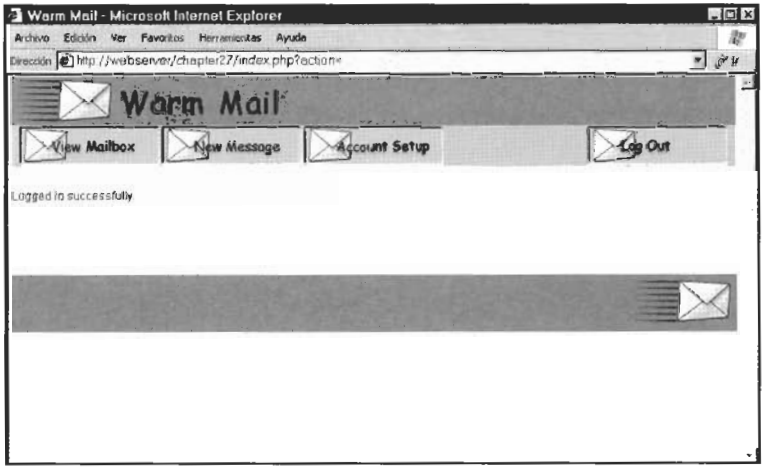


Figura 27.3. Después de conectarse satisfactoriamente, el usuario puede empezar a utilizar la aplicación

En esta ejecución de la secuencia de comandos, activaremos distintas secciones del código. El formulario de inicio de sesión tiene dos campos, `$username` y `$password`. Si estos campos se han completado, se activará el siguiente segmento de código de preprocesamiento:

```
if($username||$password)
{
    if(login($username, $passwd))
    {
        $status .= '<p>Logged in successfully.</p><br /><br /><br />
                <br /><br />';
        $HTTP_SESSION_VARS["auth_user"] = $username;
        if(number_of_accounts($HTTP_SESSION_VARS["auth_user"])==1)
        {
            $accounts = get_account_list($HTTP_SESSION_VARS['auth_user']);
            $HTTP_SESSION_VARS['selected_account'] = $accounts[0];
        }
```

```
    }
    else
    {
        $status .= '<p>Sorry, we could not log you in with that
              username and password.</p><br /><br /><br /><br /><br /><br />';
    }
}
```

Como puede comprobar, el código invoca la función `login()`, similar a la que hemos utilizado en capítulos anteriores. Si todo funciona correctamente, registramos el nombre de usuario en la variable de sesión `auth_user`.

Además de configurar los botones que vemos cuando no estamos conectados, añadiremos otro botón para permitir que el usuario pueda desconectarse de nuevo, como se indica a continuación:

```
if(check_auth_user())
{
      $buttons[4] = 'log-out';
}
```

Puede ver el botón **Log Out** en la figura 27.3.

En la fase del encabezado, mostramos de nuevo el encabezado y los botones. En el cuerpo, mostramos el mensaje de estado que definimos anteriormente:

```
echo  $status;
```

Tras ello, basta con imprimir el pie y esperar a ver qué hace el usuario.

Configuración de cuentas

Cuando un usuario utiliza por primera vez el sistema Warm Mail, tendrá que configurar alguna cuenta de correo. Si el usuario pulsa el botón **Account Setup**, se define la variable `action` como `account-setup` y se vuelve a invocar la secuencia de comandos `index.php`. Tras ello, el usuario verá el resultado que mostramos en la figura 27.4.

Volvamos a la secuencia de comandos del listado 27.2. En esta ocasión, debido al valor de `$action`, obtendremos un comportamiento diferente.

El comportamiento obtenido será ligeramente diferente, como se indica a continuación:

```
do_html_header($HTTP_SESSION_VARS['auth_user'], 'Warm Mail - '.
        format_action($action), $HTTP_SESSION_VARS['selected_account']);
```

Lo que es más importante, obtenemos un cuerpo diferente:

```
case 'store-settings' :
case 'account-setup' :
case 'delete-account' :
{
```

```
    display_account_setup($HTTP_SESSION_VARS['auth_user']);
    break;
}
```



Figura 27.4. El usuario tiene que configurar su cuenta de correo electrónico antes de que pueda leer su correo

Es el modelo habitual: cada comando invoca una función. En este caso, invocamos la función `display_account_setup()`, cuyo código se muestra en el listado 27.3.

Listado 27.3. Función display_account_setup() de output_fns.php. Función para obtener y mostrar detalles de las cuentas

```
function display_account_setup($auth_user)
{
    //muestre un formulario 'new account' vacío

    display_account_form($auth_user);
    $list = get_accounts($auth_user);
    $accounts = sizeof($list);

    // muestre todas las cuentas almacenadas
    foreach($list as $key => $account)
    {
        // muestre un formulario para los detalles de cada cuenta.
        // fíjese en que vamos a enviar la contraseña de todas las cuentas en
        // el HTML
        // no es aconsejable
        display_account_form($auth_user, $account['accountid'],
$account['server'],
                $account['remoteuser'], $account['remotepassword'],
                $account['type'], $account['port']);
    }
}
```

Al invocar esta función, muestra un formulario en blanco para añadir una nueva cuenta, seguido por formularios modificables que contienen cada una de las cuentas de correo actuales del usuario. La función `display_account_form()` mostrará el formulario de la figura 27.4. Comprobará que lo utilizamos de dos formas diferentes. Lo utilizamos sin parámetros para mostrar un formulario vacío y lo utilizamos con un conjunto de parámetros para representar un registro existente. Esta función se encuentra en la biblioteca `output_fns.php` y simplemente representa código HTML, por lo que no la analizaremos.

La función que recupera todas las cuentas existentes es `get_accounts()`, de la biblioteca `mail_fns.php`. En el listado 27.4 se incluye esta función.

Listado 27.4. Función get_accounts() de mail_fns.php. Función para recuperar todos los detalles de cuentas de un determinado usuario

```
function  get_accounts($auth_user)
{
   $list = array();
   if(db_connect())
   {
      $query = "select * from accounts where username = '$auth_user'";
      $result = mysql_query($query);
         if($result)
      {
         while($settings = mysql_fetch_array($result))
            array_push( $list, $settings);
      }
      else
         return false;
   }
   return $list;
}
```

Como puede comprobar, esta función se conecta a la base de datos, recupera todas las cuentas de un determinado usuario y los devuelve en forma de matriz.

Creación de una nueva cuenta

Si un usuario completa el formulario de cuentas y pulsa el botón **Save Changes**, se activa la acción `store-settings`. A continuación veremos el código de control de eventos para ello desde `index.php`. En la fase de preprocesamiento, ejecutamos el siguiente código:

```
case 'store-settings' :
{
   store_account_settings($HTTP_SESSION_VARS['auth_user'],
$HTTP_POST_VARS);
   break;
}
```

La función `store_account_settings()` escribe los detalles de la nueva cuenta en la base de datos. Su código se describe en el listado 27.5.

Listado 27.5. Función store_account_settings() de mail_fns.php. Función para guardar nuevos detalles de cuentas de un usuario

```
function store_account_settings($auth_user, $settings)
{
   if(!filled_out($settings))
   {
      echo 'All fields must be filled in.  Try again.<br /><br />';
      return false;
   }
   else
   {
      if($settings['account']>0)
         $query = "update accounts  set server = '$settings[server]',
               port = $settings[port], type = '$settings[type]',
               remoteuser = '$settings[remoteuser]',
               remotepassword = '$settings[remotepassword]'
         where accountid = $settings[account]
            and username = '$auth_user'";
   else
         $query = "insert into accounts values ('$auth_user',
               '$settings[server]', $settings[port],
               '$settings[type]', '$settings[remoteuser]',
               '$settings[remotepassword]', NULL)";
      if(db_connect() && mysql_query($query))
      {
         return true;
      }
      else
      {
         echo 'could not store changes.<br /><br /><br /><br /><br /><br />';
         return false;
      }
   }
}
```

Como puede apreciar, dos de las opciones de esta función permiten añadir una nueva cuenta o actualizar una cuenta ya existente. La función ejecuta la correspondiente consulta para guardar los detalles de la cuenta.

Después de almacenar los detalles de la cuenta, volvemos a `index.php`, a la fase del cuerpo principal:

```
case 'store-settings' :
case 'account-setup' :
case 'delete-account' :
{
   display_account_setup($HTTP_SESSION_VARS['auth_user']);
   break;
}
```

Tras ello, ejecutamos la función `display_account_setup()` como antes para enumerar los detalles de la cuenta del usuario. Seguidamente, se añadirá la nueva cuenta.

Modificación de una cuenta existente

El proceso de modificación de una cuenta existente es muy similar. El usuario puede cambiar los detalles de la cuenta y pulsar el botón **Save Changes**. De nuevo, se desencadenará la acción `store-settings` pero, en esta ocasión, actualizará los detalles de la cuenta en lugar de añadirlos.

Eliminación de una cuenta

Para eliminar una cuenta, el usuario puede pulsar el botón **Delete Account** que se encuentra debajo de cada listado de cuenta. De esta forma se activa la función `delete-account`.

En la sección de preprocesamiento de la secuencia de comandos `index.php`, ejecutamos el siguiente código:

```
case 'delete-account' :
{
   delete_account($HTTP_SESSION_VARS['auth_user'], $account);
   break;
}
```

Este código invoca la función `delete_account()`. El código de esta función se recoge en el listado 27.6. Es necesario controlar la eliminación de cuentas antes del encabezado, ya que la opción de qué cuenta se va a utilizar se encuentra dentro del encabezado. También será necesario actualizar la lista de cuentas antes de poderlo dibujar correctamente.

Listado 27.6. Función delete_account() de mail_fns.php. Función para eliminar los detalles de una cuenta

```
function delete_account($auth_user, $accountid)
{
   //elimine una de las cuentas de este usuario de la base de datos

   $query = "delete from accounts where accountid='$accountid'
             and username ='$auth_user'";
   if(db_connect())
   {
      $result = mysql_query($query);
   }
   return $result;
}
```

Después de que la ejecución vuelve a `index.php`, la fase del cuerpo ejecuta el siguiente código:

```
case 'store-settings' :
case 'account-setup' :
case 'delete-account' :
```

```
{
   display_account_setup($HTTP_SESSION_VARS['auth_user']);
   break;
}
```

Comprobará que este código es el mismo que ejecutamos anteriormente; simplemente muestra la lista de las cuentas del usuario.

Lectura de correo

Después de que el usuario ha configurado algunas cuentas, podemos pasar a la parte principal, la conexión a dichas cuentas y la lectura del correo.

Selección de una cuenta

Tendremos que seleccionar una de las cuentas del usuario para leer el correo de la misma. La cuenta actualmente seleccionada se almacena en la variable de sesión `$selected_account`.

Si el usuario sólo tiene una cuenta registrada en el sistema, se seleccionará automáticamente cuando se conecte, como se indica a continuación:

```
if(number_of_accounts($HTTP_SESSION_VARS['auth_user'])==1)
{
   $accounts = get_account_list($HTTP_SESSION_VARS['auth_user']);
   $HTTP_SESSION_VARS['selected_account'] = $accounts[0];
}
```

La función `number_of_accounts()`, de `mail_fns.php`, se utiliza para determinar si el usuario tiene más de una cuenta. La función `get_account_list()` recupera una matriz de los nombres de las cuentas del usuario. En este caso sólo hay una, por lo que podemos acceder a la misma como el valor `0` de la matriz.

La función `number_of_accounts()` se muestra en el listado 27.7.

Listado 27.7. Función number_of_accounts de mail_fns.php. Función para determinar el número de cuentas que ha registrado un usuario

```
function number_of_accounts($auth_user)
{
   // obtenga el número de cuentas de este usuario

   $query = "select count(*) from accounts where username = '$auth_user'";

   if(db_connect())
   {
      $result = mysql_query($query);
         if($result)
            return mysql_result($result, 0, 0);
   }
```

```
    return 0;
}
```

La función `get_account_list()` es similar a la función `get_accounts()` que mencionamos anteriormente a excepción de que sólo recupera los nombres de las cuentas.

Si un usuario ha registrado varias cuentas, tendrá que seleccionar la que quiere utilizar.

En este caso, los encabezados contendrán una instrucción `SELECT` que enumera todos los buzones disponibles.

Al seleccionar el buzón apropiado, se muestra el correspondiente a esa cuenta, como puede apreciar en la figura 27.5.

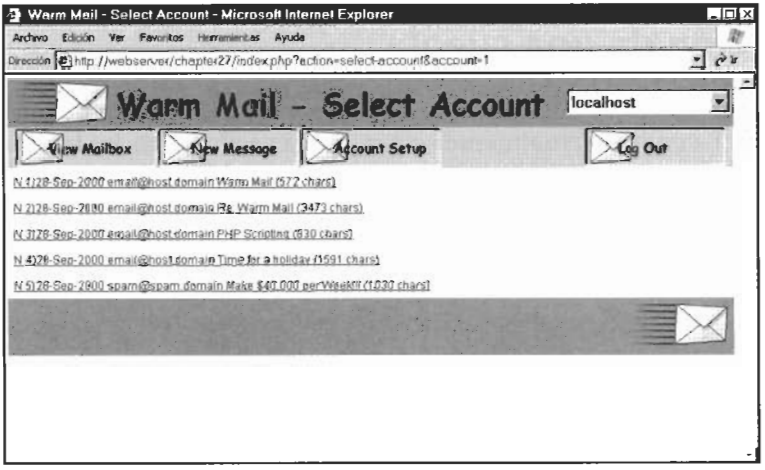


Figura 27.5. Al seleccionar la cuenta del equipo local en el cuadro SELECT, se descarga y se muestra el correo de dicha cuenta

Esta opción `SELECT` se genera en la función `do_html_header()` de `output_fns.php`, como se muestra en el siguiente fragmento de código:

```
// incluya el buzón de la cuenta seleccionada si el usuario tiene más de una cuenta
if(number_of_accounts($auth_user)>1)
{
    echo '<form target="index.php?action=open-mailbox" method="post">';
    echo '<td bgcolor="#ff6600" align="right" valign="middle">';
    display_account_select($auth_user, $selected_account);
    echo '</td>';
    echo '</form>';
}
```

Hemos evitado tener que analizar el HTML utilizado en los ejemplos del libro, pero el que genera la función `display_account_select()` merece una mención especial.

En función de las cuentas que tenga el usuario actual, `display_account_select()` generará un HTML como el que mostramos a continuación:

```
<select onchange="window.location=this.options[selectedIndex].value"
        name="account">
   <option value="0" selected>
      Choose Account</a>
   <option value="index.php?action=select-account&account=10">
      mail.domain.com
   </option>
   <option value="index.php?action=select-account&account=11">
      mail.server.com
   </option>
   <option value="index.php?action=select-account&account=9">
      localhost
   </option>
</select>
```

La mayor parte de este código es simplemente un elemento de selección HTML pero también incluye cierta cantidad de JavaScript. Del mismo modo que PHP puede generar HTML, también puede generar secuencias de comandos del lado del cliente.

Siempre que se produzca un evento de cambio en este elemento, JavaScript define `window.location` con el valor de la opción. Si el usuario selecciona la primera opción, `window.location` se define como `'index.php?action=select-account&account=10'`. Como resultado, se carga este URL. Evidentemente, si el navegador del usuario no es compatible con JavaScript o el usuario lo ha desactivado, el código no tendrá efecto alguno. La función `display_account_select()`, de `output_fns.php`, se utiliza para obtener la lista de cuentas disponibles y para mostrar la opción `SELECT`. También utiliza la función `get_account_list()` que describimos anteriormente. Al seleccionar una de las opciones de `SELECT` se activa el evento `select_account`. Si se fija en el URL de la figura 27.5, comprobará que se ha adjuntado al final del URL, junto con el Id. de la cuenta seleccionada.

Esto tiene dos efectos. En primer lugar, en la fase de preprocesamiento de `index.php`, la cuenta seleccionada se almacena en la variable de sesión `$selected_account`, como se indica a continuación:

```
case 'select-account' :
{
   // si se ha seleccionado una cuenta válida, almacénela como variable de
   // sesión
   if($account&&account_exists($HTTP_SESSION_VARS['auth_user'],
$account))
   {
      $HTTP_SESSION_VARS['selected_account'] = $account;
   }
}
```

Por otra parte, al ejecutar la fase de cuerpo de la secuencia de comandos, se ejecuta el siguiente código:

```
case 'select-account' :
case  'view-mailbox'  :
```

```
{
   // si se acaba de seleccionar mailbox o view mailbox, muestre el buzón
   display_list($HTTP_SESSION_VARS['auth_user'],
         $HTTP_SESSION_VARS['selected_account']);
   break;
}
```

Como puede comprobar, se realiza la misma acción que si el usuario hubiera seleccionado la opción **View Mailbox**. Lo veremos a continuación.

Cómo ver los contenidos del buzón de correo

Los contenidos del buzón de correo se pueden ver por medio de la función `display_list()`. De esta forma se mostrará una lista de todos los mensajes del buzón de correo. El código de esta función se incluye en el listado 27.8.

Listado 27.8. Función display_list() de output_fns.php. Función para mostrar todos los mensajes del buzón

```
function  display_list($auth_user,  $accountid)
{
   // muestre la lista de mensajes de este buzón

   global $table_width;

   if(!$accountid)
   {
      echo 'No mailbox selected<br /><br /><br /><br /><br /><br />.';
   }
   else
   {

      $imap = open_mailbox($auth_user, $accountid);

      if($imap)
      {
         echo "<table width=$table_width' cellspacing='0'
               cellpadding='6'  border='0'>";

         $headers = imap_headers($imap);
         // podríamos volver a aplicar formato a estos datos u obtener otros
         // detalles por
         // medio de
         // imap_fetchheaders, pero como no es un mal resumen, simplemente
         // duplicaremos cada uno

         $messages = sizeof($headers);
         for($i  =  0;  $i<$messages;  $i++)
         {
            echo '<tr><td bgcolor = "';
            if($i%2)
               echo  '#ffffff';
            else
```

```
            echo '#ffffcc';
         echo '"><a href="index.php?action=view-
            message&messageid='.($i+1).'">';
         echo $headers[$i];
         echo "</a></td></tr>\n";
      }
      echo  '</table>';
   }
   else
   {
      $account  =  get_account_settings($auth_user,  $accountid);
      echo 'could not open mail box '.$account['server'].
         '.<br /><br /><br /><br />';
   }
}
}
```

En esta función empezamos a utilizar las funciones IMAP de PHP. Las dos partes principales de esta función son la apertura del correo y la lectura de los encabezados de mensaje.

Para abrir el buzón de la cuenta de un usuario se invoca la función `open_mailbox()` que hemos escrito en `mail_fns.php`. Esta función se muestra en el listado 27.9.

Listado 27.9. Función open_mailbox() de mail_fns.php. Esta función se conecta al buzón de un usuario

```
function open_mailbox($auth_user, $accountid)
{
   global $HTTP_SESSION_VARS;

   // seleccione el buzón si sólo hay uno
   if(number_of_accounts($auth_user)==1)
   {
      $accounts = get_account_list($auth_user);
      $HTTP_SESSION_VARS['selected_account'] = $accounts[0];
      $accountid = $accounts[0];
   }

   // realice la conexión al servidor POP3 o IMAP que haya seleccionado el usuario
   $settings = get_account_settings($auth_user, $accountid);
   if(!sizeof($settings)) return 0;
   $mailbox = '{'.$settings[server];
   if($settings[type]=='POP3')
      $mailbox .= '/pop3';

   $mailbox .= ':'.$settings[port].'}INBOX';

   // elimine la advertencia y no olvide comprobar el valor devuelto
@  $imap  =  imap_open($mailbox,  $settings['remoteuser'],
      $settings['remotepassword']);

   return   $imap;
}
```

De hecho hemos abierto el buzón con la función `imap_open()`, que tiene el siguiente prototipo:

```
int imap_open (string buzón, string nombre de usuario, string contraseña [,
int opciones])
```

Los parámetros que debemos pasar a la función son los siguientes:

- `buzón`. Esta cadena debe incluir el nombre del servidor y el nombre del buzón y, opcionalmente, un número de puerto y un protocolo. El formato de la cadena es

  ```
  {nombre del servidor/protocolo:puerto}nombre del buzón
  ```

  Si no especificamos un protocolo, se utiliza de forma predeterminada IMAP. En el código que hemos escrito, comprobará que especificamos POP3 si el usuario ha especificado dicho protocolo para una determinada cuenta.

  Por ejemplo, para leer correo electrónico desde el equipo local, utilizando los puertos predeterminados, podríamos utilizar el siguiente nombre de buzón para IMAP:

  ```
  {localhost:143}INBOX
  ```

  y este otro para POP3:

  ```
  {localhost/pop3:110}INBOX
  ```

- `nombre de usuario`. El nombre de usuario de la cuenta
- `contraseña`. La contraseña de la cuenta

También podemos pasar indicadores opcionales para especificar opciones como "abrir el buzón en modo de sólo lectura".

Conviene mencionar que hemos construido la cadena de buzón paso a paso con el operador de concatenación antes de pasarla a `imap_open()`. Tendrá que prestar especial atención a la forma de construir esta cadena ya que las cadenas que contienen `{ $` generan problemas en PHP 4.

La invocación de esta función devuelve un flujo IMAP si el buzón se puede abrir y false si no se puede abrir.

Al terminar con un flujo IMAP, se puede cerrar por medio de `imap_close(flujo_imap)`.

En nuestra función, el flujo IMAP se pasa de nuevo al programa principal. Tras ello utilizamos la función `imap_headers()` para obtener los encabezados de correo y representarlos:

```
$headers = imap_headers($imap);
```

Esta función devuelve información de encabezado para todos los mensajes de correo del buzón al que nos hayamos conectado. La información se devuelve en forma de matriz, una línea por mensaje. No hemos aplicado ningún formato. Simplemente

devuelve una línea por mensaje, por lo que si se fija en la figura 27.5 podrá ver el resultado.

Se puede obtener más información sobre encabezados de correo si utiliza `imap_header()`, del mismo nombre pero confusa. En este caso, la función `imap_headers()` nos ofrece suficiente información para nuestros objetivos.

Lectura de un mensaje de correo

Hemos configurado todos los mensajes de la función `display_list()` anterior para vincularlos a determinados mensajes de correo electrónico.

Cada uno de estos vínculos tiene la siguiente forma:

```
index.php?action=view-message&messageid=6
```

`messageid` es el número de secuencia utilizado en los encabezados que recuperamos anteriormente. Comprobará que los mensajes IMAP están numerados a partir del 1, no del 0.

Si el usuario pulsa sobre uno de estos enlaces, verá el resultado que mostramos en la figura 27.6.

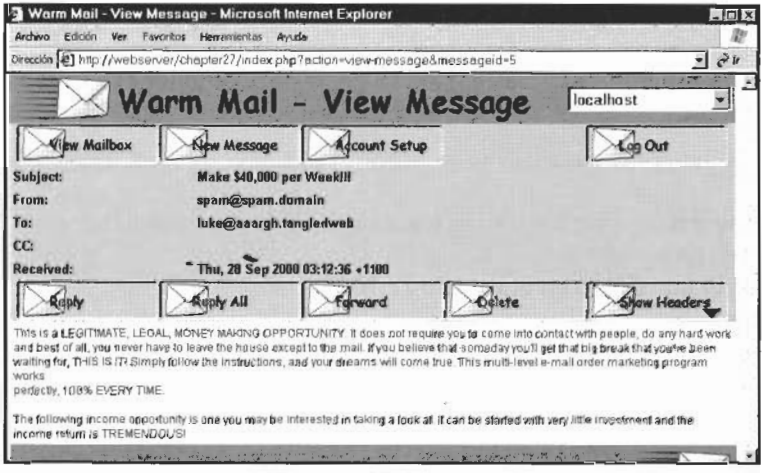


Figura 27.6. Al utilizar la acción view-message podemos ver un determinado mensaje, en este caso, un correo basura

Al introducir estos parámetros en la secuencia de comandos `index.php`, ejecutamos el siguiente código:

```
case 'show-headers' :
case 'hide-headers' :
case 'view-message' :
{
   // si acabamos de obtener un mensaje de la lista o estamos analizando
```

```
   // un mensaje y optamos por ocultar o ver encabezados, cargue un mensaje
   $fullheaders = ($action=='show-headers');
   display_message ($HTTP_SESSION_VARS['auth_user'],
                    $HTTP_SESSION_VARS['selected_account'],
                    $messageid, $fullheaders);
   break;
}
```

Apreciará que se comprueba si el valor de `$action` es igual a `'show-headers'`.

En este caso, será `false`, y `$fullheaders` también se definirá como `false`. Describiremos la acción `'show-headers'` en breve.

La línea

```
$fullheaders = ($action=='show-headers');
```

se podría escribir de forma más clara como

```
if($action=='show-headers')
   $fullheaders  =  true;
else
   $fullheaders  =  false;
```

Tras ello, invocamos la función `display_message()`.

Básicamente representa código HTML sencillo, por lo que no la describiremos.

Invoca la función `retrieve_message()` para obtener el correspondiente mensaje del buzón:

```
$message = retrieve_message($auth_user, $accountid, $messageid, $fullheaders);
```

La función `retrieve_message()` se encuentra en la biblioteca `mail_fns.php`.

Puede ver su código en el listado 27.10.

Listado 27.10. Función retrieve_message() de mail_fns.php. Esta función recupera un mensaje concreto de un buzón

```
function retrieve_message($auth_user, $accountid, $messageid, $fullheaders)
{
   $message = array();

   if(!($auth_user && $messageid && $accountid))
      return false;

   $imap = open_mailbox($auth_user, $accountid);
   if(!$imap)
      return false;

   $header = imap_header($imap, $messageid);

   if(!$header)
```

```
        return false;

    $message['body'] = imap_body($imap, $messageid);
    if(!$message['body'])
        $message['body'] = "[This message has no body]\n\n\n\n\n\n";

    if($fullheaders)
        $message['fullheaders'] = imap_fetchheader($imap, $messageid);
    else
        $message['fullheaders'] = '';

    $message['subject'] = $header->subject;
    $message['fromaddress']  =      $header->fromaddress;
    $message['toaddress'] =   $header->toaddress;
    $message['ccaddress'] =   $header->ccaddress;
    $message['date'] =   $header->date;

    // fíjese en que podemos obtener información más detallada si utilizamos from y to
    // en lugar de fromaddress y toaddress, pero éstos son más sencillos

    imap_close($imap);
    return $message;
}
```

Como en el caso anterior, hemos utilizado `open_mailbox()` para abrir el buzón del usuario.

Sin embargo, en esta ocasión, buscamos un mensaje concreto. Por medio de esta biblioteca de funciones descargamos de forma separada los encabezados del mensaje y el cuerpo del mismo.

Las tres funciones IMAP que hemos utilizado son `imap_header()`, `imap_fetchheader()` e `imap_body()`.

Las dos funciones de encabezado son distintas a `imap_headers()`, la que utilizamos previamente.

Los nombres pueden resultarle confusos.

En resumen:

- `imap_headers()`. Devuelve un resumen de los encabezados de todos los mensajes de un buzón. Los devuelve en forma de matriz con un elemento por mensaje.
- `imap_header()`. Devuelve los encabezados de un mensaje concreto en forma de objeto.
- `imap_fetchheader()`. Devuelve los encabezados de un mensaje concreto en forma de cadena.

En este caso utilizamos `imap_header()` para completar campos de encabezado específicos e `imap_header()` para mostrar los encabezados completos al usuario en caso de que lo solicite (como veremos en breve).

Utilizamos `imap_header()` e `imap_body()` para generar una matriz que contenga todos los elementos del mensaje en el que estemos interesados.

Invocamos `imap_header()` de la siguiente forma:

```
$header = imap_header($imap, $messageid);
```

Tras ello podemos extraer del objeto todos los campos que nos interesen:

```
$message['subject'] = $header->subject;
```

Invocamos `imap_body()` para añadir el cuerpo del mensaje a la matriz:

```
$message['body'] = imap_body($imap, $messageid);
```

Por último, cerramos el buzón con `imap_close()` y devolvemos la matriz que hemos generado. De esta forma, la función `display_message()` puede mostrar los campos del mensaje como se indica en la figura 27.6.

Cómo ver encabezados de mensaje

Como puede comprobar en la figura 27.6, hay un botón **Show Headers**. Este botón activa la opción `show-headers`, que añade todos los encabezados de mensaje a la pantalla del mensaje. Si el usuario pulsa este botón, verá un resultado similar al de la figura 27.7.

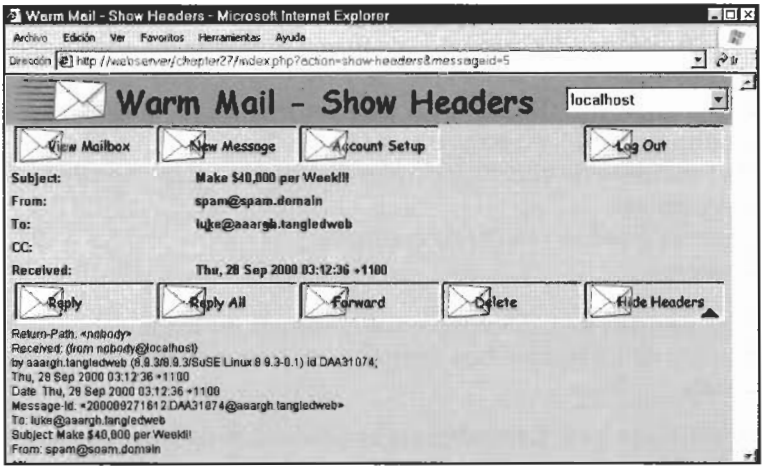


Figura 27.7. El uso de show-headers para ver los encabezados completos de este mensaje permite al usuario localizar el origen del correo basura

Como habrá comprobado, el control de eventos de `view-message` también engloba `show-headers` (y su antónimo `hide-headers`). Si se selecciona esta opción, realizamos las mismas acciones que en casos anteriores. Pero en `retrieve-messages()` también obtenemos el texto completo de los encabezados, como se indica a continuación:

```
if($full_headers)
```

```
$message['fullheaders'] = imap_fetchheader($imap, $messageid);
```

Tras ello podemos mostrar estos encabezados al usuario.

Eliminación de correo

Si un usuario pulsa el botón **Delete** de un determinado correo electrónico, activará la acción `"delete"`, que ejecuta el siguiente código de `index.php`:

```
case 'delete' :
{
   delete_message($HTTP_SESSION_VARS['auth_user'],
                  $HTTP_SESSION_VARS['selected_account'], $messageid);
   //fíjese en que no hay 'salto' - continuaremos con el siguiente caso
}
case  'select-account'  :
case  'view-mailbox'  :
{
   // si se selecciona mailbox o view mailbox, muestre el buzón
   display_list( $HTTP_SESSION_VARS['auth_user'],
                 $HTTP_SESSION_VARS['selected_account']);
   break;
}
```

Como puede comprobar, el mensaje se borra por medio de la función `delete_message()` y, tras ello, se muestra el buzón resultante como mencionamos anteriormente.

El código de la función `delete_message()` se muestra en el listado 27.11.

Listado 27.11. Función delete_message de mail_fns.php. Esta función borra un mensaje concreto de un buzón

```
function  delete_message($auth_user,  $accountid,  $message_id)
{
   // borre un solo mensaje del servidor

   $imap  =  open_mailbox($auth_user,  $accountid);
   if($imap)
   {
      imap_delete($imap, $message_id);
      imap_expunge($imap);
      imap_close($imap);
      return true;
   }
   return false;
}
```

Como puede apreciar, esta función utiliza una serie de funciones IMAP. Las nuevas son `imap_delete()` e `imap_expunge()`. `imap_delete()` solamente marca mensajes para su eliminación.

Puede marcar todos los mensajes que desee. La invocación de `imap_expunge()` elimina realmente los mensajes.

Envío de correo

Por último llegamos a la parte de envío del correo. Por medio de esta secuencia de comandos, el envío se puede realizar de varias formas: el usuario puede enviar un nuevo mensaje, contestarlo o reenviarlo. Veamos cómo funciona.

Envío de un nuevo mensaje

El usuario puede seleccionar esta opción si pulsa el botón **New Message**. Se activará la acción `'new-message'`, que ejecuta el siguiente código en `index.php`:

```
case 'new-message' :
{
   display_new_message_form($HTTP_SESSION_VARS['auth_user'],
                            $to, $cc, $subject, $body);
   break;
}
```

El formulario de nuevo mensaje es simplemente un formulario para enviar correo. Puede ver su aspecto en la figura 27.8. En realidad, en esta figura se muestra el reenvío de un mensaje, no el envío de un nuevo correo, pero el formulario es el mismo. En apartados posteriores veremos cómo se responde y se reenvía correo.

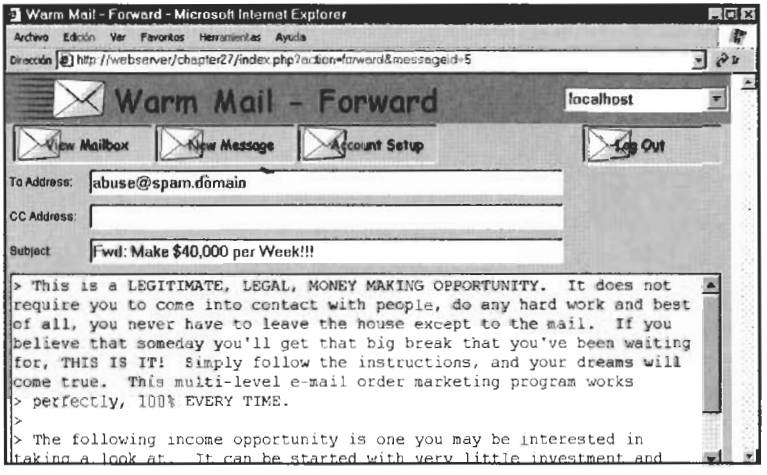


Figura 27.8. Al utilizar el reenvío de correo podemos informar del correo basura

Al pulsar el botón **Send Message** se invoca la acción `"send-message"` y se ejecuta el siguiente código:

```
case 'send-message' :
{
```

```
    if(send_message($to, $cc, $subject, $message))
       echo "<p>Message sent.</p><br /><br /><br /><br /><br /><br />";
    else
       echo "<p>Could not send message.</p><br /><br /><br /><br /><br /><br />";
    break;
}
```

Este código invoca la función `send_message()`, que se encarga de enviar el correo.

El código de esta función se recoge en el listado 27.12.

Listado 27.12. Función send_message() de mail_fns.php. Esta función envía el mensaje que haya escrito el usuario

```
function send_message($to, $cc, $subject, $message)
{
   //envíe un correo a través de PHP
   global $HTTP_SESSION_VARS;

   if (!db_connect())
   {
      return false;
   }
   $query ='select address from users where username=\''.
           $HTTP_SESSION_VARS['auth_user']."'";

   $result = mysql_query($query);
   if (!$result)
   {
      return false;
   }
   else if (mysql_num_rows($result)==0)
   {
      return false;
   }
   else
   {
      $other = 'From: '.mysql_result($result, 0, 'address')."\r\nCc: $cc";
      if (mail($to, $subject, $message, $other))
      {
         return true;
      }
      else
      {
         return false;
      }
   }
}
```

Como puede comprobar, esta función utiliza `mail()` para enviar el correo electrónico.

Sin embargo, primero carga la dirección de correo del usuario desde la base de datos para utilizarla en el campo **From** del mensaje.

Cómo responder o reenviar correo

Las funciones Reply, Reply All y Forward envían correo de la misma forma que New Message. La diferencia de funcionamiento radica en que completan las partes del formulario de nuevo mensaje antes de mostrárselo al usuario. Volvamos a la figura 27.8. El mensaje que vamos a reenviar se sangra con el símbolo > y a la línea **Subject** se le ha añadido **To** como prefijo. Del mismo modo, las opciones **Reply** y **Reply All** se completan con los datos de los receptores, la línea del asunto y el mensaje.

El código para realizar esta operación se activa en la sección del cuerpo de `index.php`, como se indica a continuación:

```
case  'reply-all'  :
{
   //establezca cc como línea cc antigua
   if(!$imap)
      $imap = open_mailbox( $HTTP_SESSION_VARS['auth_user'],
                            $HTTP_SESSION_VARS['selected_account']);
   if($imap)
   {
      $header = imap_header($imap, $messageid);
      if($header->reply_toaddress)
         $to = $header->reply_toaddress;
      else
         $to = $header->fromaddress;
      $cc = $header->ccaddress;
      $subject = 'Re: '.$header->subject;
      $body = add_quoting(stripslashes(imap_body($imap, $messageid)));
      imap_close($imap);

      display_new_message_form( $HTTP_SESSION_VARS['auth_user'],
                                $to, $cc, $subject, $body);
   }
   break;
}
case 'reply' :
{
   //defina address como reply-to o from del mensaje actual
   if(!$imap)
      $imap = open_mailbox($HTTP_SESSION_VARS['auth_user'],
                           $HTTP_SESSION_VARS['selected_account']);
   if($imap)
   {
      $header = imap_header($imap, $messageid);
      if($header->reply_toaddress)
         $to  =  $header->reply_toaddress;
      else
         $to = $header->fromaddress;
      $subject = 'Re: '.$header->subject;
      $body = add_quoting(stripslashes(imap_body($imap, $messageid)));
      imap_close($imap);

      display_new_message_form($HTTP_SESSION_VARS['auth_user'],
```

```
                                $to, $cc, $subject, $body);
    }

    break;
}
case 'forward' :
{
    //defina el mensaje como cuerpo del mensaje actual
    if(!$imap)
        $imap = open_mailbox( $HTTP_SESSION_VARS['auth_user'],
                              $HTTP_SESSION_VARS['selected_account']);
    if($imap)
    {
        $header = imap_header($imap, $messageid);
        $body = add_quoting(stripslashes(imap_body($imap, $messageid)));
        $subject = 'Fwd: '.$header->subject;
        imap_close($imap);

        display_new_message_form( $HTTP_SESSION_VARS['auth_user'],
                                  $to, $cc, $subject, $body);
    }
    break;
}
```

Puede comprobar que cada una de estas opciones define los correspondientes encabezados, aplica un formato en caso de que sea necesario e invoca la función `display_new_message_form()` para configurar el formulario.

Con esto hemos completado la funcionalidad de nuestro lector de correo Web.

Ampliación del proyecto

En este proyecto podemos realizar numerosas ampliaciones y mejoras. Puede utilizar como referencia el lector de correo que utilice normalmente, pero entre algunas de las posibles mejoras destacamos las siguientes:

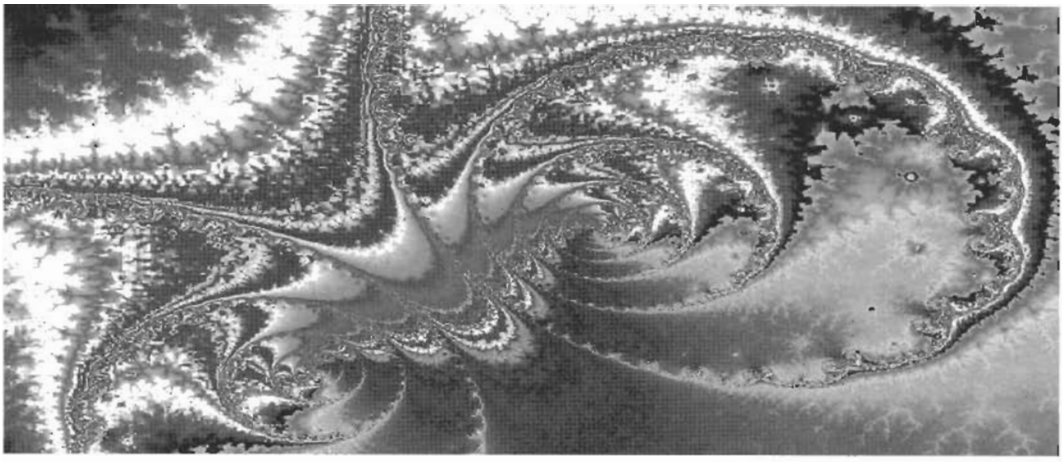
- Permitir que los usuarios se registren en el sitio (puede reutilizar parte del código de capítulos anteriores sobre autenticación y personalización de usuarios).
- Muchos usuarios tienen más de una dirección de correo electrónico, por ejemplo una personal y otra de trabajo. Al cambiar sus direcciones de correo almacenadas de la tabla de usuarios a la tabla de cuentas, les permitirá utilizar varias direcciones y bastaría con cambiar una pequeña cantidad de código. El formulario de envío de correo debería incluir un cuadro desplegable para seleccionar la dirección que se va a utilizar.
- Se podría permitir enviar, recibir y ver correo con archivos adjuntos. Si los usuarios pudieran enviar archivos adjuntos, tendríamos que incorporar op-

ciones de carga de archivos como las descritas en un capítulo anterior. El envío de correo con archivos adjuntos se analizará en un capítulo posterior.

- Se podrían añadir opciones de libreta de direcciones.
- Se podrían añadir opciones de lectura de noticias de red. La lectura desde un servidor NNTP por medio de funciones IMAP es idéntica a la lectura desde un buzón. Basta con especificar un número de puerto y un protocolo diferentes en la invocación de `imap_open()`. En lugar de asignar un nombre como BANDEJA DE ENTRADA a un buzón, se le asigna un grupo de noticias desde el que se realiza la lectura. Podría combinarlo con opciones de generación de cadenas del proyecto de uno de los siguientes capítulos para crear un lector de noticias basado en subprocesos Web.

A continuación

En el siguiente capítulo crearemos otro proyecto relacionado con el correo electrónico. En esta ocasión diseñaremos una aplicación que admita el envío de boletines informativos sobre diferentes temas a los usuarios suscritos a nuestro sitio.



28
Creación de un gestor de listas de correo

Después de diseñar una base de suscriptores a nuestro sitio Web, una opción muy acertada consistiría en mantenernos en contacto con ellos y enviarles un boletín informativo. En este capítulo implementaremos una interfaz para un gestor de listas de correo (o MLM). Algunos MLM permiten que los suscriptores puedan enviar mensajes a otros suscriptores. Nuestro programa será un sistema de boletines informativos, en el que solamente el administrador podrá enviar mensajes. El nombre de nuestro sistema será Pyramid-MLM.

El sistema será similar a muchos de los que ya existen en el mercado. Para hacerse una idea de lo que pretendemos puede visitar `http://www.topica.com`.

Nuestra aplicación permitirá al administrador crear varias listas de correo y enviar boletines informativos a cada una de ellas individualmente. Esta aplicación utilizará la carga de archivos para que el administrador pueda cargar versiones en texto y HTML de los boletines que haya creado. Esto significa que los administradores pueden utilizar cualquier programa para crear los boletines.

Los usuarios podrán suscribirse a cualquiera de las listas de nuestro sitio e indicar si quieren recibir los boletines informativos en texto o en HTML.

El problema

Queremos diseñar un sistema en línea de creación y envío de boletines informativos. Este sistema permitirá crear y enviar distintos boletines a los usuarios y que los usuarios se suscriban a uno o a varios de estos boletines.

En concreto, los requisitos de este sistema son los siguientes:

- A los administradores se les debe permitir definir y modificar los boletines informativos.
- A los administradores se les debe permitir enviar boletines en texto y en HTML a todos los suscriptores de una lista de correo.
- A los usuarios se les debe permitir registrarse para utilizar el sitio, así como introducir y modificar sus datos.
- A los usuarios se les debe permitir suscribirse a cualquiera de las listas del sitio.
- A los usuarios se les debe permitir anular la suscripción a las listas a las que estén suscritos.
- A los usuarios se les debe permitir almacenar su preferencia con respecto a boletines informativos en formato HTML o en texto.
- Por motivos de seguridad, a los usuarios no se les debe permitir enviar correo a las listas o ver las direcciones de correo electrónico de otros usuarios.
- A los usuarios y administradores se les debe permitir ver información sobre las listas de correo.
- A los usuarios y administradores se les debe permitir ver boletines informativos atrasados que se hayan enviado a una lista (el archivo).

Componentes de la solución

Para cumplir los requisitos anteriores, necesitaremos una serie de componentes. Los principales consisten en definir una base de datos de listas, suscriptores y boletines informativos archivados; la carga de los boletines creados fuera del sistema en línea y el envío de correo con archivos adjuntos.

Definición de una base de datos de listas y suscriptores

Realizaremos el seguimiento del nombre de usuario y la contraseña de todos los usuarios del sistema, así como de una lista de las listas a las que se hayan suscrito. También almacenaremos la preferencia de cada usuario para recibir correo electrónico en HTML o en texto, para poder enviarle la versión adecuada del boletín informativo.

Un administrador será un usuario especializado con capacidad para crear nuevas listas de correo y para enviar boletines informativos a dichas listas.

Sería aconsejable dotar al sistema de un archivo de boletines informativos anteriores. Puede que los suscriptores no guarden boletines pasados pero que les interese buscar algo en concreto. También se puede utilizar el archivo como herramienta de marketing de los boletines ya que los suscriptores potenciales pueden ver el aspecto de los mismos.

La definición de una base de datos en MySQL y la creación de una interfaz para la misma en PHP no resultará difícil ni es nada nuevo.

Carga de archivos

Necesitamos una interfaz para permitir al administrador enviar boletines de noticias, como indicamos previamente. Sin embargo no hemos mencionado cómo los crearán. Podríamos proporcionarles un formulario en el que puedan redactar o pegar el contenido del boletín informativo. No obstante, si permitimos que los administradores creen el boletín en su editor de texto favorito y que carguen el archivo al servidor Web, aumentaremos el atractivo del sistema.

Para ello podemos utilizar las opciones de carga de archivos que analizamos en un capítulo anterior.

Será necesario utilizar un formulario más complicado que el que hemos empleado previamente. El administrador tendrá que cargar versiones del boletín informativo tanto en texto como en HTML, junto con las imágenes que quiera incluir en HTML.

Una vez cargado el boletín, tendremos que crear una interfaz para que el administrador pueda ver una vista previa del mismo antes de enviarlo. De esta forma, puede confirmar que todos los archivos se cargan correctamente.

Debe saber que almacenaremos todos estos archivos en un directorio para que los usuarios puedan leer ejemplares atrasados de los boletines informativos. Es necesario que el usuario bajo el que se ejecute su servidor Web pueda escribir en este archivo.

La secuencia de comandos de carga intentará escribir los boletines informativos en `./archivo/` por lo que debe crear este directorio y definir correctamente los permisos del mismo.

Envío de correo con archivos adjuntos

En este proyecto, queremos enviar a los usuarios un boletín informativo en texto sencillo o en una "atractiva" versión en HTML, en función de lo que prefieran.

Para enviar un archivo HTML con imágenes incrustadas, tendremos que buscar la forma de mandar archivos adjuntos. La sencilla función `mail()` de PHP no admite el envío de archivos adjuntos. En su lugar, utilizaremos el magnífico paquete Mail_Mime de PEAR, creado por Richard Heyes, que permite procesar archivos adjuntos HTML y también se puede utilizar para adjuntar cualquier imagen incluida en el archivo HTML.

Las instrucciones de instalación de este paquete se incluyen en el apéndice A.

Presentación de la solución

En este proyecto, utilizaremos de nuevo un enfoque dirigido por eventos para escribir nuestro código, como hicimos en el capítulo anterior.

En primer lugar diseñamos una serie de diagramas de flujo de sistema para mostrar las rutas que los usuarios pueden tomar a lo largo del sistema. En este caso, hemos diseñado tres diagramas para representar los tres conjuntos de interacciones que los usuarios pueden realizar con el sistema. Los usuarios dispondrán de distintas opciones en función de si no están conectados, de si están conectados como usuarios convencionales o de si están conectados como administradores. En las figuras 28.1, 28.2 y 28.3 se describen cada una de estas acciones.

En la figura 28.1 se muestran las acciones que puede realizar un usuario si no está conectado. Como puede comprobar, el usuario se puede conectar (si ya dispone de una cuenta), crear una cuenta (en caso que no tenga una) o ver las listas de correo a las que se puede suscribir (como táctica de marketing).

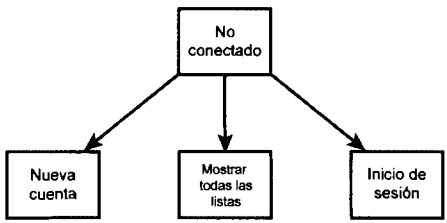


Figura 28.1. Cuando un usuario no está conectado, solamente puede seleccionar un número limitado de acciones

En la figura 28.2 vemos la acción que puede realizar un usuario una vez conectado. Puede cambiar la configuración de su cuenta (dirección de correo electrónico y preferencias), cambiar su contraseña y cambiar las listas a las que está suscrito.

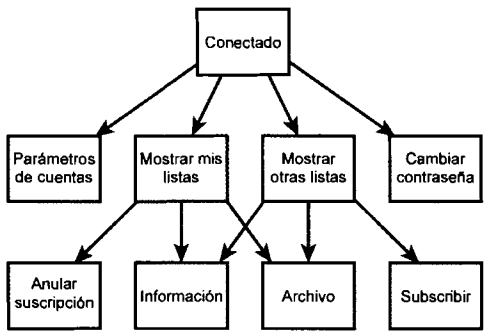


Figura 28.2. Una vez conectados, los usuarios pueden cambiar sus preferencias por medio de distintas opciones

En la figura 28.3 se muestran las acciones disponibles cuando se conecta un administrador. Como puede comprobar, un administrador accede prácticamente a la misma funcionalidad que un usuario, así como a opciones adicionales. También puede crear nuevas listas de correo, crear nuevos mensajes para una lista de correo si carga archivos y previsualizar mensajes antes de enviarlos.

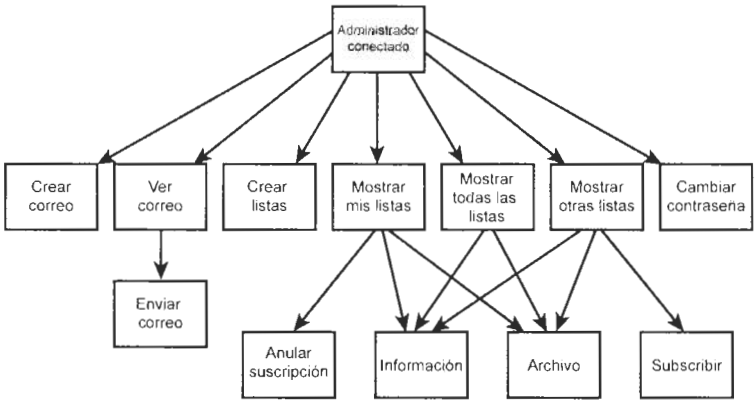


Figura 28.3. Los administradores disponen de acciones adicionales

Como hemos recurrido a un enfoque dirigido por eventos, la parte principal de la aplicación se incluye en un solo archivo, `index.php`, que invoca un conjunto de bibliotecas de funciones. En la tabla 28.1 se incluye un resumen de los archivos de esta aplicación.

Tabla 28.1. Archivos de la aplicación de gestión de listas de correo

Nombre de archivo	Tipo	Descripción
`index.php`	Aplicación	Secuencia de comandos principal que ejecuta toda la aplicación.
`include_fns.php`	Funciones	Colección de archivos de inclusión de esta aplicación.
`data_valid_fns.php`	Funciones	Colección de funciones para validar datos introducidos.
`db_fns.php`	Funciones	Colección de funciones para realizar la conexión a la base de datos `mlm`.
`mlm_fns.php`	Funciones	Colección de funciones específicas de esta aplicación.
`output_fns.php`	Funciones	Colección de funciones para re presentar HTML.

Nombre de archivo	Tipo	Descripción
upload.php	Componente	Secuencia de comandos que gestiona el componente de carga de archivos del administrador. Está separado para facilitar la seguridad.
user_auth_fns.php	Funciones	Colección de funciones para autenticar usuarios
create_database.sql	SQL	SQL para definir la base de datos mlm y para configurar un usuario Web y un usuario administrativo.

A continuación describiremos la implementación del proyecto, comenzado con la base de datos en la que almacenaremos información sobre suscriptores y listas.

Definición de la base de datos

En esta aplicación, tendremos que almacenar detalles de:

- Listas: listas de correo a las que se puede realizar la suscripción.
- Suscriptores: usuarios del sistema y sus preferencias.
- Listas de suscriptores: registro de qué usuarios se han suscrito a qué listas (una relación varios a varios).
- Correo: un registro de los mensajes de correo enviados.
- Imágenes: como queremos enviar mensajes de correo electrónico formados por varios archivos (es decir, texto y HTML más una serie de imágenes), también tendremos que saber qué imágenes se incluyen en cada correo electrónico.

El código SQL que hemos escrito para crear esta base de datos se recoge en el listado 28.1.

Listado 28.1. create_database.sql. SQL para crear la base de datos mlm

```
create database mlm;

use mlm;

create table lists
(
   listid int auto_increment not null primary key,
   listname char(20) not null,
   blurb varchar(255)
```

```
);

create  table  subscribers
(
   email  char(100)  not  null  primary  key,
   realname char(100) not null,
   mimetype  char(1)  not  null,
   password char(16) not null,
   admin tinyint not null
);

# almacena una relación entre un suscriptor y una lista
create table sub_lists
(
   email char(100) not null,
   listid int not null
);

create table mail
(
   mailid int auto_increment not null primary key,
   email char(100) not null,
   subject char(100) not null,
   listid int not null,
   status char(10) not null,
   sent datetime,
   modified timestamp
);

#almacena las imágenes que corresponden a un determinado correo
create table images
(
   mailid int not null,
   path char(100) not null,
   mimetype char(100) not null
);

grant  select,  insert,  update,  delete
on  mlm.*
to mlm@localhost identified by 'password';

insert into subscribers values

('admin@localhost', 'Administrative User', 'H', password('admin'), 1);
```

Recuerde que puede ejecutar este SQL si escribe

```
mysql -u root -p < create_database.sql
```

Tendrá que proporcionar su contraseña de usuario raíz (también podría ejecutar esta secuencia de comandos a través de cualquier usuario de MySQL que tenga los privilegios correctos; simplemente hemos utilizado el usuario raíz por motivos de simplicidad).

Debería cambiar la contraseña del usuario `mlm` y el administrador de su secuencia de comandos antes de ejecutarla.

Conviene explicar con mayor detalle alguno de los campos de la base de datos.

La tabla `lists` contiene un `listid` y un `listname`. También contiene un `blurb`, una descripción de qué es la lista.

La tabla `suscribers` contiene las direcciones de correo (`email`) y los nombres (`realname`) de los suscriptores. También almacena su contraseña (`password`) y un indicador (`admin`) para indicar si el usuario es un administrador o no. En `mimetype` almacenaremos también el tipo de correo que prefieren recibir. Puede tratarse de `H` para HTML o de `T` para texto.

La tabla `sublists` contiene direcciones de correo electrónico (`email`) de la tabla `suscribers` y `listid` de la tabla `lists`.

La tabla `mail` contiene información sobre todos los mensajes de correo que se envían a través del sistema.

Almacena un identificador exclusivo (`mailid`), la dirección desde la que se envía el correo (`email`), la línea de asunto (`subject`) y el `listid` de la lista a la que se ha enviado o la que se va a enviar. El texto o HTML del mensaje puede ser un archivo de gran tamaño, por lo que almacenaremos el archivo de los mensajes fuera de la base de datos.

También realizaremos el seguimiento de la información de estado general: si el mensaje se ha enviado o no (`status`), cuándo se ha enviado (`sent`) y una marca de tiempo para mostrar cuándo se ha modificado por última vez (`modified`).

Por último, utilizamos la tabla `images` para realizar el seguimiento de todas las imágenes asociadas a mensajes HTML. De nuevo, estas imágenes pueden tener un gran tamaño, por lo que las almacenaremos fuera de la base de datos. En su lugar, utilizaremos el `mailid` al que están asociadas, la ruta (`path`) a la ubicación en la que se encuentra almacenada la imagen y el tipo MIME de la imagen (`mimetype`), por ejemplo, `image/gif`.

El SQL que mostramos anteriormente también define un usuario para que se conecte PHP y un usuario administrativo para el sistema.

Arquitectura de la secuencia de comandos

Al igual que en el proyecto anterior, hemos empleado un enfoque dirigido por eventos. La parte central de la aplicación es el archivo `index.php`.

Esta secuencia de comandos tiene cuatro segmentos principales, que son los siguientes:

1. Preprocesamiento: es necesario realizar cualquier procesamiento antes de enviar los encabezados.
2. Configuración y definición de encabezados: crear y enviar el inicio de la página HTML.
3. Ejecución de la acción: responder al evento que se ha pasado. Como en el último ejemplo, el evento se incluye en la variable `$action`.

4. Envío de pies de página.

La práctica totalidad del procesamiento de la aplicación se lleva a cabo en este archivo.

La aplicación también utiliza las bibliotecas de funciones enumeradas en la tabla 28.1, como mencionamos anteriormente.

El código completo de la secuencia de comandos `index.php` se recoge en el listado 28.2.

