Manual de Referencia

Código en línea gratuito

PHP

- Diseñe, depure e implemente aplicaciones Web con características enriquecidas.
- Trabaje contenido HTML, XML, bases de datos y multimedia.
- Aprenda técnicas avanzadas para Web 2.0 como AJAX y RSS.

Mc Graw Hill

Steven Holzner

PHP:
Manual de referencia

Acerca del autor

Steven Holzner es autor galardonado de 100 libros de computación, incluidos tres "bestsellers" sobre PHP. Fue editor de *PC Magazine*, imparte clases de programación en compañías de Fortune 500 y ha sido catedrático de la Universidad Cornell y de MIT.

Acerca del editor técnico

Chris Cornutt ha sido miembro de la comunidad PHP por 8 o 9 años. Poco después de descubrir el lenguaje inició su sitio, PHPDeveloper.org, para compartir acontecimientos y opiniones de otros adeptos a PHP en todo el mundo. Chris ha escrito para publicaciones sobre PHP, como *php | architect* y la revista *International PHP Magazine*, sobre temas que van de la geocodificación a "trackbacks" (o vínculos inversos).

Chris vive en Dallas, Texas, con su esposa e hijo y trabaja para un importante distribuidor de gas natural, dando mantenimiento a su sitio Web y desarrollando aplicaciones basadas en PHP.

PHP: Manual de referencia

Steven Holzner

Traducción

Juan Carlos Vega Fagoaga
MCSE
Ingeniero en sistemas, I.T.A.M.
Instituto Anglo-Mexicano de Cultura

MÉXICO • BOGOTÁ • BUENOS AIRES • CARACAS • GUATEMALA • LISBOA • MADRID
NUEVA YORK • SAN JUAN • SANTIAGO • AUCKLAND • LONDRES • MILÁN
MONTREAL • NUEVA DELHI • SAN FRANCISCO • SINGAPUR • SAN LUIS • SIDNEY •
TORONTO

Director editorial: Fernando Castellanos Rodríguez
Editor de desarrollo: Miguel Ángel Luna Ponce
Supervisora de producción: Jacqueline Brieño Álvarez
Formación: María Alejandra Bolaños Avila

PHP: MANUAL DE REFERENCIA

Prohibida la reproducción total o parcial de esta obra,
por cualquier medio, sin la autorización escrita del editor.

Educación

DERECHOS RESERVADOS © 2009, respecto a la primera edición en español por
McGRAW-HILL/INTERAMERICANA EDITORES, S.A. DE C.V.
A Subsidiary of The McGraw-Hill Companies, Inc.
Corporativo Punta Santa Fe
Prolongación Paseo de la Reforma 1015, Torre A,
Piso 17, Colonia Desarrollo Santa Fe,
Delegación Álvaro Obregón,
C.P. 01376, México, D.F.
Miembro de la Cámara Nacional de la Industria Editorial Mexicana, Reg. Núm. 736

ISBN-13: 978-970-10-6757-4
ISBN: 970-10-6757-6

Traducido de la primera edición de
PHP: The Complete Reference
By: Steven Holzner
Copyright © MMVIII by The McGraw-Hill Companies. All rights reserved

ISBN: 978-0-07-150854-4

1234567890 8765432109

Impreso en México *Printed in Mexico*

A Nancy

Contenido breve

1	Fundamentos de PHP	1
2	Operadores y control de flujo	41
3	Cadenas y matrices	81
4	Creación de funciones	123
5	Lectura de datos en páginas Web	161
6	Poder de manejo de navegadores de PHP	203
7	Programación orientada a objetos	245
8	Programación avanzada orientada a objetos	281
9	Manejo de archivos	319
10	Trabajo con bases de datos	361
11	Sesiones, cookies y FTP	395
12	Ajax	433
13	Ajax avanzado	467
14	Trazo de imágenes en el servidor	501
15	XML y RSS	537
	Índice	575