Listado 28.2. index.php. Archivo principal de la aplicación Pyramid-MLM

```
<?php

/*************************************************************************
* Sección 1 : preprocesamiento
*************************************************************************/

   include ('include_fns.php');
   session_start();

   $action = $HTTP_GET_VARS['action'];
   $buttons = array();

   //se adjunta a esta cadena si hay contenidos antes de procesar el encabezado
   $status = '';

   // es necesario procesar antes que nada las solicitudes de conexión o desconexión
   if($HTTP_POST_VARS['email']&&$HTTP_POST_VARS['password'])
   {
      $login = login($HTTP_POST_VARS['email'], $HTTP_POST_VARS['password']);

      if($login == 'admin')
      {
         $status .= "<p><b>".get_real_name($HTTP_POST_VARS['email']).
            "</b> logged in"." successfully as <b>Administrator</b></p>
            <br /><br /><br /><br /><br />";
         $HTTP_SESSION_VARS['admin_user'] = $HTTP_POST_VARS['email'];
      }
      else if($login == 'normal')
      {
         $status .= "<p><b>".get_real_name($HTTP_POST_VARS['email'])."</b>
            logged in"
            ." successfully.</p><br /><br />";
         $HTTP_SESSION_VARS['normal_user'] = $HTTP_POST_VARS['email'];
      }
      else
      {
         $status .= "<p>Sorry, we could not log you in with that
               email address and password.</p><br />";
      }
   }

   if($action == 'log-out')
```

```
    {
        unset($action);
        unset($HTTP_SESSION_VARS);
        session_destroy();
    }

/*************************************************************************
* Sección 2: definición y configuración de encabezados
*************************************************************************/

    // defina los botones de la barra de herramientas
    if(check_normal_user())
    {
        // si se trata de un usuario normal
        $buttons[0] = 'change-password';
        $buttons[1] = 'account-settings';
        $buttons[2] = 'show-my-lists';
        $buttons[3] = 'show-other-lists';
        $buttons[4] = 'log-out';
    }
    else if(check_admin_user())
    {
        // si se trata de un administrador
        $buttons[0] = 'change-password';
        $buttons[1] = 'create-list';
        $buttons[2] = 'create-mail';
        $buttons[3] = 'view-mail';
        $buttons[4] = 'log-out';
        $buttons[5] = 'show-all-lists';
        $buttons[6] = 'show-my-lists';
        $buttons[7] = 'show-other-lists';
    }
    else
    {
        // si no se ha conectado
        $buttons[0] = 'new-account';
        $buttons[1] = 'show-all-lists';
        $buttons[4] = 'log-in';
    }

    if($action)
    {
        // muestre el encabezado con el nombre de la aplicación y una
        // descripción de la página
        // o acción do_html_header('Pyramid-MLM - '.format_action($action));
    }
    else
    {
        // muestre el encabezado sólo con el nombre de la aplicación
        do_html_header('Pyramid-MLM');
    }

    display_toolbar($buttons);

    //muestre cualquier texto generado por las funciones invocadas antes del encabezado
```

```
    echo $status;

/*************************************************************************
* Sección 3: ejecución de la acción
*************************************************************************/

    // sólo se pueden realizar estas acciones si no está conectado
    switch ( $action )
    {
       case 'new-account' :
       {
          // elimine las variables de sesión
          session_destroy();
          display_account_form();
          break;
       }
       case 'store-account' :
       {
          if (store_account($HTTP_SESSION_VARS['normal_user'],
                $HTTP_SESSION_VARS['admin_user'], $HTTP_POST_VARS))
             $action = '';
          if(!check_logged_in())
             display_login_form($action);
          break;
       }
       case 'log-in' :
       case '':
       {
          if(!check_logged_in())
             display_login_form($action);
          break;
       }
       case 'show-all-lists' :
       {
          display_items('All Lists', get_all_lists(), 'information',
                        'show-archive','');
          break;
       }
       case 'show-archive' :
       {
          display_items('Archive For '.get_list_name($HTTP_GET_VARS['id']),
                        get_archive($HTTP_GET_VARS['id']), 'view-html',
                        'view-text', '');
          break;
       }
       case 'information' :
       {
          display_information($HTTP_GET_VARS['id']);
          break;
       }

    }

    //todas las acciones restantes requieren que el usuario esté conectado
    if(check_logged_in())
    {
```

```
switch ( $action )
{
   case 'account-settings' :
   {
      display_account_form(get_email(),
            get_real_name(get_email()), get_mimetype(get_email()));
      break;
   }
   case 'show-other-lists' :
   {
      display_items('Unsubscribed Lists',
                    get_unsubscribed_lists(get_email()), 'information',
                    'show-archive', 'subscribe');
      break;
   }
   case 'subscribe' :
   {
      subscribe(get_email(), $HTTP_GET_VARS['id']);
      display_items('Subscribed Lists', get_subscribed_lists(get_email()),
                       'information', 'show-archive', 'unsubscribe');
      break;
   }
   case 'unsubscribe' :
   {
      unsubscribe(get_email(), $HTTP_GET_VARS['id']);
      display_items('Subscribed Lists', get_subscribed_lists(get_email()),
                       'information', 'show-archive', 'unsubscribe');
      break;
   }
   case '':
   case 'show-my-lists' :
   {
      display_items('Subscribed Lists', get_subscribed_lists(get_email()),
                       'information', 'show-archive', 'unsubscribe');
      break;
   }
   case 'change-password' :
   {
      display_password_form();
      break;
   }
   case 'store-change-password' :
   {
      if(change_password(get_email(),  $HTTP_POST_VARS['old_passwd'],
      $HTTP_POST_VARS['new_passwd'], $HTTP_POST_VARS['new_passwd2']))
      {
         echo '<p>OK: Password changed.</p>
               <br /><br /><br /><br /><br /><br />';
      }
      else
      {
         echo '<p>Sorry, your password could not be changed.</p>';
         display_password_form();
      }
      break;
   }
```

```
    }
  }
  // Las siguientes acciones sólo las puede realizar el usuario administrativo
  if(check_admin_user())
  {
    switch ( $action )
    {
      case 'create-mail' :
      {
        display_mail_form(get_email());
        break;
      }
      case 'create-list' :
      {
        display_list_form(get_email());
        break;
      }
      case 'store-list' :
      {
        if(store_list($HTTP_SESSION_VARS['admin_user'], $HTTP_POST_VARS))
        {
          echo '<p>New list added</p><br />';
          display_items('All Lists', get_all_lists(), 'information',
                        'show-archive','');
        }
        else
          echo '<p>List could not be stored, please try '
              .'again.</p><br /><br /><br /><br /><br />';

        break;
      }
      case 'send' :
      {
        send($HTTP_GET_VARS['id'], $HTTP_SESSION_VARS['admin_user']);
        break;
      }
      case 'view-mail' :
      {
        display_items('Unsent Mail', get_unsent_mail(get_email()),
                      'preview-html', 'preview-text', 'send');
        break;
      }
    }
  }
/*************************************************************************
* Sección 4: mostrar pie de página
*************************************************************************/

  do_html_footer();
?>
```

Comprobará que en el listado se han separado claramente los cuatro segmentos del código. En la fase de preprocesamiento, establecemos la sesión y procesamos todas las acciones que se deban realizar antes de enviar los encabezados. En este caso, se incluye la conexión y la desconexión.

En la fase de encabezados, definimos los botones de menú que verá el usuario y mostramos los correspondientes encabezados por medio de la función `do_html_header()` de `output_fns.php`.

Esta función simplemente muestra la barra de encabezados y los menús, por lo que no la describiremos con detalle.

En la sección principal de la secuencia de comandos, respondemos a la acción seleccionada por el usuario.

Estas acciones se dividen en tres subconjuntos: acciones que se pueden realizar sin estar conectado, acciones que pueden realizar los usuarios convencionales y acciones que pueden realizar los administradores.

Por medio de las funciones `ckech_logged_in()` y `check_admin_user()` comprobamos si se permite el acceso a los dos últimos conjuntos de acciones.

Estas funciones se encuentran en la biblioteca de funciones `user_auth.fns`.

El código de las mismas, y también de la función `check_normal_user()`, se muestra en el listado 28.3.

Listado 28.3. Funciones de user_auth_fns.php. Comprueban si el usuario se ha conectado o no y a qué nivel

```
function check_normal_user()
// compruebe si hay alguien conectado y, en caso contrario, notifíqueselo
{
   global  $HTTP_SESSION_VARS;

   if (isset($HTTP_SESSION_VARS['normal_user']))
      return true;
   else
      return false;
}

function  check_admin_user()
// compruebe si hay alguien conectado y, en caso contrario, notifíqueselo
{
   global $HTTP_SESSION_VARS;

   if (isset($HTTP_SESSION_VARS['admin_user']))
      return true;
   else
      return false;
}


function check_logged_in()
{
   return ( check_normal_user() || check_admin_user() );
}
```

Como puede comprobar, estas funciones utilizan las variables de sesión `normal_user` y `admin_user` para comprobar si el usuario se ha conectado o no.

Veremos la configuración de estas variables más adelante.

En la sección final de la secuencia de comandos, enviamos un pie de página HTML con ayuda de la función `do_html_footer()` de `output_fns.php`.

A continuación, en la tabla 28.2, se enumeran brevemente las posibles acciones del sistema.

Tabla 28.2. Posibles acciones en la aplicación de gestión de listas de correo

Acción	Utilizada por	Descripción
`log-in`	Todos	Ofrece al usuario un formulario de inicio de sesión
`log-out`	Todos	Finaliza una sesión
`new-account`	Todos	Crea una nueva cuenta para un usuario
`store-account`	Todos	Almacena detalles de cuentas
`show-all-lists`	Todos	Muestra una lista de las listas de correo disponibles
`show-archive`	Todos	Muestra boletines informativos archivados de una lista concreta
`information`	Todos	Muestra información básica sobre una lista concreta
`account-settings`	Usuarios conectados	Muestra la configuración de la cuenta del usuario
`show-other-lists`	Usuarios conectados	Muestra las listas de correo a las que no está suscrito el usuario
`show-my-lists`	Usuarios conectados	Muestra las listas de correo a las que está suscrito el usuario
`subscribe`	Usuarios conectados	Suscribe un usuario a una lista determinada
`unsubscribe`	Usuarios conectados	Anula la suscripción de un usuario a una lista determinada

Acción	Utilizada por	Descripción
`change-password`	Usuarios conectados	Muestra el formulario de cambio de contraseña
`store-change-password`	Usuarios conectados	Actualiza la contraseña del usuario en la base de datos
`create-mail`	Administradores	Muestra un formulario para cargar boletines informativos
`create-list`	Administradores	Muestra un formulario para poder crear nuevas listas de correo
`store-list`	Administradores	Almacena detalles de las listas de correo en la base de datos
`view-mail`	Administradores	Muestra boletines informativos que se han cargado pero que no se han enviado
`send`	Administradores	envía boletines informativos a los suscriptores

Probablemente habrá apreciado que en la tabla falta una opción como `store-mail`, es decir, una acción que cargue los boletines informativos introducidos a través de `create-mail` por los administradores. De hecho, esta funcionalidad concreta se encuentra en un archivo diferente, `upload.php`. La incluimos en un archivo diferente ya que resulta más sencillo para los programadores realizar el seguimiento de los aspectos de seguridad. Analizaremos la implementación de estas acciones en los tres grupos indicados en la tabla 28.2, es decir, acciones para usuarios no conectados, acciones para usuarios conectados y acciones para administradores.

Implementación del inicio de sesión

Cuando un nuevo usuario accede a nuestro sitio, podemos esperar que realice tres acciones. Primero, que vea lo que ofrecemos; segundo, que se registre y, por último, que se conecte. Veremos cada una de ellas de forma separada. En la figura 28.4 se muestra la pantalla a la que acceden los usuarios cuando visitan el sitio por primera vez.

A continuación veremos cómo se crea una nueva cuenta y cómo se realiza la conexión, y volveremos a la operación de ver detalles de las listas en un apartado posterior.

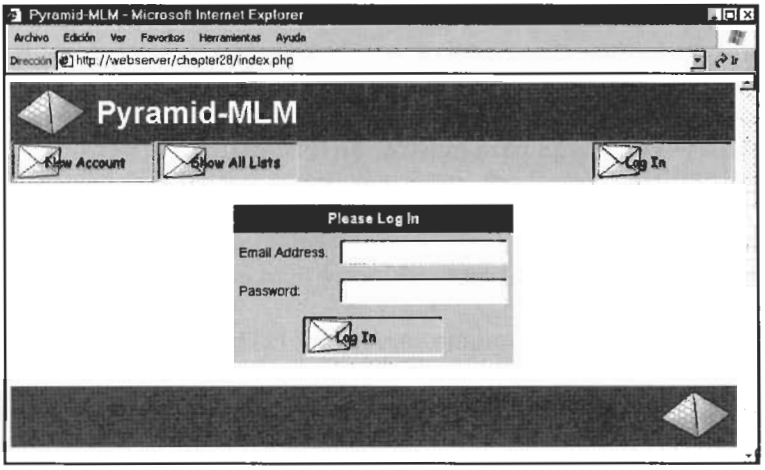


Figura 28.4. Al llegar, los usuarios pueden crear una nueva cuenta, ver las listas disponibles o simplemente conectarse

Creación de una nueva cuenta

Si un usuario selecciona la opción de menú New Account, se activa la acción `new-account` que, a su vez, activa el siguiente código en `index.php`:

```
case 'new-account' :
      {
         // elimine las variables de sesión
         session_destroy();
         display_account_form();
         break;
      }
```

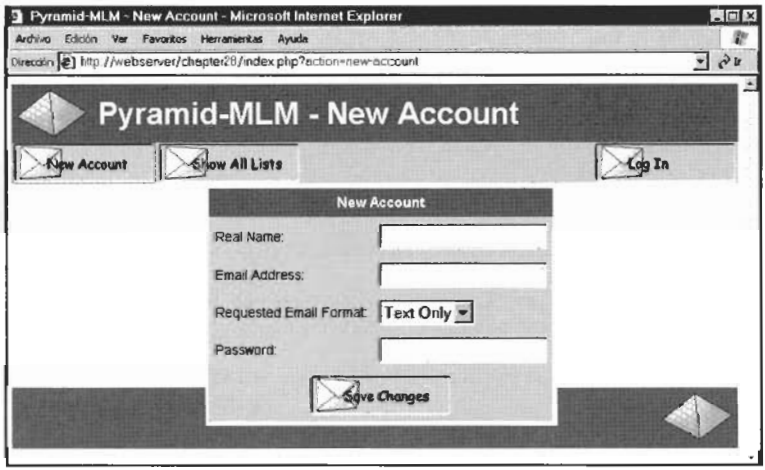


Figura 28.5. El formulario para crear nuevas cuentas permite a los usuarios introducir sus datos

Este código desconecta a un usuario si ya se encuentra conectado y muestra el formulario de detalles de cuenta (figura 28.5).

Este formulario se genera por medio de la función `display_account_form()` de la biblioteca `output_fns.php`. Esta función se utiliza tanto con esta acción como con `account-settings` para mostrar un formulario que permita al usuario configurar una cuenta.

Si la función se invoca desde la acción `account-settings`, el formulario se completa con los datos de cuenta existentes del usuario.

En este caso el formulario está en blanco, listo para introducir nuevos detalles de cuentas.

Como esta función únicamente representa HTML, no la analizaremos aquí.

El botón de envío de este formulario invoca la acción `store-account`, cuyo código mostramos a continuación:

```
case 'store-account' :
      {
         if  (store_account($HTTP_SESSION_VARS['normal_user'],
                  $HTTP_SESSION_VARS['admin_user'], $HTTP_POST_VARS))
            $action = '';
         if(!check_logged_in())
            display_login_form($action);
         break;
      }
```

La función `store_account()` escribe los detalles de la cuenta en la base de datos. Su código se recoge en el listado 28.4.

Listado 28.4. Función store_account() de mlm_fns.php. Estas funciones añaden un nuevo usuario o modifican un usuario ya existente en la base de datos

```
// añada un nuevo suscriptor a la base de datos o permita que el
// usuario modifique sus datos
function store_account($normal_user, $admin_user, $details)
{
   if(!filled_out($details))
   {
      echo 'All fields must be filled in.  Try again.<br /><br />';
      return false;
   }
   else
   {
      if(subscriber_exists($details['email']))
      {
         //compruebe que se han conectado como el usuario que tratan de cambiar
         if(get_email()==$details['email'])
         {
            $query ="update subscribers set  realname ='$details[realname]',
                                             mimetype = '$details[mimetype]'
                     where email = '" . $details[email] . "'";
            if(db_connect() && mysql_query($query))
            {
               return true;
```

```
            }
            else
            {
               echo 'could not store changes.<br /><br /><br /><br /><br />';
               return false;
            }
         }
         else
         {
            echo '<p>Sorry, that email address is already registered here.</p>';
            echo '<p>You will need to log in with that address to change '
               .' its settings.</p>';
            return false;
         }
      }
      else // nueva cuenta
      {
         $query = "insert into subscribers
                        values ('$details[email]',
                        '$details[realname]',
                        '$details[mimetype]',
                        password('$details[new_password]'),
                                                0)";
         if(db_connect() && mysql_query($query))
         {
            return true;
         }
         else
         {
            echo 'Could not store new account.<br /><br /><br /><br /><br /><br />';
            return false;
         }
      }
   }
}
```

En primer lugar, la función comprueba que el usuario ha introducido los datos requeridos.

Si son correctos, la función crea un nuevo usuario o actualiza los detalles de la cuenta si el usuario ya existe.

Un usuario sólo puede actualizar los detalles de la cuenta del usuario bajo el que se ha conectado.

Para realizar la comprobación se utiliza la función `get_mail()`, que recupera la dirección de correo electrónico del usuario conectado actualmente. Volveremos a este aspecto más adelante, ya que utiliza variables de sesión que se configuran cuando se conecta el usuario.

Conexión

Si un usuario completa el formulario de inicio de sesión que vimos en la figura 28.4 y pulsa el botón **Log In**, activará la secuencia de comandos `index.php` con las

variables `email` y `password` configuradas. De esta forma se activa el código de conexión, que se corresponde a la fase de preprocesamiento de la secuencia de comandos y que mostramos a continuación:

```
// es necesario procesar antes que nada las solicitudes de conexión y desconexión
if($HTTP_POST_VARS['email']&&$HTTP_POST_VARS['password'])
{
   $login = login($HTTP_POST_VARS['email'], $HTTP_POST_VARS['password']);

   if($login == 'admin')
   {
      $status .= '<p><b>'.get_real_name($HTTP_POST_VARS['email']).'</b> logged in'.
                 ' successfully as <b>Administrator</b></p>
                 '<br /><br /><br /><br /><br />';
      $HTTP_SESSION_VARS['admin_user'] = $HTTP_POST_VARS['email'];
   }
   else if($login == 'normal')
   {
      $status .= "<p><b>".get_real_name($HTTP_POST_VARS['email'])."</b> logged in"
               ." successfully.</p><br /><br />";
      $HTTP_SESSION_VARS['normal_user'] = $HTTP_POST_VARS['email'];
   }
   else
   {
      $status .= "<p>Sorry, we could not log you in with that
                  email address and password.</p><br />";
   }
}
```

Como puede comprobar, primero intentamos conectarlo con la función `login()` de la biblioteca `user_auth_fns.php`.

Es ligeramente distinta a las funciones de inicio de sesión que hemos utilizado anteriormente, por lo que la analizaremos con más detalle.

En el listado 28.5 puede ver el código correspondiente a esta función.

Listado 28.5. Función login() de user_auth_fns.php. Comprobación de los detalles de conexión de un usuario

```
function login($email, $password)
// compruebe el nombre de usuario y la contraseña en la base de datos
// si coinciden, devuelva el tipo de conexión
// en caso contrario, devuelva false
{
   // conéctese a la base de datos
   $conn = db_connect();
   if (!$conn)
      return 0;

   $query = "select admin from subscribers
                          where email='$email'
                          and password = password('$password')";
```

```
    //duplique $query;
    $result = mysql_query($query);
    if (!$result)
       return false;

    if (mysql_num_rows($result)<1)
       return false;

    if(mysql_result($result, 0, 0) == 1)
       return 'admin';
    else
       return 'normal';
}
```

Anteriormente con las funciones de conexión devolvíamos `true` si la conexión era satisfactoria y `false` si no lo era. En este caso, devolvemos `false` si la conexión no ha sido correcta pero, en caso afirmativo, devolvemos el tipo de usuario, `'admin'` o `'normal'`.

Comprobamos el tipo de usuario recuperando el valor almacenado en la columna `admin` de la tabla de suscriptores, para una combinación concreta de direcciones de correo electrónico y contraseñas.

Si no se recupera ningún resultado, devolvemos `false`. Si el usuario es un administrador, este valor será `1` (`true`) y devolvemos `'admin'`. En caso contrario, devolvemos `'normal'`.

Volvamos a la línea de ejecución principal. Registramos una variable de sesión para realizar el seguimiento de nuestro usuario. Será `admin_user` si se trata de un administrador o `normal_user` si es un usuario convencional. Sea cual sea la variable, contendrá la dirección de correo electrónico del usuario. Para simplificar la comprobación de la misma, utilizamos la función `get_mail()` que mencionamos anteriormente.

Esta función se describe en el listado 28.6.

Listado 28.6. Función get_email() de user_auth_fns.php. Devuelve una dirección de correo electrónico del usuario conectado

```
function  get_email()
{
   global $HTTP_SESSION_VARS;

   if (isset($HTTP_SESSION_VARS['normal_user']))
      return $HTTP_SESSION_VARS['normal_user'];
   if (isset($HTTP_SESSION_VARS['admin_user']))
    return $HTTP_SESSION_VARS['admin_user'];

   return false;
}
```

De nuevo en el programa principal, informamos al usuario si está conectado o no, y a qué nivel.

El resultado de un intento de conexión se muestra en la figura 28.6.

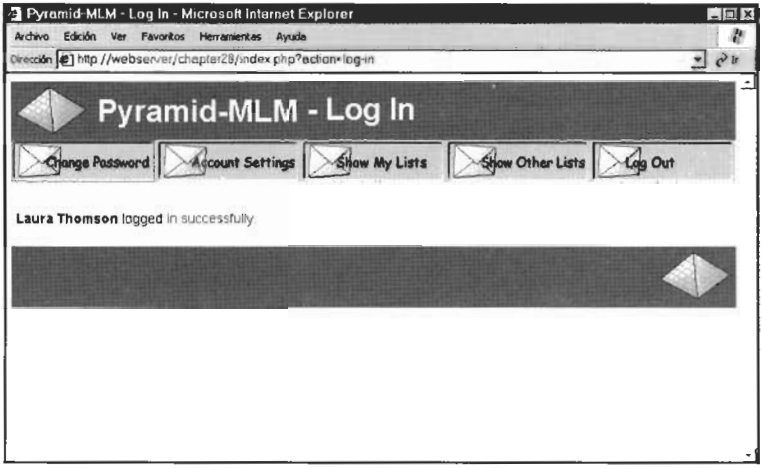


Figura 28.6. El sistema informa al usuario de que la conexión ha sido correcta

Una vez conectado el usuario, podemos pasar a las funciones de usuario.

Implementación de funciones de usuario

Después de conectarse, los usuarios pueden realizar cinco operaciones:

- Ver las listas disponibles a las que se pueden suscribir
- Suscribirse y anular la suscripción a una lista
- Cambiar la configuración de sus cuentas
- Cambiar sus contraseñas
- Desconectarse

En la figura 28.6 hemos visto la mayoría de estas opciones. A continuación analizaremos la implementación de cada una de ellas.

Cómo ver las listas

Implementaremos una serie de opciones para poder ver las listas disponibles y los detalles de las mismas.

En la figura 28.6 puede ver dos de estas opciones: Show My Lists, que recupera las listas a las que está suscrito el usuario y Show Other Lists, que recupera las listas a las que no está suscrito.

Si se fija de nuevo en la figura 28.4, comprobará que existe otra opción, Show All Lists, que recupera todas las listas de correo disponibles en el sistema. Para que el

sistema sea realmente escalable, será necesario añadir funcionalidad de paginación (para mostrar, por ejemplo, 10 resultados por página). No lo hemos hecho por motivos de brevedad.

Estas tres opciones de menú activan las acciones `show-my-lists`, `show-other-lists` y `show-all-lists` respectivamente. Como habrá apreciado, las tres acciones tienen un funcionamiento similar. A continuación se incluye el código de las mismas:

```
case 'show-all-lists' :
{
   display_items( 'All Lists', get_all_lists(), 'information',
                  'show-archive','');
   break;
}


case 'show-other-lists' :
{
   display_items( 'Unsubscribed Lists',
                  get_unsubscribed_lists(get_email()), 'information',
                  'show-archive', 'subscribe');
   break;
}

case '':
case 'show-my-lists' :
{
   display_items('Subscribed Lists', get_subscribed_lists(get_email()),
                  'information', 'show-archive', 'unsubscribe');
   break;
}
```

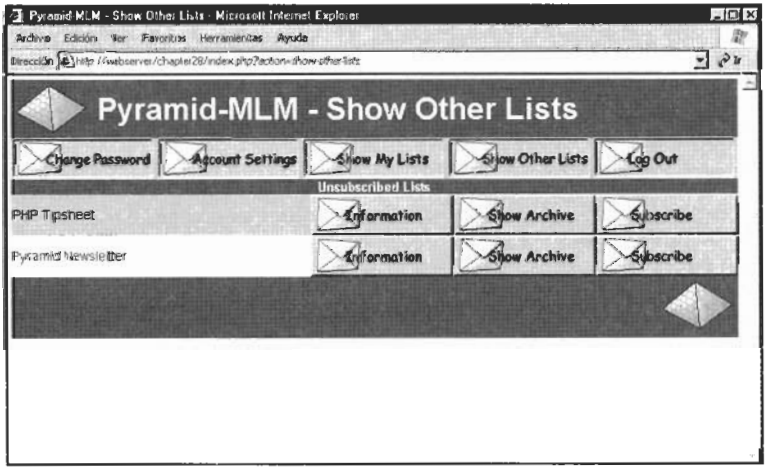


Figura 28.7. La función display_items() se ha utilizado para mostrar una lista de las listas a las que no está suscrito el usuario

Como ve, las tres acciones invocan la función `display_item()` de la biblioteca `ouput_fns.php`.

Cada una lo hace con diferentes parámetros.

También utilizan la función `get_mail()` que hemos mencionado anteriormente para obtener la dirección de correo electrónico correspondiente a cada usuario.

En la figura 28.7 puede ver lo que hace esta función.

Ésta es la página Show Other Lists.

Veamos a continuación el código de la función `display_items()`, que se incluye en el listado 28.7.

Listado 28.7. Función display_items de output_fns.php. Esta función se utiliza para mostrar una lista de elementos con sus correspondientes acciones

```
function display_items($title, $list, $action1='', $action2='', $action3='')
{
   global  $table_width;
   echo "<table width=\"$table_width\" cellspacing=\"0\" cellpadding=\"0\"
         border=\"0\">";

   // cuente el número de acciones
   $actions = (($action1!='') + ($action2!='') + ($action3!=''));

   echo  '<tr>
         <th colspan="'. (1+$actions) .'" bgcolor="#5B69A6">'. $title .'</th>
         </tr>';

   // cuente el número de artículos
   $items = sizeof($list);

   if($items == 0)
      echo  '<tr>
            <td colspan="'.(1+$actions).'" align="center">No Items to Display</td>
            </tr>';
   else
   {
      // imprima todas las filas
      for($i = 0; $items; $i++)
      {
         if($i%2) // los colores de fondo se alternan
            $bgcolor = "'#ffffff'";
         else
            $bgcolor = "'#ccccff'";
         echo "<tr>
               <td bgcolor=$bgcolor
               width=\"".($table_width - ($actions*149)).'">';
         echo $list[$i][1];
         if($list[$i][2])
            echo ' - '.$list[$i][2];
         echo  '</td>';

         // cree botones para una máximo de tres acciones por línea
```

```
            for($j = 1; $j<=3; $j++)
            {
               $var = 'action'.$j;
               if($$var)
               {
               echo "<td bgcolor=\"$bgcolor\" width=\"149\">";
               // los botones view/preview son un caso especial ya que están
               // vinculados a un archivo
               if($$var == 'preview-html'||$$var == 'view-html'||
                  $$var == 'preview-text'||$$var == 'view-text')
                  display_preview_button($list[$i][3], $list[$i][0], $$var);
                  else
                     display_button( $$var, '&id=' . $list[$i][0] );
                  echo '</td>';
               }
            }
            echo  "</tr>\n";
         }
         echo '</table>';
      }
   }
```

Esta función representa una tabla de elementos, cada uno con hasta tres botones de acción asociados. La función espera cinco parámetros que, en orden, son los siguientes:

- `$title` es el título que aparece en la parte superior de la tabla. En el caso ilustrado en la figura 28.7, pasamos el título Unsubscribed Lists, como indicamos en el fragmento de código descrito anteriormente para la acción "Show Other Lists".
- `$list` es una matriz de elementos que se muestran en cada fila de la tabla. En este caso, se trata de una matriz de las listas a las que el usuario no está suscrito actualmente. Generamos la matriz (en este caso) en la función `get_unsubscribed_lists()`, que veremos en breve. Se trata de una matriz multidimensional en la que cada fila puede contener hasta cuatro fragmentos de datos sobre cada fila. En orden, son los siguientes:
  - `$list [n] [0]` contiene el Id. del elemento, que normalmente es un número de fila. Proporciona a los botones de acción el Id. de la fila en la que actúan. En nuestro caso, utilizaremos el Id. de la base de datos, como veremos en breve.
  - `$list [n] [1]` contiene el nombre del elemento. Será el texto que aparezca para un determinado elemento. Por ejemplo, en el caso de la figura 27.8, el nombre del elemento en la primera fila de la tabla es PHP Tipsheet.
  - `$list [n] [2]` y `$list [n] [3]` son opcionales. Los utilizaremos para indicar que hay más información. Se corresponden, respectivamente, al texto y al Id. de dicha información adicional. Veremos un ejemplo en el que se utilizan estos dos parámetros cuando analicemos la acción `View Mail`, en un apartado posterior.

Como alternativa, podríamos haber utilizado palabras clave como índices de la matriz.

El tercer, cuarto y quinto parámetros de la función se utilizan para pasar tres acciones que se mostrarán en botones correspondientes a cada elemento. En la figura 28.7, se trata de los tres botones de opción representados como **Information**, **Show Archive** y **Subscribe**.

Obtenemos estos tres botones para la página Show All Lists al pasar los nombres de acción, `information`, `show-archive` y `subscribe`.

Al utilizar la función `display_button()`, estas acciones se han convertido en botones con dichas palabras sobre los mismos, y se les ha asignado la correspondiente acción.

Todas las acciones Show invocan la función `display_item` de forma diferente, como puede comprobar si observa sus acciones. Además de tener títulos y botones de acción diferentes, cada una de ellas utiliza una función distinta para generar la matriz de elementos que se van a representar. Show All Lists utiliza la función `get_all_lists()`, Show Other Lists utiliza la función `get_unsubscribed_lists()` y Show My Lists, la función `get_subscribed_lists()`. El funcionamiento de las tres funciones es similar y todas provienen de la biblioteca `mlm_fns.php`.

Describiremos `get_unsubscribed_lists()`, ya que es el ejemplo que hemos seguido hasta ahora. En el listado 28.8 se muestra el código de esa función.

Listado 28.8. Función get_unsubscribed_lists() de mlm_fns.php. Generación de una matriz de listas de correo a las que el usuario no está suscrito

```
function get_unsubscribed_lists($email)
{
   $list = array();

   $query = "select lists.listid, listname, email from lists left join sub_lists
                  on lists.listid = sub_lists.listid
                  and email='$email' where email is NULL order by listname";
   if(db_connect())
   {
      $result = mysql_query($query);
      if(!$result)
         echo '<p>Unable to get list from database.</p>';
      $num = mysql_numrows($result);
      for($i = 0; $i<$num; $i++)
      {
         array_push($list, array(mysql_result($result, $i, 0),
                     mysql_result($result, $i, 1)));
      }
   }
   return $list;
}
```

Como puede comprobar, esta función requiere que se pase una dirección de correo electrónico. Debería ser la dirección de correo del suscriptor que estamos utilizando.

La función `get_suscribed_lists()` también requiere una dirección de correo como parámetro, pero `get_all_lists()` no, por razones evidentes.

Tras indicar la dirección de correo del usuario, nos conectamos a la base de datos y obtenemos todas las listas a las que el suscriptor no está suscrito. Como suele ser habitual con este tipo de consulta en MySQL, utilizamos `LEFT JOIN` para buscar elementos que no coincidan.

Realizamos un bucle por el resultado y generamos la matriz fila a fila por medio de la función incorporada `array_push()`.

Una vez que sabemos cómo se genera esta lista, veremos los botones de acción asociados.

Cómo ver información de listas

El botón **Information** de la figura 28.7 desencadena la acción `'information'`, que mostramos a continuación:

```
case 'information' :
{
   display_information($HTTP_GET_VARS['id']);
   break;
}
```

Para ver lo que hace la función `display_information()`, observe la figura 28.8.

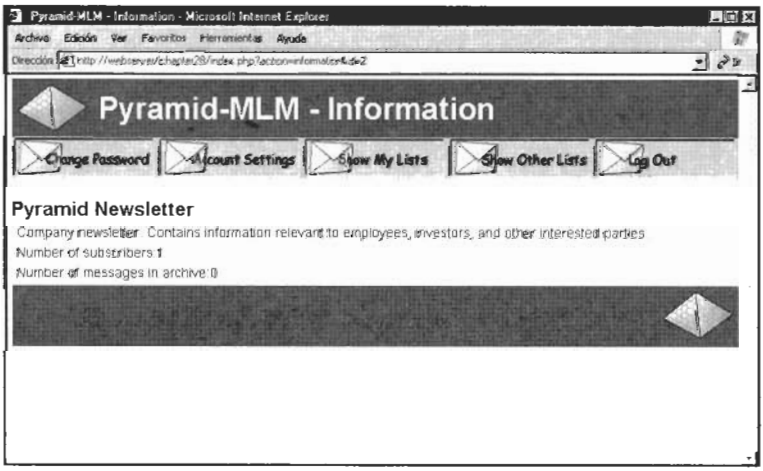


Figura 28.8. La función display_information() muestra un comentario sobre una lista de correo

La función muestra información general sobre una determinada lista de correo, así como el número de suscriptores y el número de boletines informativos que se han

enviado a dicha lista y que se encuentran disponibles en el archivo (que veremos en breve). El código de esta función se recoge en el listado 28.9.

Listado 28.9. Función display_information() de output_fns.php. Muestra información de la lista

```
function display_information($listid)
{
   if(!$listid)
      return false;

   $info  =  load_list_info($listid);

   if($info)
   {
      echo '<h2>'.pretty($info[listname]).'</h2>';
      echo '<p>'.pretty($info[blurb])'</p>;
      echo '<p>Number of subscribers:' . $info[subscribers]'</p>;
      echo '<p>Number of messages in archive:' . $info[archive]'</p>;
   }
}
```

La función `display_information()` utiliza otras dos funciones para conseguir sus propósitos Web: `load_list_info()` y `pretty()`. La función `load_list_info()` se encarga de recuperar datos de la base de datos. La función `pretty()` simplemente aplica formato a los datos recuperados, elimina las barras invertidas, convierte nuevos párrafos en saltos de línea HTML, etc.

Veamos brevemente la función `load_list_info()`, que se encuentra en la biblioteca de funciones `mlm_fns.php`. Su código se incluye en el listado 28.10.

Listado 28.10. Función load_list_info() de mlm_fns.php. Generación de una matriz de información de lista

```
function load_list_info($listid)
{
   if(!$listid)
      return false;

   if(!db_connect())
      return false;

   $query = "select listname, blurb from lists where listid = $listid";
   $result = mysql_query($query);
        if(!$result)
   {
      echo 'Cannot retrieve this list';
      return false;
   }
   $info = mysql_fetch_array($result);

   $query = "select count(*) from sub_lists where listid = $listid";
   $result = mysql_query($query);
      if($result)
```

```
    {
        $info['subscribers']  =  mysql_result($result,  0,  0);
    }
    $query = "select count(*) from mail where listid = $listid
            and status = 'SENT'";
    $result = mysql_query($query);
        if($result)
    {
        $info['archive'] = mysql_result($result, 0, 0);
    }
    return  $info;
}
```

Esta función ejecuta tres consultas de bases de datos para obtener el nombre y el comentario de una lista de la tabla `lists`, el número de suscriptores de la tabla `sub_lists` y el número de boletines informativos enviados de la tabla `mail`.

Cómo ver archivos de listas

Además de poder ver el comentario de la lista, los usuarios pueden comprobar todo el correo que se ha enviado a una determinada lista si pulsan el botón **Show Archive**. Se activará la acción `show-archive`, que desencadena el siguiente código:

```
case 'show-archive' :
{
    display_items('Archive For '.get_list_name($HTTP_GET_VARS['id']),
                get_archive($HTTP_GET_VARS['id']), 'view-html', 'view-text', '');
    break;
}
```

De nuevo, esta función utiliza la función `display_items()` para enumerar los distintos elementos de correo que se han enviado a la lista. Estos elementos se recuperan por medio de la función `get_archive()` de `mlm_fns.php`, cuyo código reproducimos en el listado 28.11.

Listado 28.11. Función get_archive() de mlm_fns.php. Generación de una matriz de boletines informativos archivados para una determinada lista

```
function get_archive($listid)
{
    //devuelve una matriz del correo archivado para esta lista
    //la matriz tiene filas como (Id de correo, asunto)

    $list = array();
    $listname  =  get_list_name($listid);

    $query = "select mailid, subject, listid from mail
            where listid = $listid and status = 'SENT' order by sent";

    if(db_connect())
```

```
    {
        $result = mysql_query($query);
        if(!$result)
        {
            echo "<p>Unable to get list from database - $query.</p>";
            return false;
        }
        $num = mysql_numrows($result);
        for($i = 0; $i<$num; $i++)
        {
            $row = array(mysql_result($result, $i, 0),
                   mysql_result($result, $i, 1), $listname, $listid);
            array_push($list, $row);
        }
    }
    return $list;
}
```

Como en el caso anterior, esta función recupera la información necesaria de la base de datos, los detalles de correo que se han enviado, y genera la correspondiente matriz que se pasará a la función `display_items`.

Suscripciones y anulación de suscripciones

En la lista de listas de correo de la figura 28.7, cada lista dispone de un botón que permite a los usuarios suscribirse a la misma. Del mismo modo, si los usuarios recurren a la opción **Show My Lists** para ver las listas a las que ya están suscritos, verán un botón **Unsubscribe** junto a cada una de las listas.

Estos botones activan las acciones `subscribe` y `unsubscribe`, que desencadenan los dos siguientes fragmentos de código, respectivamente:

```
case 'subscribe' :
{
    subscribe(get_email(), $HTTP_GET_VARS['id']);
    display_items('Subscribed Lists', get_subscribed_lists(get_email()),
                  'information', 'show-archive', 'unsubscribe');
    break;
}
case 'unsubscribe' :
{
    unsubscribe(get_email(), $HTTP_GET_VARS['id']);
    display_items('Subscribed Lists', get_subscribed_lists(get_email()),
                  'information', 'show-archive', 'unsubscribe');
    break;
}
```

En cada caso, invocamos una función (`subscribe()` o `unsubscribe()`) y, tras ello, volvemos a mostrar una lista de listas de correo a las que el usuario se ha suscrito por medio de la función `display_items()`.

Las funciones `subscribe()` y `unsubscribe()` se muestran en el listado 28.12.

Listado 28.12. Funciones subscribe() y unsubscribe() de mlm_fns.php. Estas funciones añaden o borran suscripciones de un usuario

```
function subscribe($email, $listid)
{
   if(!$email||!$listid||!list_exists($listid)||!subscriber_exists($email))
      return  false;

   //si ya está suscrito, salga
   if(subscribed($email,  $listid))
      return false;

   if(!db_connect())
      return false;

   $query = "insert into sub_lists values ('$email', $listid)";

   $result  =  mysql_query($query);
      return $result;
}

function unsubscribe($email, $listid)
{
   if(!$email||!$listid)
      return false;

   if(!db_connect())
      return false;

   $query = "delete from sub_lists where email = '$email' and listid = $listid";

   $result = mysql_query($query);
   return $result;
}
```

La función `subscribe()` añade una fila a la tabla `sub_lists` que se corresponde a la suscripción; la función `unsubscribe()` elimina esta fila.

Modificación de la configuración de una cuenta

Al pulsar el botón **Account Settings** se activa la acción `account-settings`, cuyo código mostramos a continuación:

```
case 'account-settings' :
{
   display_account_form(get_email(),
      get_real_name(get_email()), get_mimetype(get_email()));
   break;
}
```

Como puede comprobar, volvemos a utilizar la función `display_account_form()` que utilizamos para crear la cuenta inicialmente. Sin embargo, en esta oca-

sión pasamos los detalles actuales del usuario, que se representarán en el formulario para facilitar la modificación. Cuando el usuario pulsa sobre el botón de envío del formulario, se activa la acción `store-account` como mencionamos previamente.

Modificación de contraseñas

Al pulsar el botón **Change Password** se activa la acción `change-password`, que desencadena el siguiente código:

```
case 'change-password' :
{
   display_password_form();
   break;
}
```

La función `display_password_form()` (de la biblioteca de funciones `output_fns.php`) simplemente muestra un formulario en el que el usuario puede cambiar su contraseña. Este formulario se ilustra en la figura 28.9.

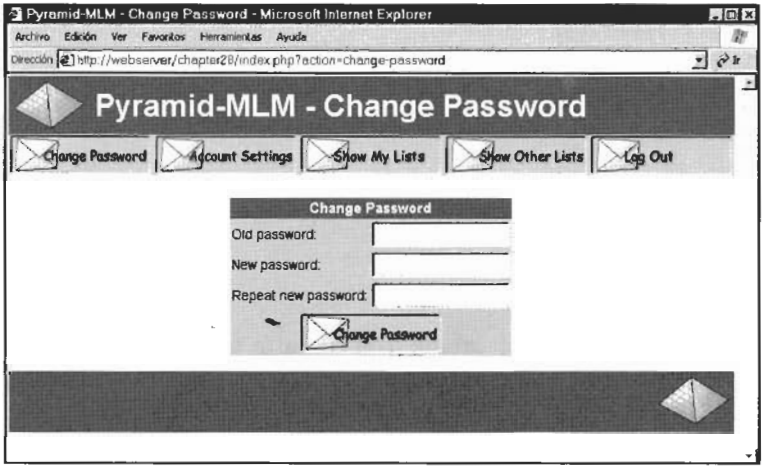


Figura 28.9. La función display_password() permite a los usuarios cambiar sus contraseñas

Cuando el usuario pulsa el botón **Change Password**, que se encuentra en la parte inferior del formulario, se activa la acción `store-change-password`, cuyo código es el siguiente:

```
case 'store-change-password' :
{
   if(change_password(get_email(), $HTTP_POST_VARS['old_passwd'],
      $HTTP_POST_VARS['new_passwd'], $HTTP_POST_VARS['new_passwd2']))
   {
      echo '<p>OK: Password changed.</p><br /><br /><br /><br /><br /><br />';
   }
```

```
    else
    {
        echo '<p>Sorry, your password could not be changed.</p>';
        display_password_form();
    }
    break;
}
```

Como puede comprobar, este código intenta cambiar la contraseña por medio de la función `change_password()` e indica al usuario si la operación ha sido satisfactoria o no.

La función `change_password()` se encuentra en la biblioteca de funciones `user_auth_fns.php` y su código se reproduce en el listado 28.13.

Listado 28.13. Función change_password() de user_auth_fns.php. Esta función valida y actualiza la contraseña de un usuario

```
function change_password($email, $old_password, $new_password,
                         $new_password_conf)
// cambie la contraseña del correo electrónico/old_password por new_password
// devuelva true o false
{

    // si la contraseña antigua es correcta
    // cambie su contraseña por new_password y devuelva true
    // en caso contrario, devuelva false
    if (login($email, $old_password))
    {
        if($new_password==$new_password_conf)
        {
            if (!($conn = db_connect()))
                return false;
            $query = "update subscribers
                      set password = password('$new_password')
                      where email = '$email'";
            $result = mysql_query($query);
            return $result;
        }
        else
            echo '<p> Your passwords do not match.</p>;
    }
    else
        echo '<p> Your old password is incorrect.</p>';

    return false; // la contraseña antigua era incorrecta
}
```

Esta función es similar a otras funciones de configuración y modificación de contraseñas que ya hemos descrito.

Compara las dos nuevas contraseñas introducidas por el usuario para garantizar que son idénticas y, en caso de que lo sean, intenta actualizar la contraseña del usuario en la base de datos.

Desconexión

Cuando un usuario pulsa el botón **Log Out**, se desencadena la acción `log-out`. El código ejecutado por esta acción en la secuencia de comandos principal se encuentra de hecho en la sección de preprocesamiento, como se indica a continuación:

```
if($action == 'log-out')
{
   unset($HTTP_SESSION_VARS);
   session_destroy();
}
```

Este fragmento de código elimina las variables de sesión y destruye la sesión. También anula la variable `action`, lo que significa que introducimos la instrucción de caso principal sin una acción, con lo que se desencadena el siguiente código:

```
case '':
{
   if(!check_logged_in())
      display_login_form($action);
   break;
}
```

De esta forma se podrá conectar otro usuario o el usuario podrá conectarse como alguien diferente.

Implementación de funciones administrativas

Si un usuario se conecta como administrador, tendrá acceso a opciones de menú adicionales, como puede comprobar en la figura 28.10.

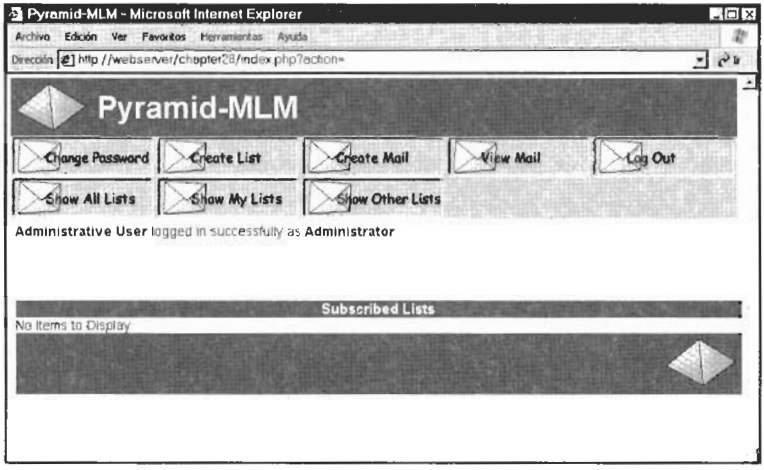


Figura 28.10. El menú de administración permite crear y mantener listas de correo

Las opciones adicionales son Create List (para crear una nueva lista de correo), Create Mail (para crear un nuevo boletín informativo) y View Mail (para ver y enviar boletines informativos creados pero todavía sin enviar). Veremos cada una de estas opciones individualmente.

Creación de una nueva lista

Si el administrador opta por definir una nueva lista y pulsa el botón **Create List**, activará la acción `create-list`, que está asociada al siguiente código:

```
case 'create-list' :
{
   display_list_form(get_email());
   break;
}
```

La función `display_list_form()` muestra un formulario que permite al administrador introducir los detalles de la nueva lista. Se encuentra en la biblioteca `output_fns.php` y simplemente representa HTML por lo que no la describiremos. El resultado de esta función se muestra en la figura 28.11.

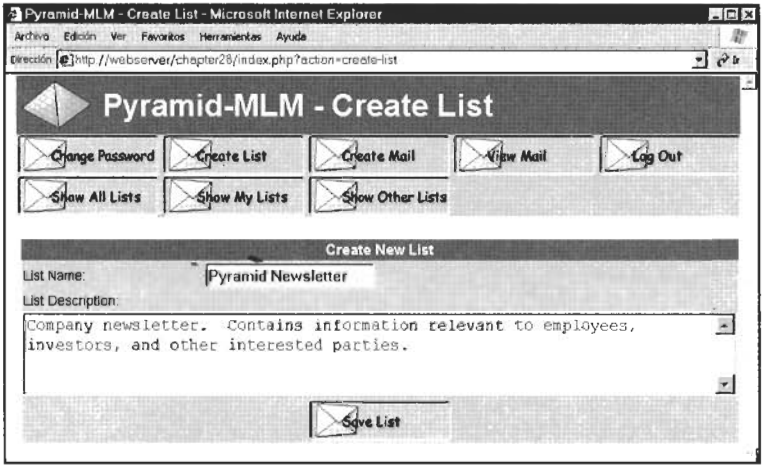


Figura 28.11. La opción Create List requiere que el administrador introduzca un nombre y una descripción (o comentario) sobre la nueva lista

Cuando el administrador pulsa el botón **Save List**, se activa la acción `store-list`, que desencadena el siguiente código en `index.php`:

```
case 'store-list' :
{
   if(store_list($HTTP_SESSION_VARS['admin_user'], $HTTP_POST_VARS))
   {
      echo '<p>New list added</p><br />';
```

```
        display_items('All Lists', get_all_lists(), 'information',
                      'show-archive','');
    }
    else
        echo '<p>List could not be stored, please try '
            .'again.</p><br /><br /><br /><br /><br />';

    break;
}
```

Como puede comprobar, el código intenta almacenar los detalles de la nueva lista y, tras ello, muestra la nueva lista de listas.

Los detalles de la lista se almacenan por medio de la función `store_list()`, cuyo código reproducimos en el listado 28.14.

Listado 28.14. Función store_list() de mlm_fns.php. Esta función añade una nueva lista de correo a la base de datos

```
function store_list($admin_user, $details)
{
    if(!filled_out($details))
    {
        echo 'All fields must be filled in.  Try again.<br /><br />';
        return  false;
    }
    else
    {
        if(!check_admin_user($admin_user))
            return false;
            // ¿cómo pudo invocar esta función alguien no conectado como administrador?

        if(!db_connect())
        {
            return false;
        }

        $query = "select count(*) from lists where listname = '"
                .$details['name']."'";
        $result = mysql_query($query);
        if(mysql_result($result, 0, 0) > 0)
        {
            echo 'Sorry, there is already a list with this name.';
            return false;
        }

        $query = "insert into lists values(NULL,
                                           '".$details['name']."',
                                           '".$details['blurb']."')";

        $result = mysql_query($query);
        return  $result;
    }
}
```

Esta función realiza algunas comprobaciones de validación antes de escribir en la base de datos. Comprueba que se han introducido todos los detalles, que el usuario actual es un administrador y que el nombre de la lista es exclusivo. Si todo funciona correctamente, la lista se añade a la tabla `lists` de la base de datos.

Carga de un nuevo boletín informativo

Por último llegamos a la parte principal de la aplicación: la carga y el envío de boletines informativos a listas de correo.

Cuando un administrador pulsa el botón **Create Mail**, se activa la acción `create-mail`, como se indica a continuación:

```
case 'create-mail' :
{
   display_mail_form(get_email());
   break;
}
```

El administrador verá el formulario que mostramos en la figura 28.12.

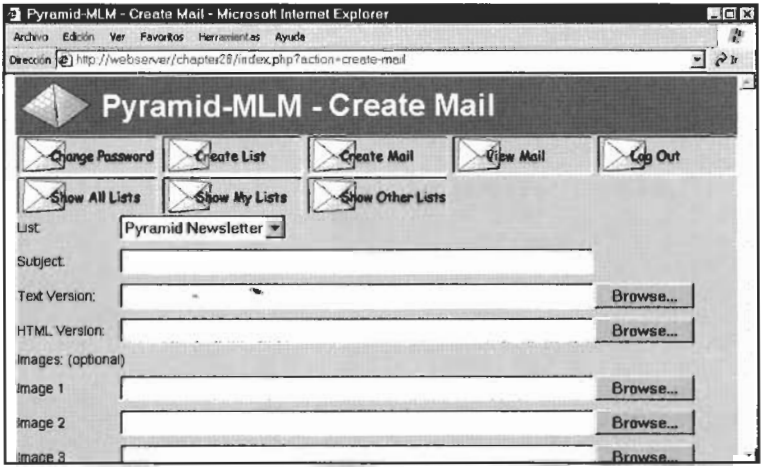


Figura 28.12. La opción Create Mail ofrece al administrador una interfaz para cargar archivos de boletines informativos

Recuerde que para esta aplicación asumimos que el administrador ha creado un boletín informativo en formato HTML y en texto, y que cargará ambas versiones antes de enviarlo. Hemos optado por implementarlo de esta forma para que los administradores puedan utilizar su software preferido para crear los boletines informativos. De esta forma la aplicación resulta más accesible.

Comprobará que el formulario cuenta con una serie de campos que el administrador debe completar. En la parte superior se incluye un cuadro desplegable para seleccionar la lista de correo. El administrador también tiene que indicar el asunto

del boletín informativo (lo que se utilizará como línea de asunto del correo electrónico posterior).

Los campos restantes son campos de carga de archivos, como indica el botón **Browse** junto a cada uno de ellos. Para poder enviar un boletín informativo, el administrador debe incluir tanto la versión en texto como la versión HTML del mismo (aunque lo puede cambiar para ajustarlo a sus necesidades). También hay una serie de campos de imagen opcionales en los que el administrador puede cargar todas las imágenes que haya incrustado en el HTML. Será necesario especificar y cargar individualmente cada uno de estos archivos.

El formulario es similar a cualquier formulario de carga de archivos convencional a excepción de que, en este caso, lo utilizamos para cargar varios archivos. Por ello habrá que realizar algunas modificaciones en la sintaxis del formulario y en la forma de procesar los archivos cargados.

El código de la función `display_mail_form()` se recoge en el listado 28.15.

Listado 28.15. Función display_mail_form() de output_fns.php. Esta función muestra el formulario de carga de archivos

```
function display_mail_form($email, $listid=0)
{
   // muestre un formulario html para cargar un nuevo mensaje
   global $table_width;
   $list = get_all_lists();
   $lists = sizeof($list);
?>
   <table cellpadding="4" cellspacing="0" border="0"
        width="<?php echo $table_width?>">
   <form enctype='multipart/form-data' action='upload.php' method='post'>
   <tr>
      <td bgcolor="#cccccc">
         List:
      </td>
      <td  bgcolor="#cccccc">
         <select name="list">
         <?php
         for($i = 0; $i<$lists; $i++)
         {
            echo  '<option  value  =  '.$list[$i][0];
            if ($listid== $list[$i][0]) echo ' selected';
            echo '>'.$list[$i][1]."</option>\n";
         }
         ?>
         </select>
      </td>
   </tr>
   <tr>
      <td bgcolor="#cccccc">
         Subject:
      </td>
      <td bgcolor="#cccccc">
         <input type="text" name="subject" value="<?php echo $subject?>"
          size  =  60  ></td>
```

```
    </tr>
    <tr><td bgcolor="#cccccc">
       Text Version:
    </td><td bgcolor="#cccccc">
       <input type=file name='userfile[0]' size = 60>
    </td></tr>
    <tr><td bgcolor="#cccccc">
       HTML Version:
    </td><td bgcolor="#cccccc">
       <input type=file name='userfile[1]' size = 60>
    </td></tr>
    <tr><td bgcolor="#cccccc" colspan="2">Images: (optional)

<?php
    $max_images = 10;
    for($i = 0; $i<10; $i++)
    {
       echo  "<tr><td bgcolor='#cccccc'>Image ". ($i+1) .' </td>';
       echo  "<td bgcolor='#cccccc'>";
       echo  "<input type='file' name='userfile[".($i+2)."]' size='60'></td></tr>";
    }
?>
    <tr><td colspan="2" bgcolor="#cccccc" align="center">
    <input type="hidden" name="max_images" value="<?php echo $max_images?>">
    <input type="hidden" name="listid"  value="<?php echo $listid?>">
    <?php display_form_button('upload-files'); ?>
    </td>
    </form>
    </tr>
    </table>
<?php
}
```

Conviene destacar que los archivos que queremos cargar introducirán sus nombres en una serie de entradas de tipo `file` y que los nombres estarán comprendidos entre `userfile[0]` y `userfile[n]`.

En definitiva, tratamos a estos campos del formulario de la misma forma que a casillas de verificación y les asignamos nombres utilizando una convención de matrices.

Si quiere cargar varios archivos por medio de una secuencia de comandos PHP, tendrá que utilizar esta convención.

En la secuencia de comandos que procesa este formulario, obtendremos tres matrices.

A continuación analizaremos dicha secuencia de comandos.

Procesamiento de la carga de varios archivos

Recordará que el código de carga de archivos se encuentra en un archivo independiente.

El listado completo de dicho archivo se muestra en el listado 28.16.

Listado 28.16. upload.php. Esta secuencia de comandos carga todos los archivos necesarios para un boletín informativo

```
<?php
   // esta funcionalidad se incluye en un archivo distinto, lo que nos permite ser
   // más paranoicos al respecto

   // si algo va mal, saldremos

   $max_size = 50000;

   include ('include_fns.php');
   session_start();

   // sólo los usuarios de administración pueden cargar archivos
   if(!check_admin_user())
   {
      echo 'You do not seem to be authorized to use this page.';
      exit;
   }

   // cree los botones de la barra de herramientas de administración
   $buttons = array();
   $buttons[0]= 'change-password';
   $buttons[1]= 'create-list';
   $buttons[2]= 'create-mail';
   $buttons[3]= 'view-mail';
   $buttons[4]= 'log-out';
   $buttons[5]= 'show-all-lists';
   $buttons[6]= 'show-my-lists';
   $buttons[7]= 'show-other-lists';

   do_html_header('Pyramid-MLM - Upload Files');

   display_toolbar($buttons);

   // compruebe que la página se invoca con los datos necesarios
   if(!$HTTP_POST_FILES['userfile']['name'][0]
      ||!$HTTP_POST_FILES['userfile']['name'][1]
      ||!$HTTP_POST_VARS['subject']||!$HTTP_POST_VARS['list'])
   {
       echo 'Problem: You did not fill out the form fully. The images are the
            only optional fields. Each message needs a subject, text version
            and an HTML version.';
       do_html_footer();
       exit;
   }

   $list = $HTTP_POST_VARS['list'];
   $subject = $HTTP_POST_VARS['subject'];

   if(!db_connect())
   {
       echo '<p>Could not connect to db</p>';
       do_html_footer();
```

```
    exit;
}

// añada los detalles del correo a la base de datos
$query = "insert into mail values (NULL,
                                   '".$HTTP_SESSION_VARS['admin_user']."',
                                   '".$subject."',
                                   '".$list."',
                                   'STORED', NULL, NULL)";
$result = mysql_query($query);
if(!$result)
{
   do_html_footer();
   exit;
}

//obtenga el Id. MySQL asignado a este correo
$mailid  =  mysql_insert_id();

if(!$mailid)
{
   do_html_footer();
   exit;
}


// si éste no es el primer mensaje archivado, fallará la creación del directorio
// es correcto
@ mkdir('archive/'.$list, 0700);

// si la creación del directorio específico para este correo falla, es un problema
if(!mkdir('archive/'.$list."/$mailid", 0700))
{
   do_html_footer();
   exit;
}

// procese una iteración en la matriz de archivos cargados
$i = 0;
while ($HTTP_POST_FILES['userfile']['name'][$i]&&
       $$HTTP_POST_FILES['userfile']['name'][$i]!='none')
{
   echo '<p>Uploading '.$HTTP_POST_FILES['userfile']['name'][$i].' - ';
   echo $HTTP_POST_FILES['userfile']['size'][$i].' bytes.</p>';
   if  ($HTTP_POST_FILES['userfile']['size'][$i]==0)
   {
      echo  'Problem: '.$HTTP_POST_FILES['userfile']['name'][$i].
            ' is zero length';
      $i++;
      continue;
   }

   if ($HTTP_POST_FILES['userfile']['size'][$i]>$max_size)
   {
      echo 'Problem: '.$HTTP_POST_FILES['userfile']['name'][$i].' is over '
           .$max_size.' bytes';
```

```
            $i++;
            continue;
        }

        // deberíamos comprobar que la imagen cargada es una imagen
        // si getimagesize() puede determinar su tamaño, probablemente lo sea.
        if($i>1&&!getimagesize($HTTP_POST_FILES['userfile']['tmp_name'][$i]))
        {
            echo  'Problem: '.$HTTP_POST_FILES['userfile']['name'][$i].
                  ' is corrupt, or not a gif, jpeg or png';
            $i++;
            continue;
        }

        // file 0 (el mensaje de texto) y file 1 (el mensaje html) son casos especiales
        if($i==0)
            $destination = "archive/$list/$mailid/text.txt";
        else if($i == 1)
            $destination = "archive/$list/$mailid/index.html";
        else
        {
            $destination ="archive/$list/$mailid/"
                          .$HTTP_POST_FILES['userfile']['name'][$i];
            $query = "insert into images values ($mailid,
                                  '".$HTTP_POST_FILES['userfile']['name'][$i]."',
                                  '".$HTTP_POST_FILES['userfile']['type'][$i]."')";
            $result = mysql_query($query);
        }

        //si utilizamos PHP versión >= 4.03

        if (!is_uploaded_file($HTTP_POST_FILES['userfile']['tmp_name'][$i]))
        {
            // posible ataque de carga de archivos detectado
            echo  'Something funny happening with '
                  .$HTTP_POST_FILES['userfile']['name'].', not uploading.';
            do_html_footer();
            exit;
        }

        move_uploaded_file( $HTTP_POST_FILES['userfile']['tmp_name'][$i],
                            $destination);
/*
        // si es la versión <= 4.02
        copy ($userfile[$i], $destination);

        unlink($userfile[$i]);
*/

        $i++;
    }

    display_preview_button($list, $mailid, 'preview-html');
    display_preview_button($list, $mailid, 'preview-text');
```

```
    display_button('send', "&id=$mailid");

    echo '<br /><br /><br /><br /><br />';
    do_html_footer();
?>
```

Veamos los pasos de este listado.

En primer lugar, iniciamos una sesión y comprobamos si el usuario se ha conectado como administrador, ya que no queremos que nadie más pueda cargar archivos.

Estrictamente hablando, también deberíamos comprobar si en las variables `list` y `mailid` hay caracteres no deseados, pero lo hemos omitido por motivos de brevedad.

Seguidamente, definimos y enviamos los encabezados de la página, y comprobamos que el formulario se haya completado correctamente.

En este caso es muy importante, ya el formulario resulta difícil de completar para el usuario.

Tras ello creamos una entrada para este correo en la base de datos y creamos un directorio en el archivo para almacenar el correo.

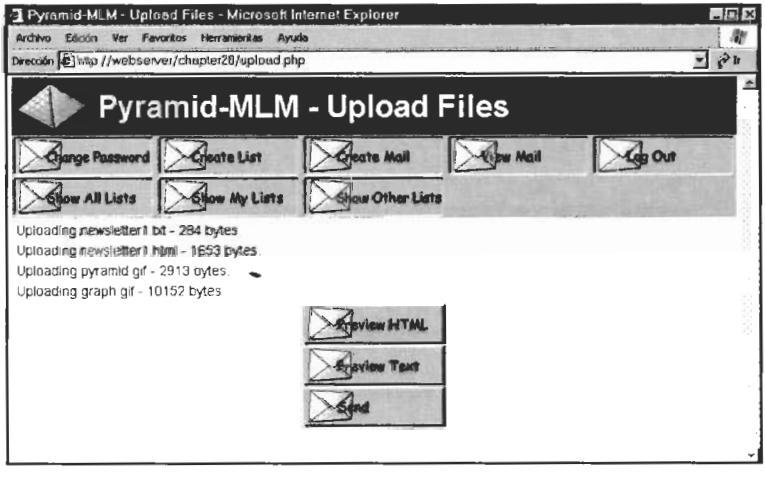


Figura 28.13. La secuencia de comandos de carga informa de los archivos cargados así como de su tamaño

A continuación, la parte principal de la secuencia de comandos, en la que se comprueba y se mueve cada uno de los archivos cargados. Esta parte es diferente si se cargan varios archivos. En este momento debemos procesar cuatro matrices, con los nombres `$HTTP_POST_FILES['userfile']['name']`, `$HTTP_POST_FILES['userfile']['tmp_name']`, `$HTTP_POST_FILES['userfile']['size']` y `$HTTP_POST_FILES['userfile']['type']`. Se corresponden a los equivalentes de mismo nombre de la carga de un solo archivo a excepción de que cada una es

una matriz. El primer archivo del formulario se detalla en `$HTTP_POST_FILES['userfile']['tmp_name'][0]`, `$HTTP_POST_FILES['userfile']['name'][0]`, `$HTTP_POST_FILES['userfile']['size'][0]` y `$HTTP_POST_FILES['userfile']['type'][0]`.

Con estas tres matrices, realizamos las habituales comprobaciones de seguridad y añadimos los archivos al archivo principal.

Por último, le ofrecemos una serie de botones al administrador que puede utilizar para previsualizar el boletín informativo cargado antes de enviarlo, así como un botón para enviarlo. El resultado de `upload.php` se muestra en la figura 28.13.

Vista previa del boletín informativo

El administrador dispone de dos formas de acceder a una vista previa de un boletín informativo antes de enviarlo. Puede acceder a las funciones de vista previa desde la pantalla de carga si quiere previsualizar el boletín inmediatamente después de cargarlo. También puede pulsar el botón **View Mail**, que le mostrará todos los boletines informativos sin enviar que hay en el sistema, para que los pueda previsualizar y enviar más tarde. El botón **View Mail** activa la acción `view-mail`, que desencadena el siguiente código:

```
case 'view-mail' :
{
   display_items( 'Unsent Mail', get_unsent_mail(get_email()),
                  'preview-html', 'preview-text', 'send');
   break;
}
```

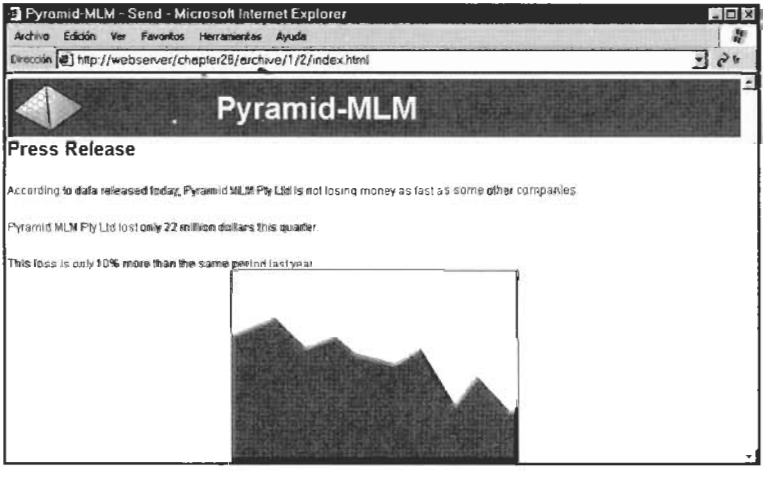


Figura 28.14. Una vista previa de un boletín informativo HTML, con imágenes

Como puede comprobar, se utiliza de nuevo la función `display_items()` con botones para las acciones `preview-html`, `preview-text` y `send`.

Un aspecto interesante que conviene destacar es que los botones "Preview" no desencadenan realmente una acción, sino que realizan un enlace directo al boletín informativo en el archivo.

Si se fija en los listados 28.7 y 28.8 apreciará que utilizamos la función `display_preview_button()` para crear estos botones, en lugar de la función `display_button()` habitual.

La función `display_button()` crea un enlace de imagen a una secuencia de comandos con parámetros `GET` cuando sea necesario; la función `display_preview_button()` ofrece un simple enlace al archivo.

Este enlace aparecerá en una nueva ventana, que se consigue por medio del atributo `target=new` de la etiqueta de ancla HTML.

El resultado de la vista previa de la versión HTML de un boletín informativo se representa en la figura 28.14.

Envío del mensaje

Al pulsar el botón **Send** de un boletín informativo se activa la acción `send`, que desencadena el siguiente código:

```
case 'send' :
{
   send($HTTP_GET_VARS['id'], $HTTP_SESSION_VARS['admin_user']);
   break;
}
```

De esta forma se invoca la función `send()`, incluida en la biblioteca de funciones `mlm_fns.php`. Es una función bastante extensa y en la que utilizaremos la clase `Mail_mime`.

El código de nuestra función se muestra en el listado 28.17.

Listado 28.17. Función send() de mlm_fns.php. Esta función envía un boletín informativo

```
// cree el mensaje a partir de las entradas y archivos almacenados en la base de datos
// envíe mensajes de prueba al administrador o mensajes reales a toda la lista
function send($mailid, $admin_user)
{
   if(!check_admin_user($admin_user))
      return false;

   if(!($info = load_mail_info($mailid)))
   {
      echo "Cannot load list information for message $mailid";
      return false;
   }
   $subject = $info[0];
   $listid = $info[1];
   $status  =  $info[2];
   $sent = $info[3];
```

```
$from_name = 'Pyramid MLM';

$from_address = 'return@address';

$query = "select email from sub_lists where listid = $listid";

$result = mysql_query($query);
if (!$result)
{
   echo $query;
   return false;
}
else if (mysql_num_rows($result)==0)
{
   echo "There is nobody subscribed to list number $listid";
   return false;
}
else
{
   // incluya clases de correo de PEAR
   include('Mail.php');
   include('Mail/mime.php');

   // instancie una clase MIME y pásela al carácter de retorno de carro/nueva línea
   // utilizado en este sistema
   $message = new Mail_mime("\r\n");

   // lea la versión en texto del boletín informativo
   $textfilename = "archive/$listid/$mailid/text.txt";
   $tfp = fopen($textfilename, "r");
   $text = fread($tfp, filesize($textfilename));
   fclose($tfp);

   // lea la versión HTML del boletín informativo
   $htmlfilename = "archive/$listid/$mailid/index.html";
   $hfp = fopen($htmlfilename, "r");
   $html = fread($hfp, filesize($htmlfilename));
   fclose($hfp);

   // añada HTML y texto al objeto mimemail
   $message->setTXTBody($text);
   $message->setHTMLBody($html);

   // obtenga la lista de imágenes relacionadas con este mensaje
   $query = "select path, mimetype from images where mailid = $mailid";
   if(db_connect())
   {
      $result = mysql_query($query);
      if(!$result)
      {
         echo '<p>Unable to get image list from database.</p>';
         return false;
      }
      $num = mysql_numrows($result);
```

```
    for($i = 0; $i<$num; $i++)
    {
        //cargue todas las imágenes desde disco
        $imgfilename = "archive/$listid/$mailid/".mysql_result($result, $i, 0);
        $imgtype = mysql_result($result, $i, 1);

        // añada todas las imágenes al objeto
        $message->addHTMLImage($imgfilename, $imgtype, $imgfilename, true);
    }
}

// cree el cuerpo del mensaje
$body = $message->get();

// cree los encabezados del mensaje
$from = '"'.get_real_name($admin_user).'" <'.$admin_user.'>';
$hdrarray = array(
                'From'      => $from,
                'Subject' => $subject);

$hdrs = $message->headers($hdrarray);

// cree el objeto que se va a enviar
$sender =& Mail::factory('mail');

if($status == 'STORED')
{

    // envíe el mensaje HTML al administrador
    $sender->send($admin_user, $hdrs, $body);

    // envíe la versión en texto del mensaje al administrador
    mail($admin_user, $subject, $text, 'From: "'
        .get_real_name($admin_user).'" <'.$admin_user.">");

    echo "Mail sent to $admin_user";

    // marque el boletín informativo como leído
    $query = "update mail set status = 'TESTED' where mailid = $mailid";
    if(db_connect())
    {
        $result = mysql_query($query);
    }

    echo '<p>Press send again to send mail to whole list.</p><center>';
    display_button('send', "&id=$mailid");
    echo '</center>';
}
else if($status == 'TESTED')
{
    //envíelo a la lista

    $query = "select  subscribers.realname, sub_lists.email,
                      subscribers.mimetype
```

```
                from sub_lists, subscribers
                where listid = $listid and
                      sub_lists.email = subscribers.email";

        if(!db_connect())
           return false;

        $result = mysql_query($query);
        if(!$result)
           echo '<p>Error getting subscriber list</p>';

        $count = 0;
        // para cada suscriptor
        while( $subscriber = mysql_fetch_row($result) )
        {
           if($subscriber[2]=='H')
           {
              //envíe una versión HTML a los que la soliciten
              $sender->send($subscriber[1], $hdrs, $body);
           }
           else
           {
              //envíe una versión en texto a los que no quieran correo
        // HTML
              mail($subscriber[1], $subject, $text,
                    'From: "'.get_real_name($admin_user).'" <'.$admin_user.">");
           }
           $count++;
        }

        $query = "update mail set status = 'SENT', sent = now()
                where mailid = $mailid";
        if(db_connect())
        {
           $result = mysql_query($query);
        }
        echo "<p>A total of $count messages were sent.</p>";
     }
     else if($status == 'SENT')
     {
        echo '<p>This mail has already been sent.</p>';
     }
  }
}
```

Esta función realiza diferentes operaciones.

Remite pruebas del boletín informativo al administrador antes de enviarlo.

Realiza el seguimiento por medio del estado de un correo en la base de datos.

Cuando la secuencia de comandos de carga de archivos carga un correo, establece el estado inicial del mismo como "STORED" (almacenado).

Si la función `send()` encuentra un correo con el estado "STORED", lo actualiza a "TESTED" (probado) y lo envía al administrador. El estado "TESTED" significa que

se ha mandado una prueba del correo al administrador. Si el estado es "TESTED", se cambia por "CHANGED" (modificado) y se envía a la lista.

Este significa que todos los correos se envían dos veces: una en modo de prueba y otra en modo real.

La función también envía dos tipos de correo electrónico diferentes: la versión en texto, que se envía por medio de la función `mail()` de PHP, y la versión HTML, que se envía por medio de la clase `Mail_mime`.

En este libro hemos utilizado `mail()` muchas veces, por lo que nos centraremos en el uso de la clase `Mail_mime`.

No analizaremos la clase exhaustivamente sino que explicaremos cómo la hemos utilizado en esta aplicación.

En primer lugar incluimos los archivos de clase y creamos una instancia de la clase `Mail_mime`:

```
// incluya clases de correo de PEAR
include('Mail.php');
include('Mail/mime.php');

// instancie una clase MIME y pásela al carácter de retorno de carro/nueva línea
// utilizado en este sistema
$message = new Mail_mime("\r\n");
```

Apreciará que hemos incluido dos archivos de clase. Posteriormente en esta secuencia de comandos utilizaremos la clase Mail genérica de PEAR para enviar el correo. Esta clase se incluye en su instalación de PEAR.

La clase `Mail_mime` se utiliza para crear el mensaje en formato MIME que se enviará.

Tras ello leemos las versiones en texto y en HTML del correo y las añadimos a la clase `Mail_mime`:

```
// lea la versión en texto del boletín informativo
$textfilename = "archive/$listid/$mailid/text.txt";
$tfp = fopen($textfilename, "r");
$text = fread($tfp, filesize($textfilename));
fclose($tfp);

// lea la versión HTML del boletín informativo
$htmlfilename = "archive/$listid/$mailid/index.html";
$hfp = fopen($htmlfilename, "r");
$html = fread($hfp, filesize($htmlfilename));
fclose($hfp);

// añada HTML y texto al objeto mimemail
$message->setTXTBody($text);
$message->setHTMLBody($html);
```

Seguidamente cargamos los detalles de las imágenes desde la base de datos y realizamos un bucle por las mismas, añadiendo cada una de ellas al correo que queremos enviar:

```
// cargue todas las imágenes desde disco
```

```
$imgfilename = "archive/$listid/$mailid/".mysql_result($result, $i, 0);
$imgtype = mysql_result($result, $i, 1);

// añada todas las imágenes al objeto
$message->addHTMLImage($imgfilename, $imgtype, $imgfilename, true);
```

Los parámetros que pasamos a `addHTMLImage()` son el nombre del archivo de imagen (también podríamos pasar los datos de la imagen), el tipo MIME de la imagen, el nombre del archivo de nuevo y `true` para indicar que el primer parámetro es un nombre de archivo en lugar de datos de archivo (si quisiéramos pasar sólo datos de imagen, pasaríamos los datos, el tipo MIME, un parámetro vacío y `false`). Estos parámetros son ligeramente complicados.

A continuación será necesario crear el cuerpo del mensaje, antes de poder definir los encabezados del mensaje. El cuerpo se crea de esta forma:

```
// cree el cuerpo del mensaje
$body = $message->get();
```

Tras ello podemos crear los encabezados del mensaje con una llamada a la función `headers()` de `Mail_mime`:

```
// cree los encabezados del mensaje
$from = '"'.get_real_name($admin_user).'" <'.$admin_user.'>';
$hdrarray = array(
          'From'      => $from,
          'Subject'   => $subject);
$hdrs = $message->headers($hdrarray);
```

Por último, después de configurar el mensaje, podemos enviarlo.

Para llevarlo a cabo, será necesario instanciar la clase `Mail` de PEAR y pasarla al mensaje que hemos creado. En primer lugar, instanciamos la clase:

```
// cree el objeto que se va a enviar
$sender =& Mail::factory('mail');
```

(En este caso, el parámetro `'mail'` indica a la clase `Mail` que utilice la función `mail()` de PHP para enviar mensajes.

También podríamos utilizar `'sendmail'` o `'smtp'` como valor de este parámetro con los resultados obvios.)

Seguidamente, enviamos el correo a todos nuestros suscriptores.

Para ello, recuperamos y realizamos un bucle por todos los usuarios suscritos a esta lista, utilizando `Mail send()` o la convencional `mail()`, en función de la preferencia de tipo MIME del usuario:

```
if($subscriber[2]=='H')
{
   //envíe la versión HTML a los que la hayan solicitado
   $sender->send($subscriber[1], $hdrs, $body);
}
else
{
```

```
    //envíe la versión en texto a los que no quieran correo HTML
    mail($subscriber[1], $subject, $text,
         'From: "'.get_real_name($admin_user).'" <'.$admin_user.">");
}
```

El primer parámetro de `$sender->send()` debería ser la dirección de correo electrónico del usuario; el segundo, los encabezados y el tercero, el cuerpo del mensaje.

Eso es todo. Hemos terminado la creación de la aplicación de listas de correo.

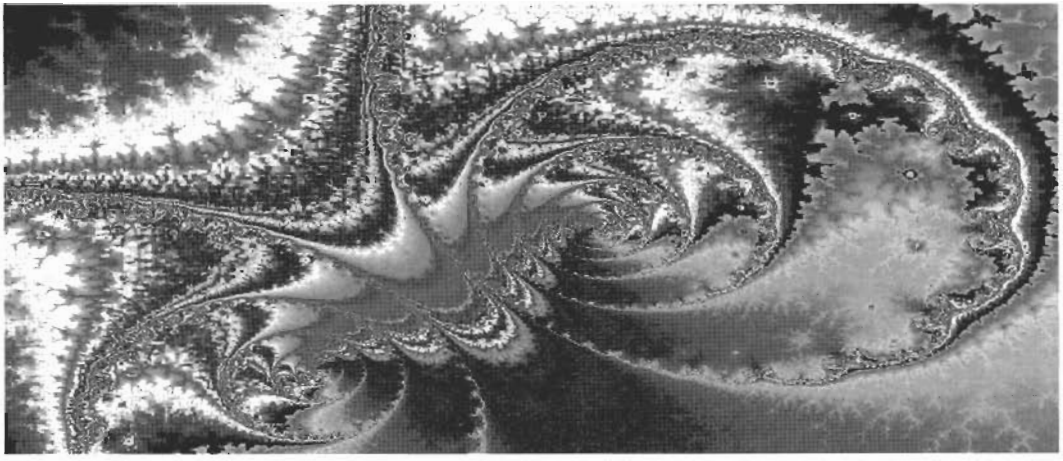
Ampliación del proyecto

Como suele ser habitual en estos proyectos, su funcionalidad se puede ampliar de distintas formas, como indicamos a continuación:

- Confirmar la pertenencia con los suscriptores, para que nadie se pueda suscribir sin su permiso. Para ello, se suele enviar un correo electrónico a sus cuentas y se elimina a los que no respondan. Este enfoque también nos permite borrar de la base de datos todas las direcciones de correo incorrectas.
- Se puede permitir al administrador que admita o rechace a los usuarios que se quieran suscribir a sus listas.
- Se puede añadir funcionalidad de listas abiertas que permita a cualquier miembro enviar correo electrónico a la lista.
- Permitir que sólo los miembros registrados puedan ver el archivo de una determinada lista de correo.
- Permitir a los usuarios buscar listas que coincidan con determinados criterios. Por ejemplo, los usuarios pueden estar interesados en boletines informativos sobre golf. Si se llega a un número concreto de boletines informativos, la búsqueda permitiría localizar ejemplares concretos.
- Si tiene pensado utilizar una lista de correo de gran tamaño, una aplicación PHP no es la mejor forma de enviar los mensajes. Un gestor de listas de correo específico como exmlm resultaría más eficaz que la invocación repetida de `mail()` en PHP. Evidentemente, siempre puede generar la interfaz de usuario en PHP, pero hacer que exmlm se encargue del trabajo duro.

A continuación

En el siguiente capítulo implementaremos una aplicación de foros Web que permitirá a los usuarios realizar debates en línea estructurados por tema y cadenas de conversación.



29
Creación de foros Web

Un método muy indicado para conseguir que los usuarios vuelvan a visitar nuestro sitio consiste en ofrecer foros Web. Se pueden utilizar para cosas tan variadas como grupos de debate sobre filosofía o como asistencia técnica de productos.

En este capítulo implementaremos un foro Web en PHP.

Una posible alternativa consiste en utilizar un paquete ya existente como Phorum para crear nuestros foros.

Los foros Web en ocasiones se denominan *tablones de debate* o *grupos de debate encadenados*.

La idea de los foros es que los usuarios puedan añadir artículos o preguntas y que otros usuarios las lean y las respondan. Cada uno de los temas de debate de un foro se denomina *cadena*.

Implementaremos un foro Web con el nombre blah-blah que tendrá la funcionalidad que describimos a continuación.

Los usuarios podrán:

- Iniciar nuevas cadenas de debate y añadir artículos
- Añadir artículos en respuesta a artículos existentes
- Ver artículos que hayan enviado
- Ver las cadenas de conversación del foro
- Ver la relación entre artículos, es decir, ver qué artículos responden a otros artículos

El problema

La configuración de un foro plantea de hecho un interesante problema. Necesitaremos alguna forma de almacenar los artículos en la base de datos, con el autor, el título, la fecha e información sobre el contenido. A primera vista, se asemeja bastante a la base de datos Book-O-Rama.

Sin embargo, el funcionamiento de la mayoría del software de foros se basa, además de en mostrar los artículos disponibles, en mostrar las relaciones entre artículos. Es decir, podemos ver qué artículos son respuestas a otros artículos (y de qué artículo provienen) y qué artículos son nuevos temas de debate.

Encontrará muchos ejemplos de foros de debate que implementan estas características, como por ejemplo Slashdot (`http://slashdot.org`).

Debe planificar cuidadosamente la forma de mostrar estas relaciones. En este sistema, los usuarios deben poder ver un mensaje concreto, una cadena de conversación en la que se incluyan las relaciones o todas las cadenas de un sistema.

También debemos permitir que los usuarios inicien nuevos temas o que respondan. Será la parte fácil.

Componentes de la solución

Como hemos mencionado anteriormente, el almacenamiento y recuperación del autor y del texto de un mensaje es sencillo.

La parte más compleja de esta aplicación es la búsqueda de una estructura de base de datos que almacene la información que queremos así como una forma de desplazarnos eficazmente por dicha estructura. La estructura de los artículos de un debate puede ser similar a la que mostramos en la figura 29.1.

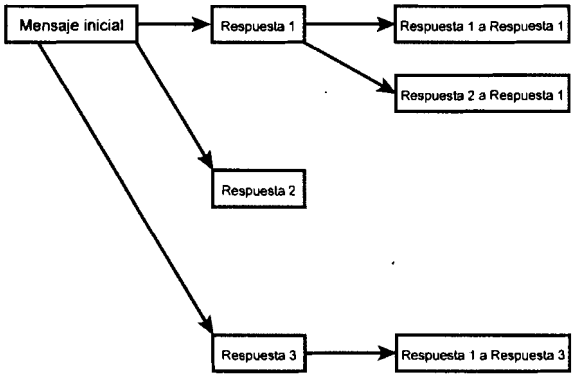


Figura 29.1. Un artículo en un debate encadenado puede ser el primer artículo de un nuevo tema, pero normalmente es una respuesta a otro artículo

En este diagrama, apreciará que hay un envío inicial que empieza un tema, con tres respuestas. Algunas de las respuestas tienen sus propias respuestas. Éstas podrían tener sus propias respuestas y así sucesivamente.

Mediante el análisis del diagrama podemos determinar la forma de almacenar y recuperar los datos de los artículos así como los enlaces entre los mismos. Este diagrama muestra una *estructura de árbol*. Si ya tiene experiencia en el mundo de la programación, sabrá que es una de las principales estructuras de datos que se utilizan. En el diagrama hay *nodos*, o artículos, y *enlaces*, o relaciones entre artículos, al igual que en cualquier estructura de árbol (si no está familiarizado con este tipo de estructura de datos, no se preocupe, ya que analizaremos sus fundamentos sobre la marcha).

Los trucos para conseguir que funcione son los siguientes:

1. Determinar la forma de asignar esta estructura de árbol al almacenamiento, en nuestro caso, en una base de datos MySQL
2. Determinar una forma de reconstruir los datos según sea necesario

Comenzaremos con la implementación de una base de datos MySQL que nos permita almacenar artículos, para lo que crearemos sencillas interfaces.

Al cargar la lista de artículos que se pueden ver, también cargaremos los encabezados de cada artículo en una clase `tree_node` de PHP. Cada una de estas clases contendrá los encabezados de un artículo así como un conjunto de respuestas para el mismo.

Las respuestas se almacenan en una matriz. Cada respuesta será a su vez un `tree_node`, que puede contener una matriz de respuestas a dicho artículo que, a su vez, son `tree_nodes` y así sucesivamente. El proceso continúa hasta llegar a lo que se denominan *nodos de hoja* del árbol, los nodos que no tienen respuestas. De esta forma obtendremos una estructura de árbol similar a la de la figura 29.1.

Veamos algunos términos. El mensaje al que contestamos se denomina nodo *principal* del nodo actual. Todas las respuestas al mensaje se denominan nodos *secundarios* del principal. Le resultará más sencillo recordarlo si se imagina esta estructura de árbol como si fuera un árbol genealógico.

El primer artículo de esta estructura de árbol, el que no tiene nodo principal, en ocasiones se denomina nodo *raíz*.

Nota

Puede que no sea demasiado intuitivo, ya que normalmente dibujamos el nodo raíz en la parte superior del diagrama, al contrario que las raíces de un árbol real.

Para generar y representar esta estructura de árbol, escribiremos funciones recursivas (que analizamos en un capítulo anterior).

Hemos optado por utilizar una clase para esta estructura ya que es la forma más sencilla de crear una estructura de datos compleja y que se pueda ampliar

dinámicamente para esta aplicación. Al mismo tiempo, conseguimos un código sencillo y elegante para realizar operaciones de cierta dificultad.

Presentación de la solución

Para comprender realmente lo que hemos hecho con este proyecto, lo más aconsejable es analizar pormenorizadamente el código, como haremos en breve. En esta aplicación, el código es menos extenso que en las anteriores, pero su complejidad es mayor.

La aplicación cuenta únicamente con tres páginas reales.

Tendremos una página de índice principal que muestre todos los artículos de los foros en forma de enlaces a los artículos. Desde aquí, podremos añadir un nuevo artículo, ver cualquier artículo de la lista o cambiar la forma en que se ve el artículo desplegando o replegando las ramas del árbol (como veremos a continuación).

Desde la vista del artículo, podrá enviar una respuesta para el mismo o ver las que ya existen.

La página de nuevo artículo nos permite introducir un nuevo mensaje, ya sea una respuesta a un mensaje existente o un nuevo mensaje que no tenga relación con los anteriores.

El diagrama de flujo del sistema se representa en la figura 29.2.

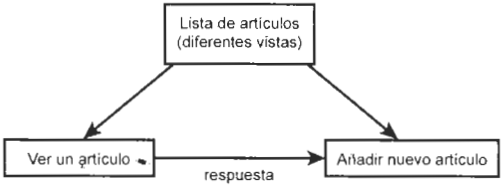


Figura 29.2. El sistema de foros blah-blah consta de tres partes principales

En la tabla 29.1 se recoge un resumen de los archivos de esta aplicación.

Tabla 29.1. Archivos de la aplicación de foros Web

Nombre	Tipo	Descripción
`index.php`	Aplicación	La página principal que verán los usuarios cuando accedan al sitio. Contiene una lista de todos los artículos del sitio, lista que pue den abrir o cerrar.
`new_post.php`	Aplicación	Formulario utilizado para enviar nuevos artículos.

Nombre	Tipo	Descripción
store_new_post.php	Aplicación	Almacena artículos introducidos en el formulario new_post.php.
view_post.php	Aplicación	Muestra un tema concreto y una lista de respuestas a dicho tema.
treenode_class.php	Biblioteca	Contiene la clase treenode, que utilizaremos para mostrar la jerarquía de los temas.
include_fns.php	Biblioteca	Agrupa a todas las restantes bibliotecas de funciones de la aplicación (el resto de archivos de tipo biblioteca que enumeramos).
data_valid_fns.php	Biblioteca	Funciones de validación de datos.
db_fns.php	Biblioteca	Funciones de conectividad de base de datos.
discussion_fns.php	Biblioteca	Funciones para almacenar y recuperar mensajes.
output_fns.php	Biblioteca	Funciones para representar HTML.
create_database.sql	SQL	SQL para definir la base de datos que necesitamos en la aplicación.

A continuación nos detendremos en la implementación.

Diseño de la base de datos

Será necesario almacenar una serie de atributos sobre cada uno de los artículos que se muestren en el foro: la persona que lo haya escrito, el título del artículo, cuándo se ha enviado y el cuerpo del artículo. Por esta razón, necesitaremos una tabla de artículos. Crearemos un Id. exclusivo para cada uno de los artículos, denominado postid. En cada artículo será necesario incluir cierta información con respecto a su posición dentro de la jerarquía. Podríamos almacenar en el artículo información sobre los nodos secundarios del mismo. Sin embargo, cada artículo puede tener numerosas respuestas, lo que puede provocar problemas a la hora de construir la base de datos. Como cada artículo sólo puede ser respuesta de otro artículo, resulta más sencillo almacenar una referencia al artículo principal, es decir, al artículo al que responde este artículo.

Por lo tanto, disponemos de los siguientes datos para almacenar en cada uno de los artículos:

- postid. Un Id. exclusivo para cada artículo

- `parent`. El `postid` del artículo principal
- `poster`. El autor de este artículo
- `title`. El título de este artículo
- `posted`. Fecha y hora del artículo que hemos enviado
- `message`. El cuerpo del artículo

Añadiremos algunas optimizaciones a todo esto.

Al tratar de determinar si un artículo tiene o no respuestas, tendremos que ejecutar una consulta para ver si hay otros artículos que lo tengan como principal. Necesitaremos esta información en todos los mensajes que mostremos. Cuantas menos consultas tengamos que realizar, con mayor rapidez se ejecutará el código. Podemos suprimir la necesidad de dichas consultas si añadimos un campo que muestre si hay o no alguna respuesta. Este campo será `children` y será booleano, es decir, su valor será 1 si el nodo tiene nodos secundarios y 0 en caso contrario.

Siempre hay que pagar un precio por las optimizaciones. En este caso hemos optado por almacenar datos redundantes. Como vamos a almacenar datos de dos formas, tendremos que prestar especial atención y asegurarnos de que las dos representaciones coinciden entre sí. Al añadir secundarios tendremos que actualizar el principal. Si permitimos que se borren secundarios, tendremos que actualizar el principal para que la base de datos sea coherente. En este proyecto no vamos a incluir una función para eliminar artículos, por lo que evitaremos parte del problema. Si tiene pensado ampliar este código, recuerde este aspecto.

Conviene mencionar que algunas bases de datos nos serán de más ayuda que otras para nuestros propósitos. Si utilizamos Oracle, mantendrá la integridad referencial por nosotros. Si utilizamos MySQL, que no es compatible con desencadenadores ni restricciones de clave secundaria, tendremos que escribir nuestras propias comprobaciones para asegurarnos de que los datos tienen sentido cada vez que añadamos o eliminemos un registro.

Realizaremos otra optimización más. Separaremos los cuerpos del mensaje del resto de los datos y los almacenaremos en una tabla independiente, ya que este atributo tendrá el tipo `text` de MySQL. Al tener este tipo en una tabla, se reduce la velocidad de las consultas a la misma. Como realizaremos numerosas consultas para generar la estructura de árbol, el proceso se ralentizará considerablemente. Si almacenamos los cuerpos de los mensajes en una tabla independiente, podremos recuperarlos cuando un usuario quiera acceder a un mensaje concreto.

MySQL puede buscar con mayor rapidez registros con un tamaño fijo que registros de tamaño variable. Si es necesario utilizar datos de tamaño variable podemos crear índices en los campos que se utilicen para buscar en la base de datos. En algunos proyectos, nos bastaría con dejar el campo de texto en el mismo registro en el que se encuentre todo lo demás y especificar índices en todas las columnas en las que realizaremos la búsqueda. Sin embargo, los índices tardan un tiempo en generarse y es muy probable que los datos de nuestros foros cambien continuamente, por lo que tendremos que volver a generarlos a menudo.

También añadiremos un atributo `area` en caso de que posteriormente decidamos implementar varias conversaciones con la aplicación.

En este caso no lo implementare-mos ya que esta opción la reservamos para utilizarla más adelante.

Una vez realizadas todas estas consideraciones, en el listado 29.1 se muestra el SQL necesario para crear la base de datos de foros.

Listado 29.1. create_database.sql. SQL para crear la base de datos de debate

```
create  database  discussion;

use discussion;

create table header
(
   parent int not null,
   poster char(20) not null,
   title char(20) not null,
   children int default 0 not null,
   area int default 1 not null,
   posted datetime not null,
   postid int unsigned not null auto_increment primary key
);

create table body
(
   postid int unsigned not null primary key,
   message text
);


grant select, insert, update, delete
on discussion.*
to discussion@localhost identified by 'password';
```

Puede crear esta estructura de base de datos si ejecuta esta secuencia de comandos a través de MySQL como se indica a continuación:

```
mysql -u root -p < create_database.sql
```

Tendrá que introducir su contraseña raíz. También debería cambiar la contraseña que hemos asignado al usuario.

Para comprender cómo se incluyen los artículos y las relaciones entre ellos en esta estructura, puede recurrir a la figura 29.3.

Como puede comprobar, el campo principal de cada artículo de la base de datos contiene el `postid` del artículo situado por encima en el árbol. El artículo principal es el artículo al que se responde.

También verá que el nodo raíz, `postid` 1, no tiene principal. Todos los nuevos temas de debate estarán en esta posición. En artículos de este tipo, almacenamos su principal como 0 en la base de datos.

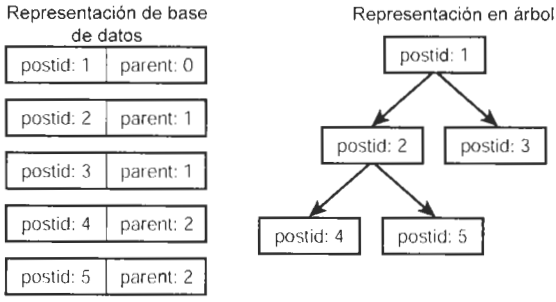


Figura 29.3. La base de datos cuenta con una estructura de árbol de forma relacional plana

Cómo ver el árbol de artículos

A continuación, necesitamos algún modo de obtener información de la base de datos y de representarla en la estructura de árbol. Lo haremos en la página principal, `index.php`. Para los objetivos de esta explicación, hemos introducido algunos envíos de ejemplo con ayuda de las secuencias de comandos `new_post.php` y `store_new_post.php`, que analizaremos en el siguiente apartado.

En primer lugar describiremos la lista de artículos ya que es la parte central del sitio. Tras ello, todo lo demás resultará más sencillo.

En la figura 29.4 se muestra la vista inicial de los artículos del sitio que verá el usuario.

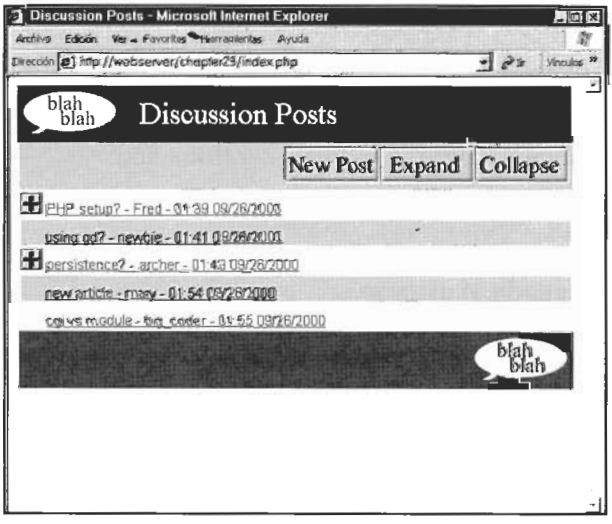


Figura 29.4. La vista inicial de la lista de artículos muestra los artículos de forma "replegada"

Aquí vemos todos los artículos iniciales. Ninguno tiene respuestas ya que todos son los primeros artículos de un determinado tema.

Comprobará que disponemos de una serie de opciones. Hay una barra de menú que nos permite añadir un nuevo envío así como desplegar o replegar la vista de los artículos.

Para comprender lo que significa, nos detendremos en los envíos. Algunos de ellos tienen símbolos más (+).

Significa que estos artículos cuentan con respuestas.

Para ver las respuestas a un determinado artículo, basta con pulsar sobre dicho símbolo.

El resultado se muestra en la figura 29.5.

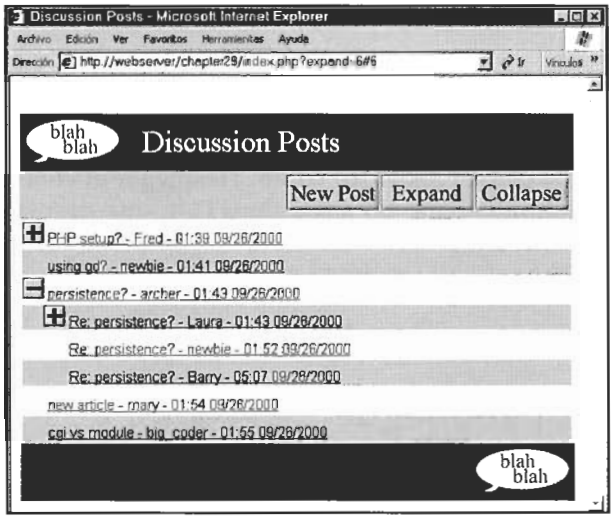


Figura 29.5. La cadena de debate sobre persistencia se ha desplegado

Como puede comprobar, al pulsar sobre el símbolo más (+) se despliegan las respuestas al primer artículo y el símbolo se convierte en un símbolo menos (-). Si lo pulsamos, se repliegan todos los artículos de esta cadena, lo que nos lleva de nuevo a la vista inicial.

También comprobará que una de las respuestas tiene un símbolo más, lo que significa que cuenta con sus propias respuestas. Esto puede continuar de forma arbitraria y puede ver cada conjunto de respuestas si pulsa sobre el correspondiente símbolo más. Las dos opciones de la barra de menús, Expand y Collapse, permiten desplegar y replegar todas las cadenas, respectivamente. En la figura 29.6 puede ver el resultado de pulsar el botón **Expand**.

Si se fija atentamente en las figuras 29.5 y 29.6 verá que hemos pasado determinados parámetros de index.php en la línea de comandos. En la figura 29.5 el URL es `http://webserver/chapter29/index.php?expand=6#6`.

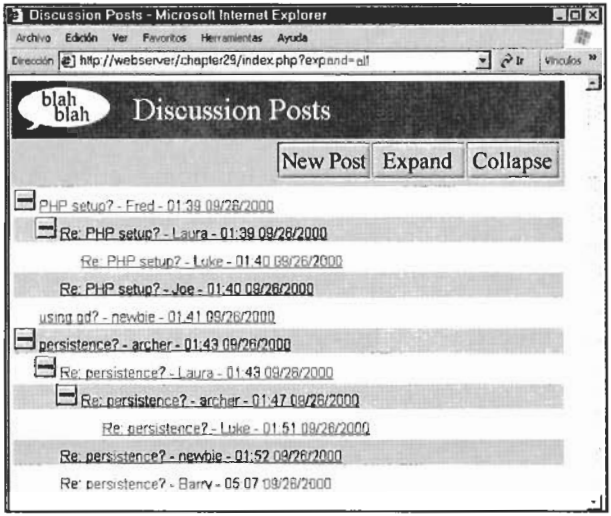


Figura 29.6. Todas las cadenas se han desplegado

La secuencia de comandos lo lee como "Desplegar el elemento con el postid 6". EL símbolo # es simplemente un ancla de HTML que desplaza la página hasta la parte que se acaba de desplegar.

En la figura 29.6, el URL es `http://webserver/chapter29/index.php?expand=all`.

Al pulsar el botón **Expand** se ha pasado el parámetro `expand` con el valor `all`.

Despliegue y repliegue

Si analizamos la secuencia de comandos `index.php` (listado 29.2) sabremos cómo se lleva a cabo esta operación.

Listado 29.2. index.php. Secuencia de comandos para crear la vista del artículo en la página principal de la aplicación

```
<?php
   include ('include_fns.php');
   session_start();

   // compruebe si se ha creado la variable de sesión
   if(!isset($HTTP_SESSION_VARS['expanded']))
   {
      $HTTP_SESSION_VARS['expanded'] = array();
   }

   // compruebe si se ha pulsado un botón de despliegue
   // expand debe ser igual a 'all' o a un postid o no estar configurado
   if(isset($HTTP_GET_VARS['expand']))
   {
```

```
    if($HTTP_GET_VARS['expand']  ==  'all')
       expand_all($HTTP_SESSION_VARS['expanded']);
    else
       $HTTP_SESSION_VARS['expanded'][$HTTP_GET_VARS['expand']] = true;
 }

 // compruebe si se ha pulsado un botón de repliegue
 // collapse puede ser igual a all o a un postid o no estar configurado
 if(isset($HTTP_GET_VARS['collapse']))
 {
    if($HTTP_GET_VARS['collapse']=='all')
       $HTTP_SESSION_VARS['expanded'] = array();
    else
    unset($HTTP_SESSION_VARS['expanded'][$HTTP_GET_VARS['collapse']]);
 }

 do_html_header('Discussion  Posts');

 display_index_toolbar();

 // muestre la versión en árbol de las conversaciones
 display_tree($HTTP_SESSION_VARS['expanded']);

 do_html_footer();
?>
```

La secuencia de comandos utiliza tres variables para realizar su trabajo, como indicamos a continuación:

- La variable de sesión `expanded`, que realiza el seguimiento de las cadenas desplegadas. Se puede conservar de una vista a otra, para poder disponer de múltiples cadenas desplegadas. Esta variable es una matriz asociativa que contiene el `postid` de los artículos cuyas respuestas se muestran desplegadas.
- El parámetro `expand`, que indica a la secuencia de comandos qué nuevas cadenas se deben desplegar.
- El parámetro `collapse`, que indica a la secuencia de comandos qué cadenas se deben replegar.

Al pulsar un símbolo más o menos, o los botones **Expand** o **Collapse**, se invoca de nuevo la secuencia `index.php` con nuevos parámetros para `expand` o `collapse`. Utilizamos `expanded` de una página a otra para saber qué cadenas deben desplegarse en una vista concreta.

Inicialmente, la secuencia de comandos inicia una sesión y añade la variable `expanded` como variable de sesión en caso de que no se haya hecho previamente.

Tras ello, la secuencia de comandos comprueba si se ha pasado un parámetro `expand` o `collapse` y modifica la matriz `expanded` en consecuencia. El código del parámetro `expand` es el siguiente:

```
if(isset($HTTP_GET_VARS['expand']))
```

```
{
   if($HTTP_GET_VARS['expand'] == 'all')
      expand_all($HTTP_SESSION_VARS['expanded']);
   else
      $HTTP_SESSION_VARS['expanded'][$HTTP_GET_VARS['expand']] = true;
}
```

Si hemos pulsado el botón **Expand**, se invoca la función `expand_all()` para añadir todas las cadenas que tengan respuestas a la matriz `expanded`, como veremos más adelante.

Si quisiéramos desplegar una cadena concreta, tendríamos que pasar un `postid` a través de expand. Para reflejarlo, añadimos una nueva entrada a la matriz expanded.

La función `expand_all()` se muestra en el listado 29.3.

Listado 29.3. Función expand_all() de discussion_fns.php. Procesa la matriz $expanded para desplegar todas las cadenas del foro

```
function  expand_all(&$expanded)
{
   // marque todas las cadenas con secundarios cuando se muestren desplegadas
   $conn = db_connect();
   $query = 'select postid from header where children = 1';
   $result = mysql_query($query);
   $num = mysql_numrows($result);
   for($i = 0; $i<$num; $i++)
   {
      $expanded[mysql_result($result, $i, 0)]=true;
   }
}
```

Esta función ejecuta una consulta de base de datos para determinar qué cadenas del foro cuentan con respuestas, como se aprecia a continuación:

```
select postid from header where children = 1
```

Todos los artículos devueltos se añaden a la matriz `expanded`. Ejecutamos esta consulta para ahorrar tiempo más adelante. Bastaría con añadir todos los artículos a la lista desplegada, pero sería inútil intentar procesar respuestas que no existen.

El repliegue de artículos funciona de forma similar pero al contrario, como se indica a continuación:

```
if(isset($HTTP_GET_VARS['collapse']))
{
   if($HTTP_GET_VARS['collapse']=='all')
      $HTTP_SESSION_VARS['expanded']  =  array();
   else
      unset($HTTP_SESSION_VARS['expanded'][$HTTP_GET_VARS['collapse']]);
}
```

Puede eliminar elementos de la matriz `expanded` si anula su configuración. Eliminamos la cadena que se va a replegar o anulamos toda la matriz si toda la página se va a replegar.

Todas estas operaciones son de preprocesamiento, por lo que sabemos qué artículos se deben mostrar y cuáles no. La parte principal de esta secuencia de comandos es la invocación de `display_tree($HTTP_SESSION_VARS['expanded'])`, que genera realmente el árbol de artículos mostrados.

Cómo mostrar los artículos

A continuación analizaremos la función `display_tree()`, que mostramos en el listado 29.4.

Listado 29.4. Función display_tree() de output_fns.php. Crea el nodo raíz de la estructura de árbol

```
function display_tree($expanded, $row = 0, $start = 0)
{
   // muestre la vista en árbol de las conversaciones

   global $table_width;
   echo "<table width=\"$table_width\">;

   // compruebe si se muestra toda la lista o una sublista
   if($start>0)
      $sublist = true;
   else
      $sublist = false;

   // construya la estructura de árbol para representar el resumen de las conversaciones
   $tree = new treenode($start, '', '', '', 1, true, -1, $expanded, $sublist);

   // indique al árbol que se muestre
   $tree->display($row,  $sublist);

   echo '</table>';
}
```

El papel principal de esta función consiste en crear el nodo raíz de la estructura de árbol. La utilizaremos tanto para mostrar el índice completo como para crear subárboles de respuestas en la página `view_post.php`.

Como puede comprobar, adopta tres parámetros.

El primero, `$expanded`, es la lista de `postid` de artículos que se representan desplegados.

El segundo, `$row`, es un indicador del número de fila que se utilizará para determinar los colores alternos de las filas de la lista.

El tercer parámetro, `&start`, indica a la función dónde debe empezar a mostrar los artículos. Se trata del `postid` del nodo raíz del árbol que se va a crear y a representar.

Si lo mostráramos todo, como hicimos en la página principal, sería 0 (cero), lo que significa que se mostrarían todos los artículos sin principal.

Si este parámetro es 0, establecemos `$sublist` como `false` y mostramos todo el árbol.

Si el parámetro es mayor que cero, lo utilizamos como nodo raíz del árbol que se va a mostrar, definimos `$sublist` como `true` y generamos y mostramos sólo parte del árbol (lo utilizaremos en la secuencia de comandos `view_post.php`).

Lo más importante que hace esta función es instanciar una instancia de la clase `treenode` que representa la raíz del árbol.

Realmente no es un artículo pero actúa como principal de todos los artículos de primer nivel, que no tienen principal.

Una vez construido el árbol, basta con invocar su función de representación para mostrar la lista de artículos.

Uso de la clase treenode

El código de la clase `treenode` se recoge en el listado 29.5 (en capítulos anteriores se describe el funcionamiento de las clases).

Listado 29.5. Clase treenode de treenode_fns.php. Parte principal de la aplicación

```
<?php
// las funciones para cargar, construir y
// representar el árbol se encuentran en este archivo

class treenode
{
   // cada nodo del árbol tiene variables miembro que contienen
   // todos los datos de un mensaje a excepción del cuerpo
   var $m_postid;
   var $m_title;
   var $m_poster;
   var $m_posted;
   var $m_children;
   var $m_childlist;
   var $m_depth;

   function treenode($postid, $title, $poster, $posted, $children,
                     $expand, $depth, $expanded, $sublist)
   {
      // el constructor define las variables miembro, pero
      // crea recursivamente las partes inferiores del árbol
      $this->m_postid = $postid;
      $this->m_title = $title;
      $this->m_poster = $poster;
      $this->m_posted = $posted;
      $this->m_children =$children;
      $this->m_childlist = array();
      $this->m_depth = $depth;

      // sólo nos importa lo que aparece por debajo de este nodo si
      // tiene secundarios y se ha marcado para ser desplegado
      // las sublistas siempre se despliegan
```

```
    if(($sublist||$expand) && $children)
    {
        $conn = db_connect();

        $query = "select * from header where parent = $postid order by posted";
        $result = mysql_query($query);

        for ($count=0; $row = @mysql_fetch_array($result); $count++)
        {
            if($sublist||$expanded[ $row['postid'] ] == true)
                $expand = true;
            else
                $expand = false;
            $this->m_childlist[$count]= new treenode($row['postid'],$row['title'],
                                        $row['poster'],$row['posted'],
                                        $row['children'], $expand,
                                        $depth+1, $expanded, $sublist);
        }
    }
}


function display($row, $sublist = false)
{
    // como se trata de un objeto, se encarga de mostrarse a sí mismo

    // $row nos indica en qué fila nos encontramos
    // para saber qué color le corresponde

    // $sublist nos indica si estamos en la página principal
    // o en la de mensajes. En las páginas de mensaje
    // $sublist = true.
    // En una sublista, todos los mensajes se despliegan y no hay
    // símbolos "+" o "-".

    // si se trata del nodo de raíz vacío, no lo muestre
    if($this->m_depth>-1)
    {
        //filas de colores alternos
        echo '<tr><td bgcolor="';
        if ($row%2)
            echo '#cccccc">';
        else
            echo '#ffffff">';

        // el sangrado indica la profundidad de la estructura de anidación
        for($i = 0; $i<$this->m_depth; $i++)
        {
            echo '<img src="images/spacer.gif" height="22"
                        width="22" alt="" valign="bottom">';
        }

        // muestre  + o -, o un espaciador
        if ( !$sublist && $this->m_children && sizeof($this->m_childlist))
```

```
        // estamos en la página principal, con algunos secundarios, todos desplegados
        {
           // si está desplegado, ofrezca un botón para replegar
           echo '<a href="index.php?collapse='.
                        $this->m_postid.'#'.$this->m_postid.'"
              ><img src="images/minus.gif" valign="bottom"
              height="22" width="22" alt="Collapse Thread" border="0"></a>';
        }
        else if(!$sublist && $this->m_children)
        {
           // si está replegado, ofrezca un botón para desplegar
           echo '<a href="index.php?expand='.
                 $this->m_postid.'#'.$this->m_postid.'"><img src="images/plus.gif"
                 height="22" width="22" alt="Expand Thread" border="0"></a>';
        }
        else
        {
           // no hay secundarios o se trata de una sublista; no ofrezca ningún
           // botón
           echo '<img src="images/spacer.gif" height="22" width="22"
                       alt="" valign="bottom">';
        }


        echo " <a name = $this->m_postid ><a href =
                 'view_post.php?postid=$this->m_postid'>$this->m_title -
                 $this->m_poster - ".reformat_date($this->m_posted).'</a>';
        echo '</td></tr>';

        // incremente el contador de filas para alternar colores
        $row++;
     }
     // invoque la representación de todos los secundarios de este nodo
     // un nodo sólo tiene secundarios en su lista si está desplegado
     $num_children = sizeof($this->m_childlist);
     for($i = 0; $i<$num_children; $i++)
     {
        $row = $this->m_childlist[$i]->display($row, $sublist);
     }
     return $row;
  }

};

?>
```

Esta clase contiene la funcionalidad que controla la vista de árbol en la aplicación.

Una instancia de la clase `treenode` contiene detalles sobre un determinado envío y enlaces a todas las respuestas de dicha clase. Esto nos proporciona las siguientes variables miembro:

```
var $m_postid;
```

```
var $m_title;
var $m_poster;
var $m_posted;
var $m_children;
var $m_childlist;
var $m_depth;
```

Comprobará que `treenode` no contiene el cuerpo de un artículo. No es necesario cargarlo hasta que el usuario acceda a la secuencia de comandos `view_post.php`. Tendremos que hacerlo con cierta rapidez ya que estamos realizando una extensiva manipulación de datos para mostrar la lista de árbol y tendremos que volver a calcular cuando se actualiza la página o cuando se pulsa un botón.