Contenido

Introducción xvii

1 Fundamentos de PHP 1
Introducción a PHP 1
Cómo obtener PHP 3
PHP en Internet 4
PHP en su máquina local 5
Creación de su entorno de desarrollo 6
Creación de su primera página en PHP 8
Ejecución de su primera página en PHP 9
Cómo corregir algunos problemas 9
Combinación de HTML y PHP 10
Impresión de texto 14
Impresión de HTML 16
Mayor poder de echo 17
Uso de documentos "here" de PHP 19
PHP en la línea de comandos 20
Adición de comentarios a código de PHP 24
Trabajo con variables 26
Almacenaje de datos en variables 27
Interpolación de cadenas 31
Creación de variables alternas 33
Creación de constantes 35
Entendamos los tipos de datos internos de PHP 37

2 Operadores y control de flujo 41
Operadores matemáticos de PHP 41
Trabajando con los operadores de asignación 46
Incremento y disminución de valores 48
Operadores de cadena de PHP 50
Operadores orientados a bits 51
Operador de ejecución 52
Precedencia de operadores de PHP 53
Uso de la instrucción if 55
Operadores de comparación de PHP 59
Operadores lógicos de PHP 61

Instrucción else 63
Instrucción elseif 65
Operador ternario 66
Instrucción switch 67
Uso de ciclos for 69
Uso de ciclos while 72
Uso de ciclos do...while 74
Uso del ciclo foreach 76
Terminación prematura de ciclos 77
Omisión de iteraciones 78
Sintaxis alterna de PHP 80

3 Cadenas y matrices 81
Funciones de cadena 81
Conversión a y desde cadenas 87
Formateo de cadenas de texto 88
Construya sus propias matrices 92
Modificación de datos en matrices 95
Eliminación de elementos de una matriz 97
Manejo de matrices con ciclos 99
Ciclo for 99
Función print_r 100
Ciclo foreach 101
Ciclo while 103
Funciones de matriz de PHP 104
Conversión entre cadenas y matrices utilizando implode y explode 106
Extracción de datos de matrices 107
Clasificación de matrices 109
Uso de operadores de matriz de PHP 110
Comparación de unas matrices con otras 112
Manejo de matrices multidimensionales 112
Uso de matrices multidimensionales en ciclos 114
Desplazamiento por matrices 116
Separación y combinación de matrices 117
Otras funciones de matriz 119

4 Creación de funciones 123
Creación de funciones en PHP 123
Paso de datos a funciones 125
Paso de matrices a funciones 127
Paso por referencia 130
Uso de argumentos predeterminados 132
Paso de números variables de argumentos 133

Retorno de datos de funciones 135
Retorno de matrices 137
Retorno de listas 139
Retorno de referencias 141
Introducción de un ámbito variable en PHP 143
Acceso a datos globales 145
Trabajo con variables estáticas 147
Funciones condicionales de PHP 150
Funciones variables de PHP 153
Anidación de funciones 156
Creación de archivos de inclusión 157
Retorno de errores de funciones 158

5 Lectura de datos en páginas Web 161
Configuración de páginas Web para comunicación con PHP 161
Manejo de campos de texto 164
Manejo de áreas de texto 167
Manejo de casillas de selección 170
Manejo de botones de radio 173
Manejo de cuadros de lista 175
Manejo de controles de contraseña 179
Manejo de controles ocultos 182
Manejo de mapas de imagen 184
Manejo de cargas de archivos 187
Manejo de botones 191
Cómo hacer que persistan los datos de botones 192
Uso de botones de envío como botones HTML 195