El modelo de notación de estas variables utiliza un modelo muy empleado en aplicaciones OO: las variables empiezan por `m_` para recordarnos que se trata de variables miembro de la clase.

La mayoría de estas variables se corresponde directamente a filas de la tabla `header` de nuestra base de datos.

Las excepciones son `$m_childlist` y `$m_depth`. Utilizaremos la variable `$m_childlist` para almacenar las respuestas a este artículo. La variable `$m_depth` almacenará el número de niveles de árbol en el que nos encontramos, que utilizaremos para crear la representación.

La función de construcción determina los valores de todas las variables, como se indica a continuación:

```
function treenode($postid, $title, $poster, $posted, $children,
                  $expand, $depth, $expanded, $sublist)
{
   // el constructor define las variables miembro, pero
   // crea recursivamente las partes inferiores del árbol
   $this->m_postid = $postid;
   $this->m_title  =  $title;
   $this->m_poster  =  $poster;
   $this->m_posted = $posted;
   $this->m_children =$children;
   $this->m_childlist = array();
   $this->m_depth = $depth;
```

Al construir el `treenode` raíz a partir de `display_tree()` de la página principal, lo que estamos creando en realidad es un nodo ficticio sin artículos asociados. Pasamos algunos valores iniciales:

```
$tree = new treenode($start, '', '', '', 1, true, -1, $expanded, $sublist);
```

De esta forma se crea un nodo raíz con un `$postid` cero, que se puede utilizar para buscar todos los envíos de primer nivel que tengan un principal cero. Definimos la profundidad como -1 ya que realmente este nodo no forma parte de la representación.

Todos los envíos de primer nivel tendrán una profundidad cero y se situarán en la izquierda de la pantalla. Los niveles inferiores siguientes se desplazan hacia la derecha.

En este constructor, lo más importante es que se instancian los nodos secundarios de este nodo. El proceso se inicia comprobando si resulta necesario desplegar los nodos secundarios.

Solamente se llevará a cabo este proceso si un nodo tiene nodos secundarios y hemos optado por mostrarlos:

```
if(($sublist||$expand) && $children)
{
   $conn = db_connect();
```

Tras ello nos conectamos a la base de datos y recuperamos todos los envíos secundarios, como se indica a continuación:

```
$query = "select * from header where parent = $postid order by posted";
$result = mysql_query($query);
```

Seguidamente completamos la matriz `$m_childlist` con instancias de la clase `treenode`, que contienen las respuestas a los envíos almacenados en este `treenode`, como se indica a continuación:

```
for ($count=0; $row = @mysql_fetch_array($result); $count++)
{
   if($sublist||$expanded[ $row['postid'] ] == true)
      $expand = true;
   else
      $expand = false;
   $this->m_childlist[$count]= new treenode($row['postid'],$row['title'],
                               $row['poster'],$row['posted'],
                               $row['children'], $expand,
                               $depth+1, $expanded, $sublist);
}
```

Esta última línea crea el nuevo `treenode` y sigue exactamente el mismo proceso que acabamos de describir pero en el siguiente nivel inferior.

Es la parte recursiva. Un nodo de árbol principal invoca el constructor `treenode`, le pasa su propio `postid` como principal y añade uno a su propia profundidad antes de pasarlo.

Tras ello se crea cada uno de los `treenode`, así como sus propios secundarios, hasta que ya no haya respuestas o niveles que desplegar.

Finalmente, invocamos la función de representación del `treenode` raíz (de nuevo en `display_tree()`), como se indica a continuación:

```
$tree->display($row, $sublist);
```

La función `display()` comprueba en un principio si se trata del nodo raíz ficticio:

```
if($this->m_depth>-1)
```

De esta forma, podemos eliminar el nodo ficticio de la pantalla. Sin embargo, no queremos ignorar por completo el nodo raíz. No queremos que aparezca pero tiene que notificar a sus nodos secundarios que se muestren.

Tras ello, la función empieza a crear la tabla que contiene los artículos. Utiliza el operador de módulo (%) para determinar el color de fondo que tendrá esta fila (de forma que sean alternos):

```
// filas de colores alternos
echo '<tr><td bgcolor = ';
if ($row%2)
   echo "'#cccccc'>";
else
   echo  "'#ffffff'>";
```

Seguidamente utiliza la variable miembro `$m_depth` para determinar el sangrado del elemento actual.

Si se fija en las figuras anteriores, comprobará que cuanto más profundo es el nivel de una respuesta, mayor es el sangrado.

Para ello, utilizamos lo siguiente:

```
// el sangrado indica la profundidad de la estructura de anidación
for($i = 0; $i<$this->m_depth; $i++)
{
   echo '<img src="images/spacer.gif" height="22"
                  width="22" alt="" valign="bottom">';
}
```

La siguiente parte de la función determina si se incluye un botón más (+) o menos (-), o no se incluye nada:

```
// muestre + o -, o un espaciador
if ( !$sublist && $this->m_children && sizeof($this->m_childlist))
// estamos en la página principal, hay algunos secundarios, todos desplegados
{
   // si está desplegado, ofrezca un botón para replegar
        echo '<a href="index.php?collapse='.
              $this->m_postid.'#'.$this->m_postid.'"
        ><img src="images/minus.gif" valign="bottom"
        height="22" width="22" alt="Collapse Thread" border="0"></a>';
}
else if(!$sublist && $this->m_children)
{
   // si está replegado, ofrezca un botón para desplegar
   echo '<a href = "index.php?expand='.
        $this->m_postid.'#'.$this->m_postid.'"><img src="images/plus.gif"
        height="22" width="22" alt="Expand Thread" border="0"></a>';
}
else
{
   // no hay secundarios o se trata de un sublista; no ofrecemos ningún botón
   echo '<img src="images/spacer.gif" height="22" width="22"
              alt="" valign="bottom">';
}
```

A continuación, mostramos los detalles de este nodo:

```
echo " <a name = $this->m_postid ><a href =
          'view_post.php?postid=$this->m_postid'>$this->m_title -
```

```
            $this->m_poster - ".reformat_date($this->m_posted).'</a>';
echo '</td></tr>';
```

Cambiamos el color de la siguiente fila:

```
// incremente el contador de filas para alternar colores
$row++;
```

Tras ello, todos los `treenode` ejecutan un código, incluyendo el nodo raíz:

```
// invoque la representación en todos los secundarios de este nodo
// un nodo sólo tiene secundarios en su lista si está desplegado
$num_children = sizeof($this->m_childlist);
for($i = 0; $i<$num_children; $i++)
{
   $row = $this->m_childlist[$i]->display($row, $sublist);
}
return  $row;
```

Se trata de nuevo de una función recursiva que invoca todos los secundarios de este nodo para representarlos. Les pasamos el color de fila actual y conseguimos que lo devuelvan cuando terminen, para poder realizar el seguimiento del color alterno. Eso es todo para la clase. El código es bastante complejo. Puede experimentar y probar la aplicación, y después volver a estudiarlo si está conforme con los resultados que genera.

Cómo ver artículos individuales

La invocación de `display_tree()` nos ofrece una serie de enlaces a un conjunto de artículos.

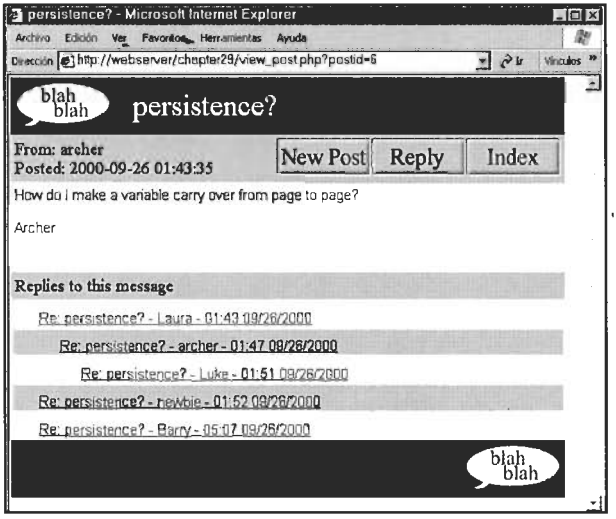


Figura 29.7. Ahora podemos ver el cuerpo del mensaje de este envío

Si pulsamos sobre uno de estos artículos, accederemos a la secuencia de comandos `view_post.php`, con un parámetro del `postid` del artículo que queremos ver. En la figura 29.7 se muestra un ejemplo de esta secuencia de comandos.

Esta secuencia de comandos muestra el cuerpo del mensaje así como todas las respuestas al mismo. Comprobará que, de nuevo, las respuestas se muestran en forma de árbol, pero totalmente desplegadas y sin botones más o menos. Es el efecto de `$sublist`.

El código de `view_post.php` se recoge en el listado 29.6.

Listado 29.6. view_post.php. Muestra el cuerpo de un mensaje

```
<?php
   // incluya bibliotecas de funciones
   include ('include_fns.php');

   $postid = $HTTP_GET_VARS['postid'];
   // obtenga los detalles del mensaje
   $post = get_post($postid);

   do_html_header($post['title']);

   // muestre el mensaje
   display_post($post);

   // si el mensaje tiene respuestas, muéstrelas en vista de árbol
   if($post['children'])
   {
      echo '<br /><br />';
      display_replies_line();
      display_tree($HTTP_SESSION_VARS['expanded'], 0, $postid);
   }

   do_html_footer();
?>
```

Esta secuencia de comandos utiliza tres llamadas de función principales para realizar su trabajo:

La función `get_post()` obtiene los detalles de la función de la base de datos. En el listado 29.7 puede ver su código.

Listado 29.7. Función get_post() de discussion_fns.php. Recupera un mensaje de la base de datos

```
function get_post($postid)
{
   // extraiga un mensaje de la base de datos y devuélvalo como matriz

   if(!$postid) return false;

   $conn = db_connect();

   // obtenga de 'header' toda la información de los encabezados
```

```
    $query = "select * from header where postid = $postid";
    $result = mysql_query($query);
    if(mysql_numrows($result)!=1)
       return  false;
    $post  =  mysql_fetch_array($result);

    // obtenga el mensaje del cuerpo y añádalo al resultado anterior
    $query = "select * from body where postid = $postid";
    $result2 = mysql_query($query);
    if(mysql_numrows($result2)>0)
    {
       $body = mysql_fetch_array($result2);
       if($body)
       {
          $post['message'] = $body['message'];
       }
    }
    return $post;
}
```

Esta función, al asignarle un `postid`, ejecuta las dos consultas necesarias para recuperar el encabezado y el cuerpo del mensaje, y los combina en una sola matriz asociativa que devuelve como resultado.

Tras ello, se pasan los resultados de esta función a la función `display_post()` de `ouput_fns.php`. Simplemente imprime la matriz con formato HTML, por lo que no la hemos incluido.

Por último, la secuencia de comandos `view_post.php` comprueba si hay alguna respuesta para este artículo e invoca `display_tree()` para mostrarlas en formato de sublista, es decir, completamente desplegadas sin símbolos más o menos.

Cómo añadir nuevos artículos

Después de todo esto, pasaremos a ver cómo se añade un tema al foro. Un usuario puede hacerlo de dos formas: en primer lugar, si pulsa el botón **New Post** de la página de índice y, por otra parte, si pulsa el botón **Reply** en la página `view_post.php`.

Ambas acciones activan la misma secuencia de comandos, `new_post.php`, pero con distintos parámetros. En la figura 29.8 puede ver el resultado de `new_post.php` cuando se accede a esta secuencia de comandos tras pulsar el botón **Reply**.

En primer lugar, fíjese en el URL: `http://webserver/chapter29/new_post.php?parent=5`.

El parámetro pasado como `parent` será el `postid` principal del nuevo envío. Si pulsa **New Post** en lugar de **Reply**, obtendrá `parent=0` en el URL.

Seguidamente, comprobará que en una respuesta, el texto del mensaje original se añade y se marca con un carácter ">", al igual que en la mayoría de programas de lectura de correo o de noticias. Por último, de forma predeterminada el título de este mensaje adopta el título del mensaje original con el prefijo "Re:".

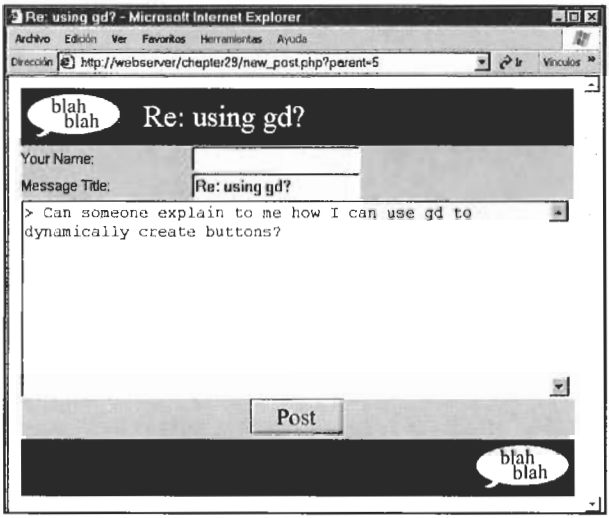


Figura 29.8. En las respuestas se añade y se marca automáticamente el texto del original

El código que genera este resultado se reproduce en el listado 29.8.

Listado 29.8. new_post.php. Permite a un usuario introducir un nuevo tema o responder a uno ya existente

```
<?php
   include ('include_fns.php');

   $title   =  $HTTP_POST_VARS['title'];
   $poster  = $HTTP_POST_VARS['poster'];
   $message = $HTTP_POST_VARS['message'];

   if(isset($HTTP_GET_VARS['parent']))
      $parent = $HTTP_GET_VARS['parent'];
   else
      $parent = $HTTP_POST_VARS['parent'];


   if(!$area)
      $area = 1;

   if(!$error)
   {
      if(!$parent)
      {
         $parent = 0;
         if(!$title)
            $title = 'New Post';
      }
      else
      {
```

```
        // obtenga el nombre del mensaje enviado
        $title = get_post_title($parent);

        // adjunte Re:
        if(strstr($title, 'Re: ') == false )
           $title = 'Re: '.$title;

        //compruebe que el título encaja en la base de datos
        $title = substr($title, 0, 20);

        //adjunte un estilo de comillas al mensaje al que esté respondiendo
        $message = add_quoting(get_post_message($parent));
     }
  }
  do_html_header($title);

  display_new_post_form($parent, $area, $title, $message, $poster);

  if($error)
  {
     echo 'Your message was not stored.
          Make sure you have filled in all fields and try again.';
  }

  do_html_footer();
?>
```

Después de la configuración inicial, la secuencia de comandos comprueba si el principal es cero o no. Si es cero, se trata de un nuevo tema y requiere acciones adicionales. Si se trata de una respuesta (`$parent` es el `postid` de un artículo existente), la secuencia de comandos prosigue y define el título y el texto del mensaje original:

```
// obtenga el nombre del mensaje enviado
$title = get_post_title($parent);

// adjunte Re:
if(strstr($title, 'Re: ') == false )
   $title = 'Re: '.$title;

//compruebe que el título encaja en la base de datos
$title = substr($title, 0, 20);

//adjunte un estilo de comillas al mensaje al que esté respondiendo
$message = add_quoting(get_post_message($parent));
```

Las funciones que utiliza son `get_post_title()`, `get_post_message()` y `add_quoting()`. Todas ellas se encuentran en la biblioteca de funciones `discussion_fns.php`. A continuación las mostramos en los listados 29.9, 29.10 y 29.11, respectivamente.

Listado 29.9. Función get_post_title() de discussion_fns.php. Recupera de la base de datos el título de un mensaje

```
function get_post_title($postid)
```

```
{
   // extraiga de la base de datos el nombre de un mensaje

   if(!$postid)  return  '';

   $conn = db_connect();

   //obtenga de 'header' toda la información sobre los encabezados
   $query = "select title from header where postid = $postid";
   $result = mysql_query($query);
   if(mysql_numrows($result)!=1)
      return '';
   return  mysql_result($result, 0, 0);

}
```

Listado 29.10. Función get_post_message() de discussion_fns.php. Recupera de la base de datos el cuerpo de un mensaje

```
function get_post_message($postid)
{
   // extraiga de la base de datos el contenido de un mensaje

   if(!$postid) return '';


   $conn = db_connect();

   $query = "select message from body where postid = $postid";
   $result = mysql_query($query);
   if(mysql_numrows($result)>0)
   {
      return  mysql_result($result,0,0);
   }
}
```

Las dos primeras funciones recuperan el encabezado y el cuerpo de un mensaje de la base de datos.

Listado 29.11. Función add_quoting() de discussion_fns.php. Sangra el texto de un mensaje con símbolos ">"

```
function  add_quoting($string,  $pattern  =  '>  ')
{
   // añada un estilo de comillas para marcar el texto que aparecerá
   // entrecomillado en la respuesta
   return $pattern.str_replace("\n", "\n$pattern", $string);
}
```

La función `add_quoting()` vuelve a aplicar formato a la cadena para que cada línea del texto original empiece con un símbolo, que de forma predeterminada es >.

Después de que el usuario introduzca su respuesta y pulse el botón **Post**, accede a la secuencia de comandos `store_new_post.php`. En la figura 29.9 puede ver un ejemplo del resultado de esta secuencia.

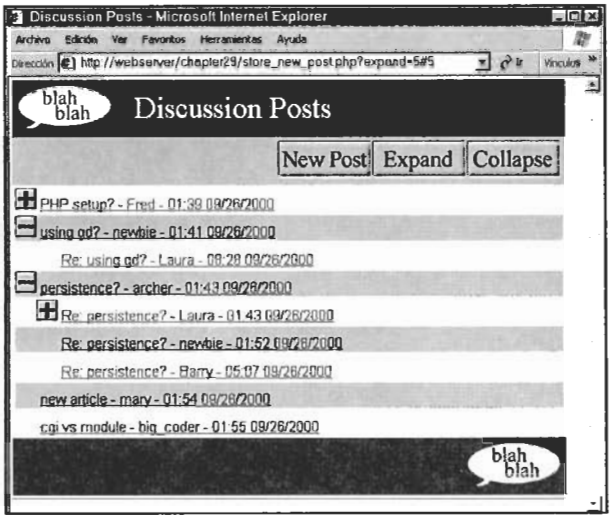


Figura 29.9. Ahora se puede ver el nuevo tema en el árbol

El nuevo tema aparece en la imagen, debajo de `Re: using gd? - Laura - 08:28 09/26/2000`. Aparte de esto, la página es idéntica a la página `index.php` convencional.

El código de `store_new_post.php` se muestra en el listado 29.12.

Listado 29.12. store_new_post.php. Añade el nuevo tema a la base de datos

```
<?php
   include ('include_fns.php');
   if($id = store_new_post($HTTP_POST_VARS))
   {
      include ('index.php');
   }
   else
   {
      $error = true;
      include ('new_post.php');
   }

?>
```

Como puede comprobar, es una secuencia de comandos breve. Su tarea principal consiste en invocar la función `store_new_post()`. Esta página carece de contenido visual propio.

Si el almacenamiento es satisfactorio, veremos la página de índice.

En caso contrario, volveremos a la página `new_post.php` para que el usuario lo intente de nuevo.

La función `store_new_post()` se muestra en el listado 29.13.

Listado 29.13. Función store_new_post() de discussion_fns.php. Valida y almacena el nuevo tema en la base de datos

```
function store_new_post($post)
{
   // valide y almacene un nuevo mensaje enviado

   $conn = db_connect();
   // compruebe que no hay campos vacíos
   if(!filled_out($post))
   {
      return false;
   }
   $post = clean_all($post);

   //asegúrese de que existe el principal
   if($post['parent']!=0)
   {
      $query = "select postid from header where postid = '".$post['parent']."'";
      $result = mysql_query($query);
      if(mysql_numrows($result)!=1)
      {
         return false;
      }
   }

   // compruebe que no es un duplicado
   $query ="select header.postid from header, body where
            header.postid = body.postid and
            header.parent = ".$post['parent']." and
            header.poster = '".$post['poster']."' and
            header.title = '".$post['title']."' and
            header.area = ".$post['area']." and
            body.message = '".$post['message']."'";
   $result = mysql_query($query);
   if (!$result)
   {
      return false;
   }
   if(mysql_numrows($result)>0)
      return mysql_result($result, 0, 0);

   $query ="insert into header values
            ('".$post['parent']."',
            '".$post['poster']."',
            '".$post['title']."',
            0,
            '".$post['area']."',
            now(),
            NULL
            )";
   $result = mysql_query($query);
   if (!$result)
```

```
    {
        return false;
    }

    // fíjese en que ahora el principal tiene un secundario
    $query = 'update header set children = 1 where postid = '.$post['parent'];
    $result = mysql_query($query);
    if (!$result)
    {
        return false;
    }

    // busque el Id. del mensaje. Puede haber varios encabezados
    // idénticos a excepción del Id. y probablemente de la fecha de envío
    $query = "select header.postid from header left join body
            on header.postid = body.postid
                    where parent = '".$post['parent']."'
                    and poster = '".$post['poster']."'
                    and title = '".$post['title']."'
                    and body.postid is NULL";
    $result = mysql_query($query);
    if (!$result)
    {
        return false;
    }
    if(mysql_numrows($result)>0)
        $id = mysql_result($result, 0, 0);

    if($id)
    {
        $query = "insert into body values ($id, '".$post['message']."')";
        $result = mysql_query($query);
        if (!$result)
        {
            return false;
        }

        return $id;
    }

}
```

Es una función extensa, pero no excesivamente compleja.

La longitud se debe a que para añadir un tema es necesario añadir entradas a las tablas de encabezado y de cuerpo, así como actualizar la fila del artículo principal en la tabla de encabezados para indicar que ahora tiene secundarios.

Con esto llegamos al final del código para la aplicación de foros Web.

Ampliaciones

Existen distintas ampliaciones que se pueden realizar en este proyecto:

- Se puede añadir navegación a las opciones de vista, para que desde un tema se pueda acceder al siguiente mensaje, al mensaje anterior, al mensaje de la siguiente cadena o al de la cadena anterior.
- Se puede añadir una interfaz de administración para crear nuevos foros y añadir temas antiguos.
- Se puede añadir autenticación de usuarios para que solamente los usuarios registrados puedan enviar temas.
- Se puede añadir algún tipo de moderador o de mecanismo censor.

Puede recurrir a sistemas existentes si necesita nuevas ideas.

Uso de un sistema existente

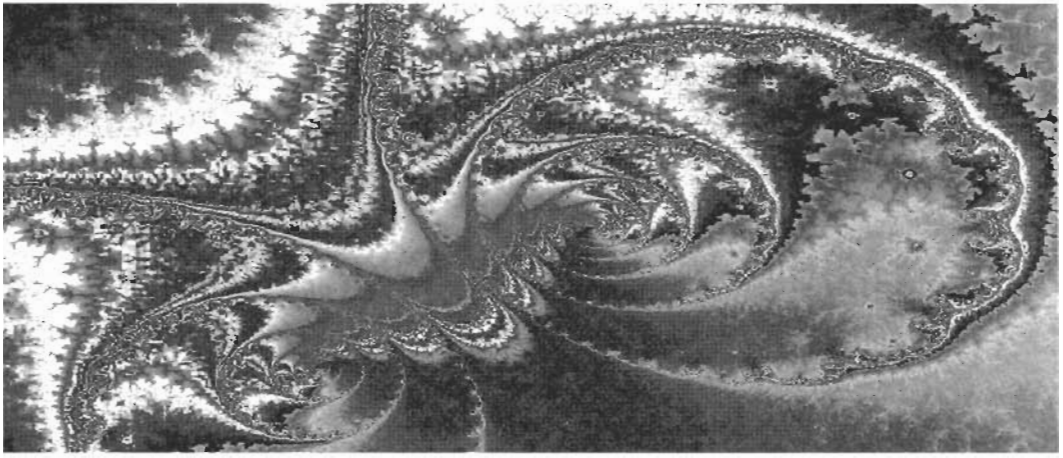
Existen algunos sistemas que conviene destacar.

Phorum es un proyecto de foros Web de código abierto. La navegación y la semántica son diferentes de la de nuestro sistema pero su estructura resulta bastante sencillo personalizar para que se ajuste a la de nuestro sitio. Una de las características más notables de phorum es que el usuario lo puede configurar bien en vista con cadenas o en vista plana. Encontrará más información al respecto en la dirección `http://www.phorum.org`.

Otro proyecto muy interesante es phpslash. Se trata de un puerto del software utilizado para ejecutar los foros de debate Slashdot. Aunque el software original está escrito en Perl, existe una versión disponible en PHP. La puede obtener en `http://www.phpslash.org`.

A continuación

En el siguiente capítulo utilizaremos el formato de documento PDF para crear documentos atractivos, que se impriman de forma coherente y que, de algún modo, no se puedan falsificar. Lo podremos aplicar a una amplia gama de aplicaciones basadas en servicios, como por ejemplo para generar contratos en línea.



30
Generación de documentos personalizados en PDF

En sitios dirigidos por servicios, en ocasiones resulta necesario entregar documentos personalizados generados en respuesta a entradas de nuestros visitantes. Se pueden utilizar para ofrecer un formulario completado de forma automática o para generar documentos personalizados, como por ejemplo documentos legales, cartas o certificados.

En el ejemplo de este capítulo presentaremos al usuario una página en línea de evaluación de conocimientos y generaremos un certificado.

Nos centraremos en los siguientes aspectos:

- Uso de procesamiento de cadenas PHP para integrar una plantilla con los datos de un usuario para crear documentos RTF (Formato de texto enriquecido).
- Uso de un enfoque similar para generar un documento PDF (Formato de documento portable).
- Uso de las funciones PDFlib de PHP para generar un documento PDF similar.

El problema

Nuestra intención es proporcionar a los usuarios un examen con una serie de preguntas. Si responden correctamente a la mayoría de las preguntas, generaremos un certificado que muestre que han aprobado el examen.

Para que un ordenador pueda calificarlos, las preguntas constarán de varias opciones, es decir, estarán formadas por una pregunta y una serie de posibles respuestas.

Sólo una de las posibles respuestas será correcta.

Si un usuario obtiene una nota de aprobado, se le otorgará un certificado.

El formato de archivo ideal de nuestro certificado debería ser el siguiente:

1. Debería ser fácil de diseñar
2. Debería incluir varios elementos diferentes, como imágenes en mapa de bits o vectoriales
3. La impresión del mismo tendrá que ser de gran calidad
4. Debería descargarse en un pequeño archivo
5. Debería generarse casi al instante
6. El coste de producción debería ser bajo
7. Debería funcionar en la mayoría de los sistemas
8. Debería resultar difícil de duplicar o modificar fraudulentamente
9. No debería necesitarse ningún software especial para verlo o imprimirlo
10. Debería mostrarse e imprimirse coherentemente para todos los receptores

Al igual que muchas otras decisiones que debemos tomar, intentaremos conseguir un formato que cumpla el mayor número posible de estos diez atributos.

Evaluación de formatos de documento

La decisión más importante que debemos tomar es el formato que tendrá el certificado. Entre las opciones posibles tenemos el papel, texto ASCII, HTML, Microsoft Word o el formato de cualquier otro procesador de texto, formato RTF, PostScript y PDF. Con los diez atributos enumerados anteriormente, podemos comparar algunas de estas opciones.

Papel

La entrega del certificado en papel tiene algunas ventajas evidentes. Tenemos control total sobre el proceso. Podemos saber el aspecto exacto de cada certificado antes de enviarlo al receptor. No tenemos que preocuparnos del software ni del ancho de banda y el certificado se puede imprimir con medidas antifalsificación.

Cumpliría todos nuestros requisitos a excepción del 5 y el 6. El certificado no se podría crear y enviar de de forma rápida. El envío postal podría demorarse durante días o semanas, en función de nuestra ubicación y la del receptor.

Cada certificado tendrá unos costes de impresión, de franqueo y también de entrega. La entrega electrónica automática resultaría más económica.

ASCII

La entrega de documentos en texto ASCII o texto sencillo tiene sus propias ventajas. La compatibilidad no será un problema. El ancho de banda necesario no será muy elevado, por lo que los costes serán bajos. La sencillez del resultado final facilitará el diseño y acelerará la creación de una secuencia de comandos.

No obstante, si ofrecemos a nuestros usuarios un archivo ASCII, el control sobre el aspecto del certificado será mínimo. No podemos controlar las fuentes ni los saltos de página. Solamente podremos incluir texto y no controlaremos el formato. No podremos evitar que el receptor duplique o modifique el documento. Es el método con el que resulta más fácil alterar fraudulentamente el certificado.

HTML

Una opción obvia para entregar un documento en la Web es HTML. El Lenguaje de marcado de hipertexto se ha diseñado específicamente para esta función. Como sin duda sabrá, cuenta con control del formato, sintaxis para añadir objetos como imágenes y es compatible (con algunos cambios) con distintos tipos de sistemas operativos y software. Es relativamente sencillo, por lo que resulta fácil de diseñar y las secuencias de comandos lo pueden generar y entregar sin dificultad.

Entre los inconvenientes del uso de HTML en esta aplicación podemos destacar la limitada compatibilidad con información de impresión como los saltos de página, incoherencias a la hora de representarlo en distintas plataformas y programas, y una calidad de impresión variable. Además, aunque HTML puede incluir cualquier tipo de elemento externo, la capacidad de un navegador para representar o utilizar dichos elementos no se puede garantizar si se trata de tipos poco habituales.

Formatos de procesadores de texto

La entrega de documentos como documentos de procesador de texto, en especial en proyectos de redes internas, tiene bastante sentido. Sin embargo, en proyectos de Internet, el uso de un formato de procesador de texto propietario puede excluir a algunos usuarios, pero si consideramos su importancia en el mercado, Microsoft Word es una buena opción. La mayoría de los usuarios tiene acceso a Word o a procesadores de texto que puedan leer archivos de Word.

Los usuarios de Windows que no dispongan de Word pueden descargar el programa gratuito WordViewer de la dirección `http://www.microsoft.com/office/000/viewers.asp`.

La generación de un documento como documento de Microsoft Word tiene sus ventajas. El diseño es muy sencillo, siempre que disponga de una copia de Word. El

control sobre el aspecto impreso del documento es considerable y también disponemos de gran flexibilidad en lo que respecta a los contenidos. También podemos impedir que el usuario modifique el documento si indicamos que Word solicite una contraseña. Desafortunadamente, los archivos de Word pueden tener un gran tamaño, sobre todo si contienen imágenes u otros elementos complejos. Tampoco resulta fácil generarlos dinámicamente en PHP. El formato está documentado, pero es un formato binario y la documentación de formato se incluye con las condiciones del contrato. Se pueden generar documentos de Word con un objeto COM, pero no es sencillo. Otra posibilidad que puede tener en cuenta es OpenOfficeWriter, que tiene dos ventajas: no es un programa propietario y utiliza un formato de archivos basado en XML.

Formato de texto enriquecido

El Formato de texto enriquecido o RTF nos ofrece la mayoría de las ventajas de Word pero los archivos son más fáciles de generar. Disponemos de gran flexibilidad sobre el diseño y el formato de la página impresa. Podemos incluir elementos como imágenes vectoriales o en mapa de bits. Y tenemos la seguridad de que el usuario obtendrá un resultado similar cuando vea o imprima el documento.

RTF es un formato de texto de Microsoft Word. Se ideó como formato de intercambio para transferir documentos de unos programas a otros. Tiene algunas semejanzas con HTML. Utiliza sintaxis y palabras clave en lugar de datos binarios para transmitir la información de formato. Por lo tanto, resulta legible.

El formato está bien documentado. La especificación está disponible gratuitamente y la puede encontrar en la dirección `http://msdn.microsoft.com/library/specs/rtfspec.htm`.

La forma más sencilla de generar un documento RTF consiste en seleccionar la opción Guardar como RTF en un procesador de texto. Como los archivos RTF sólo contienen texto, se pueden generar directamente y los ya existentes resultan fáciles de modificar.

Como el formato está documentado y disponible gratuitamente, son más los programas que pueden leer RTF que el formato binario de Word. No obstante, debe saber que los usuarios que abran un archivo RTF complejo en versiones antiguas de Word o en diferentes procesadores verán resultados muy distintos. En cada nueva versión de Word se añaden nuevas palabras clave para RTF, por lo que las implementaciones más antiguas ignorarán los controles que no entiendan o que hayan optado por no implementar.

En función de nuestra lista, resultará muy sencillo diseñar un certificado RTF con ayuda de Word o de cualquier otro procesador de texto; puede contener distintos elementos como imágenes vectoriales o en mapa de bits; tiene una gran calidad de impresión; se puede generar sencilla y rápidamente, y se puede enviar electrónicamente con un coste bajo.

Funciona con diferentes tipos de aplicaciones y sistemas operativos, aunque con resultados variables. Entre los inconvenientes, diremos que un documento RTF lo

puede modificar cualquiera, lo que resulta un problema para nuestro certificado y para otros tipos de documentos. En documentos complejos, el tamaño del archivo puede ser demasiado elevado. RTF es una opción muy aconsejable para muchas aplicaciones de envío de documentos, por lo que lo utilizaremos en este ejemplo.

PostScript

PostScript, de Adobe, es un lenguaje de descripción de páginas. Se trata de un potente y complejo lenguaje de comunicación ideado para representar documentos independientemente del dispositivo, es decir, una descripción que genera resultados coherentes entre diferentes dispositivos, como por ejemplo impresoras y pantallas. Está muy bien documentado. Existen varios manuales al respecto así como innumerables sitios Web sobre el tema. Un documento PostScript puede incluir formato de gran precisión, texto, imágenes, fuentes incrustadas y otros elementos. Resulta muy sencillo generar un documento PostScript desde una aplicación si se imprime a través de un controlador de impresora PostScript. Si le interesa, podría incluso aprender a programarlo directamente.

Los documentos PostScript son bastante portables. Ofrecen impresiones de gran calidad en distintos dispositivos y sistemas operativos.

Sin embargo, a la hora de utilizar PostScript para distribuir documentos encontramos algunos inconvenientes:

- El tamaño de los archivos puede ser excesivo
- Los usuarios tendrán que descargar software adicional para utilizarlos

La mayoría de los usuarios de Unix podrán procesar archivos PostScript, pero los de Windows tendrán que descargar un visor como GSview, que utiliza el intérprete Ghostscript PostScript. Este software está disponible para una amplia variedad de plataformas. Aunque es gratuito, no es aconsejable obligar a los usuarios a que descarguen programas adicionales.

Encontrará más información sobre Ghostscript en `http://www.ghostscript.com` y lo puede descargar de `http://www.cs.wisc.ed/ghost`.

Para nuestra aplicación, PostScript ofrece un resultado de gran calidad pero no cumple el resto de los requisitos.

Formato de documento portable

Afortunadamente, existe un formato con la mayoría de las prestaciones de PostScript, pero con algunas ventajas significativas. El Formato de documento portable (también de Adobe) se diseñó para distribuir documentos que se comportaran de forma coherente en las distintas plataformas, así como para generar resultados de gran calidad tanto en papel como en pantalla.

Adobe describe PDF como "el estándar abierto para la distribución de documentos electrónicos a escala mundial. Adobe PDF es un formato de archivo universal que

conserva todas las fuentes, formato, colores y gráficos de cualquier documento original, independientemente de la aplicación y la plataforma utilizadas para crearlo. Los archivos PDF son compactos y se pueden compartir, ver, examinar e imprimir como desee cualquier usuario por medio de Adobe Acrobat Reader".

PDF es un formato abierto y la documentación correspondiente la puede encontrar en `http://partners.adobe.com/asn/developer/technotes/acrobatpdf.html`, así como en numerosos sitios Web y en el libro oficial.

Si lo comparamos con nuestra lista de atributos, PDF parece muy indicado.

Los documentos PDF ofrecen un resultado coherente de gran calidad y son capaces de incluir elementos como imágenes vectoriales y de mapa de bits, pueden utilizar compresión para crear un archivo más pequeño, se pueden enviar electrónicamente y a bajo coste, se pueden utilizar en la mayoría de los sistemas operativos y pueden incluir controles de seguridad.

Uno de los inconvenientes de trabajar con PDF es que la mayoría de los programas utilizados para crear este tipo de documentos son comerciales.

Se necesita un lector para ver archivos PDF pero Acrobat Reader se ofrece gratuitamente para Windows, Unix y Macintosh. Muchos visitantes de nuestro sitio estarán familiarizados con la extensión `.pdf` y probablemente tenga instalado el lector. Los archivos PDF resultan muy indicados para distribuir documentos atractivos y que se puedan imprimir, en especial los que no queremos que se puedan modificar por parte de los lectores. Veremos dos formas distintas de generar un certificado PDF.

Componentes de la solución

Para que el sistema funcione, tendremos que examinar el conocimiento de los usuarios y, suponiendo que pasen la prueba, generar un certificado que informe de su rendimiento. Experimentaremos con tres formas para generar este certificado: dos con PDF y una con RTF.

A continuación veremos con más detenimiento los requisitos de cada uno de estos componentes.

Sistema de preguntas y respuestas

La creación de un sistema flexible de evaluación en línea que admita distintos tipos de preguntas, distintos tipos de medios para admitir la información, datos útiles sobre preguntas erróneas así como un acertado informe estadístico es una tarea de gran complejidad.

En este capítulo, nos centraremos básicamente en el reto que supone generar documentos personalizados para enviarlos por la Web, por lo que únicamente crearemos un sencillo sistema de preguntas.

El examen no depende de ningún programa especial. Utiliza HTML para realizar las preguntas y PHP para procesar las respuestas, lo que hemos hecho desde el primer capítulo.

Software de generación de documentos

No se necesita software adicional en el servidor Web para generar documentos RTF o PDF a partir de plantillas, pero necesitaremos algún programa para crear la plantilla. Para poder utilizar las funciones de creación PDF de PHP, tendremos que haber compilado la compatibilidad con PDF en PHP (como veremos en breve).

Software para crear una plantilla RTF

Puede utilizar cualquier procesador de texto para generar archivos RTF. Para crear nuestra plantilla de certificado hemos utilizado Microsoft Word. La plantilla de certificado se incluye en el correspondiente directorio del CD-ROM.

Si prefiere utilizar otro procesador de texto, es aconsejable que pruebe el resultado en Word ya que es el programa que utilizará la mayoría de nuestros usuarios.

Software para crear una plantilla PDF

Los documentos PDF resultan más difíciles de generar. La forma más sencilla consiste es adquirir Adobe Acrobat, programa que le permite crear archivos PDF de gran calidad a partir de diferentes aplicaciones. En este proyecto hemos utilizado Acrobat para crear el archivo de plantilla.

Para crear el archivo, hemos utilizado Microsoft Word para diseñar un documento. Una de las herramientas del paquete Acrobat es Adobe Distiller. Dentro de Distiller tendremos que seleccionar algunas opciones no predeterminadas. El archivo se debe guardar en formato ASCII y es necesario desactivar la compresión. Una vez definidas estas opciones, basta con imprimir el archivo PDF.

Encontrará más información sobre Acrobat en la dirección `http://www.adobe.com/products/acrobat` y lo puede comprar en línea o en su proveedor de software habitual.

Otra opción para crear PDF es el programa de conversión `ps2pdf` que, como indica su nombre, convierte archivos PostScript en archivos PDF. Su principal ventaja es que es gratuito pero no siempre genera buenos resultados en documentos con imágenes o fuentes no estándar. Este conversor se incluye con el paquete Ghostscript que mencionamos anteriormente. Evidentemente, si vamos a crear un archivo PDF de esta forma, primero habrá que crear un archivo PostScript. Para ello, los usuarios de Unix disponen de las utilidades `a2ps` o `dvips`.

Si trabaja en un entorno Windows, también puede crear archivos PostScript sin Adobe Distiller, aunque a través de un proceso más complicado. Tendrá que instalar un controlador de impresora PostScript. Puede utilizar, por ejemplo, el controlador Apple LaserWriter IINT. Si no tiene instalado ningún controlador PostScript, puede descargar uno de Adobe en `http://www.adobe.com/support/downloads/product.jsp?product=44&platform=Windows`.

Para crear su archivo PostScript, tendrá que seleccionar esta impresora y ejecutar la opción **Imprimir a archivo**, que se incluye en el cuadro de diálogo **Imprimir**.

La mayoría de las aplicaciones de Windows generan un archivo con extensión `.prn`, que debería ser un archivo PostScript. Debería cambiar el nombre del archivo a

.ps. Lo podrá ver por medio de GSview u otro visor PostScript, o crear un archivo PDF con ayuda de la utilidad ps2pdf.

Debe saber que los distintos controladores de impresión generan resultados PostScript de calidad variable. Seguramente descubra que algunos de los archivos PostScript dan problemas cuando se ejecutan con la utilidad ps2pdf. Le sugerimos que utilice un controlador de impresora diferente.

Si solamente tiene pensado crear un número reducido de archivos PDF, puede que le interese el servicio en línea de Adobe. Por sólo 9,99 dólares al mes, puede cargar archivos en distintos formatos y descargar un archivo PDF. Este servicio fue suficiente para nuestro certificado, pero no le permite seleccionar opciones que son importantes para el proyecto. El PDF creado se almacenará como archivo binario y se comprimirá. Por esta razón resulta muy complicado modificarlo.

Puede encontrar este servicio en http://createpdf.adobe.com/.

Existe una opción de prueba gratuita que puede utilizar.

También puede encontrar una interfaz gratuita para ps2pdf basada en ftp en Net Distillery, http://www.babinszki.com/distiller/.

Software para crear PDF mediante programación

PHP incorpora compatibilidad con la creación de documentos PDF. Existen dos bibliotecas de funciones diferentes, con objetivos similares. Como dependen de bibliotecas externas, ninguna de ellas se compila de forma predeterminada en PHP.

Las funciones PDFlib de PHP utilizan la biblioteca PDFlib, disponible en http://www.pdflib.com. Las funciones ClibPDF utilizan la biblioteca ClibPDF, disponible en http://www.fastio.com. Ambas bibliotecas son similares. Proporcionan un API de funciones para generar un documento PDF. Hemos optado por utilizar PDFlib ya que parece que se actualiza con mayor regularidad.

Conviene mencionar que ninguna de estas bibliotecas es software gratuito. Ambas permiten cierto uso no comercial sin cargo, pero si tiene pensado ofrecer un servicio comercial que las utilice, tendrá que abonar una cuota.

Para ver si PDFlib ya está instalada en su sistema, compruebe el resultado de la función phpinfo(). Por debajo del título pdf verá si se activado la compatibilidad con PDFlib, así como la versión que se utiliza.

Si tiene pensado utilizar imágenes TIFF o JPEG en sus documentos PDF, tendrá que instalar también la biblioteca PDF, disponible en http://www.libtiff.org/ y la biblioteca JPEG, que encontrará en ftp://ftp.uu.net/graphics/jpeg/.

En un sistema Unix, este software se instala de forma habitual, por medio de configure y make. Tendrá que volver a compilar PHP con el conmutador –with-pdflib.

En un servidor Windows, la DLL PDFlib se incluye en el archivo PHP Zip, por lo que basta con anular el comentario de la extensión en su archivo php.ini.

Presentación de la solución

Crearemos un sistema con tres posibles resultados.

Como se indica en la figura 30.1, realizaremos una serie de preguntas, evaluaremos las respuestas y, tras ello, generaremos un certificado de una de estas formas:

- Generaremos un documento RTF a partir de una plantilla en blanco.
- Generaremos un documento PDF a partir de una plantilla en blanco.
- Generaremos un documento PDF mediante programación por medio de PDFlib.

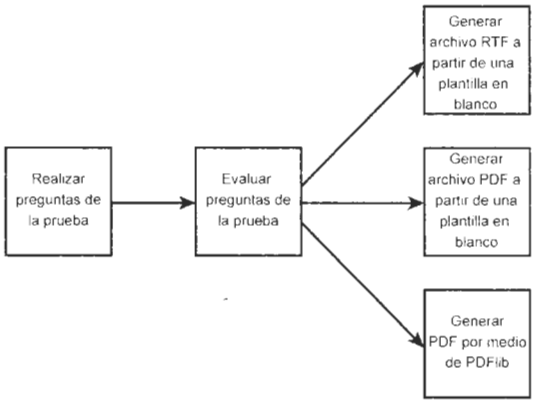


Figura 30.1. Nuestro sistema de certificación generará uno de estos tres certificados

En la tabla 30.1 se recoge un resumen de los archivos del proyecto de certificación.

Tabla 30.1. Archivos de la aplicación de certificación

Nombre	Tipo	Descripción
`index.html`	Página HTML	Formulario HTML que contiene las preguntas de la prueba
`score.php`	Aplicación	Secuencia de comandos para evaluar las respuestas de los usuarios
`rtf.php`	Aplicación	Secuencia de comandos para generar un certificado RTF a partir de una plantilla
`pdf.php`	Aplicación	Secuencia de comandos para generar un certificado PDF a partir de una plantilla
`pdflib.php`	Aplicación	Secuencia de comandos para generar un certificado PDF por medio de PDFlib
`signature.png`	imagen	Imagen de mapa de bits de la firma que se incluirá en el certificado PDFlib
`PHPCertification.rtf`	RTF	Plantilla de certificado RTF
`PHPCertification.pdf`	PDF	Plantilla de certificado PDF

Cómo responder a las preguntas

El archivo `index.html` es muy sencillo. Debe incluir un formulario HTML que pregunte el nombre del usuario así como las respuestas a una serie de preguntas.

En una aplicación de evaluación real, probablemente tendríamos que recuperar estas preguntas de una base de datos.

En nuestro caso nos centramos en la creación del certificado, por lo que codificaremos específicamente algunas preguntas en el código HTML. El campo `name` es un campo de entrada de texto.

Cada pregunta cuenta con tres botones de opción para que el usuario indique la pregunta que desee. El formulario tiene un botón de imagen y un botón de envío. El código de esta página se recoge en el listado 30.1.

Listado 30.1. index.html. Página HTML que contiene las preguntas de la prueba

```
<html>
 <body>
    <h1><p align="center">
          <img src="rosette.gif" alt="">
          Certification
          <img src="rosette.gif" alt=""></p></h1>
    <p>You too can earn your highly respected PHP certification
       from the world famous Fictional Institute of PHP Certification.</p>
    <p>Simply answer the questions below:</p>

    <form  action="score.php"  method="post">

       <p>Your  Name  <input  type="text"  name="name"></p>

       <p>What  does  the  PHP  statement  echo  do?</p>
       <ol>
          <li><input  type="radio"  name="q1"  value="1">
                Outputs strings.</li>
          <li><input type="radio" name="q1" value="2">
                Adds two numbers together.</li>
          <li><input type="radio" name="q1" value="3">
                Creates a magical elf to finish writing your code.</li>
       </ol>

       <p>What  does  the  PHP  function  cos()  do?</p>
       <ol>
          <li><input  type="radio"  name="q2"  value="1">
                Calculates a cosine in radians.</li>
          <li><input type="radio" name="q2" value="2">
                Calculates a tangent in radians. </li>
          <li><input type="radio" name="q2" value="3">
                It is not a PHP function it is a lettuce. </li>
       </ol>

       <p>What  does  the  PHP  function  mail()  do?</p>
       <ol>
          <li><input type="radio" name="q3" value="1">
```

```
            Sends a mail message.
        <li><input type="radio" name="q3" value="2">
            Checks for new mail.
        <li><input type="radio" name="q3" value="3">
            Toggles PHP between male and female mode.
    </ol>

    <p align="center"><input type="image" src="certify-me.gif" border="0"></p>

    </form>
  </body>
</html>
```

Al cargar `index.html` en un navegador Web, se produce el resultado que mostramos en la figura 30.2

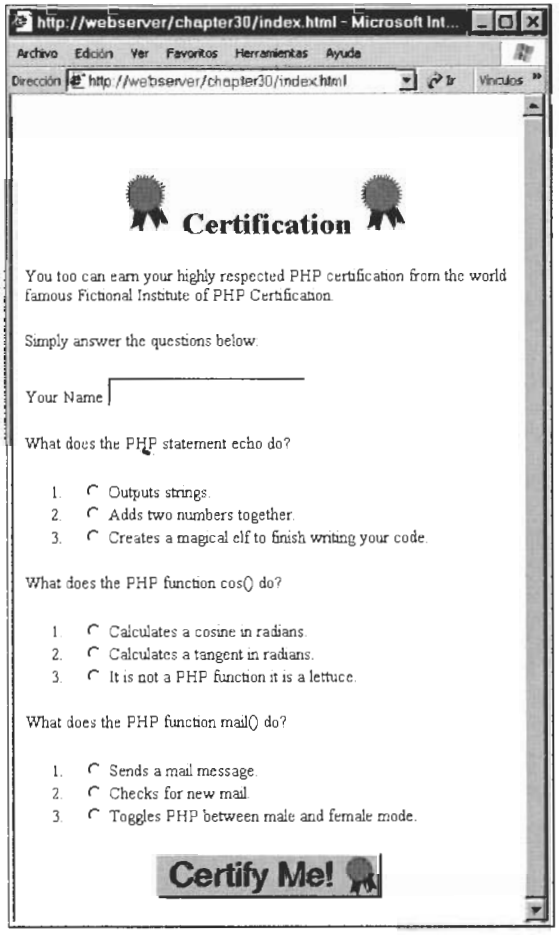


Figura 30.2. index.html pide al usuario que responda a las preguntas de la prueba

Cómo calificar las respuestas

Cuando el usuario envía sus respuestas en index.html, tendremos que calificarlas y calcular una puntuación.

Para ello utilizamos la secuencia de comandos `score.php`, cuyo código se muestra en el listado 30.2.

Listado 30.2. score.php.
Secuencia para calificar los exámenes

```
<?php
   //cree nombres de variables cortos
   $q1  =  $HTTP_POST_VARS['q1'];
   $q2  =  $HTTP_POST_VARS['q2'];
   $q3  =  $HTTP_POST_VARS['q3'];
   $name  =  $HTTP_POST_VARS['name'];

   // compruebe que se han recibido todos los datos
   if($q1==''||$q2==''||$q3==''||$name=='')
   {
      echo '<h1><p align = center><img src="rosette.gif" alt="">
                                    Sorry:
                                    <img src="rosette.gif" alt=""></p></h1>';
      echo '<p>You need to fill in your name and answer all questions</p>';
   }
   else
   {
      //sume las puntuaciones
      $score = 0;
      if($q1 == 1) // la respuesta correcta de q1 es 1
         $score++;
      if($q2 == 1) // la respuesta correcta de q2 es 1
         $score++;
      if($q3 == 1) // la respuesta correcta de q3 es 1
         $score++;

      //convierta la puntuación en un porcentaje
      $score = $score / 3 * 100;

      if($score < 50)
      {
         // esta persona ha suspendido
         echo '<h1 align="center"><img src="rosette.gif" alt="" />
                                    Sorry:
                                    <img src="rosette.gif" alt="" /></h1>';
         echo '<p>You need to score at least 50% to pass the exam</p>';
      }
      else
      {
         // cree una cadena que contenga la puntuación con un decimal
         $score = number_format($score, 1);
```

```
        echo '<h1 align="center"><img src="rosette.gif" alt="" />
                                  Congratulations
                                  <img src="rosette.gif" alt="" /></h1>';
        echo "<p>Well done $name, with a score of $score%,
              you have passed the exam.</p>";

        // incluya enlaces a secuencias de comandos que generen los certificados
        echo '<p>Please click here to download your certificate as
                 a Microsoft Word (RTF) file.</p>';
        echo '<form action="rtf.php" method="post">';
        echo '<center>
              <input type="image" src="certificate.gif" border="0">
           </center>';
        echo '<input type="hidden" name="score" value="'.$score.'">';
        echo '<input type="hidden" name="name" value="'.$name.'">';
        echo '</form>';

        echo '<p>Please click here to download your certificate as
                 a Portable Document Format (PDF) file.</p>';
        echo '<form action="pdf.php" method="post">';
        echo '<center>
                 <input type="image" src="certificate.gif" border="0">
           </center>';
        echo '<input type="hidden" name="score" value="'.$score.'">';
        echo '<input type="hidden" name="name" value="'.$name'.">';
        echo '</form>';

        echo '<p>Please click here to download your certificate as
              a Portable Document Format (PDF) file generated with PDFLib.</p>';
        echo '<form action="pdflib.php" method="post">';
        echo '<center>
                 <input type="image" src="certificate.gif" border="0">
              </center>';
        echo '<input type="hidden" name="score" value="'.$score.'">';
        echo '<input type="hidden" name="name" value="'.$name.'">';
        echo '</form>';
    }
  }
?>
```

Esta secuencia de comandos mostrará un mensaje si el usuario no ha respondido a todas las preguntas o si su puntuación es inferior a la calificación de aprobado que hayamos seleccionado.

Si el usuario responde correctamente a las preguntas, podrá generar un certificado.

Véase la figura 30.3.

A partir de aquí, el usuario dispone de tres opciones.

Puede conseguir un certificado RTF o uno de los certificados PDF.

Veremos cada una de las secuencias de comandos correspondientes a cada opción.

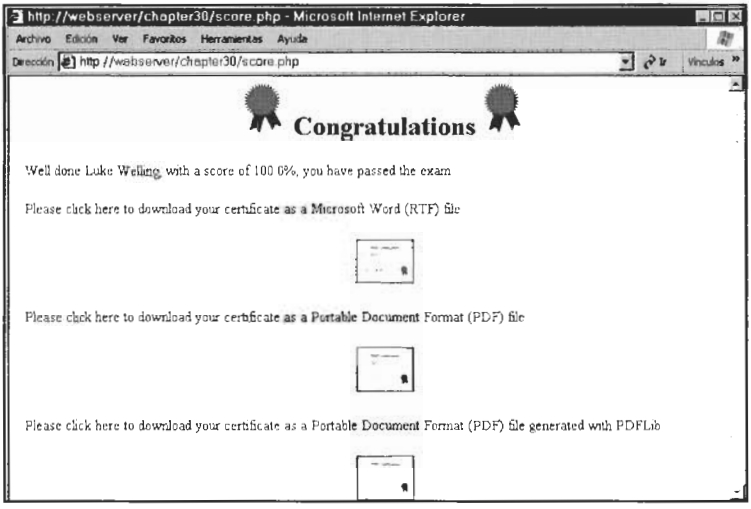


Figura 30.3. score.php ofrece a los visitantes que hayan realizado la prueba correctamente la opción de generar un certificado de tres formas

Generación de un certificado RTF

No hay nada que nos impida generar un documento RTF si escribimos texto ASCII en un archivo o en una variable de cadena, pero para ello tendríamos que aprender otra sintaxis. Veamos un documento RTF muy sencillo:

```
{\rtf1
{\fonttbl {\f0 Arial;}{\f1 Times New Roman;}}
\f0\fs28 Heading\par
\f1\fs20 This is an rtf document.\par
}
```

Este documento establece una tabla de fuentes con dos fuentes: Arial, a la que se hará referencia como `f0` y Times New Roman, a la que se hará referencia como `f1`. El encabezado utiliza `f0` (Arial) con un tamaño de 28 (14 puntos). El control `\par` indica un salto de párrafo. Tras ello, escribimos `This is a rtf document` (Éste es un documento rft) en `f1` (Times New Roman) con un tamaño de 20 (10 puntos).

Podríamos generar este documento manualmente, pero PHP no cuenta con funciones que nos ahorren trabajo a la hora de crear las partes más complejas, como por ejemplo en la incorporación de gráficos. Afortunadamente, en muchos documentos, la estructura, el estilo y la mayor parte del texto son estáticos y solamente cambian algunas partes de un usuario a otro. Una forma más eficaz de generar un documento consiste en utilizar una plantilla.

Con ayuda de un procesador de texto, podemos generar un documento complejo como el de la figura 30.4.

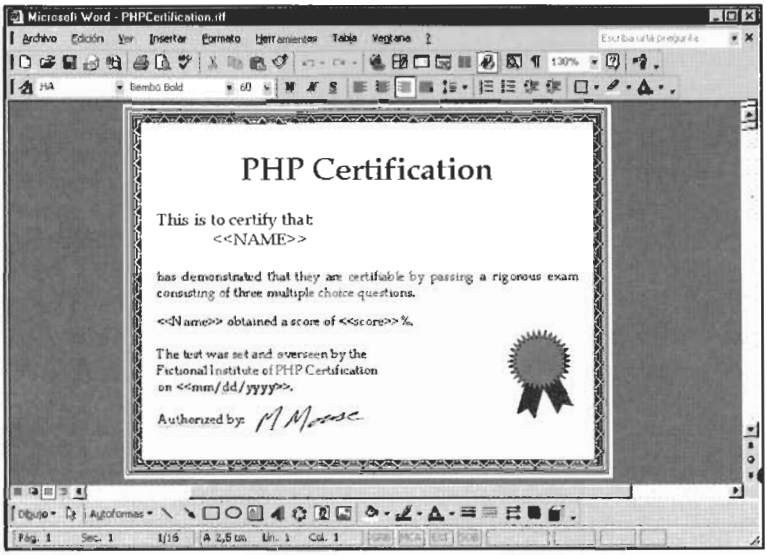


Figura 30.4. Por medio de un procesador de texto podemos crear una sencilla y atractiva plantilla

Nuestra plantilla incluye marcadores de posición como `<<NAME>>` para indicar dónde se añadirán los datos dinámicos. El aspecto de estos marcadores no es relevante. Utilizaremos una descripción con sentido dentro de las comillas angulosas. Es muy importante que seleccionemos marcadores de posición que no aparezcan de forma accidental en el resto del documento. También le ayudará a diseñar la plantilla si los marcadores de posición tienen aproximadamente la misma longitud que los datos por lo que los vamos a reemplazar. Los marcadores de posición de este documento son `<<NAME>>`, `<<Name>>`, `<<score>>` y `<<mm/dd/yyyy>>`. Utilizaremos tanto NAME como Name, ya que emplearemos un método que discrimina entre mayúsculas y minúsculas para reemplazarlos.

Una vez diseñada la plantilla, necesitamos una secuencia de comandos para personalizarla. Esta secuencia es `rtf.php` y su código se muestra en el listado 30.3.

Listado 30.3. rtf.php. Secuencia de comandos para crear un certificado RTF personalizado

```
<?php
   //cree nombres cortos de variables
   $name  =  $HTTP_POST_VARS['name'];
   $score  =  $HTTP_POST_VARS['score'];

   //  compruebe que dispone de los parámetros necesarios
   if( !$name || !$score )
```

```
    {
        echo '<h1>Error:</h1>This page was called incorrectly';
    }
    else
    {
        //genere los encabezados para que el navegador seleccione la aplicación correcta
        header( 'Content-type: application/msword' );
        header( 'Content-Disposition: inline, filename=cert.rtf');

        $date = date( 'F d, Y' );

        // abra el archivo de plantilla
        $filename = 'PHPCertification.rtf';
        $fp = fopen ( $filename, 'r' );

        //lea la plantilla en una variable
        $output = fread( $fp, filesize( $filename ) );

        fclose ( $fp );

        // sustituya los marcadores de posición en la plantilla por los datos
        $output = str_replace( '<<NAME>>', strtoupper( $name ), $output );
        $output = str_replace( '<<Name>>', $name, $output );
        $output = str_replace( '<<score>>', $score, $output );
        $output = str_replace( '<<mm/dd/yyyy>>', $date, $output );

        // envíe el documento generado al navegador
        echo $output;
    }
?>
```

Esta secuencia de comandos realiza una comprobación de errores básica para garantizar que se han pasado todos los detalles del usuario y, tras ello, pasa a crear el certificado.

El resultado de esta secuencia de comandos será un archivo RTF en lugar de un archivo HTML, por lo que tendremos que indicárselo al navegador del usuario.

De esta forma, el navegador intentará abrir el archivo con la aplicación correcta o proporcionará un cuadro de diálogo Guardar como si no reconoce la extensión RTF.

Especificamos el tipo MIME del archivo por medio de la función `header()` de PHP para enviar el correspondiente encabezado HTTP como indicamos a continuación:

```
header('Content-type: application/msword');
header('Content-Disposition: inline, filename=cert.rtf');
```

El primer encabezado indica al navegador que vamos a enviar un archivo de Microsoft Word (no es del todo cierto, pero es la aplicación que probablemente se utilice para abrir el archivo RTF).

El segundo encabezado indica al navegador que muestre automáticamente los contenidos del archivo y que el nombre sugerido del mismo es `cert.rtf`. Es el nombre de archivo predeterminado que el usuario verá si trata de guardar el archivo desde el navegador.

Una vez enviados los encabezados, abrimos y leemos nuestro archivo de plantilla RTF en la variable `$output` y utilizamos la función `str_replace()` para reemplazar nuestros marcadores de posición con los datos que queremos que aparezcan en el archivo. La línea

```
$output = str_replace( '<<Name>>', $name, $output );
```

reemplazará cualquier presencia del marcador de posición `<<Name>>` por los contenidos de la variable `$name`.

Después de realizar las sustituciones, basta con duplicar el resultado en el navegador. En la figura 30.5 puede ver un ejemplo de esta secuencia de comandos.

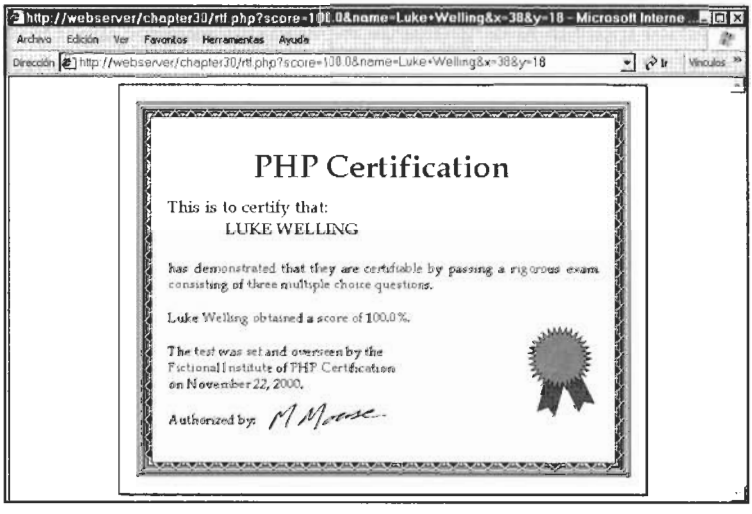


Figura 30.5. rtf.php genera un certificado a partir de la plantilla RTF

Este enfoque funciona a la perfección. Las invocaciones de `str_replace()` se ejecutan con rapidez, incluso a través de nuestra plantilla y, por tanto, los contenidos de `$output` son considerablemente extensos. El principal problema desde el punto de vista de esta aplicación es que el usuario cargará el certificado en su procesador de texto para poder imprimirlo, lo que puede tentarle a que lo modifique. RTF no permite crear documentos de sólo lectura.

Generación de un certificado PDF a partir de una plantilla

El proceso de generación de un certificado PDF a partir de una plantilla es muy similar.

La principal diferencia es que al crear al archivo PDF, algunos de los marcadores de posición se pueden mezclar con códigos de formato. Por ejemplo, si nos fijamos en

el archivo de plantilla de certificado que hemos creado, veremos que el aspecto de los marcadores de posición es el siguiente:

```
<<N)-13(AME)-10(>)-6(>
<<Na)-9(m)0(e)-18(>>
<)-11(<)1(sc)-17(or)-6(e)-6(>)-11(>
<)-11(<)1(m)-12(m)0(/d)-6(d)-19(/)1(yy)-13(yy)-13(>>
```

Comprobará que, a lo largo del archivo, al contrario de lo que sucede con RTF, no es un formato que nos resulte legible.

Hay varias formas de solucionar este aspecto.

Podríamos ir marcador por marcador y eliminar los códigos de formato. De hecho, apenas influye en el aspecto final del documento ya que los códigos incrustados en la plantilla anterior indican la cantidad de espacio que se debe dejar entre las letras de los marcadores que, de todas formas, vamos a sustituir. No obstante, si adoptamos este enfoque, tendremos que modificar manualmente el archivo PDF y repetirlo cada vez que cambiemos o actualicemos el archivo. No resulta complicado si sólo tenemos cuatro marcadores de posición, pero se puede convertir en una verdadera pesadilla si por ejemplo tenemos varios documentos con multitud de marcadores de posición y decidimos cambiar la letra en todos los documentos.

Podemos evitar este problema si utilizamos otra técnica. Podemos utilizar Adobe Acrobat para crear un formulario PDF, similar a un formulario HTML con campos en blanco. Tras ello, utilizamos una secuencia de comandos PHP para crear lo que se denomina un archivo FDF (Formato de datos de formularios) que básicamente es un conjunto de datos que se combinarán con una plantilla. Se pueden crear FDF por medio de la biblioteca de funciones FDF de PHP, en concreto la función `fdf_create()` para crear un archivo, la función `fdf_set_value()` para definir los valores de los campos y la función `fdf_set_file()` para definir el archivo de formulario de plantilla asociado. Seguidamente se puede pasar este archivo al navegador con el correspondiente tipo MIME, en este caso, `vnd.fdf` y el complemento Acrobat Reader del navegador debería sustituir los datos del formulario.

Es la forma correcta de hacer las cosas, pero tiene dos limitaciones. Por un lado, asume que disponemos de una copia de Acrobat. Por otra parte, es complicado sustituir texto que está en línea en lugar de en campos de un formulario. Esto puede constituir o no un problema, en función de lo que queramos hacer. Hemos utilizado la creación de PDF para crear cartas en las que era necesario cambiar numerosos elementos en línea. Los FDF no funcionan correctamente para este tipo de cosas. Si por ejemplo quiere completar el formulario automáticamente, como en el caso de un formulario de impuestos en línea, no será problema alguno.

Encontrará más información sobre el formato FDF en el sitio de Adobe

También debería consultar la documentación FDF del manual de PHP si tiene pensado utilizar este enfoque

A continuación aplicaremos la solución PDF al problema anterior.

Podemos buscar y reemplazar los marcadores de posición de nuestro archivo PDF si reconocemos que los códigos de formato adicionales solamente están formados por guiones, dígitos y paréntesis y que, por lo tanto, pueden coincidir con una

expresión regular. Hemos escrito una función, `pdf_replace`, para generar automáticamente una expresión regular de coincidencias para un marcador de posición y reemplazar dicho marcador con el texto correcto.

Aparte de esta novedad, el código para generar el certificado por medio de una plantilla PDF es muy similar al de la versión RTF. Esta secuencia de comandos se muestra en el listado 30.4.

Listado 30.4. pdf.php. Secuencia de comandos para crear un certificado PDF personalizado a través de una plantilla

```
<?php
   set_time_limit( 180 ); // esta secuencia de comandos puede ser muy lenta

   //cree nombres de variables cortos
   $name = $HTTP_POST_VARS['name'];
   $score = $HTTP_POST_VARS['score'];

   function pdf_replace( $pattern, $replacement, $string )
   {
      $len = strlen( $pattern );
      $regexp = '';
      for ( $i = 0; $i<$len; $i++ )
      {
         $regexp .= $pattern[$i];
         if ($i<$len-1)
            $regexp .= "(\)\-{0,1}[0-9]*\(){0,1}";
      }
      return ereg_replace ( $regexp, $replacement, $string );
   }

   if(!$name||!$score)
   {
      echo '<h1>Error:</h1>This page was called incorrectly';
   }
   else
   {
      // genere los encabezados para que el navegador seleccione la aplicación correcta
      header( 'Content-Disposition:  filename=cert.pdf');
      header( 'Content-type: application/pdf' );

      $date = date( 'F d, Y' );

      // abra el archivo de plantilla
      $filename = 'PHPCertification.pdf';
      $fp = fopen ( $filename, 'r' );

      //lee la plantilla en una variable
      $output = fread( $fp, filesize( $filename ) );

      fclose ( $fp );

      // sustituya los marcadores de posición en la plantilla por los datos
```

```
        $output = pdf_replace( '<<NAME>>', strtoupper( $name ), $output );
        $output = pdf_replace( '<<Name>>', $name, $output );
        $output = pdf_replace( '<<score>>', $score, $output );
        $output = pdf_replace( '<<mm/dd/yyyy>>', $date, $output );

        // envíe el documento generado al navegador
        echo $output;
    }
?>
```

Esta secuencia de comandos genera una versión personalizada de nuestro documento PDF.

El documento, que mostramos en la figura 30.6, se imprimirá correctamente en distintos sistemas y resultará muy complicado modificarlo o cambiarlo.

Comprobará que el documento PDF de la figura 30.6 es prácticamente idéntico al documento RTF de la figura 30.5.

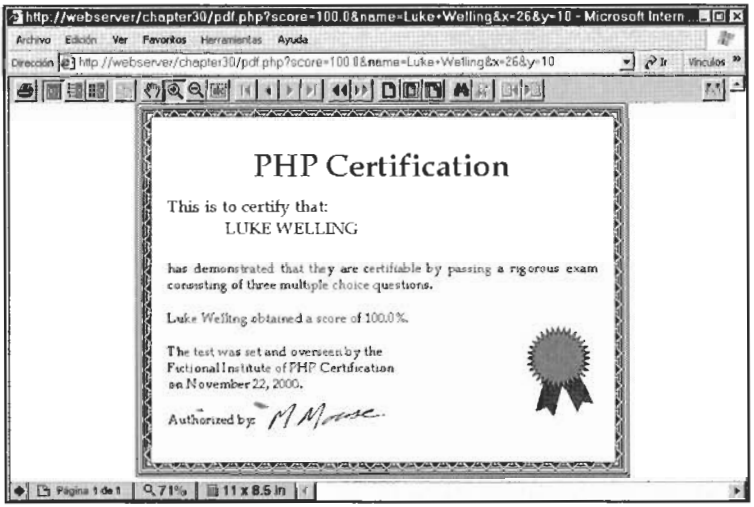


Figura 30.6. pdf.php genera un certificado a partir de una plantilla PDF

Un problema asociado a este enfoque es que el código se ejecuta lentamente debido a la expresión regular de coincidencias que necesitamos. La ejecución de las expresiones regulares es mucho más lenta que la de `str_replace()`, que utilizamos en la versión RTF.

Si quiere que coincida un elevado número de marcadores de posición o quiere generar muchos de estos documentos en el mismo servidor, podría considerar enfoques adicionales.

En el caso de una plantilla más sencilla, no supone ningún problema.

El grueso de este archivo se corresponde a los datos que representan las imágenes.

Generación de un documento PDF por medio de PDFlib

PDFlib se ha ideado para generar documentos PDF dinámicos a través de la Web.

Estrictamente no forma parte de PHP sino que es una biblioteca independiente con un gran número de funciones que se pueden invocar desde una amplia variedad de lenguajes de programación.

Existen vinculaciones de lenguaje disponibles para C, C++, Java, Perl, Python, Tcl y ActiveX/COM.

Desde PHP 4.0.5, oficialmente PDFlib GmbH es compatible con PDFLib, lo que significa que puede remitirse a la documentación PHP en `http://www.php.net/manual/ref.pdf.php` o descargar la documentación oficial de pdflib.com.

Una secuencia de comandos Hello World para PDFlib

Una vez instalado PHP con PDFlib activado, puede probarlo con un sencillo programa como el del ejemplo Hello World que mostramos en el listado 30.5.

Listado 30.5. testpdf.php. Ejemplo Hello World clásico que utiliza PDFlib a través de PHP

```
<?php
   //cree el archivo
   $fp = fopen('hello.pdf', 'w');
   if(!$fp)
   {
      echo "Error: could not create the PDF file";
      exit;
   }

   // inicie el documento pdf
   $pdf = pdf_open($fp);
   pdf_set_info($pdf, "Creator", "pdftest.php");
   pdf_set_info($pdf, "Author", "Luke Welling and Laura Thomson");
   pdf_set_info($pdf, "Title", "Hello World (PHP)");

   // la carta US es 11" x 8.5" y hay aproximadamente 72 puntos por pulgada
   pdf_begin_page($pdf, 8.5*72, 11*72);
   pdf_add_outline($pdf, 'Page 1');
   $font = pdf_findfont($pdf, 'Times-Roman', 'host', 0);
   pdf_setfont($pdf, $font, 24);
   pdf_set_text_pos($pdf, 50, 700);

   // escriba el texto
   pdf_show($pdf,'Hello,world!');
   pdf_continue_text($pdf,'(says PHP)');

   // finalice el documento
   pdf_end_page($pdf);
   pdf_close($pdf);
```

```
    fclose($fp);

    // muestre un enlace de descarga
    echo "download the pdf <a href = 'hello.pdf'>here</a>";
?>
```

Si esta secuencia de comandos falla, el error que probablemente vea es el siguiente:

```
Fatal error: Call to undefined function: pdf_open() in
/home/book/public_html/chapter30/pdftest.php on line 6
```

Significa que no ha compilado la extensión PDFlib en PHP.

La instalación es muy sencilla, pero algunos detalles cambian en función de las versiones de PHP y de PDFlib que utilice. Puede revisar las sugerencias incluidas en las notas de los usuarios de la página PDFlib, en el manual comentado de PHP.

Una vez ejecutada esta secuencia de comandos en su sistema, es necesario analizar su funcionamiento.

La primera sección del código, en la que se incluye la línea

```
$fp = fopen('hello.pdf', 'w');
```

crea un archivo que se puede escribir.

Conviene indicar que en este caso el código se escribe directamente en el directorio actual aunque ya hemos mencionado por qué no es aconsejable configurar permisos para que PHP pueda escribir en el árbol Web.

La línea

```
$pdf = pdf_open($fp);
```

inicializa un documento PDF utilizando el archivo que ya hemos abierto. También se puede invocar `pdf_open()` sin parámetros para crear un documento en memoria que se represente directamente en el navegador. En cualquier caso, tendrá que capturar el valor devuelto de `pdf_open()` ya que lo utilizarán todas las llamadas posteriores a una función PDF.

La función `pdf_set_info()` nos permite etiquetar el documento con un asunto, un título, un creador, un autor, una lista de palabras clave y un campo personalizado definido por el usuario.

En nuestro caso definimos un creador, un autor y un título. Los seis campos de información son opcionales.

```
pdf_set_info($pdf, 'Creator', 'pdftest.php');
pdf_set_info($pdf, 'Author', 'Luke Welling and Laura Thomson');
pdf_set_info($pdf, 'Title', 'Hello World (PHP)');
```

Un documento PDF está formado por una serie de páginas. Para iniciar una nueva página, tendremos que invocar `pdf_begin_page()`. Al igual que el identificador devuelto por `pdf_open()`, `pdf_begin_page()` requiere las dimensiones de la página. Cada página de un documento puede tener un tamaño diferente, pero a menos que tenga una buena razón para no hacerlo, conviene utilizar un tamaño de papel común.

PDFlib funciona en puntos, tanto para el tamaño de las páginas como para localizar coordenadas en cada una de las páginas. Como referencia, diremos que A4 mide aproximadamente 595 por 842 puntos y que el papel de carta U.S. tiene 612 por 792 puntos. Esto significa que nuestra línea

```
pdf_begin_page($pdf, 8.5*72, 11*72);
```

crea una página en nuestro documento con el tamaño del papel de carta U.S.

No es necesario que un documento PDF sea simplemente un documento que se pueda imprimir. Se pueden añadir numerosas funciones PDF al documento, como por ejemplo hipervínculos o marcadores. La función `pdf_add_outline()` añade un marcador al contorno del documento. Los marcadores de un documento aparecen en un panel independiente de Acrobat Reader, lo que nos permite pasar de una sección a otra.

Esta línea

```
pdf_add_outline($pdf, 'Page 1');
```

añade una entrada del contorno con la etiqueta Page 1, que hace referencia a la página actual.

Las fuentes disponibles en el sistema varían de un sistema operativo a otro e incluso de un equipo a otro. Para garantizar la consistencia de los resultados, existe un conjunto de fuentes básicas que funcionan con todos los lectores PDF. Las 14 fuentes principales son las siguientes:

- Courier
- Courier-Bold
- Courier-Oblique
- Courier-BoldOblique
- Helvetica
- Helvetica-Bold
- Helvetica-Oblique
- Helvetica-BoldOblique
- Times-Roman
- Times-Bold
- Times-Italic
- Times-BoldItalic
- Symbol
- ZapfDingbats

Se pueden añadir fuentes distintas a las de este conjunto, pero aumentará el tamaño del archivo y puede que no se acepten en función de la licencia que tenga de esa

determinada fuente. Podemos seleccionar una fuente, su tamaño y una codificación de caracteres, como indicamos a continuación:

```
$font = pdf_findfont($pdf, 'Times-Roman', 'host', 0);
pdf_setfont($pdf, $font, 24);
```

El tamaño de las fuentes se especifica en puntos. Como codificación de caracteres hemos seleccionado host. Los valores admitidos son winansi, builtin, macroman, ebcdic o host, cuyo significado se muestra a continuación:

- winansi. ISO 8859-1 más caracteres especiales añadidos por Microsoft, como por ejemplo el símbolo del euro.
- macroman. Codificación Mac Roman. Es el conjunto de caracteres predeterminado de Macintosh.
- ebcdic. EBCDIC como el utilizado en sistemas IBM AS/400.
- builtin. Utiliza la codificación incorporada en la fuente. Se suele utilizar con fuentes y símbolos no Latinos.
- host. Selecciona automáticamente macroman en Mac, ebcdic en sistemas basados en EBCDIC y winansi en los sistemas restantes.

Si no necesita incluir caracteres especiales, no es importante la codificación que escoja.

Un documento PDF no es como un documento HTML o un documento de un procesador de texto. De forma predeterminada, el texto no comienza en la parte superior izquierda y fluye hasta otras líneas según queramos. Tendremos que indicar dónde se situarán todas las líneas de texto. Como ya hemos mencionado, PDF utiliza puntos para indicar ubicaciones. El origen (la coordenada x, y [0, 0]) se encuentra en la esquina inferior izquierda de la página.

Si nuestra página es de 612 por 792 puntos, el punto (50, 700) se encuentra a dos tercios de pulgada con respecto a la parte izquierda de la página y a una pulgada y un tercio con respecto a la parte superior. Para definir la posición del texto en este punto, utilizamos

```
pdf_set_text_pos($pdf, 50, 700);
```

Por último, una vez configurada la página, podemos escribir texto en la misma. Para añadir un texto a la posición actual con la fuente actual, utilizamos `pdf_show()`.

La línea

```
pdf_show($pdf,'Hello,world!');
```

añade el texto "Hello World!" a nuestro documento.

Para pasar a la siguiente línea y escribir más texto, utilizamos `pdf_continue_text()`. Para añadir la cadena "`(says PHP)`", utilizamos

```
pdf_continue_text($pdf,'(says PHP)');
```

La ubicación exacta en la que aparecerá dependerá de la fuente y del tamaño que hayamos seleccionado. Si en lugar de líneas o frases utiliza párrafos continuos, la función `pdf_show_boxed()` le resultará de mayor utilidad. Le permite declarar un cuadro de texto y añadir texto al mismo.

Después de añadir los elementos a la página, tendremos que invocar `pdf_end_apge()` de la siguiente forma:

```
pdf_end_page($pdf);
```

Una vez terminado el documento PDF, tendremos que cerrarlo por medio de `pdf_close()`. Al generar un archivo también es necesario cerrarlo.

Las líneas

```
pdf_close($pdf);
fclose($fp);
```

completan la generación de nuestro documento Hello World. Lo único que falta es proporcionar alguno modo de descargarlo.

```
echo 'download the pdf <a href="hello.pdf">here</a>';
```

Este ejemplo se deriva del ejemplo de C de la documentación PDFlib y debería servirle como punto de partida.

El documento que queremos crear para el certificado es más complicado, ya que incluye un borde, una imagen vectorial y una imagen de mapa de bits. Con las otras dos técnicas, añadimos estos elementos con nuestro procesador de texto. Con PDFlib, tendremos que añadirlos manualmente.

Generación de nuestro certificado con PDFlib

Para poder utilizar PDFlib, hemos tomado algunas decisiones. Aunque se pueda duplicar exactamente el certificado que utilizamos anteriormente, resultaría más costoso generar y ubicar manualmente todos los elementos que utilizar una herramienta como Microsoft Word para maquetar el documento.

Utilizaremos el mismo texto que antes, incluyendo el sello rojo y la firma en mapa de bits, pero no vamos a duplicar el borde tan complejo.

El código completo de esta secuencia de comandos se incluye en el listado 30.6.

Listado 30.6. pdflib.php. Generación de nuestro certificado por medio de PDFlib

```
<?php
   // cree nombre de variables cortos
   $name = $HTTP_POST_VARS['name'];
   $score  =  $HTTP_POST_VARS['score'];

   if(!$name||!$score)
   {
      echo '<h1>Error:</h1>This page was called incorrectly';
      exit;
```

```
}
else
{
   $date = date( 'F d, Y' );

   // cree un documento pdf en memoria
   $pdf = pdf_new();
   pdf_open_file($pdf);

   // defina el nombre de la fuente que se utilizará posteriormente
   $fontname = 'Times-Roman';

   // defina el tamaño de la página y créela
   // la carta US es 11" x 8.5" y hay aproximadamente 72 puntos por pulgada
   $width = 11*72;
   $height = 8.5*72;
   pdf_begin_page($pdf,  $width,  $height);

   // dibuje los bordes
   $inset = 20; // espacio entre el borde y el borde de la página
   $border = 10; // anchura de la línea del borde principal
   $inner = 2; // espacio dentro del borde

   //dibuje el borde exterior
   pdf_rect($pdf,$inset-$inner,
                 $inset-$inner,
                 $width-2*($inset-$inner),
                 $height-2*($inset-$inner));
   pdf_stroke($pdf);

   // dibuje el borde principal con una anchura de $border puntos
   pdf_setlinewidth($pdf, $border);
   pdf_rect($pdf,$inset+$border/2,
                 $inset+$border/2,
                 $width-2*($inset+$border/2),
                 $height-2*($inset+$border/2));
   pdf_stroke($pdf);
   pdf_setlinewidth($pdf, 1.0);

   // dibuje el borde interior
   pdf_rect($pdf,$inset+$border+$inner,
                 $inset+$border+$inner,
                 $width-2*($inset+$border+$inner),
                 $height-2*($inset+$border+$inner));
   pdf_stroke($pdf);

   // añada el título
   $font = pdf_findfont($pdf, $fontname, 'host', 0);
   if  ($font)
      pdf_setfont($pdf, $font, 48);
   $startx = ($width - pdf_stringwidth($pdf, 'PHP Certification'))/2;
   pdf_show_xy($pdf, 'PHP Certification', $startx, 490);

   // añada texto
   $font  =  pdf_findfont($pdf,  $fontname,  'host',  0);
   if ($font)
```

```
   pdf_setfont($pdf, $font, 26);
$startx = 70;
pdf_show_xy($pdf, 'This is to certify that:', $startx, 430);
pdf_show_xy($pdf, strtoupper($name), $startx+90, 391);

$font = pdf_findfont($pdf, $fontname, 'host', 0);
if ($font)
   pdf_setfont($pdf, $font, 20);

pdf_show_xy($pdf, 'has demonstrated that they are certifiable '.
                  'by passing a rigorous exam', $startx, 340);
pdf_show_xy($pdf, 'consisting of three multiple choice questions.',
                  $startx, 310);

pdf_show_xy($pdf, "$name obtained a score of $score".'%.', $startx, 260);

pdf_show_xy($pdf, 'The test was set and overseen by the ', $startx, 210);
pdf_show_xy($pdf, 'Fictional Institute of PHP Certification',
                  $startx, 180);
pdf_show_xy($pdf, "on $date.", $startx, 150);
pdf_show_xy($pdf, 'Authorised by:', $startx, 100);

// añada la imagen en mapa de bits de la firma
$signature = pdf_open_image_file($pdf, 'png', 'signature.png');
pdf_place_image($pdf, $signature, 200, 75, 1);
pdf_close_image($pdf, $signature);

// defina los colores del sello
pdf_setrgbcolor_fill($pdf, 0, 0, .4);  //azul oscuro
pdf_setrgbcolor_stroke($pdf, 0, 0, 0);  // negro

// dibuje la cinta 1
pdf_moveto($pdf, 630, 150);
pdf_lineto($pdf, 610, 55);
pdf_lineto($pdf, 632, 69);
pdf_lineto($pdf, 646, 49);
pdf_lineto($pdf, 666, 150);
pdf_closepath($pdf);
pdf_fill($pdf);

// contorno de la cinta 1
pdf_moveto($pdf, 630, 150);
pdf_lineto($pdf, 610, 55);
pdf_lineto($pdf, 632, 69);
pdf_lineto($pdf, 646, 49);
pdf_lineto($pdf,  666,  150);
pdf_closepath($pdf);
pdf_stroke($pdf);

// dibuje la cinta 2
pdf_moveto($pdf, 660, 150);
pdf_lineto($pdf,  680,  49);
pdf_lineto($pdf,  695,  69);
pdf_lineto($pdf, 716, 55);
pdf_lineto($pdf, 696, 150);
pdf_closepath($pdf);
```

```
    pdf_fill($pdf);

    // contorno de la cinta 2
    pdf_moveto($pdf, 660, 150);
    pdf_lineto($pdf, 680, 49);
    pdf_lineto($pdf, 695, 69);
    pdf_lineto($pdf, 716, 55);
    pdf_lineto($pdf, 696, 150);
    pdf_closepath($pdf);
    pdf_stroke($pdf);

    pdf_setrgbcolor_fill($pdf, .8, 0, 0);  //red

    //dibuje el sello
    draw_star(665, 175, 32, 57, 10, $pdf, true);

    //contorno del sello
    draw_star(665, 175, 32, 57, 10, $pdf, false);

    // termine la página y prepare su representación
    pdf_end_page($pdf);
    pdf_close($pdf);
    $data = pdf_get_buffer($pdf);

    // genere los encabezados para que el navegador seleccione la aplicación correcta
    header('Content-type: application/pdf');
    header('Content-disposition: inline; filename=test.pdf');
    header('Content-length: ' . strlen($data));

    // represente el PDF
    echo $data;
}

function draw_star( $centerx, $centery, $points, $radius,
                    $point_size, $pdf, $filled)
{
    $inner_radius = $radius-$point_size;

    for ($i = 0; $i<=$points*2; $i++ )
    {
        $angle= ($i*2*pi())/($points*2);

        if($i%2)
        {
            $x = $radius*cos($angle) + $centerx;
            $y = $radius*sin($angle) + $centery;
        }
        else
        {
            $x = $inner_radius*cos($angle) + $centerx;
            $y = $inner_radius*sin($angle) + $centery;
        }
        f($i==0)
            pdf_moveto($pdf, $x, $y);
```

```
            else if($i==$points*2)
               pdf_closepath($pdf);
            else
               pdf_lineto($pdf, $x, $y);
         }
         if($filled)
            pdf_fill_stroke($pdf);
         else
            pdf_stroke($pdf);
      }
?>
```

El certificado que se obtiene al utilizar esta secuencia de comandos se muestra en la figura 30.7.

Como puede comprobar, es bastante similar a los otros, a excepción de que el borde es más sencillo y la estrella tiene un aspecto diferente.

Se debe a que los hemos dibujado en el documento en lugar de utiliza un archivo existente.

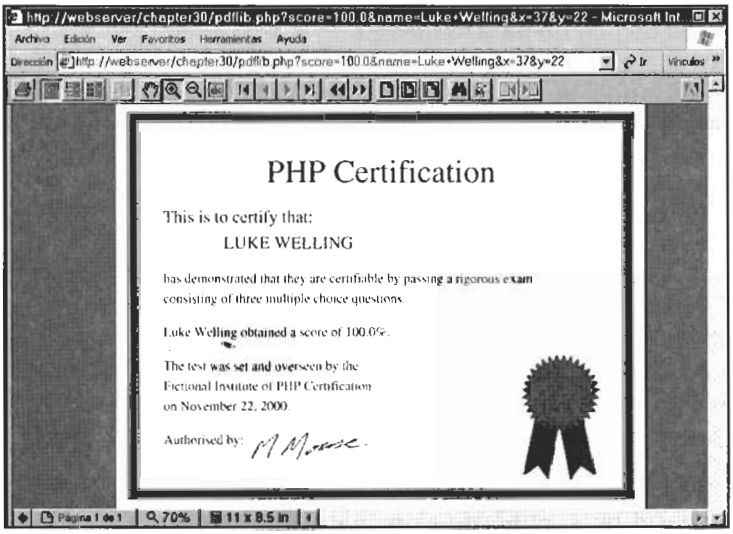


Figura 30.7. pdflib.php dibuja el certificado en un documento PDF

Analizaremos las partes de esta secuencia de comandos que son diferentes a las de los ejemplos anteriores.

Los visitantes tienen que obtener sus datos en un certificado, por lo que crearemos el documento en la memoria en lugar de un archivo.

Si lo escribiéramos en un archivo, tendríamos que preocuparnos por mecanismos para crear nombres de archivo exclusivos, para evitar que se vean los certificados de otros usuarios y para determinar la forma de eliminar los archivos antiguos para liberar espacio en el servidor.

Para crear un documento en memoria, invocamos `pdf_new()` sin parámetros seguido de una llamada a `pdf_open_file()`, como se indica a continuación:

```
$pdf = pdf_new();
pdf_open_file($pdf);
```

Nuestro borde simplificado estará formado por tres barras: un borde grueso y dos más estrechos, uno dentro del borde principal y otro exterior. Los dibujaremos como rectángulos.

Para situar los bordes de forma que se pueda modificar el tamaño de la página o el aspecto de los bordes, basaremos todas las posiciones de los mismos en las variables que ya tenemos, `$width` y `$height`, y en algunas nuevas: `$inset`, `$border` y `$inner`. Utilizaremos `$inset` para especificar la anchura en puntos del borde con respecto al borde de la página, `$border` para especificar el grosor del borde principal y `$inner` para especificar la anchura del espacio entre el borde principal y los más estrechos,

Si ha dibujado con otro API gráfico, el uso de PDFlib le deparará algunas sorpresas. Si no ha leído uno de los capítulos anteriores sobre generación de imágenes, debería hacerlo ahora ya que la generación de imágenes con la biblioteca `gd` es muy similar al uso de PDFlib para lo mismo.

Los bordes estrechos son sencillos. Para crear un rectángulo, utilizamos `pdf_rect()`, que requiere como parámetros el identificador del documento PDF, la coordenada x e y de la esquina inferior izquierda del rectángulo y la anchura y la altura del rectángulo. Como queremos que el diseño sea flexible, calcularemos estos valores a partir de las variables que hemos definido:

```
pdf_rect($pdf,$inset-$inner,
              $inset-$inner,
              $width-2*($inset-$inner),
              $height-2*($inset-$inner));
```

La invocación de `pdf_rect()` establece un trazado en forma de rectángulo. Para poder dibujar esta forma, tendremos que invocar la función `pdf_stroke()`, como se indica a continuación:

```
pdf_stroke($pdf);
```

Para poder dibujar el borde principal, tendremos que especificar la anchura de las líneas, que de forma predeterminada es de 1 punto. La siguiente llamada a `pdf_setlinewidth()` la establece en puntos `$border` (en este caso 10):

```
pdf_setlinewidth($pdf,  $border);
```

Una vez definida la anchura, creamos de nuevo un rectángulo con `pdf_rect()` e invocamos `pdf_stroke()` para dibujarlo.

```
pdf_rect($pdf,$inset+$border/2,
              $inset+$border/2,
              $width-2*($inset+$border/2),
              $height-2*($inset+$border/2));
pdf_stroke($pdf);
```

Después de trazar la línea de la anchura, tendremos que volver a establecerla en 1 con el siguiente código:

```
pdf_setlinewidth($pdf, 1.0);
```

Utilizaremos `pdf_show()` para situar cada línea de texto en el certificado. En la mayoría de las líneas de texto, empleamos un margen izquierdo que se puede configurar (`$startx`) como coordenada x y un valor seleccionado a ojo como coordenada y. Como queremos que el título aparezca centrado en la página, tendremos que conocer su anchura para situar la parte izquierda del mismo. Para obtener la anchura, utilizamos `pdf_stringwidht()`. La llamada

```
pdf_stringwidth($pdf, 'PHP Certification');
```

devuelve la anchura de la cadena `"PHP Certification"` con la fuente y el tamaño de fuente actuales.

Al igual que con las otras versiones del certificado, incluiremos una firma como mapa de bits escaneado. Las tres siguientes instrucciones

```
$signature = pdf_open_image_file($pdf, 'png', 'signature.png');
pdf_place_image($pdf, $signature, 200, 75, 1);
pdf_close_image($pdf, $signature);
```

abren un archivo PNG que contiene la firma, añaden la imagen a la página en la ubicación especificada y cierran el archivo PNG. También se pueden utilizar otros tipos de archivo. El único parámetro que quizás no esté demasiado claro es el quinto parámetro de `pdf_place_image()`. Esta función no se limita a añadir la imagen con su tamaño original. El quinto parámetro es un factor de escala. Hemos optado por mostrar la imagen a tamaño real y hemos utilizado 1 como factor de escala, pero podríamos haber utilizado un valor mayor para aumentarla o una fracción para reducirla.

El elemento más difícil de añadir al certificado es el sello. No podemos abrir automáticamente y añadir un metarchivo de Windows que contenga el sello que ya tenemos, pero podemos trazar cualquier forma.

Para trazar una forma con relleno como una de las cintas, podemos escribir el siguiente código.

En este caso definimos el trazo o el color de la línea en negro y el relleno o color interior en azul marino:

```
pdf_setrgbcolor_fill($pdf, 0, 0, .4);    //azul oscuro
pdf_setrgbcolor_stroke($pdf, 0, 0, 0); // negro
```

A continuación definimos un polígono de cinco lados como una de las cintas y lo rellenamos:

```
pdf_moveto($pdf, 630, 150);
pdf_lineto($pdf, 610, 55);
pdf_lineto($pdf, 632, 69);
pdf_lineto($pdf, 646, 49);
pdf_lineto($pdf, 666, 150);
```

```
pdf_closepath($pdf);
pdf_fill($pdf);
```

Como también queremos que el polígono tenga un contorno, tendremos que definir el mismo trazado por segunda vez, pero invocaremos `pdf_stroke()` en lugar de `pdf_fill()`.

Como la estrella de varias puntas es una forma compleja y repetitiva, hemos escrito una función que calcula por nosotros las posiciones del trazado. La función es `draw_star()` y requiere coordenadas x e y para el centro, el número de puntos necesarios, el radio, la longitud de los puntos, un identificador de documento PDF y un valor booleano para indicar si la forma de estrella tendrá relleno o solamente un contorno.

La función `draw_star()` utiliza conceptos de trigonometría básicos para calcular la posición de una serie de puntos para trazar nuestra estrella. En cada punto de la estrella, buscamos un punto del radio de la estrella y un punto en un círculo más pequeño `$point_size` dentro del círculo interior, y trazamos una línea entre ambos. Conviene recordar que las funciones de trigonometría de PHP como `cos()` y `sin()` trabajan con radianes y no con grados.

Por medio de una función y de algunos cálculos matemáticos podemos generar una forma compleja y repetitiva con gran precisión. Si hubiéramos necesitado un modelo de borde más complicado, podríamos haber empleado un enfoque similar.

Una vez generados todos los elementos de la página, tendremos que finalizar la página y el documento.

Problemas con los encabezados

Un aspecto que conviene mencionar en todas estas secuencias de comandos es que tendremos que indicar al navegador qué tipo de datos vamos a enviar.

Para ello enviamos un encabezado HTTP de tipo de contenido, como por ejemplo

```
header( 'Content-type: application/msword' );
```

o

```
header( 'Content-type: application/pdf' );
```

Debe saber que los navegadores procesan estos encabezados de forma poco coherente. En concreto, Internet Explorer suele ignorar el tipo MIME e intenta detectar automáticamente el tipo de archivo, aunque parece que en las últimas versiones se ha mejorado este problema.

Algunos de nuestros encabezados pueden resultar problemáticos con encabezados de control de sesión, para lo que existen distintas soluciones. Una de ellas consiste en utilizar parámetros GET en lugar de POST o parámetros de variable de sesión.

Otra solución es no utilizar PDF en línea sino que el usuario lo descargue, como indicamos en el ejemplo Hello World.

También se pueden evitar muchos de estos problemas si se escriben dos versiones del código diferentes, una para Netscape y otra para Internet Explorer.

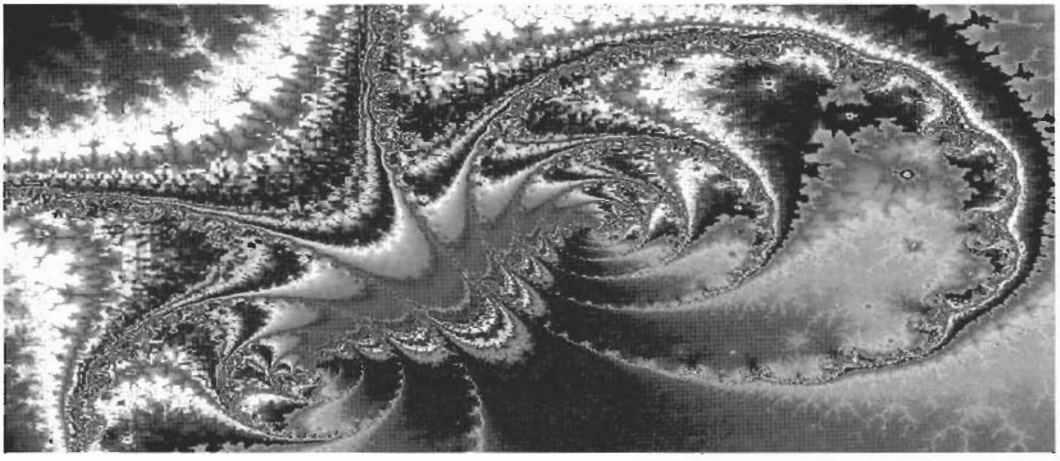
Ampliación del proyecto

La inclusión de tareas de evaluación más realistas mejoraría sin duda este proyecto, pero la intención de nuestro ejemplo es mostrar distintas técnicas de enviar documentos.

Entre los documentos personalizados que puede que le interese enviar en línea se pueden incluir documentos legales, formularios de pedido o solicitudes parcialmente completados o formularios para organismos gubernamentales.

Lecturas adicionales

Le sugerimos que visite el sitio de Adobe si necesita información adicional sobre los formatos PDF (y FDF), `http://www.adobe.com`.



31

Conexión a servicios Web con XML y SOAP

En los últimos años, XML (Lenguaje de Marcado eXtensible) se ha convertido en uno de los medios de comunicación más importantes. En este capítulo utilizaremos la nueva interfaz de servicios Web de Amazon para crear un carro de la compra en nuestro sitio Web local que utilice Amazon como sistema (llamaremos a esta aplicación Tahuayo, nombre de un afluente del Amazonas). Para ello utilizaremos dos métodos diferentes: SOAP y XML sobre HTTP. Emplearemos las bibliotecas XML (`expat`) y NuSOAP (SOAP) de PHP para implementar ambos métodos.

Nos centraremos en los siguientes aspectos:

- Fundamentos de XML y SOAP
- Uso de XML para comunicarnos con Amazon
- Análisis de XML con la biblioteca XML de PHP
- Almacenamiento en caché de las respuestas
- Comunicación con Amazon a través de NuSOAP

El problema

En este proyecto tenemos dos objetivos.

El primero de ellos consiste en comprender qué son XML y SOAP y cómo utilizarlos en PHP.

El segundo nos permitirá utilizar estas tecnologías para comunicarnos con el mundo exterior. Hemos seleccionado el nuevo programa Servicios Web de Amazon como ejemplo que puede resultarnos útil para nuestro sitio Web.

Amazon ofrece desde hace tiempo un programa que le permite anunciar productos de Amazon en su propio sitio Web. Los usuarios pueden seguir un enlace hasta la página de cada producto en el sitio de Amazon. Si pulsan sobre estos enlaces desde nuestro sitio y adquieren dicho producto, recibiremos una pequeña comisión. El programa de Servicios Web nos permite utilizar Amazon como si fuera un motor: se puede buscar en el mismo y mostrar los resultados desde nuestro propio sitio, o se puede llenar directamente el carro del usuario con los artículos que haya seleccionado mientras navegaba por nuestro sitio. En otras palabras, el cliente utiliza nuestro sitio hasta que quiera pasar por caja, operación que debe realizar a través de Amazon.

Las comunicaciones entre nosotros y Amazon se pueden establecer de dos formas diferentes. Por un lado, por medio de XML sobre HTTP. Si por ejemplo queremos realizar una búsqueda con ayuda de este método, enviamos una solicitud HTTP normal de la información que necesitemos y Amazon responderá con un documento XML en el que se incluya la información solicitada. Tras ello, se puede analizar este documento XML con la biblioteca XML de PHP y mostrar los resultados de la búsqueda al usuario final.

La segunda forma consiste en utilizar SOAP, uno de los protocolos estándar de servicios Web. Anteriormente equivalía a Protocolo de acceso a objetos simple, pero recientemente se decidió que dicho protocolo no era tan simple y que el nombre resultaba ligeramente confuso. Como resultado, el protocolo se sigue llamando SOAP, pero ya no es una sigla.

Generaremos un cliente SOAP que pueda enviar solicitudes a Amazon y recibir las correspondientes respuestas del servidor SOAP de Amazon. Tendrá la misma información que las respuestas obtenidas por medio del método XML, pero utilizaremos un enfoque diferente para extraer los datos, básicamente la biblioteca SOAP de PHP. Nuestro objetivo final en este proyecto es crear nuestro propio sitio Web de venta de libros que utilice Amazon como sistema. Crearemos dos versiones alternativas: una que utilice XML y otra que utilice SOAP.

Cómo entender XML

Nos detendremos brevemente para analizar XML y los servicios Web, en caso de que no esté familiarizado con estos conceptos.

XML es el Lenguaje de marcado extensible. Las especificaciones están disponibles en el W3C. También puede encontrar numerosa información sobre XML en el sitio XML de W3C, en la dirección `http://www.w3.org/XML/`.

XML se deriva de SGML (Lenguaje de Marcado Generalizado eStándar). Si ya conoce HTML (y suponemos que llegados a este punto del libro, sin duda lo hace), no le resultará difícil entender los conceptos de XML.

XML es un formato de texto para documentos basado en etiquetas.

Como ejemplo de documento XML, en el listado 31.1 se incluye una de las respuestas que envía Amazon como contestación a una solicitud XML sobre HTTP.

Listado 31.1. Documento XML que describe la primera edición de este libro

```
<?xml version="1.0" encoding="UTF-8"?>
<ProductInfo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation =
                                "http://xml.amazon.com/schemas2/dev-heavy.xsd">
      <Details url="http://www.amazon.com/exec/obidos/redirect?
         tag=tangledwebdesign%26creative=XXXXXXXXXXXXXX%26
         camp=2025%26link_code=xm2%26path=ASIN/0672317842">
         <Asin>0672317842</Asin>
         <ProductName>PHP and MySQL Web Development</ProductName>
         <Catalog>Book</Catalog>
         <Authors>
            <Author>Luke Welling</Author>
            <Author>Laura Thomson</Author>
         </Authors>
         <ReleaseDate>30 March, 2001</ReleaseDate>
         <Manufacturer>Sams</Manufacturer>
         <ImageUrlSmall>
            http://images.amazon.com/images/P/0672317842.01.THUMBZZZ.jpg
         </ImageUrlSmall>
         <ImageUrlMedium>
            http://images.amazon.com/images/P/0672317842.01.MZZZZZZZ.jpg
         </ImageUrlMedium>
         <ImageUrlLarge>
            http://images.amazon.com/images/P/0672317842.01.LZZZZZZZ.jpg
         </ImageUrlLarge>
         <ListPrice>$49.99</ListPrice>
         <OurPrice>$34.99</OurPrice>
         <UsedPrice>$31.95</UsedPrice>
         <ThirdPartyNewPrice>$31.75</ThirdPartyNewPrice>
         <SalesRank>312</SalesRank>
         <Lists>
            <ListId>3KZW1EV9QMB5F</ListId>
            <ListId>22YCO1IGPIZJ3</ListId>
            <ListId>Y2I9B362QXVX</ListId>
         </Lists>
         <BrowseList>
            <BrowseNode>
               <BrowseName>PHP (Computer program language</BrowseName>
            </BrowseNode>
            <BrowseNode>
               <BrowseName>SQL (Computer program language</BrowseName>
            </BrowseNode>
            <BrowseNode>
               <BrowseName>Web sites</BrowseName>
            </BrowseNode>
            <BrowseNode>
               <BrowseName>Design</BrowseName>
            </BrowseNode>
```

```
            <BrowseNode>
               <BrowseName>SQL (Computer language)</BrowseName>
            </BrowseNode>
            <BrowseNode>
               <BrowseName>Sql (Programming Language)</BrowseName>
            </BrowseNode>
            <BrowseNode>
               <BrowseName>Computer Networks</BrowseName>
            </BrowseNode>
            <BrowseNode>
               <BrowseName>Computer Bks - Languages/Programming</BrowseName>
            </BrowseNode>
            <BrowseNode>
               <BrowseName>Computers</BrowseName>
            </BrowseNode>
            <BrowseNode>
               <BrowseName>Programming Languages - General</BrowseName>
            </BrowseNode>
            <BrowseNode>
               <BrowseName>Internet - Web Site Design</BrowseName>
            </BrowseNode>
            <BrowseNode>
               <BrowseName>Database Management - SQLServer</BrowseName>
            </BrowseNode>
            <BrowseNode>
               <BrowseName>Programming Languages - SQL</BrowseName>
            </BrowseNode>
         </BrowseList>
         <Media>Paperback</Media>
         <NumMedia>1</NumMedia>
         <Isbn>0672317842</Isbn>
         <Availability>Usually ships within 24 hours</Availability>
         <SimilarProducts>
            <Product>0735709211</Product>
            <Product>1861003730</Product>
            <Product>073570970X</Product>
            <Product>1861006918</Product>
            <Product>0596000413</Product>
         </SimilarProducts>
      </Details>
   </ProductInfo>
```

El documento comienza con la siguiente línea:

```
<?xml version="1.0" encoding="UTF-8"?>
```

Es una declaración estándar que nos indica que el siguiente documento será XML y utilizará la codificación de caracteres UTF-8.

Fíjese en el cuerpo del documento.

Está enteramente formado por pares de etiquetas de apertura y de cierre como por ejemplo:

```
<ProductName>PHP and MySQL Web Development</ProductName>
```

`ProductName` es un elemento, al igual que lo sería en HTML. Y, al igual que en HTML, podemos anidar elementos:

```
 <Authors>
   <Author>Luke Welling</Author>
   <Author>Laura Thomson</Author>
<
```

Como sucede también en HTML, los elementos pueden tener atributos. Por ejemplo:

```
 <Details url="http://www.amazon.com/exec/obidos/redirect?tag=
tangledwebdesign%26creative=XXXXXXXXXXXXXX%26camp=2025%26link_code
=xm2%26path=ASIN/0672317842">
```

Este elemento `Details` tiene un único atributo `url`.

También existen diferencias con respecto a HTML. En primer lugar, todas las etiquetas de apertura deben tener una etiqueta de cierre (la única excepción es que se utilice un formato especial. Si está familiarizado con XHTML habrá comprobado que se utiliza la etiqueta `<br />` en lugar de `<br>` por esta razón). Además, todos los elementos deben anidarse correctamente.

La principal diferencia entre XML y HTML es que da la sensación de que creamos nuestras propias etiquetas sobre la marcha. Se debe a la flexibilidad de XML. Podemos estructurar nuestros documentos para que coincidan con los datos que queremos almacenar. Podemos formalizar la estructura de los documentos XML si escribimos una DTD (Definición de tipo de documento) o un Esquema XML. Ambos documentos se utilizan para describir la estructura de un determinado documento XML. Si lo prefiere, diremos que la DTD o el Esquema son como la declaración de una clase y que el documento XML es como una instancia de dicha clase.

Puede leer la DTD de Amazon para este documento en la siguiente dirección `http://xml.amazon.com/schemas2/dev-heavy.dtd`.

Puede leer el Esquema XML en `http://xml.amazon.com/schemas2/dev-heavy.xsd`.

En algunos navegadores no podrá abrir el archivo DTD, ya que tratan de analizar la DTD como XML y se producen confusiones. Sin embargo, puede descargarlo y leerlo en el editor que prefiera. El Esquema XML debería poder abrirlo directamente en su navegador.

Habrá comprobado que, aparte de la declaración XML inicial, la totalidad del cuerpo del documento se incluye dentro del elemento `ProductInfo`. Es lo que se denomina *elemento raíz* del documento. Veámoslo con más detalle:

```
<ProductInfo  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="http://xml.amazon.com/schemas2/
    dev-heavy.xsd">
```

Comprobará que tiene algunos elementos poco habituales. Se denominan *Espacios de nombres XML*. Para los objetivos de este proyecto no es necesario que comprenda los espacios de nombre, pero pueden resultar muy útiles. La idea básica es calificar

nombres de elementos y atributos con un espacio de nombres para que los nombres comunes no produzcan errores al utilizar documentos de diferentes fuentes.

Si necesita más información sobre espacios de nombres, puede leer el documento "Namespaces in XML Recommendations", en la dirección `http://www.w3.org/TR/REC-xml-names/`.

Si quiere conocer más sobre XML, existe una amplia variedad de fuentes. El sitio de W3C es un excelente punto de partida y, literalmente, existen cientos de libros y tutoriales Web. ZVON.org es uno de los mejores basado en la Web.

Servicios Web

Los servicios Web son interfaces de aplicación que se ofrecen a través de la World Wide Web. Puede considerar un servicio Web como una clase que expone sus métodos públicos a través de la Web. Los servicios Web se han hecho muy populares y algunos de los principales nombres del sector empresarial ofrecen su funcionalidad a través de este tipo de servicios.

Por ejemplo, Google ofrece en la actualidad una gran variedad de servicios Web. Después de que en este capítulo definamos un cliente a la interfaz de Amazon, comprobará que la creación de una interfaz cliente a Google resulta muy sencilla. Encontrará más información al respecto en `http://www.google.com/apis/`.

Una lista cada vez más amplia de servicios Web se encuentra disponible en `http://www.xmethods.net`.

Existen diversos protocolos básicos implicados en esta metodología de invocación de funciones remotas. Los dos más importantes son SOAP y WSDL.

SOAP

SOAP es un protocolo de mensajería dirigido por solicitudes y respuestas que permite a los clientes invocar servicios Web y permite a los servidores responder. Todos los mensajes SOAP, ya sean solicitudes o respuestas, son simples documentos XML. En el listado 31.2 se recoge un ejemplo de las solicitudes SOAP que podemos enviar a Amazon.