6 Poder de manejo de navegadores de PHP 203
Uso de variables de servidor de PHP 203
Uso de encabezados HTTP 205
Cómo determinar el tipo de navegador del usuario 206
Cómo redirigir navegadores con encabezados HTTP 209
Vaciado de todos los datos en una forma de una sola vez 212
Manejo de datos de una forma con matrices personalizadas 215
Cómo poner todo en una página 218
Ejecutar la validación de datos 221
Cómo comprobar si el usuario ingresó los datos requeridos 223
Requerir números 227
Requerir texto 230
Datos de usuario persistentes 234
Validación de datos en el cliente 237
Manejo de etiquetas HTML en la entrada del usuario 241

7 Programación orientada a objetos 245
Creación de clases 246
Creación de objetos 250
Cómo establecer el acceso a propiedades y métodos 253
Acceso público 253
Acceso privado 254
Uso de constructores para inicializar objetos 257
Uso de destructores para eliminar objetos 260
Basar una clase en otra con herencia 262
Acceso protegido 264
Constructores y herencia 266
Llamada a métodos de clases de base 267
Sustitución de métodos 271
Sobrecarga de métodos 273
Carga automática de clases 277

8 Programación avanzada orientada a objetos 281
Creación de métodos estáticos 281
Creación de un método estático 283
Paso de datos a un método estático 285
Uso de propiedades en métodos estáticos 286
Miembros estáticos y herencia 291
Creación de clases abstractas 294
Creación de interfaces 297
Soporte a la iteración de objetos 301
Comparación de objetos 304
Creación de constantes de clase 306
Uso de la palabra clave final 308
Clonación de objetos 312
Reflexión 315

9 Manejo de archivos 319
Cómo abrir archivos usando fopen 319
Recorrido cíclico del contenido de un archivo con feof 322
Lectura de texto de un archivo utilizando fgets 322
Cómo cerrar un archivo 323
Lectura del contenido de un archivo carácter por carácter con fgetc 325
Lectura de un archivo completo con file_get_contents 328
Lectura de un archivo en una matriz con file 330
Comprobar si existe un archivo con file_exists 332
Cálculo del tamaño del archivo con filesize 334
Interpretación de lecturas binarias con fread 335
Análisis gramatical de archivos con fscanf 338
Análisis gramatical de archivos ini con parse_ini_file 339

Obtención de información del archivo con stat 341
Cómo establecer la ubicación del puntero del archivo con fseek 343
Copia de archivos con copy 343
Eliminación de archivos con unlink 345
Cómo escribir en un archivo con fwrite 346
Lectura y escritura de archivos binarios 348
Anexión de información a archivos con fwrite 352
Escritura en todo el contenido del archivo con file_put_contents 355
Bloqueo de archivos 357

10 Trabajo con bases de datos 361
¿Qué es una base de datos? 362
Algunos elementos esenciales de SQL 362
Creación de una base de datos MySQL 364
Creación de una nueva tabla 367
Colocación de datos en la nueva base de datos 368
Acceso a la base de datos en PHP 370
Conexión al servidor de bases de datos 371
Conexión a la base de datos 372
Lectura de la tabla 372
Presentación de los datos de la tabla 374
Cierre de la conexión 376
Actualización de bases de datos 377
Inserción de nuevos elementos en una base de datos 380
Eliminación de registros 383
Creación de nuevas tablas 385
Creación de una nueva base de datos 389
Clasificación de sus datos 393

11 Sesiones, cookies y FTP 395
Cómo establecer una cookie 395
Lectura de una cookie 397
Cómo establecer la caducidad de una cookie 399
Eliminación de cookies 400
Trabajando con FTP 402
Descarga de archivos con FTP 406
Carga de archivos con FTP 408
Eliminación de un archivo con FTP 411
Creación y eliminación de directorios con FTP 414
Envío de correo electrónico 416
Envío de correo electrónico avanzado 418
Cómo agregar archivos adjuntos al correo electrónico 421
Almacenaje de datos en sesiones 425
Cómo escribir un contador de visitas utilizando sesiones 429