Listado 31.2. Solicitud SOAP de una búsqueda basada en ASIN

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
                  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
                  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
                  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP-ENV:Body>
      <namespl:AsinSearchRequest
```

```
            xmlns:namespl="urn:PI/DevCentral/SoapService">
         <AsinSearchRequest xsi:type="m:AsinRequest">
            <asin >0060518057</asin>
            <tag >your-associate-id</tag>
            <type >heavy</type>
            <dev-tag >your-dev-tag</dev-tag>
         </AsinSearchRequest>
      </namespl:AsinSearchRequest>
   </SOAP-ENV:Body>
<
```

El mensaje SOAP comienza con la declaración de que se trata de un documento XML. El elemento raíz de todos los mensajes SOAP es el sobre SOAP. Dentro de éste se incluye el elemento `Body`, que contiene la verdadera solicitud.

Esta solicitud es una `AnsiSearchRequest`, que pide al servidor Amazon que busque un determinado artículo en la base de datos en función del ASIN, el número de artículo estándar de Amazon.com. Se trata de un identificador exclusivo asignado a cada uno de los productos de la base de datos de Amazon.com.

`AsinSearchRequest` es una especie de llamada de función en un equipo remoto y los elementos que contiene este elemento podrían ser los parámetros que se pasan a dicha función. En este caso, pasamos un ASIN para el libro de Dilbert *Way of the Weasel*. También será necesario pasar la etiqueta, que es su Id. de asociado, el tipo de búsqueda que se va a realizar (`heavy` o `lite`) y el `dev-tag`, el valor que le asigna Amazon.com. El tipo de elemento indica al servicio si queremos información general (`lite`) o toda la información disponible (`heavy`).

La respuesta a esta solicitud es muy similar al documento XML que vimos en el listado 31.1, pero se encierra en un sobre SOAP.

Al trabajar con SOAP, tendremos que generar solicitudes SOAP e interpretar respuestas por medio de programación con ayuda de una biblioteca SOAP, independientemente del lenguaje de programación que utilicemos. Es muy positivo ya que nos evita tener que generar la solicitud SOAP e interpretar la respuesta manualmente.

WSDL

WSDL significa Lenguaje de descripción de servicios Web. Se emplea para describir la interfaz a los servicios disponibles de un determinado sitio Web. Si quiere ver el documento WSDL que describe los servicios Web de Amazon que utilizaremos en este capítulo, lo encontrará en la dirección `http://soap.amazon.com/schemas2/AmazonWebServices.wsdl`.

Como comprobará si utiliza este enlace, los documentos WSDL son considerablemente más complejos que los mensajes SOAP. Si tiene la oportunidad, tendrá que generarlos e interpretarlos mediante programación.

Si quiere saber más sobre WSDL, puede consultar el proyecto de W3C en la dirección `http://www.w3.org/TR/2002/WD-wsdl12-20020709/`.

Al cierre de la edición de este libro, WSDL no era todavía una Recomendación W3C por lo que es posible que se modifique. Sin embargo esto no ha impedido a los

programadores que lo utilicen con gran entusiasmo. Pero al igual que todas las piezas del puzzle que componen los servicios Web, siempre puede cambiar debido a su novedad.

Componentes de la solución

Para generar nuestra solución, tendremos que detenernos en diferentes aspectos.

Creación de un carro de la compra

Será necesario generar un carro de la compra como interfaz de nuestro sistema, algo que ya hemos hecho en un capítulo anterior. Como los carros de la compra no son la parte principal de este proyecto, utilizaremos una aplicación simplificada. Basta con proporcionar un carro básico para poder realizar el seguimiento de lo que quiere comprar el cliente e informar de ello a Amazon en el momento de efectuar la compra.

Uso de las interfaces de servicios Web de Amazon

Para utilizar la interfaz de servicios Web de Amazon, tendrá que descargar el paquete Amazon Web Services Developers' Kit, de la siguiente dirección:

```
http://associates.amazon.com/exec/panama/associates/join/developer/
kit.html. Este URL puede que haya cambiado.
```

También tendrá que solicitar, en el mismo sitio, un Developer Token. Se utilizará para identificar las solicitudes que remita a Amazon.

También puede obtener un Id. de asociado a Amazon, que le permitirá recibir una comisión si los usuarios realizan compras a través de su interfaz.

Una vez descargado el paquete, léalo. Incluye documentación sobre el funcionamiento de la interfaz y ejemplos de código en distintos lenguajes, incluyendo PHP.

Antes de poder descargarlo, tendrá que aceptar el contrato de licencia.

Conviene que lo lea atentamente, ya que no se trata del típico contrato de licencia. Algunas de las condiciones importantes durante la implementación son las siguientes:

- No debe realizar más de una solicitud por segundo.
- Debe almacenar en caché los datos que provengan de Amazon.
- No debe almacenar en caché los precios durante más de una hora.
- No debe almacenar en caché otros datos durante más de 24 horas.
- Debe vincular todos los datos de Amazon a la página de Amazon.

Con un nombre de dominio difícil de pronunciar, sin promoción y sin una razón aparente para utilizar Tahuayo.com en lugar de acceder directamente a Amazon.com, no tendremos que hacer nada más para conseguir menos de una solicitud por segundo.

Hemos implementado el almacenamiento en caché para cumplir las condiciones expuestas anteriormente. Almacenamos en caché las imágenes durante 24 horas y los datos de los productos (incluyendo los precios) durante una hora.

Hemos optado por ignorar el punto 5. Queremos que los artículos de la página principal estén vinculados a páginas de nuestro sitio y que solamente se vinculen a Amazon cuando estén completos. Es el funcionamiento habitual de las aplicaciones de carro de la compra y el funcionamiento de todas las aplicaciones de terceros vinculadas al sitio de Amazon. Esto nos permite creer que a Amazon realmente no le importa el incumplimiento de esta condición. Puede tomar otra decisión con total libertad. Un empleado de los servicios Web de Amazon indicó que conocían esta situación y que en el futuro tomarían medidas al respecto.

Análisis de XML

La primera interfaz que ofrece Amazon a sus servicios Web es a través de XML sobre HTTP. Esta interfaz acepta una solicitud HTTP normal y devuelve un documento XML. Para utilizar esta interfaz tendremos que analizar la respuesta XML que nos envíe Amazon. Para ello utilizaremos la biblioteca XML de PHP, que se basa en el analizador `expat` escrito por James Clark. Es el mismo analizador XML utilizado por Mozilla. Es un analizador SAX. Si necesita más información sobre la biblioteca XML de PHP, puede consultar el Manual de PHP o la documentación sobre `expat` en la dirección `http://www.jclark.com/xml/expat.html`.

Uso de SOAP con PHP

La otra interfaz que ofrece los mismos servicios Web es SOAP. Para acceder a ellos por medio de SOAP, tendremos que utilizar una de las diferentes bibliotecas SOAP de PHP. Durante la edición de este libro había tres bibliotecas SOAP de PHP principales: PHP-SOAP, PEAR SOAP y NuSOAP.

PHP-SOAP es una extensión de C a PHP y probablemente se convierta en la extensión SOAP oficial. Durante la edición de este libro seguía en desarrollo. PEAR SOAP es un conjunto de archivos de clases PHP en PEAR que permite crear clientes y servidores SOAP. NuSOAP es un conjunto de clases PHP que también permiten crear clientes y servidores SOAP.

Hemos optado por utilizar NuSOAP, ya que es la biblioteca con mayor documentación. También parece ser la más utilizada por la mayoría de los programadores de servicios Web de Amazon que utilizan PHP, en parte porque Amazon la utiliza en su código PHP de ejemplo.

NuSOAP está disponible en `http://dietrich.ganx4.com/nusoap/`, bajo Lesser GPL, es decir, la debe utilizar en aplicaciones que no sean gratuitas.

Almacenamiento en caché

Como hemos mencionado anteriormente, una de las condiciones que impone Amazon a los programadores es que los datos descargados de Amazon a través de los servicios Web se deben almacenar en caché. En nuestra solución, tendremos que encontrar una forma de almacenar y reutilizar los datos que descarguemos hasta que se pase su fecha de utilización.

Presentación de la solución

En este proyecto volveremos a utilizar un enfoque dirigido a eventos para escribir nuestro código, como hicimos en capítulos anteriores. No dibujaremos un diagrama de flujo del sistema ya que éste cuenta con escasas pantallas y los enlaces entre las mismas son muy sencillos.

Los usuarios comenzarán en la pantalla principal de Tahuayo (figura 31.1).

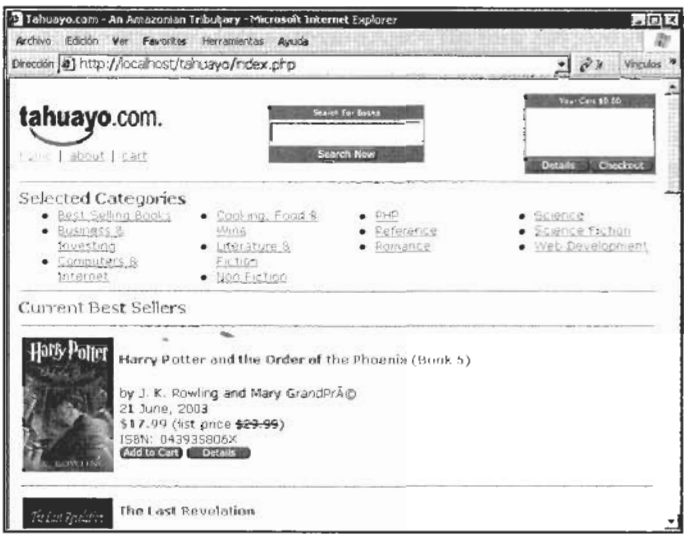


Figura 31.1. La primera pantalla de Tahuayo muestra todas las características principales del sitio: navegación entre categorías, búsquedas y el carro de la compra

Como puede comprobar, las características principales del sitio son la pantalla Selected Category y los artículos de dichas categorías. De forma predeterminada, en la página principal mostramos la categoría Current Best Sellers. Si un usuario pulsa sobre otra categoría, accederá a una pantalla similar para la misma.

Un pequeño inciso terminológico antes de continuar. Amazon se refiere a las categorías como *nodos de navegación*. Verá que esta expresión se utiliza a lo largo del código y en la documentación oficial.

La documentación proporciona una lista parcial de los nodos de navegación más populares. Además, si quiere uno en concreto, puede acceder al sitio normal de Amazon.com y leerlo desde el URL, pero no se puede obtener una lista completa.

Si se fija, verá que en la parte inferior de la página hay más libros y enlaces a páginas adicionales de los que se pueden ver en esta imagen. Mostraremos 10 libros por página, junto con enlaces a un máximo de 30 páginas. Este valor de 10 por página lo establece Amazon. El límite de 30 páginas es una decisión arbitraria personal.

Desde aquí, los usuarios pueden pulsar sobre la información detallada de cada libro. En la figura 30.2 se puede ver esta pantalla.



Figura 31.2. La página de detalles muestra información adicional sobre un determinado libro, incluyendo productos similares y críticas

Aunque no cabe en una sola figura, mostramos la práctica totalidad de la información que nos envía Amazon con una solicitud a esta página. Hemos optado por omitir las partes dirigidas a otros productos que no sean libros así como la lista de categorías en las que se incluye el libro.

Si el usuario pulsa sobre la imagen de la portada, podrá ver una versión de mayor tamaño de la misma.

Habrá reparado en el cuadro de búsqueda situado en la parte superior de la pantalla. Esta función ejecuta una búsqueda de palabras clave en nuestro sitio y en el catálogo de Amazon a través de su interfaz de servicios Web.

En la figura 31.3 se muestra un ejemplo del resultado que devuelve una de estas búsquedas.

Aunque sólo mostramos algunas categorías, los clientes pueden acceder a cualquier libro por medio de la opción de búsqueda así como desplazarse hasta libros concretos.

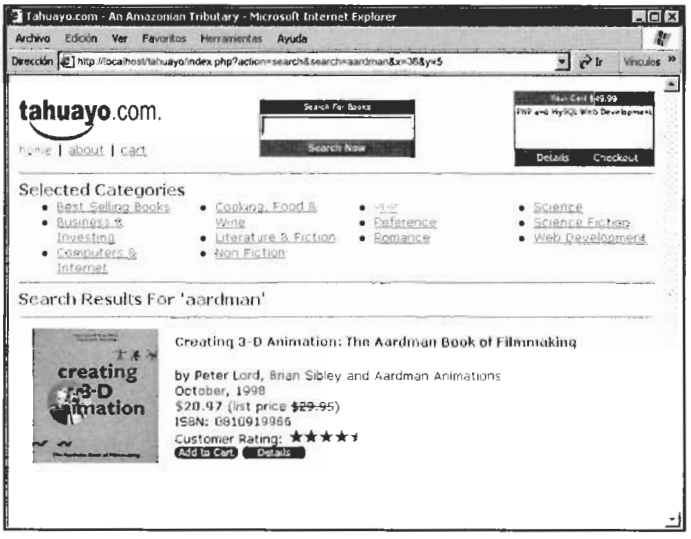


Figura 31.3. Resultados de la búsqueda aardman

Cada libro tiene un enlace **Add to Cart** (Añadir al carro). Al pulsar sobre este enlace o sobre el enlace **Details** (Detalles) se accede a los contenidos del carro, como se muestra en la figura 31.4.



Figura 31.4. Desde la página del carro de la compra se pueden borrar artículos, vaciar el carro o pasar por caja

Por último, cuando un cliente pasa por caja al pulsar sobre uno de los enlaces **Checkout**, enviamos los detalles de su carro de la compra a Amazon y le llevamos hasta allí.

Accederá a una página similar a la de la figura 31.5.

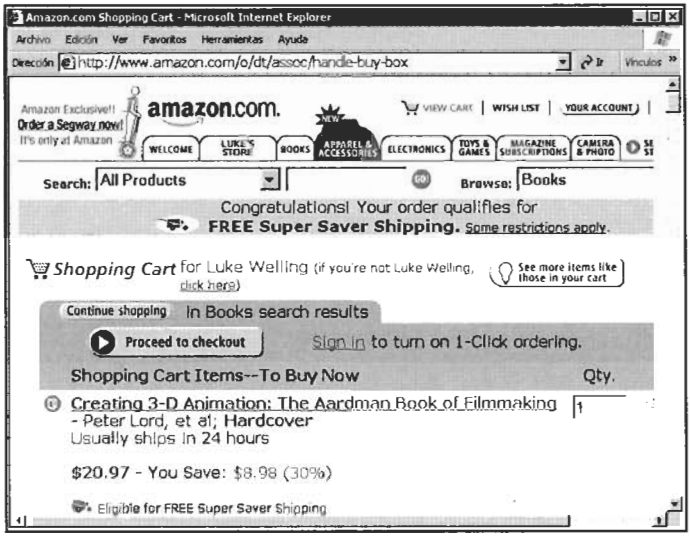


Figura 31.5. Los artículos que estaban en el carro del cliente en Tahuayo aparecen ahora en su carro de Amazon

Con esto debería comprender lo que queremos hacer al generar nuestra propia interfaz de usuario y utilizar Amazon como sistema.

Como vamos a utilizar de nuevo el enfoque dirigido por eventos, la mayor parte de la lógica de toma de decisiones se encuentra en un solo archivo, `index.php`. En la tabla 31.1 se resumen los archivos de esta aplicación.

Tabla 31.1. Archivos de la aplicación Tahuayo

Nombre del archivo	Tipo	Descripción
`index.php`	Aplicación	El archivo de aplicación principal
`about.php`	Aplicación	Muestra la página About
`constants.php`	Archivo de inclusión	Define constantes globales
`topbar.php`	Archivo de inclusión	Genera la barra de información que aparece en la parte superior de todas las páginas y la CSS
`bottom.php`	Archivo de inclusión	Genera el pie de página de todas las páginas
`AmazonResultSet.php`	Archivo de clases	Contiene la clase de PHP que almacena el resultado de cada una de las consultas Amazon

Nombre del archivo	Tipo	Descripción
`Product.php`	Archivo de clases	Contiene la clase de PHP que almacena información sobre un determinado libro
`bookdisplayfunctions.php`	Funciones	Contiene funciones que permiten mostrar un libro y una lista de libros
`cachefunctions.php`	Funciones	Contiene funciones para realizar el almacenamiento en caché requerido por Amazon
`cartfunctions.php`	Funciones	Contiene funciones relacionadas con el carro de la compra
`categoryfunctions.php`	Funciones	Contiene funciones que permiten recuperar y mostrar una categoría
`utilityfunctions.php`	Funciones	Contiene una conjunto de funciones de utilidad que se utilizan a lo largo de la aplicación

También necesitará el archivo `nusoap.php` que mencionamos anteriormente, ya que es obligatorio en estos archivos. NuSOAP se encuentra en el directorio del CD-ROM correspondiente a este capítulo, pero puede reemplazarlo por la nueva versión que encontrará en `http://dietrich.ganx4.com/nusoap/index.php`.

En primer lugar analizaremos el archivo de aplicación principal, `index.php`.

Aplicación principal

El archivo de aplicación `index.php` se muestra en el listado 31.3.

Listado 31.3. index.php. Archivo de aplicación principal

```
<?php
//utilizaremos únicamente una variable de sesión, 'cart',
//para almacenar los contenidos del carro
session_start();

require_once('constants.php');
require_once('utilityfunctions.php');
require_once('bookdisplayfunctions.php');
require_once("cartfunctions.php");
require_once('categoryfunctions.php');

// Éstas son las variables que esperamos del exterior.
```

```
// Se validarán y convertirán en globales
$external = array('action', 'ASIN', 'mode', 'browseNode', 'page', 'search');

// las variables pueden venir de Get o Post
// convierta todas las variables externas esperadas en nombres
// globales cortos
foreach ($external as $e)
{
   if(@$HTTP_POST_VARS[$e])
      $$e = $HTTP_POST_VARS[$e];
   else if(@$HTTP_GET_VARS[$e])
      $$e = $HTTP_GET_VARS[$e];
   else
      $$e = '';

   $$e = trim($$e);
}

// valores predeterminados para las variables globales
if($mode=='')
   $mode = 'books'; // No se han probado otros modos
if($browseNode=='')
   $browseNode = 1000; //1000 corresponde a libros superventas
if($page=='')
   $page = 1; // Primera página - hay 10 artículos por página

//validar/recortar entrada
if(!eregi('^[A-Z0-9]+$', $ASIN)) // los ASIN deben ser alfanuméricos
   $ASIN ='';
if(!eregi('^[a-z]+$', $mode)) // el modo debe ser alfabético
   $mode = 'books';
$page=intval($page); // las páginas y los browseNodes deben ser enteros
$browseNode = intval($browseNode);
// puede resultar confuso, pero eliminamos caracteres de
// $search; es aconsejable modificarlo ahora para que se muestre
// en el título
$search = safeString($search);

if(!isset($HTTP_SESSION_VARS['cart']))
{
   session_register('cart');
   $HTTP_SESSION_VARS['cart'] = array();
}

// tareas que debe realizar antes de mostrar la barra superior
if($action == 'addtocart')
   addToCart($HTTP_SESSION_VARS['cart'], $ASIN, $mode);
if($action == 'deletefromcart')
   deleteFromCart($HTTP_SESSION_VARS['cart'], $ASIN) ;
if($action == 'emptycart')
   $HTTP_SESSION_VARS['cart'] = array();

// muestre la barra superior
require_once ('topbar.php');

// bucle de eventos principal. Reacciona a acciones del usuario en la
```

```
// página de invocación
switch ($action)
{
   case 'detail' :
      showCategories($mode);
      showDetail($ASIN,  $mode);
   break;

   case  'addtocart'  :
   case 'deletefromcart' :
   case 'emptycart' :
   case 'showcart' :
      echo '<hr /><h1>Your Shopping Cart</h1>';
      showCart($HTTP_SESSION_VARS['cart'], $mode);
   break;

   case 'image' :
      showCategories($mode);
      echo '<h1>Large Product Image</h1>';
      showImage($ASIN, $mode);
   break;

   case 'search' :
      showCategories($mode);
      echo "<h1>Search Results For '$search'</h1>";
      showSearch($search, $page, $mode);
   break;

   case 'browsenode':
   default:
      showCategories($mode);
      $category  =  getCategoryName($browseNode);
      if(!$category||$category=='Best Selling Books')
      {
         echo '<h1>Current Best Sellers</h1>';
      }
      else
      {
         echo "<h1>Current Best Sellers in $category</h1>";
      }
      showBrowseNode($browseNode, $page, $mode) ;
   break;
}
require ('bottom.php');
? >
```

Analicemos este archivo. En primer lugar crearemos una sesión. Almacenaremos el carro de la compra del usuario como variable de sesión, como hemos hecho anteriormente. Tras ello incluiremos varios archivos.

La mayoría son funciones que describiremos más adelante, pero nos detendremos en el primero de ellos.

Este archivo, `constants.php`, define importantes constantes que se utilizarán a lo largo de la aplicación.

Los contenidos de `constants.php` se muestran en el listado 31.4.

Listado 31.4. constants.php. declaración de constantes globales clave

```
<?php
// se puede realizar la conexión a esta aplicación por medio de XML/HTTP o SOAP
// defina la versión de METHOD que quiera seleccionar.
define('METHOD', 'SOAP');
//defina('METHOD',  'XML/HTTP');

// asegúrese de crear el directorio caché, para que se pueda escribir
define('CACHE', 'cache'); // ruta a los archivos en caché
define('ASSOCIATEID', 'webservices-20); //añada aquí su Id. de asociado
define('DEVTAG', 'XXXXXXXXXXXXXX'); // añada aquí su etiqueta de desarrollador

//proporcione un error si se ejecuta el software con la devtag ficticia
if(DEVTAG=='XXXXXXXXXXXXXX')
   die ( 'You need to sign up for an Amazon.com developer tag at<a href =
         "http://associates.amazon.com/exec/panama/associates/join/
         developer/kit.html">Amazon</a>
         when you install this software.  You should probably sign up
         for an associate ID at the same time. Edit the file constants.php.');
? >
```

Esta aplicación se ha desarrollado para utilizar XML sobre HTTP o SOAP. Puede indicar cuál va a utilizar si cambia el valor de la constante `METHOD`.

La constante `CACHE` contiene la ruta a la caché de los datos que descargaremos de Amazon. Puede cambiarla por la ruta que quiera utilizar en su sistema.

La constante `ASSOCIATED` almacena el valor de su Id. de asociado. Si lo envía a Amazon con transacciones, recibirá una comisión.

Cámbielo por su Id. de asociado.

La constante `DEVTAG` almacena el valor del identificador de programador que Amazon le asigna al registrarse. Tendrá que cambiarlo por su propia etiqueta de programador o la aplicación no funcionará. Puede solicitar una etiqueta en `http://associates.amazon.com/exec/panama/associates/join/developer/kit.html`.

Volvamos a `index.php`. Contiene algunos preliminares así como el bucle de eventos principal. En primer lugar extraemos todas las variables entrantes que provengan a través de `GET` o `POST`.

Tras ello, definimos valores predeterminados para ciertas variables globales estándar que determinan qué se mostrará posteriormente, como se indica a continuación:

```
// valores predeterminados para las variables globales
if($mode=='')
   $mode = 'books'; // No se han probado otros modos
if($browseNode=='')
   $browseNode = 1000; //1000 corresponde a libros superventas
if($page=='')
   $page = 1;  // Primera página - hay 10 artículos por página
```

Hemos definido la variable `mode` con el valor `books`. Amazon admite otros muchos modos (tipos de productos) pero en esta aplicación sólo nos centraremos

en libros. No debe resultarle complicado modificar el código de este capítulo si quiere trabajar con otras categorías. El primer paso consistirá en restablecer `$mode`. Tendrá que revisar la documentación de Amazon para ver qué otros atributos se devuelven para productos que no sean libros y eliminar el lenguaje específico de los libros de la interfaz de usuario.

La variable `browseNode` se utiliza para especificar qué categoría de libros queremos mostrar. Debe configurarse si el usuario ha pulsado uno de los enlaces **Selected Categories**. Si no se configura, por ejemplo, cuando el usuario accede por primera vez al sitio, la definiremos como `1000`. Los nodos de navegación de Amazon son simples enteros que identifican a una categoría. El valor `1000` representa la categoría **Best Selling Books**, que es la que mostramos en la página inicial.

La variable `page` se utiliza para indicar a Amazon qué subconjunto de los resultados queremos mostrar dentro de una determinada categoría. La página 1 contiene los resultados del 1 al 10, la página 2 los resultados del 11 al 20 y así sucesivamente. Amazon define el número de artículos por página, algo que no podemos controlar personalmente. Evidentemente podríamos mostrar dos o más "páginas" Amazon de datos en una de cada cuatro páginas pero 10 es un número razonable.

A continuación, limpiamos los datos que hayamos recibido, ya sea a través del cuadro de búsqueda o por medio de parámetros `GET` o `POST`:

```
//validar/recortar entradas
if(!eregi('^[A-Z0-9]+$', $ASIN)) // los ASIN deben ser alfanuméricos
   $ASIN ='';
if(!eregi('^[a-z]+$', $mode)) // el modo debe ser alfabético
   $mode  =  'books';
$page=intval($page); // las páginas y los browseNodes deben ser enteros
$browseNode = intval($browseNode);
//  puede  resultar  confuso,  pero  eliminamos  caracteres  de
//  $search
// es aconsejable modificarlo ahora para que se muestre
// en el título
$search = safeString($search) ;
```

Nada nuevo. La función `safeString()` se encuentra en el archivo `utilityfunctions.php`. Simplemente elimina todos los caracteres que no sean alfanuméricos de la cadena a través de un reemplazo de función convencional. Como ya hemos mencionado esta operación anteriormente, no la describiremos.

La razón principal por la que tenemos que validar las entradas es que utilizaremos las entradas del usuario para crear nombres de archivo en la caché. Podríamos tener serios problemas si permitiéramos que los usuarios incluyeran `..` o `/` en sus entradas.

Seguidamente configuramos el carro de la compra del usuario, en caso de que no tenga uno:

```
if(!isset($HTTP_SESSION_VARS['cart']))
{
   session_register('cart');
   $HTTP_SESSION_VARS['cart'] = array();
}
```

Antes de poder mostrar la información en la barra superior de la página (figura 31.1) tendremos que ocuparnos de un par de aspectos. En esta barra superior se muestra una imagen del carro de la compra. Por lo tanto, conviene que la variable `cart` esté actualizada antes de poder representarla.

```
// tareas que deben realizarse antes de mostrar la barra superior
if($action == 'addtocart')
   addToCart($HTTP_SESSION_VARS['cart'], $ASIN, $mode);
if($action == 'deletefromcart')
   deleteFromCart($HTTP_SESSION_VARS['cart'], $ASIN);
if($action == 'emptycart')
   $HTTP_SESSION_VARS['cart'] = array();
```

En este caso, según sea necesario, añadimos o eliminamos artículos del carro antes de mostrarlo. Volveremos a estas funciones cuando analicemos el carro de la compra y el pago. Si quiere verlas ahora, las encontrará en el archivo `cartfunctions.php`. Por el momento las dejaremos de lado ya que primero conviene entender la interfaz de Amazon.

Tras ello, incluimos el archivo `topbar.php`, que simplemente contiene HTML, una hoja de estilo y una sola invocación a la llamada `ShowSmallCart()` (de `cartfunctions.php`).

De esta forma se muestra el pequeño resumen del carro de la compra que aparece en la esquina superior derecha de las imágenes. Volveremos a analizarlo cuando describamos las funciones del carro.

Por último, llegamos al bucle de control de eventos principal. En la tabla 31.2 se recoge un resumen de las posibles acciones.

Tabla 31.2. Posibles acciones en el bucle de eventos principal

Acción	Descripción
`browsenode`	Muestra libros de la categoría especificada. Es la acción predeterminada.
`detail`	Muestra los detalles de un determinado libro.
`image`	Muestra una versión ampliada de la portada del libro.
`search`	Muestra los resultados de la búsqueda de un usuario.
`addtocart`	Añade un artículo al carro de la compra del usuario.
`deletefromcart`	Elimina un artículo del carro de la compra del usuario.
`emptycart`	Vacía el carro de la compra.
`showcart`	Muestra los contenidos del carro.

Como puede comprobar, las primeras cuatro acciones de la tabla están relacionadas con la recuperación y la representación de información de Amazon. Las cuatro siguientes están relacionadas con el carro de la compra.

Todas las acciones que recuperan datos de Amazon funcionan de forma similar.

Veremos un ejemplo en el que se recuperan datos de libros en una determinada categoría (`browsenode`).

Cómo mostrar los libros de una categoría

El código que se ejecuta cuando la acción es `browsenode` (ver una categoría) es el siguiente:

```
showCategories($mode);
$category = getCategoryName($browseNode);
if(!$category||$category=='Best Selling Books')
{
   echo '<h1>Current Best Sellers</h1>';
}
else
{
   echo "<h1>Current Best Sellers in $category</h1>";
}
showBrowseNode($browseNode, $page, $mode);
```

La función `showCategories()` muestra la lista de categorías seleccionadas que aparecen en la parte superior de la mayoría de las páginas.

La función `getCategoryName()` devuelve el nombre de la categoría actual si se proporciona su número `browsenode()`.

La función `showBrowseNode()` muestra una página de libros en dicha categoría. En primer lugar nos detendremos brevemente en la función `showCategories()`.

El código de esta función se recoge en el listado 31.5.

Listado 31.5. Función showCategories() de categoryfunctions.php. Una lista de categorías

```
//muestre una lista inicial de categorías populares
function showCategories($mode)
{
   global $categoryList;
   echo '<hr /><h2>Selected Categories</h2>';

   if($mode == 'books')
   {

      asort($categoryList);

      $categories = count($categoryList);
      $columns = 4;
      $rows = ceil($categories/$columns);

      echo '<table border ="0" cellpadding ="0" cellspacing="0"
                  width ="100%"><tr>';

      reset($categoryList);

      for($col = 0; $col<$columns; $col++)
```

```
        {
        echo '<td width = "'.(100/$columns).'%" valign = "top"><ul>';
        for($row  =  0;  $row<$rows;  $row++)
        {
           $category = each($categoryList);
           if($category)
           {
              $browseNode = $category['key'];
              $name = $category['value'];
              echo  "<li><span class = 'category'><a href =
                    'index.php?action=browsnode&browseNode=$browseNode'>$name
                    </a></span></li>";

           }
           echo '</ul></td>';
        }
        echo '</tr></table><hr />';
     }
}
```

Esta función utiliza una matriz denominada `categoryList()` declarada en el paquete `categoryfunctions.php` para asignar números `browsenode` a nombres. Esta función ordena la matriz y muestra las diferentes categorías.

La función `getCategoryName()` que se invoca a continuación en el bucle de eventos principal se utiliza para buscar el nombre del `browsenode` en el que nos encontramos, para representar un título en la pantalla como por ejemplo Current Best Sellers in Business & Investing. Como hemos mencionado anteriormente, lo busca en `categoryList`.

La diversión empieza al llegar a la función `showBrowseNode()`, que mostramos en el listado 31.6.

Listado 31.6. Función showBrowseNode() de bookdisplayfunctions.php. Una lista de categorías

```
// Para un determinado browsenode, muestre una página de productos
function showBrowseNode($browseNode, $page, $mode)
{
   $ars = getARS('browse', array('browsenode'=>$browseNode,
                                 'page' => $page,
                                 'mode'=>$mode));
   showSummary($ars->products(), $page, $ars->totalResults(),
               $mode, $browseNode);
}
```

Esta función realiza dos operaciones. En primer lugar, invoca la función `getARS()` de `cachefunctions.php`. Esta función obtiene y devuelve un objeto `AmazonResultSet` (como veremos más adelante). Tras ello, invoca la función `showSummary()` de `bookdisplayfunctions.php` para mostrar la información recuperada. La función `getARS()` es fundamental para controlar la aplicación. Si analizamos el resto de las acciones del código como las de ver detalles, imágenes y búsquedas, comprobará que todas vuelven a la misma.

Obtención de un AmazonResultSet

A continuación analizaremos la función `getARS()` con más detalle. Su código se recoge en el listado 31.7.

Listado 31.7. Función getARS() de cachefunctions.php. Conjunto de resultados de una consulta

```
// Obtenga un AmazonResultSet de la caché o de una consulta
// Si se trata de una consulta, añádala a la caché
function getARS($type, $parameters)
{
   $cache = cached($type, $parameters);
   if($cache)  // si se encuentra en la caché
   {
      return  $cache;
   }
   else
   {
      $ars = new AmazonResultSet;
      if($type == 'asin')
         $ars->ASINSearch(padASIN($parameters['asin']),
$parameters['mode']);
      if($type  ==  'browse')
                  $ars->browseNodeSearch($parameters['browsenode'],
         $parameters['page'], $parameters['mode']);
      if($type == 'search')
         $ars->keywordSearch($parameters['search'], $parameters['page'],
                           $parameters['mode']);
      cache($type, $parameters, $ars);
   }
   return $ars;
}
```

Está función se ha diseñado para controlar el proceso de obtención de datos de Amazon. Se puede realizar de dos formas: bien desde la caché o directamente desde Amazon. Como Amazon requiere que los programadores almacenen en caché los datos descargados, la función busca primero los datos en caché, como veremos más adelante.

Si todavía no hemos ejecutado dicha consulta, será necesario obtener los datos directamente de Amazon. Para ello creamos una instancia de la clase `AmazonResult Set` e invocamos el método correspondiente a la consulta concreta que queramos ejecutar. El tipo de consulta lo determina el parámetro `$type`. En el ejemplo de búsqueda de categoría, pasamos `browse` como valor de este parámetro (véase el listado 31.6). Si queremos realizar una consulta sobre un determinado libro, habrá que pasar el valor `asin`, y si queremos realizar una búsqueda de palabras clave, estableceremos el parámetro como `search`.

Cada uno de estos parámetros invoca un método diferente en la clase `AmazonResultSet`. La búsqueda de un artículo concreto invoca el método `ASINSearch()`, la búsqueda de categoría llama al método `browseNodeSearch()`

y la búsqueda de palabras clave, al método `keywordSeach()`. En el listado 31.8 se incluye el código completo de esta clase.

Listado 31.8. AmazonResultScript. Clase para procesar conexiones a Amazon

```
<?php
require_once('Product.php');

// puede cambiar entre XML/HTTP y SOAP por medio de esta constante definida en
// constants.php
if(METHOD=='SOAP')
{
   include_once('nusoap/nusoap.php');
}

// Esta clase almacena el resultado de consultas
// Suele ser 1 o 10 instancias de la clase Product
class AmazonResultSet
{
   var $_browseNode;
   var $_ASIN;
   var $_page;
   var $_mode;
   var $_url;
   var $_type;
   var $_totalResults;
   var $_currentProduct = null;
   var $_products = array(); // matriz de objetos Product
   var $_names = array();
// matriz de nombres de los nodos XML que hemos introducido.
// tratada como pila. Sólo se utiliza durante el análisis.

   function products()
   {
      return $this->_products;
   }

   function totalResults()
   {
      return $this->_totalResults;
   }

   function getProduct($i)
   {
      if(isset($this->_products[$i]))
         return $this->_products[$i] ;
      else
         return false;
   }

   // Realice una consulta para obtener una página completa de productos de un browsenode
   // Cambie entre XML/HTTP y SOAP en constants.php
   // Devuelve una matriz de Products
```

```
function browseNodeSearch($browseNode, $page, $mode)
{
   if(METHOD=='SOAP')
   {
      // la clase NuSOAP genera muchos avisos. Desactívelos.
      error_reporting(error_reporting() & ~E_NOTICE);
      $soapclient = new soapclient(
         'http://soap.amazon.com/schemas2/AmazonWebServices.wsdl',
         'wsdl');
      $soap_proxy = $soapclient->getProxy();
      $parameters['mode']=$mode;
      $parameters['page']=$page;
      $parameters['type']='heavy';
      $parameters['tag']=$this->_assocID;
      $parameters['devtag']=$this->_devTag;
      $parameters['sort']='+salesrank';
      $parameters['browse_node'] = $browseNode;

      // ejecute la consulta
      $result = $soap_proxy->BrowseNodeSearchRequest($parameters);
      if(isSOAPError($result))
         return false;
      $this->_totalResults = $result['TotalResults'];
      $counter = 0;
      foreach($result['Details'] as $product)
      {
         $this->_products[$counter] = new Product;
         $this->_products[$counter]->soap = $result['Details'][$counter];
         $counter++;
      }
      unset($soapclient);
      unset($soap_proxy);
   }
   else
   {
         // cree el URL e invoque parseXML para descargarlo y analizarlo
      $this->_type = 'browse';
      $this->_browseNode = $browseNode;
      $this->_page = $page;
      $this->_mode = $mode;
      $this->_url = 'http://xml.amazon.com/onca/xml2?t='.ASSOCIATEID
                  .'&dev-t='.DEVTAG.'&BrowseNodeSearch='
                  .$this->_browseNode.'&mode='.$this->_mode
                  .'&type=heavy&page='.$this->_page.'&sort=+salesrank&f=xml';
      $this->parseXML();
   }

   return $this->_products;
}

// Con un ASIN dado, obtenga el URL de la imagen de gran tamaño
// Devuelve una cadena
function getImageUrlLarge($ASIN, $mode)
{
   if( $this->_products[0]->imageURLLarge())
```

```
        return  $this->_products[0]->imageURLLarge();
    $this->ASINSearch($ASIN, $mode);
        return $this->_products[0]->imageURLLarge();
}

// Realice una consulta para obtener un producto con el ASIN especificado
// Cambie entre XML/HTTP y SOAP en constants.php
// Devuelve un objeto Products
function ASINSearch($ASIN, $mode = 'books')
{
    $this->_type = 'ASIN';
    $this->_ASIN=$ASIN;
    $this->_mode = $mode;
    $ASIN = padASIN($ASIN);

    if(METHOD=='SOAP')
    {
        error_reporting(E_ALL & ~E_NOTICE);
        $soapclient = new soapclient (
              'http://soap.amazon.com/schemas2/AmazonWebServices.wsdl',
              'wsdl') ;
        $soap_proxy = $soapclient->getProxy();
        $parameters['asin']=$ASIN;
        $parameters['mode']=$mode;
        $parameters['type']="heavy";
        $parameters['tag']=$this->_assocID;
        $parameters['devtag']=$this->_devTag;

        $result = $soap_proxy->AsinSearchRequest($parameters);
        if(isSOAPError($result))
        {
            print_r($result);
            return false;
        }
        $this->_products[0] = new Product;
        $this->_products[0]->soap = $result['Details'][0];
        $this->_totalResults=1;
        unset($soapclient);
        unset($soap_proxy);
    }
    else
    {
        // cree el URL e invoque parseXML para descargarlo y analizarlo
        $this->_url = 'http://xml.amazon.com/onca/xml2?t='.ASSOCIATEID
                      .'&dev-t='.DEVTAG.'&AsinSearch='
                      .$this->_ASIN
                      .'&type=heavy&f=xml';
        $this->parseXML();
    }
    return $this->_products[0];
}

// Ejecute una consulta para obtener una página completa de productos con
```

```
// una búsqueda de palabras clave
// Cambie entre XML/HTTP y SOAP en index.php
// Devuelve una matriz de Products
function keywordSearch($search, $page, $mode = 'books')
{
   if(METHOD=='SOAP')
   {
      error_reporting(E_ALL & ~E_NOTICE);
      $soapclient = new soapclient(
         'http://soap.amazon.com/schemas2/AmazonWebServices.wsdl','wsdl');
      $soap_proxy = $soapclient->getProxy();
      $parameters['mode']=$mode;
      $parameters['page']=$page;
      $parameters['type']="heavy";
      $parameters['tag']=$this->_assocID;
      $parameters['devtag']=$this->_devTag;
      $parameters['sort']='+salesrank';
      $parameters['keyword'] = $search;
      // ejecute la consulta
      $result = $soap_proxy->KeywordSearchRequest($parameters);

      if(isSOAPError($result) )
         return false;
      $counter = 0;
      foreach($result['Details'] as $product)
      {
         $this->_products[$counter] = new Product;
         $this->_products[$counter]->soap = $result['Details'][$counter];
         $counter++;
      }
      $this->_totalResults = $result['TotalResults'] ;
      unset($soapclient);
      unset($soap_proxy);
   }
   else
   {
      $this->_type = 'search';
      $this->_search=$search;
      $this->_page = $page;
      $search = urlencode($search);
      $this->_mode = $mode;
      $this->_url = 'http://xml.amazon.com/onca/xml2?t='.ASSOCIATEID
                    .'&dev-t='.DEVTAG.'&KeywordSearch='
                    .$search.'&mode='.$this->_mode
                    .'&type=heavy&page='
                    .$this->_page
                    .'&sort=+salesrank&f=xml';
      $this->parseXML();
   }
   return $this->_products;
}

// Analice el XML en el objeto/objetos Product
function parseXML()
```

```
{
   $xml_parser = xml_parser_create();
   xml_parser_set_option($xml_parser,XML_OPTION_SKIP_WHITE,1);
   xml_set_object($xml_parser, $this);
   xml_set_element_handler($xml_parser,
                           "startElementHandler",
                           "endElementHandler");
   xml_set_character_data_handler($xml_parser, 'cdataHandler');

   if (!($fp = fopen($this->_url, "r")))
   {
      die("could not open XML input");
   }
   while ($data = fread($fp, 4096))
   {
      if (!xml_parse($xml_parser, $data, feof($fp)))
      {
         die(sprintf( "XML error: %s at line %d",
                     xml_error_string(xml_get_error_code($xml_parser)),
                     xml_get_current_line_number($xml_parser)));
      }
   }
   xml_parser_free($xml_parser) ;
}

// función para capturar retrollamadas cuando el analizador XML llega al inicio
// de un nuevo elemento
function startElementHandler($parser, $name, $attributes)
{
   array_push($this->_names, $name);

   if($name=='DETAILS')
   {
      $this->_currentProduct = new Product();
   }
   if($name == 'BROWSENODE')
   {
      $this->_currentProduct->_currentBrowseName++;
   }
   if($name == 'CUSTOMERREVIEW')
   {
      $this->_currentProduct->_currentReview++;
   }
}

// función para capturar retrollamadas cuando el analizador XML tiene datos
// de un elemento
function cdataHandler($parser, $cdata)
{
   $this->_currentName = array_slice($this->_names, -1, 1);
   $this->_currentName = $this->_currentName[0] ;

   switch($this->_currentName)
   {
```

```
case 'TOTALRESULTS' :
   $this->_totalResults = $cdata;
break;

case 'DETAILS' :
break;

case 'AUTHOR' :
   $this->_currentProduct->authors[] = $cdata;
break;

case 'RATING' :
case 'SUMMARY' :
case 'COMMENT' :
   @$this->_currentProduct->
      customerReviews[$this->_currentProduct->_currentReview]
      [$this->_currentName] .= $cdata;
   // los campos que contengan retornos y símbolos & deben concatenarse
   // la concatenación genera un aviso si están activados,
   // de ahí el símbolo @
break;

case 'LISTID' :
   $this->_currentProduct->listIDs[] = $cdata;
break;

case 'BROWSENAME' :
   @$this->_currentProduct->browseNames[$this->_currentProduct->_
            currentBrowseName] .= $cdata;
   // los campos que contengan retornos y símbolos & deben concatenarse
   // la concatenación genera un aviso si están activados,
   // de ahí el símbolo @

break;

case 'PRODUCT' :
   $this->_currentProduct->similarProducts[] = $cdata;
break;

// hay determinadas claves cuyos secundarios procesamos
// de forma separada, por lo que puede ignorarlas
case  'CUSTOMERREVIEW' :
case  'AUTHORS' :
case  'BROWSELIST' :
case  'BROWSENODE' :
case  'LISTS' :
case  'REVIEWS' :
case  'SIMILARPRODUCTS' :
   //no haga nada
break;

default :
   @$this->_currentProduct->nodes[$this->_currentName] .= $cdata;
break;
```

```
        }
    }

    // función para obtener retrollamadas cuando el analizador XML llega al
    // final de un elemento
    function endElementHandler($parser, $name)
    {
        if($name=='DETAILS')
        {
            //ya no son necesarias
            unset($this->_currentProduct->_currentReview);
            unset($this->_currentProduct->_currentBrowseName);

            array_push($this->_products, $this->_currentProduct);
        }
            array_pop($this->_names);
    }
}
? >
```

Esta clase tan útil encapsula la interfaz a Amazon en un cuadro de color negro. Dentro de esta clase se puede realizar la conexión a Amazon por medio de XML sobre HTTP o con ayuda del método SOAP. El método que se utilice viene determinado por la constante global `METHOD` que definimos al principio.

Comenzaremos desde el ejemplo de búsqueda de categorías. Utilizamos la clase `AmazonResultSet` como se indica a continuación:

```
$ars = new AmazonResultSet;
$ars->browseNodeSearch( $parameters['browsenode'],
                        $parameters['page'],
                        $parameters['mode']);
```

La clase carece de constructor por lo que pasaremos directamente al método `browseNodeSearch()`. Le pasamos tres parámetros: el número `browsenode` que nos interesa (que se corresponde por ejemplo a Business & Investing o Computers & Internet), el número de página (que representa los registros que queremos recuperar) y el modo (que representa el tipo de productos que nos interesa). El código de este método se muestra parcialmente en el listado 31.9.

Listado 31.9. método browseNodeSearch(). Búsqueda de categorías

```
// Ejecute una consulta para obtener una página completa de productos de un browse node
// Cambie entre XML/HTTP y SOAP en constants.php
// Devuelve una matriz de productos
function browseNodeSearch($browseNode, $page, $mode)
{
    if(METHOD=='SOAP')
    {
        // la clase NuSOAP genera numerosos avisos. Desactívelos
        error_reporting(error_reporting() & ~E_NOTICE);
        $soapclient = new soapclient(
            'http://soap.amazon.com/schemas2/AmazonWebServices.wsdl',
```

```
            'wsdl');
        $soap_proxy = $soapclient->getProxy();
        $parameters['mode']=$mode;
        $parameters['page']=$page;
        $parameters['type']='heavy';
        $parameters['tag']=$this->_assocID;
        $parameters['devtag']=$this->_devTag;
        $parameters['sort']='+salesrank';
        $parameters['browse_node'] = $browseNode;

        // ejecute la consulta
        $result = $soap_proxy->BrowseNodeSearchRequest($parameters) ;
        if(isSOAPError($result))
            return false;
        $this->_totalResults = $result['TotalResults'];
        $counter = 0;
        foreach($result['Details'] as $product)
        {
            $this->_products[$counter] = new Product;
            $this->_products[$counter]->soap = $result['Details'][$counter];
            $counter++;
        }
        unset($soapclient);
        unset($soap_proxy);
    }
    else
    {
        // cree el URL e invoque parseXML para descargarlo y analizarlo
        $this->_type = 'browse';
        $this->_browseNode = $browseNode;
        $this->_page = $page;
        $this->_mode = $mode;
        $this->_url = 'http://xml.amazon.com/onca/xml2?t='.ASSOCIATEID
                    .'&dev-t='.DEVTAG.'&BrowseNodeSearch='
                    .$this->_browseNode.'&mode='.$this->_mode
                    .'&type=heavy&page='.$this->_page.'&sort=+salesrank&f=xml';
        $this->parseXML();
    }

    return $this->_products;
}
```

En función del valor de la constante `METHOD`, este método realizará la consulta a través de XML sobre HTTP o a través de SOAP. Veremos cada una de las opciones por separado.

Uso de XML sobre HTTP

En primer lugar definimos una serie de variables miembro:

- `type`. El tipo de búsqueda que se realiza. Buscamos libros dentro de un determinado `browsenode`, por lo que definimos el valor como `browse`.

- `browse`. El valor del `browsenode` concreto que hemos pasado como parámetro.
- `page`. El número de páginas que hemos pasado como parámetro.
- `mode`. El tipo de artículos que buscamos (por ejemplo `books`) que hemos pasado como parámetro.
- `url`. El URL en Amazon al que debemos conectarnos para realizar este tipo de búsqueda.

Los URL para los que realizamos nuestras conexiones HTTP para distintos tipos de búsquedas y parámetros se encuentran en el API de servicios Web de Amazon.com y el paquete del programador. Veamos con más detalle los parámetros `GET` que pasamos en este caso:

```
$this->_url = 'http://xml.amazon.com/onca/xml2?t='.ASSOCIATEID
                .'&dev-t='.DEVTAG.'&BrowseNodeSearch='
                .$this->_browseNode.'&mode='.$this->_mode
                .'&type=heavy&page='.$this->_page
                .'&sort=+salesrank&f=xml';
```

Los parámetros que tenemos que pasar a este URL son los siguientes:

- `t`. Su Id. de asociado.
- `dev-t`. Su identificador de programador.
- `BrowseNodeSearch`. El número de `browsenode` que quiere buscar.
- `mode`. `Books` u otro tipo de producto válido.
- `type`. `Heavy` o `lite`. El primero devuelve más información.
- `page`. Grupo de diez resultados.
- `sort`. El orden en que queremos que se devuelvan los resultados. Es un parámetro opcional. En este caso, lo hemos configurado como `+salesrank` porque queremos los resultados en orden de ventas.
- `f`. El formato. Siempre contiene el valor `'xml'`.

Entre los tipos válidos de orden se encuentran los siguientes:

- Artículos destacados: `+pmrank`
- Superventas: `+salesrank`
- Opinión media del cliente: `+reviewrank`
- Precio (de menos a más): `+pricerank`
- Precio (de más a menos): `+inverse-pricerank`
- Fecha de publicación: `+daterank`
- Alfabético (A-Z): `+titlerank`

- Alfabético (Z-A): `-titlerank`

Una vez configurados todos estos parámetros, invocamos

```
$this->parseXML();
```

para realizar la operación. El método `parseXML()` se describe en el listado 31.10.

Listado 31.10. Método parseXML(). Análisis del XML devuelto por una consulta

```
// Analice el XML en el objeto/objetos Product
function parseXML()
{
   $xml_parser = xml_parser_create();
   xml_parser_set_option($xml_parser,XML_OPTION_SKIP_WHITE,1);
   xml_set_object($xml_parser, $this);
   xml_set_element_handler( $xml_parser,
                            "startElementHandler",
                            "endElementHandler");
   xml_set_character_data_handler($xml_parser, 'cdataHandler');
   if (!($fp = fopen($this->_url, "r")))
   {
      die("could not open XML input");
   }
   while ($data = fread($fp, 4096))
   {
      if (!xml_parse($xml_parser, $data, feof($fp)))
      {
         die(sprintf( "XML error: %s at line %d",
                      xml_error_string(xml_get_error_code($xml_parser)),
                      xml_get_current_line_number($xml_parser)));
      }
   }
   xml_parser_free($xml_parser);
}
```

En este método utilizamos la biblioteca XML de PHP, que se basa en `expat`. Primero invocamos la función `xml_parser_create()`, que crea una instancia del analizador y devuelve un controlador al mismo, que tendremos que pasar a otras funciones. Tras ello, será necesario configurar determinados elementos del analizador antes de poder analizar el documento XML, como se indica a continuación:

```
xml_parser_set_option($xml_parser,XML_OPTION_SKIP_WHITE,1);
xml_set_object($xml_parser, $this);
xml_set_element_handler($xml_parser,
                        "startElementHandler",
                        "endElementHandler");
xml_set_character_data_handler($xml_parser, 'cdataHandler');
```

La primera línea indica al analizador que ignore los espacios en blanco (lo que nos ahorrará trabajo). Existen otras opciones que se pueden configurar con esta función, pero la única relevante es ésta.

Las tres siguientes líneas se utilizan para definir funciones de retrollamada. Tendremos que indicar al analizador lo que debe hacer cuando encuentre datos de elementos y caracteres en el documento XML.

La invocación de `xml_set_object()` indica al analizador que estas funciones se encuentran dentro de `$this`, el objeto `AmazonResultSet` actual (si utilizáramos funciones convencionales en lugar de métodos de clase, no necesitaríamos esta invocación de función).

La invocación de `xml_set_element_handler()` indica al analizador que siempre que encuentre el principio de un elemento, debe invocar el método `starElement Handler()` y que, cuando encuentre el final de un elemento, invoque el método `endElementHandler()`.

Son algunos de los métodos que hemos escrito dentro del objeto `AmazonResult Set`.

Se denominan *retrollamadas* y su firma debe tener un determinado formato.

El método `startElementHandler()` debe aceptar tres parámetros: una referencia al analizador, el nombre del elemento que se acaba de iniciar y una matriz de los atributos del elemento.

El método `endElementHandler()` debe aceptar dos parámetros: una referencia al analizador y el nombre del elemento que se finaliza.

La invocación de `xml_set_character_data_handler()` indica al analizador lo que debe hacer cuando encuentre datos de caracteres en el documento XML. Se producirá cuando el analizador intente procesar los contenidos del elemento. Debe invocar el método `cDataHandler()`, que también se encuentra en la clase `AmazonResultSet` y que necesita una determinada firma. En este caso, debe aceptar dos parámetros: una referencia al analizador y los datos de caracteres como cadena.

Más adelante veremos el contenido de estas retrollamadas.

Después de configurar todos estos parámetros, abrimos el archivo en el URL que acabamos de recibir y analizamos el XML que encontramos.

El analizador se desplaza por el documento e invoca las funciones de retrollamada según corresponda.

Por último, será necesario limpiarlo todo y eliminar el analizador:

```
xml_parser_free($xml_parser);
```

A continuación veremos qué sucede en dichas funciones de retrollamada. Las tres funciones se muestran en el listado 31.11.

Listado 31.11. Funciones de retrollamada

```
// función para capturar retrollamadas cuando el analizador XML llega al inicio de
// un nuevo elemento
function  startElementHandler($parser,  $name,  $attributes)
{
   array_push($this->_names, $name);

   if($name=='DETAILS')
```

```
    {
        $this->_currentProduct = new Product();
    }
    if($name == 'BROWSENODE')
    {
        $this->_currentProduct->_currentBrowseName++;
    }
    if($name == 'CUSTOMERREVIEW')
    {
        $this->_currentProduct->_currentReview++;
    }
}

// función para capturar retrollamadas cuando el analizador XML tiene datos de
// un elemento
function cdataHandler($parser, $cdata)
{
    $this->_currentName = array_slice($this->_names, -1, 1);
    $this->_currentName = $this->_currentName[0];

    switch($this->_currentName)
    {
        case 'TOTALRESULTS' :
            $this->_totalResults = $cdata;
        break;

        case 'DETAILS' :
        break;

        case 'AUTHOR' :
            $this->_currentProduct->authors[] = $cdata;
        break;

        case 'RATING' :
        case 'SUMMARY' :
        case 'COMMENT' :
            @$this->_currentProduct->customerReviews[$this->_currentProduct->_
                    currentReview][$this->_currentName] .= $cdata;
            // los campos que contengan retornos o símbolos & deben concatenarse
            // la concatenación genera un aviso si están activados,
            // de ahí el símbolo @
        break;

        case 'LISTID' :
            $this->_currentProduct->listIDs[] = $cdata;
        break;

        case 'BROWSENAME' :
            @$this->_currentProduct->
                browseNames[$this->_currentProduct->_currentBrowseName] .= $cdata;
            // los campos que contengan retornos o símbolos & deben concatenarse
            // la concatenación genera un aviso si están activados,
            // de ahí el símbolo @
        break;

        case 'PRODUCT' :
```

```
            $this->_currentProduct->similarProducts[] = $cdata;
        break;

        // hay determinadas claves cuyos secundarios procesamos
        // de forma separada, por lo que puede ignorarlas
        case 'CUSTOMERREVIEW' :
        case 'AUTHORS' :
        case 'BROWSELIST' :
        case 'BROWSENODE' :
        case 'LISTS' :
        case 'REVIEWS' :
        case 'SIMILARPRODUCTS' :
        // no haga nada
        break;

        default :
            @$this->_currentProduct->nodes[$this->_currentName] .= $cdata;
        break;
    }
}

// función para obtener retrollamadas cuando el analizador XML alcanza el final
// de un elemento
function endElementHandler($parser, $name)
{
    if($name=='DETAILS')
    {
        //ya no son necesarias
        unset($this->_currentProduct->_currentReview);
        unset($this->_currentProduct->_currentBrowseName);

        array_push($this->_products, $this->_currentProduct);
    }
    array_pop($this->_names);
}
```

Al leer este código comprobará que estas funciones añaden a la matriz miembro `$_products` la información incluida en el documento XML.

Esta matriz está formada por instancias de la clase `Product`, que hemos creado específicamente para almacenar los detalles de un artículo.

El código de la clase `Product` se recoge en el listado 31.12.

Listado 31.12. Fragmento de product.php. Almacenamiento de datos sobre un determinado artículo

```
<?php
// Esta clase almacena datos sobre un solo producto de Amazon
// todos los atributos escalares se almacenan en los nodos de matriz
// los atributos del documento XML que requieren un tratamiento
// especial tienen su propia matriz

// si los datos provienen de XML/HTTP, la mayor parte se encontrará en $this->nodes
```

```
// si provienen de SOAP, _ALL_ los datos estarán en $this->soap

// El objetivo principal de esta clase es proporcionar una interfaz común a los datos
// de estas dos fuentes para que puedan compartir el código de representación
//class Product
{
   var $nodes = array();
   var $authors = array();
   var $listIDs = array();
   var $browseNames = array();
   var $customerReviews = array();
   var $similarProducts = array();
   var $_currentReview = -1;
   var $_currentBrowseName = -1;
   var $soap; // array returned by SOAP calls


   // la mayoría de los métodos de esta clase son similares
   // devuelva la variable XML o la variable SOAP
   // una sola instancia de Product tendrá una o la otra
   function similarProductCount()
   {
      if($this->soap)
         return count($this->soap['SimilarProducts']);
      else
         return  count($this->similarProducts);
   }
```

Esta clase está formada enteramente por funciones de acceso (get y set), por lo que no la describiremos.

Esta clase existe para que podamos almacenar datos de dos fuentes diferentes y acceder a los mismos a través de una única interfaz.

Esto significa que podemos tomar el resultado predeterminado de NuSOAP y un resultado sencillo de analizar de XML, y almacenarlos según los obtenemos.

También significa que no es necesario que nuestras funciones de representación, caché y manipulación sepan en qué forma se almacenan los datos o de dónde provienen.

Después de realizar todo este proceso para recuperar los datos, devolvemos al control a la función `getARS()` y de aquí a `showBrowseNode()`.

El siguiente paso es

```
showSummary( $ars->products(), $page,
             $ars->totalResults(), $mode,
             $browseNode);
```

La función `showSummary()` simplemente muestra los datos de `AmazonResult Set`, como vimos en la figura 31.1, razón por la que no incluimos esta función.

Uso de SOAP

Volvamos a la versión SOAP de `browseNodeSearch()`.

A continuación repetimos esta parte del código:

```
error_reporting(E_ALL & ~E_NOTICE);
$soapclient = new soapclient(
      'http://soap.amazon.com/schemas2/AmazonWebServices.wsdl','wsdl');
$soap_proxy = $soapclient->getProxy();
$parameters['mode']=$mode;
$parameters['page']=$page;
$parameters['type']='heavy';
$parameters['tag']=$this->_assocID;
$parameters['devtag']=$this->_devTag;
$parameters['sort']='+salesrank';
$parameters['browse_node'] = $browseNode;

$result = $soap_proxy->BrowseNodeSearchRequest($parameters);
if(isSOAPError($result))
return false;
$this->_totalResults = $result['TotalResults'];
$counter = 0;
foreach($result['Details'] as $product)
{
$this->_products[$counter] = new Product;
$this->_products[$counter]->soap = $result['Details'][$counter];
$counter++;
}
unset($soapclient);
unset($soap_proxy) ;
```

No hay funciones adicionales que analizar; el cliente SOAP se encarga de todo por nosotros.

Puede que no tenga control sobre el parámetro de informe de errores en su servidor. Si la configuración indica que se informe de los anuncios al navegador, la biblioteca NuSOAP los producirá, por lo que lo hemos desactivado por el momento.

En primer lugar creamos una instancia del cliente SOAP:

```
$soapclient = new soapclient(
              'http://soap.amazon.com/schemas2/AmazonWebServices.wsdl',
              'wsdl');
```

Proporcionamos al cliente dos parámetros. El primero es la descripción WSDL del servicio y el segundo parámetro indica al cliente SOAP que se trata de un URL WSDL. También bastaría con proporcionar un parámetro: el punto final del servicio, que es el URL directo del servidor SOAP.

Hemos optado por hacerlo de esta forma por una buena razón, como se indica en la siguiente línea de código:

```
$soap_proxy = $soapclient->getProxy();
```

Esta línea crea una clase en función de la información del documento WSDL. Esta clase, el proxy WSDL, dispondrá de métodos que se corresponden a los métodos del servicio Web, lo que facilita considerablemente las cosas. De esta forma podemos interactuar con el servicio Web como si se tratara de una clase PHP local.

A continuación, definimos una matriz de los parámetros que se deben pasar a la consulta `browsenode`:

```
$parameters['mode']=$mode;
$parameters['page']=$page;
$parameters['type']='heavy';
$parameters['tag']=$this->_assocID;
$parameters['devtag']=$this->_devTag;
$parameters['sort']='+salesrank';
$parameters['browse_node']  =  $browseNode;
```

Por medio de esta clase proxy, invocamos los métodos del servicio Web y pasamos la matriz de parámetros:

```
$result  =  $soap_proxy->BrowseNodeSearchRequest($parameters);
```

Los datos almacenados en `$result` forman una matriz que podemos almacenar directamente en nuestra matriz `_products` en la clase `AmazonResultSet`.

Ésta es la versión SOAP. Como puede comprobar, es mucho más breve que la versión XML y, por lo tanto, más aconsejable. No obstante, esperamos que haya aprendido algo sobre el análisis de XML.

Almacenamiento de datos en caché

Volvamos a la función `getARS()` para analizar el almacenamiento en caché. Como recordará, la función tiene este aspecto:

```
function getARS ($type, $parameters)
{
   $cache = cached($type, $parameters);
   if($cache)  // si se encuentra en caché
   {
      return  $cache;
   }
   else
   {
      $ars = new AmazonResultSet;
      if($type  ==  'asin')
         $ars->ASINSearch(padASIN($parameters['asin']), $parameters['mode']);
      if($type == 'browse')
         $ars->browseNodeSearch( $parameters['browsenode'],
                                 $parameters['page'], $parameters['mode']);
      if($type == 'search')
         $ars->keywordSearch($parameters['search'], $parameters['page'],
                  $parameters['mode']);
      cache($type, $parameters, $ars);
   }
   return $ars;
}
```

Todo el almacenamiento en caché SOAP o XML de la aplicación se realiza a través de esta función. Disponemos de otra para almacenar imágenes en caché. En

primer lugar, invocamos la función `cached()` para comprobar si el `AmazonResultSet` ya está en caché. Si lo está, devolvemos los datos en lugar de realizar una nueva solicitud a Amazon:

```
$cache = cached($type, $parameters);
if($cache)   // si no se encuentra en caché
{
   return  $cache;
}
```

En caso contrario, al obtener los datos de Amazon, los añadimos a la caché:

```
cache($type, $parameters, $ars);
```

Veamos estas dos funciones: `cached()` y `cache()`, que mostramos en el listado 31.13. Estas funciones implementan el almacenamiento en caché que Amazon requiere como parte de sus condiciones.

Listado 31.13. Funciones de almacenamiento en caché de cachefunctions.php

```
// compruebe si los datos de Amazon están en caché
// si están, devuélvalos
// en caso contrario, devuelva false
function  cached($type,  $parameters)
{
   if($type == 'browse')
      $filename = CACHE.'/browse.'.$parameters['browsenode'].'.
                  '.$parameters['page'].'.'.$parameters['mode'].'.dat';
   if($type == 'search')
      $filename = CACHE.'/search.'.$parameters['search'].'.
                  '.$parameters['page'].'.'.$parameters['mode'].'.dat';
   if($type == 'asin')
      $filename = CACHE.'/asin.'.$parameters['asin'].'
                  .'.$parameters['mode'].'.dat';

   // ¿faltan los datos en caché o tienen más de 1 hora?
   if(!file_exists($filename) ||
        ((mktime() - filemtime($filename)) > 60*60))
   {
      return  false;
   }
   $data = file($filename);
   $data = join($data, '');
   return unserialize($data);
}

// añada los datos de Amazon a la caché
function cache($type, $parameters, $data)
{
   if($type == 'browse')
      $filename = CACHE.'/browse.'.$parameters['browsenode'].'.'
                  .$parameters['page'].'.'.$parameters['mode'].'.dat';
   if($type == 'search')
      $filename  =  CACHE.'/search.'.$parameters['search'].'.'
```

```
            .$parameters['page'].'.'.$parameters['mode'].'.dat';
   if($type  ==  'asin')
      $filename = CACHE.'/asin.'.$parameters['asin'].'.'
.$parameters['mode'].'.dat';

   $data = serialize($data);

   $fp = fopen($filename, 'wb');
   if(!$fp||(fwrite($fp, $data)==-1) )
   {
      echo   ('<p>Error,  could  not  store  cache  file</p>');
   }
   fclose($fp);
}
```

Al analizar este código, comprobará que los archivos en caché se almacenan con un nombre de archivo formado por el tipo de la consulta seguido por los parámetros de la misma. La función `cache()` serializa los resultados para almacenarlos y la función `cached()` los deserializa.

Ésta última también sobrescribe todos los datos que tengan más de una hora, como se indica en las condiciones.

La función `serialize()` convierte datos de programa almacenados en una cadena que se puede almacenar. En este caso, creamos una representación de un objeto `AmazonResultSet` que se pueda almacenar. La invocación de `unserialize()` se encarga de lo contrario, es decir, convierte la versión almacenada en una estructura de datos en memoria.

Debe saber que al deserializar un objeto como éste, es necesario incluir la definición de la clase en el archivo para que la clase sea comprensible y utilizable después de cargarla otra vez.

En nuestra aplicación, la recuperación de un conjunto de resultados de la caché se realiza en una fracción de segundo. La ejecución directa de una nueva consulta puede tardar hasta 10 segundos.

Creación del carro de la compra

Una vez descritas todas estas opciones de consulta de Amazon, lo más obvio que podemos hacer es crear un carro de la compra. Como ya hemos descrito ampliamente esta operación en capítulos anteriores, no la analizaremos en este apartado.

Las funciones del carro de la compra se muestran en el listado 31.14.

Listado 31.14. cartfunctions.php. Implementación del carro de la compra

```
<?php
require_once('AmazonResultSet.php');

// Al utilizar la función showSummary() del archivo bookdisplay.php se muestran
// los contenidos actuales del carro de la compra
function showCart($cart, $mode)
{
```

```
    // genere la matriz que se va a pasar
    $products = array();
    foreach($cart as $ASIN=>$product)
    {
       $ars = getARS('asin', array('asin'=>$ASIN, 'mode'=>$mode));
       if($ars)
          $products[]  =  $ars->getProduct(0);
    }
    // cree el formulario que se va a vincular al carro de la compra de Amazon.com
    echo '<form method="POST"
            action="http://www.amazon.com/o/dt/assoc/handle-buy-box">';
    foreach($cart as $ASIN=>$product)
    {
       $quantity = $cart[$ASIN]['quantity'];
       echo "<input type='hidden' name='asin.$ASIN' value='$quantity'>";
    }
    echo '<input type="hidden" name="tag-value" value="ASSOCIATEID">';
    echo '<input type="hidden" name="tag_value" value="ASSOCIATEID">';
    echo '<input type="image" src="images/checkout.gif"
                          name="submit.add-to-cart"
                          value="Buy From Amazon.com">';
    echo ' When you have finished shopping press checkout to add all the
          items in your Tahuayo cart to your Amazon cart and complete
          your purchase.<br />';
    echo '</form>';

    echo '<a href = "index.php?action=emptycart"><img
          src = "images/emptycart.gif" alt = "Empty Cart" border = 0></a>
          If you have finished with this cart, you can empty it of all items.
          <br />';
    echo '<h1>Cart Contents</h1>';
    showSummary($products, 1, count($products), $mode,  0, true);

}

// muestre el resumen del carro que siempre aparece en pantalla
// muestre únicamente los tres últimos artículos añadidos
function showSmallCart()
{
    global $HTTP_SESSION_VARS;

    echo '<table border = 1 cellpadding = 1 cellspacing = 0>';
    echo '<tr><td class = cartheading>Your Cart $'
.number_format(cartPrice(), 2).
      '</td></tr>';
    echo '<tr><td class = cart>'.cartContents().'</td></tr>';

    // formulario para vincular al carro de la compra de Amazon.com
    echo '<form method="POST"
              action="http://www.amazon.com/o/dt/assoc/handle-buy-box">';
    echo '<tr><td class = cartheading><a href =
                          "index.php?action=showcart"><img
                          src="images/details.gif" border=0></a>';
    foreach($HTTP_SESSION_VARS['cart'] as $ASIN=>$product)
    {
       $quantity = $HTTP_SESSION_VARS['cart'][$ASIN]['quantity'];
```

```
      echo "<input type='hidden' name='asin.$ASIN' value='$quantity'>";
   }
   echo '<input type="hidden" name="tag-value" value="ASSOCIATEID">';
   echo '<input type="hidden" name="tag_value" value="ASSOCIATEID">';
   echo  '<input  type="image"  src="images/checkout.gif"
               name="submit.add-to-cart" value="Buy From Amazon.com">';
   echo '</td></tr>';
   echo '</form>';

   echo  '</table>';
}

// muestre los tres últimos artículos añadidos al carro
function cartContents()
{
   global $HTTP_SESSION_VARS;

   $display = array_slice($HTTP_SESSION_VARS['cart'], -3, 3);
   // queremos que aparezcan en orden cronológico inverso
   $display = array_reverse($display, true);

   $result = '';
   $counter = 0;

   // si los nombres son extensos, redúzcalos
   foreach($display as $product)
   {
      if(strlen($product['name'])<=40)
         $result  .=  $product['name'].'<br  />';
      else
         $result .= substr($product['name'], 0, 37).'...<br />';
      $counter++;
   }

   // añada líneas en blanco si el carro está prácticamente vacío para conservar
   // su representación
   for(;$counter<3; $counter++)
   {
      $result .= '<br />';
   }
   return $result;
}

// calcule el precio total de los artículos del carro
function  cartPrice()
{
   global $HTTP_SESSION_VARS;
   $total = 0.0;

   foreach($HTTP_SESSION_VARS['cart'] as $product)
   {
   $price = str_replace('$', '', $product['price']);
      $total += $price*$product['quantity'];
   }

   return $total;
```

```
}

// añada un artículo al carro
// actualmente no se puede añadir más de uno por vez
function addToCart(&$cart, $ASIN, $mode)
{
   if(isset($cart[$ASIN] ))
   {
      $cart[$ASIN]['quantity'] +=1;
   }
   else
   {
      // compruebe que el ASIN es válido y busque el precio
      $ars = new AmazonResultSet;
      $product = $ars->ASINSearch($ASIN, $mode);

      if($product->valid())
         $cart[$ASIN] = array('price'=>$product->ourPrice(),
                              'name' => $product->productName(), 'quantity' => 1) ;
   }

}

// elimine el artículo del carro
function deleteFromCart(&$cart, $ASIN)
{
   unset ($cart[$ASIN]);
}

? >
```

No obstante, hay algunas diferencias con respecto a este carro de la compra, por ejemplo en la función `addToCart()`.

Al tratar de añadir un artículo al carro, podemos comprobar que es un ASIN válido y buscar el precio actual (o al menos el almacenado en caché).

La pregunta más interesante es cómo enviar los datos a Amazon cuando los clientes terminan.

Cómo salir a Amazon

Fíjese en la función `showCart()` del listado 31.14. Veamos la parte principal de la misma:

```
// cree el formulario para vincular a un carro de la compra de Amazon.com
echo '<form method="POST"
      action="http://www.amazon.com/o/dt/assoc/handle-buy-box">';
foreach($cart as $ASIN=>$product)
{
   $quantity = $cart[$ASIN]['quantity'];
   echo "<input type='hidden' name='asin.$ASIN' value='$quantity'>";
```

```
}
echo '<input type="hidden" name="tag-value" value="ASSOCIATEID">';
echo '<input type="hidden" name="tag_value" value="ASSOCIATEID">';
echo '<input type="image" src="images/checkout.gif"
            name="submit.add-to-cart" value="Buy From Amazon.com">';
echo ' When you have finished shopping press checkout to add all the items
     in your Tahuayo cart to your Amazon cart and complete your purchase.
     <br />';
echo '</form>';
```

El botón **Checkout** es un botón de formulario que conecta el carro a un carro de la compra de Amazon.

Enviamos diferentes ASIN, cantidades y nuestro Id. de asociado a través de un variable `POST`.

El resultado final que se obtiene al pulsar este botón se muestra en la figura 31.5, al principio del capítulo.

La página resultante tendrá un aspecto diferente si el usuario no tiene almacenada una cookie de Amazon en su equipo para que Amazon lo identifique, pero el resultado final será el mismo.

Una de las dificultades de esta interfaz es que sólo funciona en una dirección.

Podemos añadir artículos al carro de Amazon, pero no eliminarlos. Esto significa que los usuarios no pueden pasar de un sitio a otro sin terminar con artículos duplicados en sus carros.

Instalación del código del proyecto

Si quiere instalar el código del proyecto correspondiente a este capítulo, tendrá que realizar una serie de pasos poco habituales.

Una vez ubicado el código correctamente en su servidor, tendrá que hacer lo siguiente:

- Crear un directorio caché
- Definir los permisos en el directorio caché para que las secuencias de comandos puedan escribir en el mismo.
- Modificar `constants.php` para que indique la ubicación de la caché.
- Solicitar una etiqueta de programador de Amazon.
- Modificar `constants.php` para que incluya esta etiqueta y, opcionalmente, su Id. de asociado.
- Comprobar que NuSOAP está instalado. En nuestro caso se incluye en el directorio Tahuayo, pero puede cambiarlo de sitio y modificar el código.
- Comprobar que ha compilado PHP con compatibilidad con XML.

Ampliación del proyecto

Puede ampliar este proyecto de formas muy divertidas:

- Puede ampliar los tipos de búsquedas disponibles a través de Tahuayo.
- Amazon también dispone de un servicio Web XSLT con el que puede probar.
- En los servicios Web de Amazon se incluyen enlaces a innovadoras aplicaciones de ejemplo, como es el caso de `http://associates.amazon.com/exec/panama/associates/ntg/browse/-/567634/`.

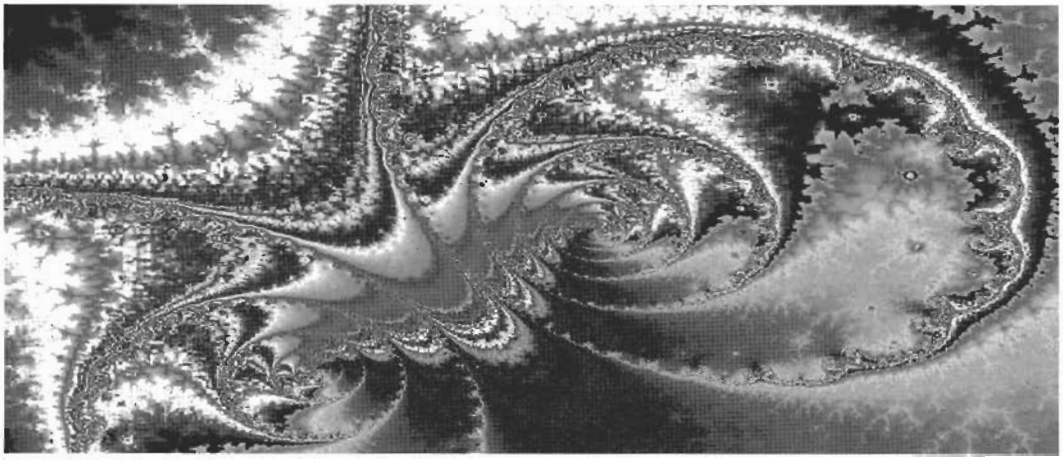
Un carro de la compra es el elemento más obvio que se puede crear con estos datos, pero no el único.

Lecturas adicionales

Existen millones de libros y recursos en línea sobre los temas de XML y servicios Web. Un punto de partida muy aconsejable es W3C, en la página del Grupo de trabajo XML `http://www.w3.org/XML/Core/` y en la página de servicios Web, `http://www.w3.org/2002/ws/`.

Parte VI

Apéndices



Apéndice A Instalación de PHP y MySQL

Existen versiones de Apache, PHP y MySQL disponibles para distintas combinaciones de sistemas operativos y servidores Web.

En este apéndice describiremos cómo puede configurar Apache, PHP y MySQL en diferentes plataformas de servidor.

Analizaremos las opciones más comunes disponibles para Unix y Windows 2000. Entre los aspectos que analizaremos destacamos los siguientes:

- Ejecución de PHP como intérprete CGI o como módulo
- Instalación de Apache, SSL, PHP y MySQL en Unix
- Instalación de Apache, SSL, PHP y MySQL en Windows
- Comprobación del funcionamiento: `phpinfo()`
- Cómo añadir PHP y MySQL a Microsoft Internet Information Server
- Cómo añadir PHP y MySQL a Microsoft Personal Web Server
- Instalación de PEAR
- Otras posibles configuraciones

El objetivo de este apéndice es ofrecerle una guía de instalación de un servidor Web que le permita alojar varios sitios Web. Algunos sitios, como en los ejemplos que hemos descrito, requieren SSL para soluciones de comercio electrónico. Y la mayoría se controlan por medio de secuencias de comandos para realizar conexiones a un servidor de base de datos (DB) y extraer y procesar datos.

Para muchos usuarios de PHP nunca será necesario instalar PHP en un equipo, razón por la que este material se incluye en un apéndice y no en el primer capítulo. La forma más sencilla de acceder a un servidor fiable con una conexión rápida a Internet y PHP instalado consiste en contratar una cuenta de uno de los miles de servicios de alojamiento existentes.

En función de la razón por la que instale PHP en un equipo, puede tomar decisiones diferentes. Si dispone de un equipo conectado permanentemente a la red que tiene pensado utilizar como servidor, el rendimiento le resultará de gran importancia. Si desea crear un servidor de desarrollo en el que generar y probar su código, el hecho de disponer de una configuración similar al servidor activo será la consideración más importante. Si pretende ejecutar ASP y PHP en el mismo equipo, tendrá que aplicar otras limitaciones.

Ejecución de PHP como intérprete CGI o como módulo

El intérprete de PHP se puede ejecutar bien como módulo o como binario CGI independiente. Por lo general, la versión modular se utiliza por motivos de rendimiento.

Sin embargo, se suele utilizar la versión CGI en servidores en los que la versión modular no está disponible o porque permite a los usuarios de Apache ejecutar distintas páginas en las que se ha activado PHP bajo diferentes Id. de usuario.

También puede optar por utilizar la versión CGI porque el instalador de Windows, de fácil manejo, la utiliza.

En este apéndice nos centraremos principalmente en la opción modular como método de ejecución de PHP en un entorno Unix y el método CGI con sistemas bajo Windows.

Instalación de Apache, PHP y MySQL en Unix

En función de sus necesidades y de su nivel de experiencia con sistemas Unix, puede optar por realizar una instalación binaria o por compilar los programas directamente desde el código fuente. Ambos enfoques tienen sus ventajas.

Para un experto, una instalación binaria sólo le llevará varios minutos y a un principiante no mucho más, pero resultará en un sistema que se encuentra una o dos versiones por detrás de las actuales y que estará configurado con las opciones definidas por alguien más.

Una instalación de código fuente tarda varias horas en descargarse, instalarse y configurarse, y la operación resulta un tanto intimidatoria las primeras veces. Sin embargo, le permite tener un control total. Seleccione lo que quiere instalar, qué versiones utilizar y qué directivas de configuración definir.

Instalación binaria

La mayor parte de las distribuciones de Linux incluyen un servidor Web Apache preconfigurado que incorpora PHP. Los detalles de lo que ofrece dependen de la distribución y la versión seleccionadas. Nos detendremos brevemente en la versión 8 de la distribución de Red Hat.

Un inconveniente de las instalaciones binarias es que raramente se obtiene la última versión del programa. Al cierre de la edición de este libro, Red Hat 8.0 incluía paquetes para instalar Apache 2.0.40, PHP 4.2.2 y MySQL 3.23.52, productos ligeramente desfasados y que carecen de los parches que se incluyen en las distribuciones de código fuente. En función de la importancia que tengan las últimas versiones de los parches, puede que en su caso no sea importante.

Un problema potencial mayor viene dado por la necesidad de utilizar un servidor http Apache 2.0, lo que sin duda tiene sus ventajas pero, en función de lo que se quiera arriesgar, conviene esperar a que se modifiquen las bibliotecas de extensiones de PHP y a que sean compatibles con subprocesos.

El mayor problema es que no se pueden seleccionar las opciones compiladas en los programas. Lo peor para los lectores de este libro es MySQL.

Probablemente consiga una configuración operativa de Linux/Apache/PHP con Red Hat 8.0 si utiliza los paquetes proporcionados. Basta con seguir algunos pasos a través de la GUI para completar el proceso. El problema con este enfoque es que el binario PHP que proporciona Red Hat se ha compilado sin compatibilidad con MySQL.

Un enfoque válido consistiría en instalar los RPM y, tras ello, instalar únicamente las partes necesarias para reparar el sistema. En el caso de Red Hat 8, implica volver a compilar únicamente PHP. Si quiere intentarlo, sigua únicamente las instrucciones que proporcionamos para una instalación completa desde el código fuente.

La solución más flexible y fiable consiste en compilar todos los programas que necesite desde sus códigos fuente. Lleva más tiempo que instalar los RPM, por lo que puede optar por utilizar los RPM u otros paquetes binarios disponibles. Incluso si los archivos binarios no están disponibles en fuentes oficiales con la configuración que necesita, sin duda encontrará otros no oficiales con ayuda de un motor de búsqueda.

Instalación desde código fuente

A continuación instalaremos Apache, PHP y MySQL en un entorno Unix. En primer lugar, será necesario decidir qué módulos adicionales cargaremos. Como en algunos de los ejemplos descritos en el libro se utiliza un servidor seguro para transacciones Web, instalaremos un servidor en el que se haya activado SSL.

Nuestra configuración PHP coincidirá aproximadamente con la configuración predeterminada pero también veremos cómo activar las dos siguientes bibliotecas en PHP:

- cURL
- PDFlib

Son dos de las muchas bibliotecas disponibles para PHP. Las incluimos para que se haga una idea de cómo se habilitan bibliotecas adicionales en PHP. La compilación de la mayoría de los programas de Unix es similar.

Habitualmente es necesario compilar PHP después de instalar una nueva biblioteca, por lo que si sabe lo que a va necesitar con antelación, instale en su equipo todas las bibliotecas necesarias y, tras ello, compile el módulo PHP.

Realizaremos la instalación en un servidor Linux Red Hat 8.0, pero es lo suficientemente genérica como para aplicarla a otros servidores Unix.

En primer lugar recopilamos los archivos necesarios para nuestra instalación, para lo que necesitaremos los siguientes programas:

- Apache (`http://www.apache.org/`). El servidor Web
- OpenSSL (`http://www.openssl.org/`). Conjunto de herramientas de código abierto que implementa SSL
- Mod_SSL (`http://www.modssl.org/`). Proporciona una interfaz modular de Apache a OpenSSL
- MySQL (`http://www.mysql.com/`). La base de datos relacional
- PHP (`http://www.php.net/`). Lenguaje de secuencia de comandos del lado del servidor
- `http://curl.haxx.se/`. Funciones de la biblioteca URL cliente
- `http://www.pdflib.com/pdflib/download/index.html`. Biblioteca para generar documentos PDF sobre la marcha

Suponemos que dispone de acceso como usuario raíz al servidor y que ha instalado las siguientes herramientas en su sistema:

- Perl (preferiblemente una versión reciente)
- gzip o gunzip
- gcc y GNU make

Cuando esté listo para iniciar el proceso de instalación, primero debe descargar todas las fuentes de archivos tar a un directorio temporal. Asegúrese de que dispone de suficiente espacio para ello. En nuestro caso, hemos seleccionado `/usr/src` como directorio temporal. Debería descargar estos archivos como usuario raíz para evitar problemas de permisos.

Instalación de MySQL

Vamos a realizar una instalación binaria de MySQL. De esta forma, los archivos se ubican automáticamente en distintos puntos. Los directorios que hemos seleccionado para el resto de los programas son los siguientes:

- `/usr/local/apache`
- `/usr/local/ssl`

Puede instalar la aplicación en directorios diferentes con tan sólo cambiar el prefijo antes de realizar la instalación.

Manos a la obra. En primer lugar, conviértase en usuario raíz por medio de `su`.

```
# su root
```

e introduzca la contraseña del usuario raíz. Cambie al directorio en el que haya almacenado los archivos fuente, por ejemplo

```
# cd /usr/src
```

MySQL recomienda actualmente que los usuarios descarguen un binario de MySQL en lugar de compilar desde cero.

Al parecer, existen problemas con determinadas versiones del compilador gcc de C.

Hemos descargado los siguientes RPM de mysql.com:

```
MySQL-3.23.53a-1.i386.rpm
MySQL-devel-3.23.53a-1.i386.rpm
MySQL-client-3.23.53a-1.i386.rpm
MySQL-shared-3.23.53a-1.i386.rpm
```

Si tiene pensado ejecutar el cliente y el servidor MySQL en este equipo y compilarlo para que sea compatible con MySQL en otros programas como PHP, los necesitará todos.

El siguiente comando instala todas las partes:

```
#   rpm -i MySQL*
```

Seguidamente puede crear las tablas `mysql`, iniciar el servidor y asignar al usuario raíz una contraseña. Asegúrese de reemplazar `new-password` en el siguiente comando con la opción que prefiera; en caso contrario, `new-password` será su contraseña raíz.

```
#  mysql_install_db
#  safe_mysqld &
#  mysqladmin -u root password 'new-password'
```

Puede comprobar que MySQL funciona si realiza algunas sencillas pruebas. El resultado será similar al que mostramos a continuación:

```
#   mysqlshow -p
Enter password:
+--------------+
|  Databases   |
+--------------+
|  mysql       |
+--------------+
```

Cuando instale MySQL, creará dos bases de datos de forma automática. Una es la tabla msyql, que controla permisos de usuarios, hosts y bases de datos en el servidor. La otra es una base de datos de prueba.

Con la siguiente línea de comandos podrá comprobar el sistema:

```
# mysql -u root -p
Enter password:
mysql> show databases;
+----------------+
| Database       |
+----------------+
| mysql          |
| test           |
+----------------+
2 rows in set (0.00 sec)
```

Escriba `quit` o `\q` para cerrar el cliente MySQL.

La configuración predeterminada de MySQL permite a cualquier usuario acceder al sistema sin necesidad de proporcionar un nombre de usuario y una contraseña, lo que evidentemente no es aconsejable.

La última parte obligatoria de MySQL consiste en eliminar el usuario anónimo. Para ello basta con abrir una línea de comandos y escribir lo siguiente:

```
# mysql -u root -p
mysql> use mysql
mysql> delete from user where User='';
mysql> quit
```

Tras ello, escriba

```
mysqladmin -u root -p reload
```

para aplicar los cambios.

Instalación de PDFlib

Si no quiere utilizar PDFlib para crear archivos PDF como vimos en un capítulo anterior, puede saltarse este apartado.

Sepa que PDFlib crea objetos compartidos para determinadas versiones de PHP. Si instala una nueva versión de PHP, puede que tenga que esperar a que se actualice PDFlib.

Para extraer los contenidos del archivo PDFlib, escriba:

```
# gunzip -c pdflib-4.0.3.tar.gz | tar xvf -
```

PDFlib dispone de numerosas opciones de configuración. Escriba

```
./configure -help
```

para obtener detalles. Como la vinculación con PHP está activada de forma predeterminada, el único indicador que asignaremos será un prefijo de ruta para especificar el directorio en el que queremos instalar los archivos.

Para instalar, siga estos pasos:

```
./configure -prefix=/usr/local/pdflib
```

```
# make
# make install
```

No utilizaremos PDFlib directamente, por lo que volveremos a este punto una vez se ejecute PHP.

Instalación de cURL

La biblioteca cURL está disponible como código fuente o como binarios precompilados para distintas plataformas. Hemos optado por la descarga de código fuente. Si no tiene pensado utilizar cURL (consulte un capítulo anterior), puede saltarse este apartado.

El primer paso consiste en descomprimir el archivo:

```
# bzip2 -d curl-7.10.2.tar.bz2
# tar xvf curl-7.10.2.tar
```

Tras ello, en el directorio fuente de cURL, configure e instale el software.

```
# cd curl-7.10.2
# ./configure
# make
# make install
```

Conviene mencionar que es necesario vincularlo a nuestra instalación de PHP, pero es todo lo necesario para instalar la biblioteca.

Instalación de PHP

Debería continuar como usuario raíz; en caso contrario, utilice `su` para convertirse en usuario raíz.

Antes de poder instalar PHP, es necesario preconfigurar Apache para que sepa dónde se encuentra todo. Volveremos a este punto más adelante cuando configuremos el servidor Apache.

Cambie al directorio en el que tenga los archivos fuente.

```
# cd /usr/src
# gunzip -c apache_1.3.27.tar.gz | tar xvf -
# cd apache_1.3.27
# ./configure -prefix=/usr/local/apache
```

Ya puede empezar a configurar PHP. Extraiga los archivos fuente y cámbielos a su directorio:

```
# cd /usr/src
# gunzip -c php-4.2.3.tar.gz | tar xvf -
# cd php-4.2.3
```

Como en el caso anterior, existen numerosas opciones para el comando `configure` de PHP. Utilice `./configure –help=short` para determinar lo que quiere añadir. En este caso, queremos añadir compatibilidad con MySQL, Apache, PDFlib y cURL.

Todo lo que viene a continuación se incluye en un solo comando. Podemos añadirlo todo en una línea o, como hemos hecho aquí, utilizar el carácter de continuación, la barra invertida (\), para poder escribir un comando en varias líneas para mejorar la legibilidad.

```
# ./configure -with-mysql=/usr \
              -with-xml -with-apache=../apache_1.3.27 \
              -with-curl=../curl \
              -enable-shared-pdflib
```

Tras ello, instalamos los binarios:

```
# make
# make install
```

Copiamos un archivo `ini` en el directorio `lib`:

```
# cp php.ini-dist /usr/local/lib/php.ini
```

o

```
# cp php.ini-recommended /usr/local/lib/php.ini
```

Las dos versiones de `php.ini` de los comandos sugeridos tienen un conjunto de opciones diferente. El primero, `php.ini-dist`,está ideado para equipos de desarrollo. Por ejemplo, configura `display_errors` con el valor `On`. De esta forma se facilita el desarrollo, pero no resulta adecuado para un equipo de producción. Cuando en el libro hemos hecho referencia al valor predeterminado de la configuración de `php.ini`, nos referíamos a su configuración en esta versión de `php.ini`. La segunda versión, `php.ini-recommended`, está indicada para equipos de producción.

Puede modificar el archivo `php.ini` para definir las opciones de PHP. Existen distintas opciones que puede configurar, pero conviene destacar algunas en concreto. Puede que tenga que establecer el valor de `sendmail_path` si quiere enviar correo electrónico por medio de secuencias de comandos. También puede optar por activar o desactivar `register_globals`.

A continuación, configuramos OpenSSL, que utilizaremos para crear certificados temporales y archivos CSR. `-prefix` especifica el directorio de instalación principal.

```
# gunzip -c openssl-0.9.6g.tar.gz | tar xvf -
# cd openssl-0.9.6g
# ./config -prefix=/usr/local/ssl
```

Tras ello, lo creamos, lo probamos y lo instalamos:

```
# make
# make test
# make install
```

Configuraremos el módulo mod_SSL y, tras ello, lo especificaremos como módulo que se pueda cargar con la configuración de Apache.

```
# cd /usr/src/
# gunzip -c mod_ssl-2.8.11-1.3.27.tar.gz |tar xvf -
# cd mod_ssl-2.8.11-1.3.27
# ./configure -with-apache=../apache_1.3.27
```

Fíjese en que podemos añadir más módulos de Apache el árbol fuente. La opción `--enable-shared=ssl` permite generar mdo_SSL como 'libssl.so' DSO. Consulte la documentación INSTALL y htdocs/manual/dso.hmtl del árbol fuente de Apache si necesita más información sobre compatibilidad DSO en Apache. Es recomendable que los ISP y los que mantengan paquetes utilicen la opción DSO para obtener una máxima flexibilidad con mod_SSL. Sin embargo, Apache no es compatible con DSO en todas las plataformas.

```
# cd ../apache_1.3.27
# SSL_BASE=../openssl-0.9.6g \
    ./configure \
    -enable-module=ssl \
    -activate-module=src/modules/php4/libphp4.a \
    -enable-module=php4 \
    -prefix=/usr/local/apache \
    -enable-shared=ssl
```

Por último, puede utilizar `make` con Apache y los certificados, e instalarlos.

```
# make
```

Si ha realizado correctamente todos los pasos, obtendrá un mensaje similar al siguiente:

```
+-------------------------------------------------------------------+
|Before you install the package you now should prepare the SSL      |
|certificate system by running the 'make certificate' command.      |
|                                 |                                 |
|For different situations the following variants are provided:      |
|                                                                   |
|% make certificate TYPE=dummy      (dummy self-signed Snake Oil cert)|
|% make certificate TYPE=test       (test cert signed by Snake Oil CA)|
|% make certificate TYPE=custom     (custom cert signed by own CA)  |
|% make certificate TYPE=existing   (existing cert)                 |
|       CRT=/path/to/your.crt  [KEY=/path/to/your.key]              |
|                                                                   |
|Use TYPE=dummy    when you're a  vendor package maintainer,        |
|the TYPE=test     when you're an admin but want to do tests only,  |
|the TYPE=custom   when you're an admin willing to run a real server|
|and TYPE=existing when you're an admin who upgrades a server.      |
|(The default is TYPE=test)                                         |
|                                                                   |
|Additionally add ALGO=RSA (default) or ALGO=DSA to select          |
|the signature algorithm used for the generated certificate.        |
|                                                                   |
|Use 'make certificate VIEW=1' to display the generated data.       |
|                                                                   |
|Thanks for using Apache & mod_ssl.      Ralf S. Engelschall        |
|                                        rse@engelschall.com        |
|                                        www.engelschall.com        |
+-------------------------------------------------------------------+
```

Ya puede crear un certificado personalizado. Esta opción requiere una ubicación, una compañía y algunos otros elementos. En cuanto a la información de contacto, conviene utilizar datos reales. En las preguntas restantes del proceso, basta con la respuesta predeterminada.

```
# make certificate TYPE=custom
```

Seguidamente, instale Apache:

```
# make install
```

Si todo sale correctamente, verá un mensaje como éste:

```
+--------------------------------------------------------+
| You now have successfully built and installed the      |
| Apache 1.3 HTTP server. To verify that Apache actually |
| works correctly you now should first check the         |
| (initially created or preserved) configuration files   |
|                                                        |
|     /usr/local/apache/conf/httpd.conf                  |
|                                                        |
| and then you should be able to immediately fire up     |
| Apache the first time by running:                      |
|                                                        |
|     /usr/local/apache/bin/apachectl start              |
|                                                        |
| Or when you want to run it with SSL enabled use:       |
|                                                        |
|     /usr/local/apache/bin/apachectl startssl           |
|                                                        |
| Thanks for using Apache.          The Apache Group     |
|                                    http://www.apache.org/ |
+--------------------------------------------------------+
```

Puede comprobar si Apache y PHP funcionan. Sin embargo, tendrá que modificar `httpd.conf` para añadir el tipo PHP a la configuración.

Fragmentos de código de httpd.conf

Fíjese en `httpd.conf` y añada las siguientes líneas. Si ha seguido las instrucciones anteriores, su archivo `httpd.conf` se encontrará en el directorio `/usr/local/apache/conf`.

El archivo incluye el `addtype` para PHP 4 con comentarios. Debería anular el comentario para que su aspecto sea el siguiente:

```
# Y para PHP 4.x, utilice:
#
AddType application/x-httpd-php .php
AddType application/x-httpd-php-source .phps
```

Ya puede iniciar el servidor Apache para comprobar si funciona. En primer lugar, iniciaremos el servidor sin compatibilidad con SSL para ver si aparece. Comproba-

mos dicha compatibilidad y, tras ello, detenemos el servidor y lo iniciamos con la compatibilidad SSL activada para comprobar que todo funciona correctamente.

`configtest` comprueba si la configuración se ha realizado correctamente:

```
#  cd  /usr/local/apache/bin
#  ./apachectl  configtest
Syntax  OK
#  ./apachectl  start
   ./apachectl  start:  httpd  started
```

Si ha funcionado correctamente, cuando nos conectemos al servidor con un navegador Web, el resultado será similar al de la figura A.1.

Nota

No se puede conectar al servidor con un nombre de dominio o por medio de la dirección IP del ordenador. Compruebe ambos casos para asegurarse de que todo funciona correctamente.

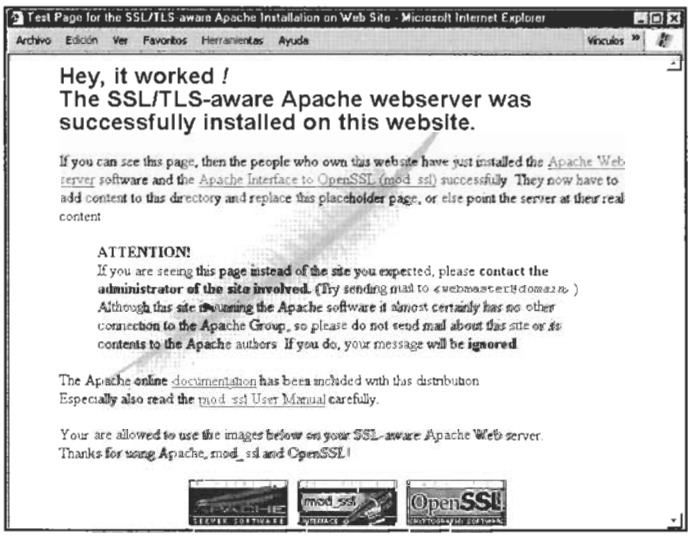


Figura A.1. Página de prueba predeterminada que proporciona Apache

¿Funciona la compatibilidad con PHP?

A continuación comprobaremos la compatibilidad con PHP. Cree un archivo con el nombre `test.php` en el que debe incluir el siguiente código.

El archivo debe estar en la ruta raíz del documento que, de forma predeterminada, será `/usr/local/apache/htdocs`, aunque depende del prefijo que haya seleccionado inicialmente.

Sin embargo, se puede cambiar en `httpd.conf`.

```
<?php phpinfo(); ?>
```

La pantalla de resultado será similar a la mostrada en la figura A.2.

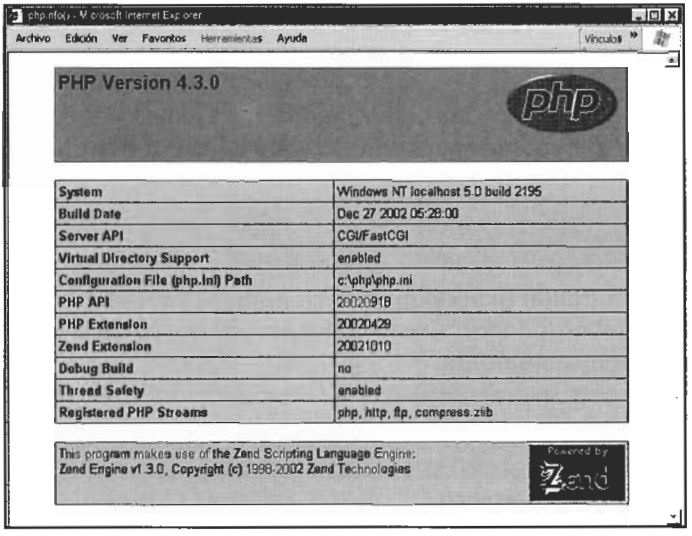


Figura A.2. La función phpinfo() proporciona información de configuración muy útil

¿Funciona SSL?

Ya podemos pasar a probar SSL. En primer lugar, detenga el servidor y reinícielo con la opción SSL activada:

```
#  /usr/local/apache/bin/apachectl  stop
#  /usr/local/apache/bin/apachectl  startssl
```

Compruebe si funciona.

Para ello debe conectarse al servidor con un navegador Web y seleccionar el protocolo `https` como se indica a continuación:

```
https://yourserver.yourdomain.com  or  http://
yoursever.yourdomain.com:443
```

Pruebe con la dirección IP de su servidor:

```
https://xxx.xxx.xxx.xxx  o  http://xxx.xxx.xxx.xxx:443
```

Si funciona, el servidor enviará el certificado al navegador para establecer una conexión segura.

Esto hará que el navegador le inste a aceptar el certificado ya firmado. Si se tratara de un certificado de una autoridad de certificación en la que el navegador ya confía,

no le preguntará nada. En nuestro caso, creamos y firmamos nuestros propios certificados. No hemos optado por adquirir uno directamente, ya que primero queríamos comprobar que todo funcionara correctamente.

Si utiliza Internet Explorer o Netscape, comprobará que aparece un símbolo en la barra de estado que indica que se ha establecido una conexión segura. El icono utilizado en Netscape se muestra en la figura A.3.

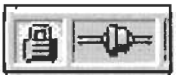


Figura A.3. Los navegadores Web muestran un icono para indicar que la página que estamos viendo proviene de una conexión SSL

Últimos pasos

Para poder ver el objeto compartido PDFlib así como otros módulos que haya instalado de esta forma, necesitará realizar algunos pasos adicionales.

Copie el archivo `libpdf_php` al directorio de extensiones PHP, que probablemente será `/usr/local/lib/php/extensions`. Añada la siguiente línea a su archivo `php.ini`:

```
extension = libpdf_php.so
```

Instalación de Apache, PHP y MySQL en Windows

En Windows, el proceso de instalación es ligeramente distinto ya que PHP se configura bien como una secuencia de comandos CGI (`php.exe`) o como módulo ISAPI (`php4isapi.dll`).

Sin embargo, Apache y MySQL se instalan de forma similar a la que hemos visto en Unix. Compruebe que ha instalado los últimos parches de servicios del sistema operativo en el equipo antes de comenzar la instalación de Windows.

En concreto, si utiliza Windows NT, compruebe que dispone del parche de servicios 3.

En primer lugar debería descargar los últimos archivos fuente a un directorio temporal con espacio suficiente.

En nuestra instalación hemos utilizado `c:\temp\download` como directorio temporal.

Si dispone de una conexión de red lenta, puede optar por utilizar las versiones del CD aunque es muy probable que se trate de versiones anteriores a la actual.

Los archivos de instalación que utilizaremos son:

```
mysql-3.23.52-win.zip
apache_1.3.27-win32-x86-no_src.msi
php-4.2.3-Win32.zip
```

Instalación de MySQL en Windows

Las siguientes instrucciones hacen referencia a Windows 2000. Si utiliza Windows NT o XP, los pasos son similares. Si utiliza Windows 95, 98 o ME, las partes directamente relacionadas con MySQL son idénticas, pero las partes que interactúan con el sistema operativo son distintas.

En primer lugar configuramos MySQL. Suponiendo que ha descargado todos los archivos necesarios, descomprima el archivo zip MySQL en el directorio temporal y ejecute el programa *Setup.exe*. El instalador es un asistente InstallShiedl convencional similar a muchos otros que haya utilizado.

Al seleccionar la instalación típica en el asistente, tendrá que indicar dónde quiere instalar MySQL. El directorio de instalación predeterminado será `C:\mysql`, aunque puede cambiarlo a un directorio distinto una vez instalado, pero con algunos pasos adicionales para garantizar que todo funciona.

Si cambia de posición MySQL y pretende ejecutar el ejecutable `mysqld`, tendrá que indicarle por medio de opciones de línea de comandos dónde se encuentra todo. Utilice `C:\mysql\bin\mysqld –help` para mostrar todas las opciones. Por ejemplo, si ha cambiado la distribución MySQL a `'D:\programs\mysql'` tendrá que iniciar `mysqld` con `'D:\programs\mysql\bin\mysqld –basedir D:\programs\mysql'`.

Si lo cambia y lo ejecuta como servicio de Windows, tendrá que crear un archivo `ini` con el nombre `my.ini` y ubicarlo en su directorio de Windows principal. Su archivo `ini` tendrá un contenido similar a éste:

```
[mysqld]
basedir=D:/programs/mysql/bin/
datadir=  D:/programs/mysql/data/
```

En la configuración de NT/2000/XP, el nombre del servidor MySQL es `mysqld-nt` y normalmente se instala como servicio. Un servicio es un programa que se ejecuta constantemente de fondo y cuya función es proporcionar servicios para otros programas.

Se ejecutan automáticamente al iniciar el equipo, lo que evita tener que iniciarlos manualmente cada vez.

Puede instalar el servidor MySQL como servicio si escribe lo siguiente en la línea de comandos de Windows:

```
cd  c:\mysql\bin
mysqld-nt  -install
```

obtendrá la siguiente respuesta

```
Service  successfully  installed.
```

Ya puede iniciar y detener el servicio MySQL desde la línea de comandos con:

```
NET  START  mysql
NET  STOP  mysql
```

Fíjese en que el nombre del ejecutable es `mysqld-nt`, pero el nombre del servicio simplemente es `mysql`. Si ejecuta NET START mysql verá el siguiente mensaje:

```
The MySql service is starting.
The MySql service was started successfully.
```

Una vez instalado el servicio, se puede detener, iniciar o configurar para que se inicie automáticamente por medio de la opción Servicios del Panel de control. Para abrir Servicios, seleccione Inicio>Configuración>Panel de control. Pulse dos veces sobre Herramientas administrativas y dos veces sobre Servicios.

La utilidad Servicios se muestra en la figura A.4. Si quiere configurar alguna de las opciones de MySQL, primero debe detener el servicio y, tras ello, especificarlas como parámetros de inicio antes de reiniciar el servicio MySQL. Este servicio se puede detener por medio de la utilidad Servicios o con los comandos `NET STOP MySQL` or `mysqladmin shutdown`.

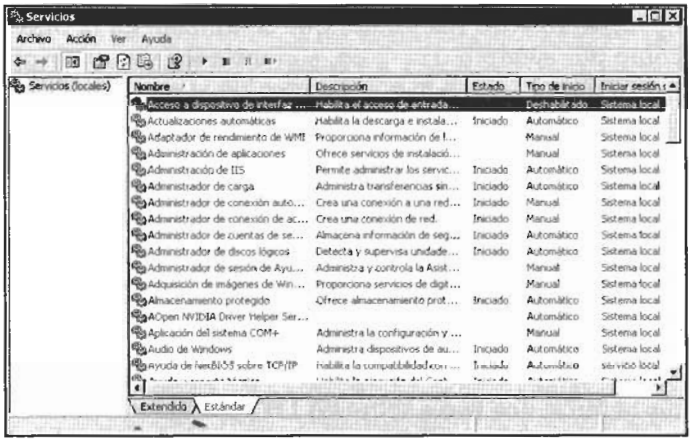


Figura A.4. La utilidad Servicios le permite configurar los servicios que se ejecutan en su equipo

Para probar si MySQL funciona o no, puede ejecutar los siguientes comandos:

```
C:\mysql\bin\mysqlshow
C:\mysql\bin\mysqlshow mysql
C:\mysql\bin\mysqladmin version status proc
```

El primer comando (`mysqlshow`) le ofrece una lista de bases de datos. Al instalar MySQL crea automáticamente dos bases de datos: `mysql` y `test`.

La base de datos `mysql` se utiliza para almacenar los permisos y accesos al servidor.

La base de datos `test` no es necesaria, pero constituye un lugar seguro para ejecutar comandos y comprobar si todo se ha configurado correctamente.

El segundo comando (`mysqlshow mysql`) le muestra las tablas de la base de datos `mysql`.

El tercer comando (`mysqladmin version status proc`) le informa sobre el estado actual de MySQL.

La configuración predeterminada no es la más idónea. Hay algunos cabos sueltos que conviene resolver:

- Definición de la ruta
- Eliminación del usuario anónimo
- Definición de la contraseña raíz

Definición de la ruta

MySQL incorpora multitud de utilidades de línea de comandos de distinta utilidad. No resulta nada sencillo obtenerlas a menos que el directorio binario de MySQL se encuentre en su `PATH`. El objetivo de esta variable de entorno es indicar a Windows dónde debe buscar los programas ejecutables.

Mucho de los comandos habituales que se utilizan en la línea de comandos de Windows, como por ejemplo `dir` y `cd`, son internos y se incorporan en `cmd.exe`. Otros, como `format` e `ipconfig`, disponen de sus propios ejecutables. No sería conveniente tener que escribir `C:\WINNT\system32\format` para aplicar formato a un disco ni tampoco `C:\mysql\bin\mysql` para ejecutar el monitor MySQL.

El directorio en el que se almacenan los ejecutables de los comandos básicos de Windows, como `format.exe`, se encuentra automáticamente en su `PATH`, por lo que basta con escribir `format`. Para disponer de la misma conveniencia función con las herramientas de línea de comandos mysql, será necesario añadirla.

Pulse **Inicio** y seleccione **Configuración>Panel de control**. Pulse dos veces sobre **Sistema** y seleccione la ficha **Opciones avanzadas**. Si pulsa el botón **Variables de entorno**, se abrirá un cuadro de diálogo en el que puede ver las variables de entorno de su sistema. Si pulsa dos veces sobre **PATH** podrá modificarla.

Añada un punto y coma al final de su ruta actual para separar la nueva entrada de la anterior y, tras ello, añada `c:\mysql\bin`. Al pulsar **Aceptar**, se almacenará en el registro del sistema. Cuando reinicie el equipo, podrá escribir mysql en lugar de `C:\mysql\bin\mysql`.

Eliminación del usuario anónimo

La configuración predeterminada de MySQL permite que cualquier usuario acceda al sistema sin necesidad de proporcionar un nombre de usuario o una contraseña, lo que no es aconsejable. Lo primero que queremos hacer es borrar el usuario anónimo. Para ello basta con abrir una línea de comandos e introducir las siguientes líneas:

```
c:\mysql\bin\mysql
use mysql
delete from user where User='';
quit
c:\mysql\bin\mysqladmin reload
```

De esta forma se elimina el usuario anónimo.

Definición de la contraseña raíz

La cuenta de superusuario, usuario raíz, todavía no tiene contraseña. Para configurarla, escriba estas líneas:

```
c:\mysql\bin\mysqladmin  -u  root  password  your_password
c:\mysql\bin\mysqladmin -u root -h your_host_name password your_password
```

Comprobará que ahora, las tareas que anteriormente no requerían un nombre de usuario y una contraseña, fallan sin esta información. Si intenta ejecutar

```
c:\mysql\bin\mysqladmin  reload
```

o

```
c:\mysql\bin\mysqladmin  shutdown
```

no lo conseguirá.

A partir de ahora, tendrá que utilizar el indicador `-u` y proporcionar un nombre de usuario, y el indicador `-p` para indicar a MySQL que dispone de una contraseña, como en este ejemplo:

```
c:\mysql\bin\mysqladmin  -u  root  -p  reload
```

Si utiliza este comando, MySQL solicitará la contraseña raíz que acaba de definir.

Si necesita más información, puede consultar el sitio Web de MySQL `http://www.mysql.com`.

Ya podemos instalar Apache en Windows. Manos a la obra.

Instalación de Apache en Windows

Apache 1.43 y posteriores se han diseñado para su ejecución en Windows NT, 2000 y XP. El instalador sólo funciona con la familia de procesadores x86, como los de Intel. Apache también se puede ejecutar en Windows 85, 98 y ME pero estos sistemas operativos no se han probado. En cualquier caso, es necesario instalar TCP/IP. Asegúrese de utilizar la biblioteca Winsock 2 si decide realizar la instalación en Windows 95 o 98.

Hemos descargado este archivo: `apache_1.3.27-win32-x86-no_src.msi`.

Contiene la versión actual (dentro de la jerarquía 1.3) para Windows, sin código fuente, empaquetado como archivo MSI. Los archivos MSI son el formato de paquete utilizado por el instalador de Windows.

A menos que tenga un problema difícil de solucionar o quiera contribuir a los esfuerzos de programación, no es aconsejable que compile personalmente el código fuente.

En este archivo se incluye el servidor Apache listo para su instalación.

Para iniciar la instalación basta con pulsar dos veces sobre el archivo descargado. El proceso de instalación le resultará familiar. Como puede apreciar en la figura A.5, es similar a muchos otros instaladores de Windows.

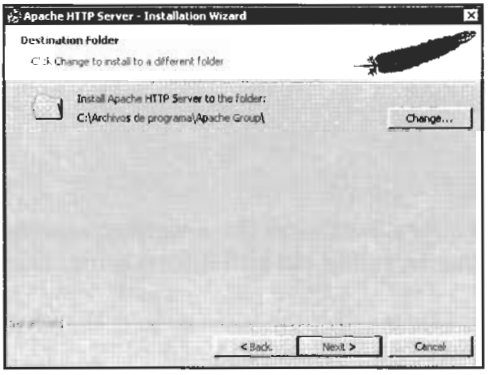


Figura A.5. El instalador de Apache es muy sencillo de utilizar

El programa de instalación le solicitará la siguiente información:

- El nombre de la red, el nombre del servidor y la dirección de correo electrónico del administrador. Si tiene pensado crear un servidor para su uso real, debería conocer las respuestas a estas preguntas. Si lo quiere crear para uso personal, esta información no es realmente importante.
- Indicar si quiere ejecutar Apache como servicio. Al igual que con MySQL, es más sencillo configurarlo de esta forma.
- El tipo de instalación. Recomendamos la opción **Complete** (Completa), pero puede elegir **Custom** (Personalizada) si quiere ignorar determinados componentes, como por ejemplo la documentación.
- El directorio en el que desea instalar Apache (el predeterminado es `C:\Archivos de programa\Apache Group\Apache`).

Una vez seleccionadas estas opciones, se instalará y se iniciará el servidor Apache.

Apache se comunica con el puerto 80 (a menos que haya cambiado las directivas Port, Listen o BindAddress en los archivos de configuración). Para conectarse al servidor y acceder a la página predeterminada, inicie un navegador e introduzca este URL: `http://localhost/`.

Accederá a una página de bienvenida similar a la de la figura A.1, con un enlace al manual de Apache. Si no sucede nada o si obtiene un error, revise el archivo `error.log` del directorio `logs`. Si su host no está conectado a Internet, puede que tenga que utilizar este otro URL: `http://127.0.0.1`. Es la dirección IP equivalente al equipo local. Si ha cambiado el número de puerto, tendrá que adjuntar *:número_puerto* al final del URL. Debe saber que Apache no puede compartir el mismo puerto con otra aplicación TCP/IP.

Puede iniciar y detener el servicio Apache desde el menú **Inicio**. Apache se adjunta a **Programas>Servidor HTTP Apache**. Debajo del título "Control Apache Server" encontrará opciones para iniciar, detener o reiniciar el servidor.

Una vez instalado, puede que sea necesario modificar los archivos de configuración del directorio `conf`. Veremos cómo se modifica el archivo de configuración `httpd.conf` cuando instalemos PHP.

Si necesita activar Apache con SSL en Windows, consulte la sección de preguntas más frecuentes de la dirección `http://tud.at/programm/apache-ssl-win32-howto.php3`, aunque no es nada sencilla.

Instalación de PHP en Windows

Existen varias formas de llevarlo a cabo. Puede instalarlo manualmente o utilizar el instalador InstallShield. En el sitio de PHP dispone de varias opciones de descarga.

Si utiliza el instalador, le resultará sorprendentemente sencillo y obtendrá una instalación funcional de PHP pero sin módulos. La instalación manual también es sencilla y le proporciona un completo conjunto de módulos. La elección es libre, pero le recomendamos el enfoque manual ya que necesitará muchos de los módulos para utilizar este libro. A continuación describiremos ambos tipos de instalación.

Uso de InstallShield

El instalador incorpora un archivo ejecutable. En primer lugar, detenga su servidor Web y, tras ello, pulse dos veces en el instalador para ejecutarlo. El instalador le preguntará si quiere utilizar la configuración estándar o la avanzada.

En la configuración estándar tendrá que responder a una serie de preguntas: en qué directorio se realiza la instalación, ubicación del servidor de coreo, dirección de correo electrónico y el tipo de servidor Web del que dispone.

En la configuración avanzada se le preguntará:

- Si quiere crear copias de seguridad durante la instalación
- Dónde quiere ubicar los archivos cargados
- Dónde quiere almacenar los datos de sesión
- Qué nivel de informe de errores prefiere
- Qué tipo de extensiones de archivo quiere que reconozca Apache

Instalación manual

En primer lugar, descomprima el archivo Zip en un directorio de su elección. La ubicación habitual suele ser c:\PHP, la que utilizaremos en nuestra descripción.

1. En el directorio principal verá un archivo con el nombre `php.exe` y otro denominado `php4ts.dll`. Son los archivos que necesita para ejecutar PHP como CGI. Si quiere ejecutarlo como módulo SAPI, tendrá que buscar en el directorio c:\PHP\sapi y copiar el correspondiente archivo DLL a C:\PHP. Si utiliza Apache, el archivo se denomina `php4apache.dll`.

Los módulos SAPI son más rápidos y más sencillos de proteger; la versión CGI le permite ejecutar PHP desde la línea de comandos. La decisión es personal.

2. Copie los contenidos de c:\PHP\dlls al directorio de sistema de Windows. En el caso de Windows NT o 2000 se encuentra en C:\winnt\system32 y en Windows XP, en C:\windows\system32.
3. Cambie el directorio c:\PHP\mibs a c:\usr\mibs.
4. Defina un archivo de configuración `php.ini`. PHP incorpora dos archivos preparados: `php.ini-dist` y `php.ini-recommended`. Le sugerimos que utilice este último. Copie este archivo y cambie su nombre por `php.ini`. Guárdelo en el directorio '%SISTEM-ROOT%'. Normalmente se encuentra en c:\winnt o c:\winnt40 en Windows NT o 2000 o en c:\windows en Windows XP.
5. Modifique su archivo `php.ini`. Incluye multitud de parámetros, la mayoría de los cuales puede ignorar. Los que debe cambiar son los siguientes:
   - Cambie la directiva `extension_dir` para que apunte al directorio en el que se encuentran sus DLL de extensión. En la instalación normal se corresponde a C:\PHP\Extensions. De esta forma, su archivo `php.ini` contendrá

     ```
     extension_dir = c:/php/extensions
     ```

   - Cambie la directiva doc_root para que apunte al directorio raíz desde el que actúa su servidor Web. Probablemente será

     ```
     doc_root = "c:/Program Files/Apache Group/Apache/htdocs"
     ```

     si utiliza Apache, o

     ```
     doc_root = "c:/Inetpub/wwwroot"
     ```

     si utiliza IIS.

   - También puede seleccionar las extensiones que quiere ejecutar. Le sugerimos que, en esta fase, simplemente ponga PHP en funcionamiento. Podrá añadir las extensiones cuando las necesite. Para añadir una extensión, consulte la lista incluida bajo "Extensiones de Windows". Verá numerosas líneas como

     ```
     ;extension=php_pdf.dll
     ```

     Para activar esta extensión, basta con eliminar el punto y coma que aparece al principio de la línea (y realizar lo contrario para desactivarla). Si posteriormente desea añadir más extensiones, tendrá que reiniciar su servidor Web después de cambiar php.ini para aplicar los cambios.

     Cierre y guarde su archivo `php.ini`.
6. Si utiliza NTFS, asegúrese de que el usuario con el que se ejecuta el servidor Web dispone de los permisos necesarios para leer su archivo `php.ini`.

Cómo añadir PHP a su configuración de Apache

Puede que sea necesario modificar uno de los archivos de configuración de Apache. Abra el archivo `httpd.conf` en su editor preferido. Habitualmente este archivo se encuentra en el directorio c:\Archivos de Programa\Apache Group\Apache\conf\. Busque las siguientes líneas:

```
ScriptAlias  /php/  "c:/php/"
AddType  application/x-httpd-php  .php
Action  application/x-httpd-php  "/php/php.exe  "
```

Si no las encuentra, añádalas al final del archivo, guárdelo y reinicie su servidor Apache.

Cómo añadir PHP y MySQL a Microsoft IIS y PWS

En este apartado veremos cómo añadir compatibilidad con PHP y MySQL a IIS con el módulo ISAPI (`php4isapi.dll`). También puede instalarlo como CGI, pero le recomendamos que utilice el módulo ISAPI ya que es más rápido. Se supone que habrá realizado los pasos descritos anteriormente en los apartados sobre instalación manual o con InstallShield. La principal diferencia es que la directiva de configuración `doc_root` será "c:/Inetpub/wwwroot".

Tras ello, tendrá que iniciar la Consola de administración de Microsoft (puede que en su sistema aparezca como Servicios de Internet Information Server). Si utiliza NT, se encuentra en el NT4 Option Pack. Si utiliza 2000 o XP, se encuentra en el **Panel de control**, bajo **Herramientas administrativas**.

Una vez abierto, en la parte izquierda verá una vista en árbol de los servicios. Pulse con el botón derecho del ratón sobre el servidor Web (normalmente con el nombre **Servidor Web predeterminado**) y seleccione **Propiedades**.

Se abrirá el cuadro de dialogo **Propiedades**, en el que se incluye gran cantidad de información, pero de la que sólo cambiaremos parte.

En la ficha **Directorio particular**, pulse el botón **Configuración**. En **Asignaciones para la aplicación**, pulse **Agregar** para añadir PHP. Le pedirá un ejecutable, por lo que debe indicar la ruta completa de la ubicación de `php4isapi.dll` (probablemente c:\php\sapi\php4isapi.dll si ha seguido las instrucciones). También verá un campo **Extensión** en el que debe introducir `.php`. Marque la casilla **Motor de secuencias de comandos** y pulse **Aceptar**.

Si quiere realizar autenticación HTTP (como hemos visto en el libro), debe utilizar la ficha **Filtros ISAPI**. Pulse **Agregar**. Se le pedirá un nombre de filtro, escriba PHP, y un ejecutable, la ruta completa al archivo `php4isapi.dll`, como en el caso anterior. Pulse **Aceptar**.

Cierre el cuadro de diálogo **Propiedades** por medio del botón **Aplicar**.

Detenga el servidor Web (o compruebe que está detenido) y reinícielo.

Puede hacerlo desde la Consola de administración/Servicios de Internet Information Server si pulsa con el botón derecho del ratón sobre el servidor Web y selecciona la opción **Detener**.

Puede reiniciarlo de la misma forma si pulsa **Iniciar**.

Pruebas

Inicie el servidor Web y compruebe si PHP funciona. Cree un archivo `test.php` y añádale las siguientes líneas:

```
<? phpinfo(); ?>
```

Asegúrese de que el archivo se encuentra en el directorio raíz de documentos (normalmente en c:\Program File\Apache Group\Apache\htdocs bajo Apache o c:\Inetpub\wwwroot bajo IIS) y ábralo en el navegador como se indica a continuación:

```
http://localhost/test.php
```

o

```
http://your-ip-number-here/test.php
```

Si accede a una página similar a la de la figura A.2, significa que PHP funciona.

Instalación de PEAR

Supuestamente, PHP 4.3 incorpora el instalador del paquete PEAR. Al cierre de la edición de este libro, en la versión de Linux se instalaba automáticamente, pero la versión de Windows no lo incluía.

Si utiliza Windows o una versión anterior de PHP (desde la 4.1 a la 4.2.3), tendrá que descargar el instalador del paquete, que puede encontrar en `http://go-pear.org`.

En este sitio hay una sola secuencia de comandos, que debe guardar como go-pear. Tendrá que ejecutarla a través de PHP desde la línea de comandos. En la línea de comandos, escriba

```
php  go-pear
```

La secuencia de comandos go-pear le preguntará dónde quiere instalar el paquete y las clases estándar de PEAR, y las descargará e instalará por nosotros.

Es necesario que cuente con una versión instalada del instalador del paquete PEAR. De esta forma, para instalar los paquetes, basta con que escriba:

```
pear  install  paquete
```

donde *paquete* se sustituye por el nombre del paquete que desee instalar.

Para ver la lista de los paquetes disponibles, escriba:

```
pear  list-all
```

Para ver los componentes instalados actualmente, pruebe con

```
pear  list
```

Para instalar el paquete de correo que utilizamos en capítulos anteriores, escriba lo siguiente:

```
pear install Mail_Mime
```

El paquete DB mencionado en uno de los primeros capítulos, se instala automáticamente, pero para comprobar que dispone de la versión más reciente, escriba:

```
pear list-upgrades
```

Si hay una nueva versión disponible, escriba

```
pear upgrade DB
```

Al cierre de la edición de este libro, resultaba complicado que el instalador del paquete de PEAR funcionara en Windows. Si en su caso no funciona el procedimiento anterior, pruebe a descargar directamente los paquetes PEAR en lugar de hacerlo a través del instalador.

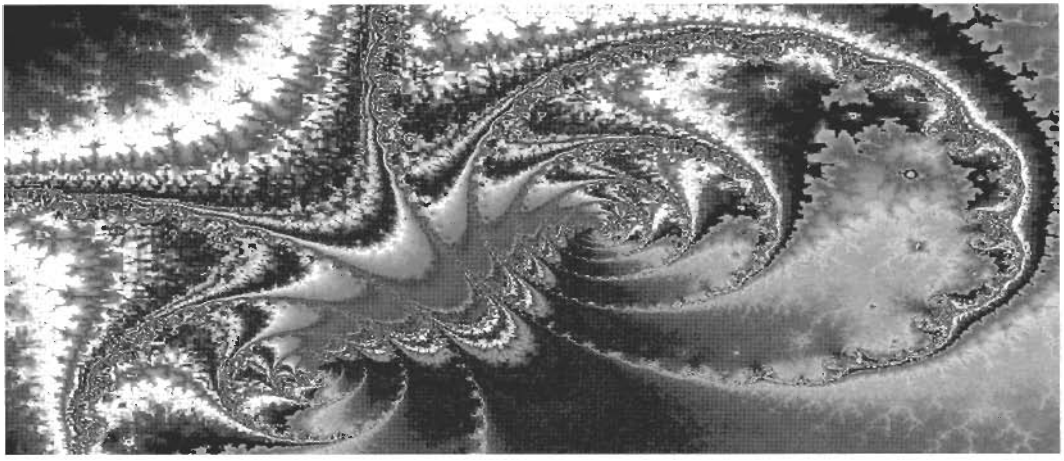
Para ello, puede visitar la página `http://pear.php.net/packages.php`.

Desde aquí, puede examinar los distintos paquetes disponibles. Por ejemplo, en este libro hemos utilizado Mail_Mime. Pulse en la página de este paquete y pulse sobre **Download Latest** para obtener una copia. Tendrá que descomprimir el archivo descargado y guardarlo en su ruta de inclusión.

También será necesario contar con un directorio c:\php\pear o similar. Si descarga manualmente los paquetes, es aconsejable que los guarde en el árbol de directorios de pear. PEAR dispone de una estructura estándar por lo que le sugerimos que almacene los componentes en la ubicación estándar, es decir, donde los guardaría el instalador. Por ejemplo, el paquete Mail_Mime corresponde a la sección Mail, por lo que en este ejemplo lo guardaríamos en el directorio c:\php\pear\Mail.

Configuraciones adicionales

Puede configurar PHP y MySQL con otros servidores Web como Omni, HTPPD y Netscape Enterprise Server. No los describiremos en este apéndice pero en los sitios Web de MySQL y PHP puede encontrar información sobre cómo se configuran, `http://www.php.net` y `http://www.mysql.com`, respectivamente.



Apéndice B
Recursos Web

En este apéndice enumeraremos algunos de los muchos recursos disponibles en la Web, que puede utilizar para buscar tutoriales, artículos y ejemplos de código PHP. Se trata de una pequeña selección. Evidentemente hay muchos más de los que podríamos incluir en un apéndice. Y otros muchos que aparecen diariamente debido al aumento del uso y la popularidad de PHP y MySQL entre los programadores Web.

Algunos de estos recursos estarán en idiomas diferentes, como alemán o francés. Le recomendamos que utilice un traductor como `http://www.syntransoft.com` para explorar el recurso Web en su idioma nativo.

Recursos PHP

PHP.Net (`http://www.php.net`). El sitio original de PHP, en el que puede descargar todas las fuentes de PHP así como una copia del manual.

ZEDN.Com (`http://www.zend.com`). Fuente del motor ZEND que dirige PHP 4. Un portal que contiene foros, artículos, tutoriales y una base de datos de clases y código de muestra que puede utilizar. No se la pierda.

PHPWizard.net (`http://www.phpwizard.net`). Fuente de numerosas y atractivas aplicaciones de PHP como phpMyAdmin; una excelente interfaz GUI para administrar servidores MySQL. También cuenta con tutoriales sobre PHP.

PHPBuilder.com (`http://www.phpbuilder.com`). El portal de tutoriales sobre PHP. En este sitio encontrará tutoriales sobre todo lo que se le ocurra. También

dispone de un foro y un tablón de anuncios en el que los usuarios pueden dejar mensajes.

DevShed.com (`http://devshed.com`). Portal que ofrece tutoriales sobre PHP, MySQL, Perl y otros lenguajes de desarrollo. Imprescindible para principiantes.

PX-PHP Code Exchange (`http://px.sklar.com`). Un buen punto de partida. Encontrará muchos ejemplos de secuencias de comandos y también funciones de gran utilidad.

PHP4 Resource (`http://www.php-resource.de`). Sitio con tutoriales, artículos y secuencias de comando. El "único" problema es que está en alemán. Le recomendamos que utilice un servicio de traducción para verlo.

WeberDev.com (`http://WeberDev.com`). Anteriormente conocido como la página de ejemplos de PHP de Berber, este sitio ha crecido y en la actualidad incluye tutoriales y código de ejemplo. El sitio está dirigido a usuarios de PHP y MySQL, y también cubre aspectos relacionados con bases de datos de seguridad y generales.

HotScripts.com (`http://www.hotscripts.com`). Excelente selección de secuencias de comandos categorizadas. Dispone de secuencias de comandos para diferentes lenguajes, como por ejemplo PHP, ASP y Perl. También cuenta con una amplia colección de secuencias de comandos de PHP. Se actualiza frecuentemente. Visita obligada si lo que busca son secuencias de comandos.

PHP Base Library (`http://phplib.sourceforge.net`). Sitio utilizado por programadores de proyectos de PHP a gran escala. Cuenta con una biblioteca con multitud de herramientas para enfoques alternativos de administración de sesión, así como con plantillas y abstracción de bases de datos.

PHP Center (`http://www.php-center.de`). Otro portal alemán utilizado para tutoriales, secuencias de comandos, consejos, trucos, publicidad, etc.

PHP Homepage (`http://www.php-homepage.de`). Otro sitio alemán sobre PHP con secuencias de comandos, artículos, noticias y mucho más. Dispone de una sección de referencia.

PHPIndex.com (`http://www.phpindex.com`). Portal francés sobre PHP con gran cantidad de contenidos relacionados con PHP. Cuenta con noticias, libros, sección de preguntas más frecuentes, artículos, etc.

WebMonkey.com (`http://webmonkey.com`). Portal con multitud de recursos Web, tutoriales reales, ejemplos de código, etc. El sitio cubre aspectos relacionados con el diseño, la programación, sistemas, multimedia y muchos otros.

PHP Club (`http://www.phpclub.net`). Ofrece una gran cantidad de recursos para principiantes de PHP. El sitio dispone de noticias, críticas de libros, ejemplos de código, foros, sección de preguntas más frecuentes y multitud de tutoriales.

Clases PHP (`http://www.phpclasses.upperdesign.com`). Sitio que distribuye clases gratuitas escritas en PHP. Es una visita obligatoria si tiene pensado desarrollar su código o si su proyecto está formado por clases. Dispone de funciones de búsqueda, para localizar con facilidad lo que necesite.

PHP Resource Index (`http://php.resourceindex.com`). Portal que ofrece secuencias de comandos, clases y documentación. Lo más atractivo de este sitio es que toda la información se agrupa en categorías, lo que nos permite ahorrar mucho tiempo.

PHP Developer (`http://www.phpdeveloper.org`). Otro portal de PHP que ofrece noticias, artículos y tutoriales sobre PHP.

Evil Walrus (`http://evilwalrus.com`). Atractivo portal en el que encontrará secuencias de comandos de PHP.

Oodie.com (`http://www.oodie.com`). Ofrece listas de proveedores de servicios de alojamiento PHP y de secuencias de comandos gratuitas.

e-gineer (`http://www.e-gineer.com`). Artículos, secuencias de comandos y una recopilación de preguntas y respuestas habituales.

Source Forge (`http://sourceforge.net`). Gran cantidad de recursos de código abierto. No sólo le permite buscar código que le resulte útil, sino que también ofrece acceso a CVS, listas de correo y equipos para programadores de código abierto.

Codewalkers (`http://codewalkers.com`). Contiene artículos, criticas de libros, tutoriales y el sorprendente Concurso PHP, en el que puede ganar por medio de sus nuevos conocimientos. Ofrece un concurso nuevo cada dos semanas.

PHP Developer's Network Unified Forums (`http://forums.devnetwork.net/index.php`). Análisis de todos los aspectos relacionados con PHP.

PHP Kitchen (`http://phpkitchen.com`). Artículos y noticias sobre PHP.

Postnuke (`http://www.postnuke.com`). Sistema de gestión de contenidos de PHP muy utilizado.

PHP Application Tools (`http://www.php-tools.de`). Conjunto de clases de PHP muy útiles.

Recursos específicos de MySQL y SQL

El sitio de MySQL (`http://www.mysql.com`). Sitio oficial de MySQL. Ofrece excelente documentación, asistencia e información. Visita obligatoria si utiliza MySQL.

SQL Course (`http://sqlcourse.com`). Cuenta con un tutorial sobre SQL con instrucciones muy sencillas de entender. Le permite practicar lo que ha aprendido en un intérprete SQL en línea. Puede encontrar una versión avanzada en `http://www.sqlcourse2.com`.

SearchDatabase.com (`http://searchdatabase.com`). Atractivo portal con multitud de información sobre DBS. Cuenta con excelentes tutoriales, consejos, sección de preguntas más frecuentes, críticas, etc. Imprescindible.

Recursos para Apache

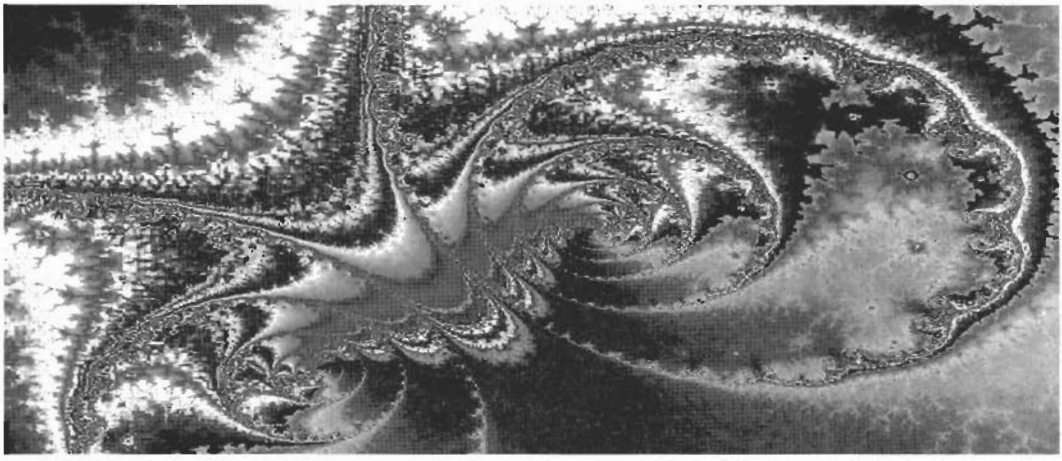
Apache Software (`http://www.apache.org`). Punto de partida si necesita descargar el código fuente o los archivos binarios. El sitio también cuenta con documentación en línea.

Apache Week (`http://www.apacheweek.com`). Revista en línea semanal que ofrece información fundamental para todo el que utilice un servidor Apache o para cualquiera que ejecute servicios Apache.

Apache Today (`http://www.apachetoday.com`). Noticias e información diaria sobre Apache. Los usuarios deben suscribirse para poder enviar preguntas.

Desarrollo Web

Philip and Alex's Guide to Web Publishing (`http://philip.greenspun.com/panda/`). Una inteligente e irreverente guía sobre ingeniería de software aplicada a la Web. Uno de los pocos libros sobre el tema escrito por un Samoyedo.



C
Contenido del CD-ROM

Contenido del CD-ROM

El CD-ROM que acompaña al libro contiene versiones completas de PHP, MySQL, Apache, varias bibliotecas gráficas, y los archivos que contienen los listados de código del libro.

Windows

En el Apéndice A se describe la configuración de Apache, MySQL y PHP en una plataforma de Windows. Hemos incluido versiones de Windows de dichos productos en el CD-ROM.

Apache 2.0.44 se encuentra en el directorio Software\Apache\Windows\Binary. Pulse dos veces sobre `apache_2.0.44-win32-x86-no_src.msi` para iniciar el instalador de Apache.

MySQL 4.0.12 se encuentra en el directorio `Software\MySQL\Windows\Binary`. Descomprima el archivo .zip y pulse dos veces sobre `SETUP.EXE` para iniciar el programa de instalación de MySQL.

PHP 4.3.1 se encuentra en el directorio `Software\PHP\Binary`. Puede utilizar el instalador simplificado (`php-4.3.1-installer.exe`) que le permite configurar PHP para utilizarlo con MySQL pero carece de opciones y módulos, o puede extraer el paquete completo (`php-4.3.1-Win32.zip`) y seguir las instrucciones detalladas en el fichero Readme del archivo Zip para configurar PHP en su sistema.

En el directorio `Libraries` encontrará diversas bibliotecas gráficas y una biblioteca PDF (PDFlib). Puede utilizar WinZip o cualquier otro descompresor para extraer el código en su ordenador.

Linux/Unix

Muchas distribuciones de Linux y algunas estaciones de trabajo de Unix están configuradas con Apache, MySQL y PHP. En el Apéndice A se describe la configuración de Apache, MySQL y PHP en una estación de trabajo Unix o Linux por si tiene que instalarlos. En el CD-ROM se incluye el código fuente para Apache, MySQL y PHP así como instaladores binarios para MySQL en Linux.

El código fuente para Apache 1.3.27 se encuentra en `Software/Apache/Unix/Source`. Si dispone de tar GNU, utilice `httpd-1.3.27.tar.gz`. En caso contrario, utilice `httpd-1.3.27.tar.Z`.

Los instaladores binarios para MySQL 3.23.54a para Linux se encuentran en `\MySQL\Unix\Binary`. Si su sistema utiliza el administrador RPM para instalar software, utilice `MySQL-3.23.54a-1.i386.rpm` para instalar la parte del servidor de MySQL y utilice `MySQL-client-3.23.54a-1.i386.rpm` para instalar la parte del cliente. Si su sistema Linux no utiliza el administrador RPM para instalar software, utilice `mysql-3.23.54a-pc-linux-i686.tar.gz` para instalar las partes de cliente y servidor de MySQL.

El código fuente para MySQL 3.23.54a para Unix se encuentra en `Software/MySQL/Unix/Source/mysql-3.23.54a.tar.gz`. Los usuarios de Solaris deben descargar tar GNU para extraer estos archivos debido a un error en la versión de Solaris del programa tar.

El código fuente para PHP 4.3.1 se encuentra en `Software/PHP/Unix/Source/php-4.3.1.tar`.

En el directorio `Libraries` encontrará diversas bibliotecas gráficas y una biblioteca PDF (PDFlib).

Léalo antes de abrirlo

Al abrir este paquete se compromete a cumplir el siguiente acuerdo:

Queda prohibida la reproducción o redistribución del CD-ROM en su totalidad. La reproducción y redistribución de los programas de software incluidos en el CD-ROM está sujeta a las condiciones impuestas por los propietarios de los derechos de dichos programas.

El instalador y el código del autor/autores son propiedad del editor y del autor/autores. Los programas y otros contenidos del CD-ROM son propiedad de los diferentes autores o propietarios de los derechos de los mismos. Algunos de los programas incluidos con este producto pueden estar sujetos a la licencia de Código abierto, lo que permite su redistribución; consulte la información de la licencia de cada uno de estos productos.

El resto de programas del CD-ROM se incluyen con permiso especial de sus correspondientes autores.

Índice alfabético

A

Acción Account-settings, 713
  administrador conectado, 703
  anular suscripción, 702
  añadir categoría, 591
  añadir libro, 591
  cambiar contraseña, 591
  crear correo, 703
  crear listas, 703
  detalles de pago, 590
  detalles del libro, 590
  enviar correo, 703
  información, 703
  information, 713
  log-in, 713
  log-out, 713
  menú de admin., 591
  mostrar mis listas, 703
  mostrar otras listas, 703
  mostrar todas las listas, 703
  new-account, 713
  procesar pago, 590
  salir, 590
  show-all-lists, 713
  show-archive, 713
  show-my-lists, 713
  show-other-lists, 713
  store-account, 713
  subscribe, 713
  unsubscribe, 713
  ver carro, 590
  ver correo, 703
Account-settings, 713
Account-setup, 670
Acrobat, 786
ACTION, 44
Activestate, 520
Activos digitales, 628
Adam, 8
Add_bm_form.php, 553
Add_bms.php, 553
Add_quoting, 775
AddSlashes, 140
Addtocart, 833

Admin.php, 592
Admin_fns.php, 593
Administrador, 233
Administrador conectado, 703
Adobe, 798
  distiller, 787
Adrian Close, 8
Alex´s Guide, 890
Alias, 260
Align, 43
ALL, 231, 304
Alnum, 151
Alpha, 151
ALTER, 230
Alter_priv, 294
AMANDA, 350
Amaya, 401
Amazon, 819
AmazonResultSet.php, 827
Ampersand, 63
Ampliación, 658
AND, 65, 656
Anular suscripción, 702
Añadir categoría, 591
  libro, 591
  marcador, 552
  nuevo artículo, 754
Apache, 871
Apachectl, 872
API, 552, 557, 788, 810
Apóstrofes, 415
Application Tools, 889
Árbol, 758
Archer, 8
Archivo add_bm_form.php, 553
  add_bms.php, 553
  admin.php, 592
  admin_fns.php, 593
  amazonResultSet.php, 827
  book_sc.sql, 593
  book_sc_fns.php, 593
  bookdisplayfunctions.php, 828
  bookmark.gif, 553
  bookmark_fns.php, 553
  bookmarks.sql, 552
  bottom, 827
  cachefunctions.php, 828
  cartfunctions.php, 828
  categoryfunctions.php, 828
  change_passwd.php, 553
  change_passwd_form.php, 553
  change_password.php, 592
  change_password_form.php, 592
  checkout.php, 592
  constants.php, 827
  create_database.sql, 639, 704, 755
  data_valid_fns, 553
  data_valid_fns.php, 593, 703, 755
  db_fns.php, 553, 593, 639, 755
  de datos, 596
  delete_bms.php, 553
  delete_book.php, 593
  delete_category.php, 593
  delete_story, 640
  discussion_fns.php, 755
  edit_book.php, 593
  edit_book_form.php, 593
  edit_category.php, 593
  edit_category_form.php, 593
  footer.php, 639
  forgot_form.php, 553
  forgot_passwd.php, 553
  header.php, 639
  headlines.php, 639
  include_fns.php, 639, 755
  index.php, 592
  insert_book.php, 593
  insert_book_form.php, 592
  insert_category.php, 592
  insert_category_form.php, 592
  keyword_add.php, 640
  keyword_delete.php, 640
  keywords.php, 640
  login.php, 552, 592, 640
  logo.gif, 639
  logout.php, 553, 592, 640
  member.php, 553
  mlm_fns.php, 703
  new_post.php, 754
  order_fns.php, 593

output_fns.php, 553, 593, 755
page.php, 639
pdflib.php, 789
phpCertification.pdf, 789
populate.sql, 593
process.php, 592
product.php, 828
publish.php, 640
publish_story.php, 640
purchase.php, 592
recommend.php, 553
register_form.php, 552
register_new.php, 553
resize_image.php, 639
rtf.php, 789
score.php, 789
search.php, 640
search_form.php, 639
select_fns.php, 639
show_book.php, 592
show_cart.php, 592
show_cat.php, 592
signature.png, 789
store_new_post.php, 755
stories.php, 640
story.php, 640
story_submit.php, 640
topbar.php, 827
treenode_class, 755
unpublish_story.php, 640
upload.php, 704
url_fns.php, 553
user_auth_fns.php, 553, 593, 639, 704
utilityfunctions.php, 828
view_post.php, 755
Arial, 794
Aritméticos, 60
Array, 114
Array_walk, 126
ASC, 262
ASCII, 390
ASIN, 821
ASINSearch, 836
Asort, 117
ASP, 41, 47
ASSOCIATED, 831
Atlanta, 7
Atributos vacíos, 219
Australia, 7
Autenticación, 490
Autenticarse, 628
AuthGroupFile, 366
Auto_append_file, 636
Auto_prepend_file, 636
AVG, 262
AWDuA3X3H.mc2, 342

B

Bandeja de entrada, 696
Bar_height, 474
Bar_spacing, 474
Barton, 8
Base de datos, 234
Base Library, 888
Bbox, 468
Berber, 888
BETWEEN, 255
Bgcolor, 79
Biblioteca de funciones, 730
BIGINT, 241
Binario, 390
BIT, 241
Bit a bit, 65
Black, 458
Blank, 151
BLOB, 244
Block, 412
BM, 579
Bm_URL, 554
Bob, 11
Bob´s Auto Parts, 42
Body, 135, 694
Boletín informativo, 742
Book Author, 626
Book Title, 626
Book_sc.sql, 593

Book_sc_fns.php, 593
Bookdisplayfunctions.php, 828
Bookmark.gif, 553
Bookmark_fns.php, 553
Bookmarks.sql, 552
Bool, 71, 102, 241
Boston Consulting Group, 7
Botón Account Setup, 676
  Add to Cart, 610
  Delete, 691
  New Message, 692
  Register, 558
  Save Changes, 605, 678
  Send Message, 692
  Submit Order, 45
Bottom, 827
Boutell.com, 479
BR, 136
Break, 668
BrowseName, 817
BrowseNode, 817, 833, 844
BSD, 390
Bucle do while, 81
  for, 80
  foreach, 80
  while, 78
BUGTRAQ, 406
Burn All Gifs, 456
Button_text, 464
Buttons, 669
Buzón de correo, 684

C

C, 41
C++, 48
CA, 345
Caché, 824
Cachefunctions.php, 828
Cadenas, 53
Cambiar contraseña, 552, 591
Canadá, 388
Card_month, 618
Card_name, 618
Card_number, 618
Card_type, 618
Card_year, 618
Carro, 604
Cartfunctions.php, 828
Case hide-headers, 687
  select-account, 683
  show-headers, 687
  view-mailbox, 683
  view-message, 687
Catálogo en línea, 588
Categorías, 598
  del sitio, 596
Category, 626
Categoryfunctions.php, 828
CC, 694
CD-ROM, 86
Cellpadding, 736
Cellspacing, 736
Center, 888
Centerx, 808
Centery, 808
Certificado PDF, 789
  rtf, 789
CGI, 89, 381
Change_passwd.php, 553
Change_passwd_form.php, 553
Change_password.php, 592
Change_password_form.php, 592
CHAR, 244, 412
Check_auth_user, 675
Checkout, 826
Checkout.php, 592
Children, 764
Chop, 136
Chown, 413
Chris Newman, 7
Chris@lightwood.net, 7
Ciencias Aplicadas, 7
  informáticas, 7
Cifrado, 360
Clase page, 198
Clases, 188
  php, 888

Class, 190
ClipPDF, 788
Cntrl, 151
Codewalkers, 889
Código, 48, 73
Columna, 236
COM, 784
Comando phpinfo, 58
COMMENT, 842
Competencia, 325
Composer, 401
Computer Networks, 396
Connection, 534
Constants, 827
Constants.php, 827
Contenido dinámico, 49
Content.html, 368
Contraseñas, 568
Control Apache Server, 880
Cookie, 482-483
Cookies, 385
Copia del libro, 604
Corp, 338
Correo electrónico, 134
Corto, 46
Cosine, 790
Coste de producción, 782
COUNT, 262
Count, 123, 128
Courier, 803
Courier-Bold, 803
Courier-BoldOblique, 803
Courier-Oblique, 803
Crear correo, 703
  listas, 703
Create_database.sql, 639, 665, 704, 755
Create_priv, 294
Created, 641
Crl_setopt, 438
Crypt, 360
CSR, 870, 347-348
CSS, 512
Cuentas, 676
Cuerpo del mensaje, 772
Cuota, 788
cURL, 379
Curl.haxx.se, 438
Current, 126
Current_dir, 408
CVS, 520
Cvshom.org, 526

D

Data_valid_fns, 553
Data_valid_fns.php, 593, 665, 703, 755
Date, 49, 446
Daterank, 845
Db, 294
Db_fns.php, 553, 593, 639, 755
DDL, 250
DEC, 242
DECIMAL, 242
Define, 58
DELETE, 230
  bm, 579
Delete-account, 670
Delete_bms.php, 553
Delete_book.php, 593, 627
Delete_category.php, 593
Delete_priv, 294
Delete_story, 640
Deletefromcart, 833
Denis, 7
Depuración, 539
DES, 343
DESC, 262, 655
Desconectar, 552
DESCRIBE, 238, 300
Description, 115
Detail, 833
Detalles de pago, 590
  del libro, 590
Developer Token, 822

Devolver valores, 182
DevShed.com, 888
DEVTAG, 831
Diccionario, 574
DICT, 438
Die, 501
Diferente, 254
Diferentes vistas, 754
Digit, 151
Dir, 412
Dirname, 412
Discussion_fns.php, 755
Diseñador gráfico, 631
Display, 205
Display_book_form, 625
Display_cart, 608
Display_errors, 542
Display_login_form, 557
Display_mail_form, 736
Display_shipping, 617
Display_site_info, 557
Display_tree, 763
Display_user_urls, 578
DisplayButton, 203
DisplayMenu, 206
Displayname, 666
Distiller, 787
DLL, 881
DML, 250
DNS, 429
Do, 81
Do_html_footer, 557, 564
Do_html_header, 708
Doctorado, 7
DOCUMENT_ROOT, 89
DOS, 520
DOUBLE, 242
Download Latest, 885
Draw_star, 812
DROP, 230
Drop_priv, 294
DTD, 819
Dump, 538
Duplicar, 782

E

E-gineer, 889
E_ALL, 541
E_COMPILE_ERROR, 540
E_COMPILE_WARNING, 540
E_CORE_ERROR, 540
E_CORE_WARNING, 540
E_ERROR, 540
E_NOTICE, 540
E_PARSE, 540
E_USER_ERROR, 541
E_USER_NOTICE, 541
E_USER_WARNING, 541
E_WARNING, 540
Each, 112, 126
EBCDIC, 804
Echo, 47, 53
Edit_book.php, 593, 627
Edit_book_form.php, 593
Edit_category.php, 593
Edit_category_form.php, 593
Editor, 632, 657
EEUU, 388
Ejecución, 505, 530
Elemento raíz, 819
Eliminar marcador, 552
Else, 73-74
Elseif, 74-75
Empty, 71
Emptycart, 833
Encabezadopag, 174
Encriptación, 342
Encrypt, 394
End, 126
Engelschall.com, 871
Enterprise, 885
ENUM, 245
Enviar correo, 703
EOF, 97
Epoch, 445
Eq_ref, 304
Equifax, 346
Equifaxsecure.com, 346

Ereg_replace, 404
ERP, 7
Error_reporting, 542
Escapeshellcmd, 417
Escopete, 101
eSec, 8
Estilo ASP, 47
  corto, 46
  script, 47
  xml, 46
Estructura de árbol, 753
Etiqueta br, 136
  img, 638
  img src, 633
  meta, 635
  td, 636
Eval, 500
Evil Walrus, 889
Exe, 415
Exit, 81, 501
EXPLAIN, 300
Explode, 142
Expresiones, 69-70
  regulares, 156
Ext_funcs, 503
Extensiones, 504-505, 882
Extensions, 503
EXTR_IF_EXISTS, 130
EXTR_OVERWRITE, 129
EXTR_PREFIX_ALL, 129
EXTR_PREFIX_IF_EXISTS, 130
EXTR_PREFIX_SAME, 129
EXTR_REFS, 130
EXTR_SKIP, 129
Extract, 129
Extreme Programming, 526

F

False, 73
Fclose, 95, 408
Feof, 97
Fgetcsv, 97
Fgets, 97
Fgetss, 97
Fifo, 412
FILE, 98, 123, 231, 231, 412, 438
File_exists, 434
File_priv, 293-294
Filesort, 305
Filetype, 412
FishCartSQL, 628
FLOAT, 242
Footer.php, 639
Fopen, 88, 96
For, 80
Foreach, 80, 503
Forgot_form.php, 553, 570
Forgot_passwd.php, 553
Form, 43
Format_action, 674
Fp, 88
Fpassthru, 98
Fragment, 428
Fraudulentamente, 782
Fread, 99
FreeType, 454
FTP, 91, 382
Fuente courier, 803
  courier-bold, 803
  courier-boldoblique, 803
  courier-oblique, 803
  helvetica, 803
  helvetica-bold, 803
  helvetica-boldoblique, 803
  helvetica-oblique, 803
  symbol, 803
  times-bold, 803
  times-bolditalic, 803
  times-italic, 803
  times-roman, 803
  zapfdingbats, 803
Fullheaders, 688
Fullheaders, 690
Función change_password, 570
  count, 123
  date, 49
  dirname, 412

display_account_select, 682
feof, 97
file, 123
file_exists, 434
filetype, 412
fread, 99
fwrite, 93-94
imap_body, 663
imap_close, 663
imap_delete, 663, 691
imap_expunge, 663, 691
imap_fetchheader, 663, 689
imap_header, 663, 689
imap_open, 663
login, 566
mail, 693
retrieve_message, 688
send_message, 693
shuffle, 120
srand, 122
store_account, 716
store_list, 734
Funciones de usuario, 720
Function, 194
Fwrite, 93-94

G

GDFONTPATH, 464
GE, 7
Generar archivo, 790
Georgia, 7
tech, 7
GET, 44, 385
Get_archive, 727
Get_extension_funcs, 504
Get_loaded_extensions, 504
Get_random_word, 572
Get_user_urls, 578
GetARS, 836
Getenv, 417
Getlastmod, 504
Ghostscript, 785
GIMP, 463
GMT, 445
Gnupg.org, 388
GNUPGHOME, 393
Google, 381
GPG, 388, 620
GRANT, 228, 292
Grant_priv, 294
Graph, 151
Greenspun, 890
GSview, 785
Guardar datos, 86
GUI, 664

H

H1, 45
H2, 45
HAVING, 264
Head, 45, 135
Header, 363, 694
Header.php, 639
Headline, 641
Headlines.php, 639
HEIGHT, 638
Hello World, 804
Helvetica, 803
Helvetica-Bold, 803
Helvetica-BoldOblique, 803
Helvetica-Oblique, 803
Herencia, 197-198
Hide-headers, 670
Highlight.bg, 506
Highlight.comment, 506
Highlight.default, 506
Highlight.html, 506
Highlight.keyword, 506
Highlight.string, 506
Home.netscape.com, 396
HomePage, 888
Host, 293-294, 428
HotScripts.com, 888
Hours, 447

Htaccess, 368
HTML, 43
Htmlspecialchars, 579
HTPPD, 885
HTTP, 91, 796
HTTP_POST_VARS, 564
HTTP_SERVER_VARS, 89
HTTP_SESSION_VARS, 564
Httpd.conf, 368, 872
HTTPS, 422

I

IBM AS/400, 804
Id, 484, 641
IDEA, 343
idenis@ureach.com, 7
IETF, 422
If, 72
Igual, 254
Igualdad, 254
IIS, 883
Image, 833
ImageArc, 478
ImageColorAllocate, 476
ImagecreatefromJPEG, 466
ImagecreatefrompGIF, 466
Imagecreatefrompng, 464
ImageDestroy, 478
ImageFilledRectangle, 476
Imágenes, 463
ImagePNG, 478
ImagePolygon, 478
ImageRectangle, 476
ImageSX, 466
ImageSY, 466
Imagettfbbox, 468
ImageTTFText, 470
IMAP, 382, 662
Imap.org, 662
Imap_close, 671, 691
Imap_delete, 691
Imap_expunge, 691
IMG, 638
  src, 633
Implode, 142
IN, 255
INBOX, 686
Include, 168-169
Include_fns.php, 639, 665, 755
INDEX, 230
Index, 304
Index.php, 592
Index_priv, 293-294
Indianapolis, 8
Inetpub, 883
Infile, 394
Información, 703
Information, 713
Ingeniería, 7
Inglaterra, 7
Ini_set, 505
Inicial, 774
Inicio de sesión, 552
INSERT, 281
Insert_book.php, 593
Insert_book_form.php, 592
Insert_category.php, 592
Insert_category_form.php, 592
Insert_order, 615
Insert_priv, 294
INSTALL, 871
Instrucción each, 112
  exit, 81
Int, 94, 241
INTEGER, 241
Interfaz de administración, 620
Internet, 7
Internet Explorer, 382
Inverse-pricerank, 845
IS NULL, 254
Is_array, 71
Is_dir, 411
Is_double, 71
Is_executable, 411
Is_file, 411
Is_float, 71
Is_int, 71

Is_integer, 71
Is_link, 411
Is_long, 71
Is_object, 71
Is_readable, 411
Is_real, 71
Is_string, 71
Is_writable, 411
ISAPI, 363
ISBN, 626
ISO 8859-1, 804
Israel Denis Jr., 7
Isset, 71
IsURLCurrentPage, 203
Italia, 7
Iteración, 77

J

James, 8
Java, 109
JavaScript, 46
Join, 142
JPEG, 454
Julie, 8

K

Keele University, 7
Keyword_add.php, 640
Keyword_delete.php, 640
Keywords.php, 640, 654
Kitchen, 889
Komodo, 520
Ksort, 117

L

Laura Thomson, 7
laura@tangledwe.com.au, 7
Lawson, 7
Layer, 116
LDAP, 438
Left_margin, 474
Lesley, 8
Lesser GPL, 823
Lettuce, 790
Lightwood Consultancy Ltd., 7
Lightwood.net, 7
LIKE, 255
LIMIT, 264
Line, 183
Link, 412
Linux, 521
Lista de artículos, 754
  de categorías, 590
Listas de correo, 732
  de suscriptores, 704
Listid, 705
Literales, 54
Localfile, 430
LOCK, 102
LOCK_EX, 102
LOCK_NB, 102
LOCK_SH, 102
LOCK_UN, 102
Log-in, 713
Log-out, 672, 713
Log_errors, 542
Lógicos, 530
Login.php, 552, 592, 640
Logo.gif, 639
Logout.php, 553, 592, 640
LONGBLOB, 245
LONGTEXT, 245
Lower, 151
Ltrim, 136
Lynx, 323
LZW, 456

M

MAC, 384, 520
Mac Roman, 804

Macintosh, 786
Macroman, 804
Mail_fns.php, 665
Mail_Mime, 885
Mailid, 705
Main_size, 474
Marcadores, 575-576
Mark Taber, 8
Martin, 8
Maryland, 350
Master en Ingeniería Electrónica, 7
MAX, 262
Max_execution_time, 505
Max_height, 637
Max_size, 738
Max_width, 637
Maxlength, 281
Maxus, 334
Mayor, 254
Mayúsculas, 172
MD5, 345
Mday, 447
MEDIUMBLOB, 245
MEDIUMINT, 241
MEDIUMTEXT, 245
Melbourne, 7
Member.php, 553
Menor, 254
Mensaje original, 772
Menú Change Password, 568
  de admin., 591
Message sent, 670
Messageid, 671
META, 635
METHOD, 43, 44, 844
Microsoft, 804
Mim_fns.php
MIME, 405
Mimetype, 705
MIN, 262
Minúsculas, 172
Minutes, 447
MIT, 344
Mixed, 148
Mkdir, 409
Mktime, 446
Mlm_fns.php, 703
Modified, 641, 705
Mon, 447
Month, 447
Mostrar encabezados, 664
  mis listas, 703
  otras listas, 703
  todas las listas, 703
Mozilla, 323
Myisamchk, 306
MyPrint, 127
MySQL, 104
Mysql_error, 534
Mysql_query, 534
Mysql_select_db, 534
Mysqladmin, 877
Mysqldump, 310

N

NAME, 795
Navegador Web, 46
NET STOP MySQL, 877
Netscape, 382
New York Times, 354
New-account, 713
New_email, 142
New_post, 754
New_post.php, 754
Newman, 7
Next, 126
NNTP, 422, 662
Nobody, 390
Nodos de navegación, 824
Noexistente, 390
Nomenclatura, 515
Nonexistent_function, 533
NOT IN, 255
NOT LIKE, 255
NT, 348
NT4, 883
NTFS, 882

Nuestro carro, 604
Nueva cuenta, 702
NUL, 140
NULL, 56
Num_children, 766
NUMERIC, 242
NuSOAP, 823
Nytimes.com, 354

O

Objetos, 188
Ocultar encabezados, 664
Old_max_execution_time, 505
Omni, 885
Oodie.com, 889
OpenOfficeWriter, 784
Opensource@sampublishing.com, 8
OpenSSL, 91
Operador NOT, 541
Option Pack, 883
OR, 65, 541, 656
Order_fns, 593
Order_fns.php, 593
Output_fns.php, 553, 593, 665, 736, 755

P

Page, 641
Page.php, 639
Panel de control, 877
Parent, 756
ParseXML, 846
Pass, 428
Passthru, 415
Passwd, 390
Password, 293, 360
PATH, 389
Path, 428
Pattern, 423
Paul, 8
PC compartido, 550
PCRE, 149
PDF, 786
Pdf_closepath, 808
Pdf_fill, 808
Pdf_lineto, 808
Pdf_moveto, 808
Pdf_stroke, 808
Pdfdoc, 56
Pdfinfo, 56
PDFlib, 788, 801
Pdflib.php, 789
PEAR, 521
  db, 286
  soap, 823
Perfil, 628
Perl, 149
PGP, 388, 620
Pgpi.org, 388
Photoshop, 463
PHP, 45
  base library, 888
  club, 888
PHP-SOAP, 823
Php-tools.de, 889
Php.ini, 51
Php_imap.dll, 662
PHP4 Resource, 888
Php4apache.dll, 881
Php4isapi.dll, 883
Php4ts.dll, 881
PHPBookmark, 550
PHPBuilder.com, 887
PHPCertification.pdf, 789
PHPCertification.rtf, 789
Phpclub.net, 888
Phpdeveloper, 889
PHPIndex.com, 888
Phpinfo, 58
PHPSESSID, 540
PHPWizard.net, 887
Picture, 641
Pie, 672
  de página, 712

Piratas, 325
Plantilla de certificado PDF, 789
de certificado RTF, 789
Plantillas, 163
Pmrank, 845
PNG, 454
POP, 382
Populate.sql, 593
Port, 428
Pos, 126
POSIX, 149, 255
Posix_getpwuid, 411
POST, 44, 385
Poster, 756
Postid, 755
PostScript, 460, 782
PRECISION, 242
Prefijo, 772
Preprocesamiento, 712
Pretty Good Privacy, 388
Prev, 126
Price, 115
Pricerank, 845
Print, 151
Printf, 137
Privilegios, 227-228
Procesamiento, 44
Procesar pago, 590
PROCESS, 231
Process.php, 592
PROCESSLIST, 302
Procter & Gamble, 7
Product.php, 828
ProductInfo, 819
Propietario, 504
Proyecto Warm Mail, 664
Pseudocódigo, 515
Publish.php, 640
Publish_story.php, 640
Published, 641
Punct, 151
Punto, 53
Purchase.php, 592
Putenv, 417
PWS, 883
PX-PHP, 888
Px.sklar.com, 888
Pyramid, 744
Pyramid-MLM, 708

Q

Query, 428

R

Radians, 790
Radio, 790
Range, 304
RATING, 842
RDBMS, 104, 211
Re, 772
READ, 310
Reader, 786
Readfile, 98
REAL, 242
Realizar preguntas, 789
Recomendaciones, 581-582
Recomendar, 552
Recommend.php, 553
Red Hat, 865
Ref, 304
References_priv, 294
REGEXP, 255
Register, 558
Register_form.php, 552
Register_globals, 51, 402
Register_new.php, 553
Registros MX, 429
RELOAD, 231
Remotefile, 430
RENAME, 266
Reply, 671
Reply-all, 671
Require, 161
Require_once, 556

Reset, 126
Resize_image.php, 637, 639
Resource Index, 888
Resourceindex.com, 888
Retrieve_message, 688
Return, 184
Reutilizar, 160
Reverse, 184
Reviewrank, 845
REVOKE, 231
RFC, 422, 662
Rfc-editor.org, 374
RFC2616, 422
RFC822, 422
RGB, 506
Right_margin, 474
Rmdir, 409
RMIT, 7
Robert, 8
ROOT, 882
Rosette.gif, 790
RPM, 865
RTF, 782
Rtf.php, 789
Rtfspec, 784

S

S-HTTP, 380
SafeString, 832
Salesrank, 845
Salir, 590
SAMBA, 350
Sams, 8
SAP, 7
SAPI, 381
Save changes, 678
Scheme, 428
School of Computer Science and Information Technology, 7
Score, 795
Score.php, 789
SCRIPT, 47
Script, 47
Search, 833
Search.php, 640
Search_form.php, 639
SearchDatabase.com, 889
Searchterm, 274
Searchtype, 274
Seconds, 447
Seguridad, 332-333
SELECT, 252, 682
Select-account, 670
Select_fns.php, 639
Select_priv, 294
Send, 714
Sendmail_path, 870
Serialize, 502
Servidor FTP, 432
Sesión, 484, 712
Session.auto_start, 490
Session.cache_expire, 490
Session.cookie_domain, 490
Session.cookie_lifetime, 490
Session.cookie_path, 490
Session.name, 490
Session.save_handler, 490
Session.save_path, 490
Session.use_cookies, 490
Session_destroy, 668
Session_start, 582
SET, 245, 264
SETUP.EXE, 893
SGML, 816
SHA, 345
Shelley Johnston, 8
Ship_address, 595
Ship_city, 595
Ship_country, 595
Ship_name, 595
Ship_state, 595
Ship_zip, 595
SHOW, 238, 300
Show-all-lists, 713
Show-archive, 713
Show-headers, 670
Show-my-lists, 708, 713

Show-other-lists, 708, 713
Show_book.php, 592
Show_cart.php, 592
Show_cat.php, 592
Showcart, 833
ShowCategories, 834
Shuffle, 120
SHUTDOWN, 231, 877
Shutdown_priv, 294
Signature.png, 789
SimilarProducts, 818
Sintácticos, 530
Sistema, 381
  operativo, 297
Sistemas informáticos, 7
Sitio Web Alex´s Guide, 890
  application tools, 889
  base library, 888
  berber, 888
  center, 888
  clases php, 888
  codewalkers, 889
  devshed.com, 888
  download latest, 885
  e-gineer, 889
  enterprise, 885
  evil walrus, 889
  greenspun, 890
  homepage, 888
  hotscripts.com, 888
  mail_mime, 885
  omni, 885
  oodie.com, 889
  option pack, 883
  php base library, 888
  php club, 888
  php-tools.de, 889
  php4 resource, 888
  phpbuilder.com, 887
  phpclub.net, 888
  phpdeveloper, 889
  phpindex.com, 888
  phpwizard.net, 887
  pws, 883
  px-php, 888
  px.sklar.com, 888
  resource index, 888
  resourceindex.com, 888
  searchdatabase.com, 889
  sqlcourse2.com, 889
  today, 890
  web kitchen, 889
  weberdev.com, 888
  webmonkey.com, 888
  week, 890
Sitios Web adaptables, 7
Sizeof, 128
Slashdot, 354
Small_size, 474
SMALLINT, 241
Smart Form, 154
Smarty, 523
SMTP, 382
SOAP, 425
Soap.amazon.com, 839
Software especial, 782
Something, 196
Sort, 117
Space, 151
Span, 164
Sprintf, 137
Sqlcourse2.com, 889
Srand, 122
SSL, 91, 380
STD, 262
STD0053, 422
STDDEV, 263
Stokeon-Trent, 7
Store-account, 713
Store-settings, 670
Store_account, 716
Store_account_settings, 678
Store_list, 734
Store_new_post.php, 755
STORED, 746
Stories.php, 640
Story.php, 640
Story_submit.php, 640
Story_text, 641
Str_replace, 148

Strcasecmp, 144
Strchr, 146
Strcmp, 144
String, 148
StripSlashes, 140
Stristr, 146
Strlen, 145
Strnatcmp, 144
Strpos, 147
Strrchr, 146
Strrpos, 147
Strstr, 146
Strtok, 142-143
Strtolower, 140
Strtoupper, 140
Subject, 694
Sublist, 770
Submit Order, 45
Subscribe, 713
Subscribir, 702
Substr, 143
Substr_replace, 148
SUM, 263
SUMMARY, 842
support@phpbookmark, 574
Suscribers, 706
Suscriptores, 704
Swatch, 444
Switch, 75
Symbol, 803
System, 304, 415

T

Taber, 8
Tablas, 234
Tahuayo, 827
Tangent, 790
Tangledweb, 554
Tar.bz2, 388
Tar.gz, 388
TCP/IP, 381
Td, 43, 636
Telnet, 382
Telstra, 7
Tera Term, 432
Ternario, 66
Test.out, 392
Test.txt.asc, 392
TESTED, 746
TEXT, 244
Text_indent, 474
Textfilename, 744
Texto, 470
Tfp, 744
Thawte, 338
Thawte.com, 338
This, 764
Thomas A. Powell, 526
Thomson, 7
Threads_cached, 302
Threads_connected, 302
Threads_running, 302
Tienda en línea, 628
TIFF, 788
Times New Roman, 794
Times-Bold, 803
Times-BoldItalic, 803
Times-Italic, 803
Times-Roman, 803
TIMESTAMP, 242
Timestamp, 296
TINYBLOB, 245
TINYINT, 241
Tireqty, 51
Title_above_line, 476
Title_size, 474
Titlerank, 845
TLS, 185
Tn_height, 638
Tn_width, 638
To, 694
Today, 890
Token, 822
Topbar.php, 827
Total Items, 604
Total Price, 604
Totales, 67

Touch, 413
Traceroute, 334
Track_errors, 542
Transacciones, 377
Treenode, 764
Treenode_class, 755
Trim, 136
True, 63
TrueType, 454
Ttf, 467
Type, 844

U

Ucfirst, 140
Ucwords, 140
UNISYS, 456
UNIX, 67
Unknown, 412
Unlink, 100-101, 394
Unpublish_story.php, 640
Unset, 71, 668
Unsubscribe, 713
UPDATE, 264
Update_priv, 294
Upload.php, 704
UPLOAD_ERR_FORM_SIZE, 405
UPLOAD_ERR_INI_SIZE, 405
UPLOAD_ERR_NO_FILE, 405
UPLOAD_ERR_PARTIAL, 405
UPLOAD_ERROR_OK, 405
Upper, 151
URL, 44, 200
URL válidos, 551
Url_array, 564
Url_fns.php, 553
USAGE, 231
User, 294, 428
User_auth_fns.php, 553, 570, 593, 639, 665, 704
User_urls, 582
Userfile, 737
Userid, 492
Usort, 127
Usuario, 642
Usuarios, 227
UTF-8, 818
Utilityfunctions.php, 828, 832

V

VARCHAR, 244
Variable admin_user, 622
  email, 574
  from, 574
  fullheaders, 688
  get, 674
  items, 604
  mesg, 574
  old_password, 570
  post, 674
  total_price, 604
Variables, 50-51
Ver carro, 590
  correo, 703
  marcadores, 552
  un artículo, 754
VeriSign, 338
Verisign.com, 338
Very, 152
View-mail, 714
View-mailbox, 670
View-message, 670
View_post.php, 755
Void, 71

W

W3.org, 816
W3C, 816
Warm Mail, 664
WBMP, 454
Wday, 447
Web fishcart, 628
  tangled Web Design, 7

WeberDev.com, 888
WebMonkey.com, 888
Webserver, 760
Week, 890
Weekday, 447
Welling, 7
WHERE, 254
While, 78-81
White, 458
WIDTH, 638
Winansi, 804
Windows, 122, 411
Winsock 2, 879
Word, 784
WordViewer, 783
WRITE, 310
Writer, 641
WSDL, 820, 851
WWW.ansi.org, 268
WWW.mysql.com, 268
WWW.tripwire.com, 336
WWWroot, 883

X

Xdigit, 151
XHTML, 136
XML, 46
Xmlns, 820
XOR, 65-66
XSLT, 859

Y

Yday, 447
Year, 447
Your.crt, 871
Your.key, 871
Yourdomain, 874
Yourserver, 874

Z

ZapfDingbats, 803
Zend, 524
Zend.com, 526
Zimmermann, 388
Zip, 436, 662
ZVON, 820