12 Ajax 433
Comience a utilizar Ajax 433
Redacción en Ajax 435
Creación del objeto XMLHttpRequest 436
Cómo abrir el objeto XMLHttpRequest 440
Manejo de datos descargados 441
Inicio de la descarga 445
Creación de objetos XMLHttpRequest 446
Ajax con algo de PHP 448
Paso de datos al servidor con GET 449
Paso de datos al servidor con POST 453
Manejo de XML 456
Manejo de XML con PHP 464

13 Ajax avanzado 467
Manejo de solicitudes concurrentes de Ajax con múltiples objetos XMLHttpRequest 467
Manejo de solicitudes concurrentes de Ajax con una matriz XMLHttpRequest 472
Manejo de solicitudes concurrentes de Ajax con funciones internas de JavaScript 475
Descarga de imágenes por medio de Ajax 479
Descarga de JavaScript con Ajax 481
Conexión a Google Suggest 484
Conexión a otros dominios utilizando Ajax 494
Inicio de sesión con Ajax y PHP 495
Obtención de datos con solicitudes de encabezados y Ajax 497

14 Trazo de imágenes en el servidor 501
Creación de una imagen 504
Presentación de imágenes en páginas HTML 506
Trazo de líneas 507
Cómo establecer el grosor de la línea 510
Trazo de rectángulos 511
Trazo de elipses 513
Trazo de arcos 514
Trazo de polígonos 516
Relleno de figuras 518
Trazo de píxeles individuales 520
Trazo de texto 522
Trazo de texto vertical 525
Trabajo con archivos de imágenes 528
Colocación de imágenes en mosaico 531
Copia de imágenes 535

15 XML y RSS **537**
Creación de XML 537
Creación de RSS 540
Uso de las funciones SimpleXML 544
Extracción de atributos 550
Uso de XPath 552
Modificación de elementos y atributos XML 555
Adición de nuevos elementos y atributos 557
Envío de XML al navegador 560
Interacción con otros paquetes XML de PHP 561
Análisis de datos con las funciones analizadoras XML 563

Índice **575**

Introducción

Este libro es su manual para PHP, se escribió con la intención de ser lo más completo y exhaustivo posible. Pone el poder de PHP a trabajar para usted, haciendo hincapié en revisar ejemplo tras ejemplo. Llevamos PHP a sus últimos efectos en este libro, con más de una centena de ejemplos, listos para ejecutarse.

PHP es un tema candente, se ha convertido en el lenguaje para servidores más popular. Una búsqueda de "PHP" en Google da como resultado un asombroso número de 2 890 000 000 coincidencias. Son dos mil ochocientos noventa millones de coincidencias, mucho más de lo que puede presumir cualquier otro lenguaje para servidor.

¿Qué hay detrás de esta increíble popularidad? PHP es rápido y fácil de usar; también, rápido y sencillo en la programación de aplicaciones. Puede combinarlo con HTML en sus páginas Web. Puede escribirlo más fácil que otros lenguajes (PHP ha aprendido de sus errores). No necesita compilarlo, como requerirían otros lenguajes, antes de ejecutarlo. Pero más que todo eso, la programación con PHP es pura diversión. Es simplemente un lenguaje fenomenal que los escritores de código para servidor realmente disfrutan. Y en este libro tratamos de traerle tal experiencia.

Las personas con sitios Web requieren más poder estos días y cada vez encuentran la respuesta con mayor frecuencia en PHP; y no contentos con limitarse a trabajar JavaScript en el navegador, desean tener el poder de escribir código para que se ejecute desde el servidor. Libros de visitantes, calendarios interactivos, bases de datos, correo electrónico de respuesta automática, blogs, salas de chat —las cosas que puede hacer con PHP son ilimitadas—. Utilizando PHP, usted tiene control total de sus aplicaciones Web (y lo bueno es que no son mucho más difíciles de escribir que una página Web típica). Puede hacer mucho con poco.

Comience a usar PHP en el momento más apropiado. El entusiasmo se encumbra y PHP vuela alto. Este libro intenta mantenerse lo más fiel posible al espíritu de esa excitación, brindándole toda la experiencia de PHP. Encontrará más PHP en este libro que en cualquier texto similar mientras captura los detalles completos de la historia de PHP.

Este libro es para usted

Éste es su libro si desea desarrollar todo el poder del que PHP es capaz y ver muestras en todos los pasos del recorrido. Por ejemplo, bien podría instalar cookies en las computadoras de otras personas, más que simplemente aceptarlas en su computadora. Podría leer datos ingresados

por usuarios en campos de texto, cuadros de lista, casillas de selección o botones de radio en su página Web. Podría conservar datos de su tienda en línea, en una base de datos en el servidor. Podría registrar a los usuarios con sesiones, obteniendo la capacidad de crear aplicaciones Web de múltiples páginas.

Cualquiera que sea su necesidad en línea, este libro es para usted.

Y este libro se escribió para que usted no necesite muchos conocimientos base para utilizarlo. De hecho, lo único que necesita saber antes de leer y trabajar con este libro es un poco de HTML. No necesita ser un experto en HTML, pero sí necesita conocer los fundamentos para abordar este libro. Si no tiene idea de lo que es HTML, éste es el momento de estudiar un tutorial en línea sobre el tema.

Utilizamos PHP 5.2 en este libro (es posible que ya lo tenga en el servidor). Si no, se enterará dónde conseguirlo sin costo en este libro y cómo instalarlo. De hecho, ni siquiera necesitará un servidor de Internet compatible con PHP para leer este libro (si lo desea, puede desarrollar y probar sus páginas PHP en la misma computadora). Por otra parte, si desea subir su código PHP a Internet, necesitará recurrir a los servicios de un PSI (proveedor de servicios de Internet) ofreciendo compatibilidad con PHP. Pregunte a su PSI si cuenta con soporte para PHP (cada vez más PSIs lo ofrecen todos los días).

Este libro se ha escrito para ser lo más completo posible y situarse en la cima de su campo. Si tiene preguntas o comentarios, por favor escríbame, me dará mucho gusto saber de usted.

¿Dónde puede conseguir el código?

Todo el código de los ejemplos de este libro está disponible en línea; de modo que usted **no** necesita escribirlo todo. Debe descomprimir esos ejemplos en su servidor; son fáciles de ejecutar (con excepción de ejemplos que necesitan contraseñas para su sistema de base de datos o conectarse con otro PSI).

Puede obtener el código de los ejemplos de este libro en http://www.mcgraw-hill-educacion.com/ Todo lo que debe hacer es descargar el archivo comprimido (zip) y descomprimirlo (todo está ahí).

Bueno, eso nos da el punto de partida que necesitamos. PHP es su puerta de acceso al poder del servidor y se integrará a un recorrido guiado de PHP en este libro. Todo el código PHP que puede caber en un libro se ha incorporado aquí. Todo lo que resta es comenzar a trabajar continuando con el capítulo 1.

CAPÍTULO 1

Fundamentos de PHP

Dé un vistazo a la Figura 1-1. Es la página principal de la Universidad Estatal de Ohio. Bastante llamativa, ¿no? Ahora observe más de cerca la URL en la barra de dirección: http://www.osu.edu/index.php. Lo que observa es una página PHP, index.php.

Aquí le presentamos otra página: http://www1.umn.edu/twincities/index.php, es de la Universidad de Minnesota, en la Figura 1-2. También es una página PHP, como se puede inferir de la URL. Nada mal.

Y ésta es otra más: la página de mapas de Yahoo!, en la Figura 1-3, http://maps.yahoo.com/index.php. ¿Desea orientación al conducir? Tan sólo ingrese sus ubicaciones inicial y final en esa página y haga clic en Go. PHP hará el resto.

Introducción a PHP

Bienvenido al mundo de PHP. Oficialmente significa "PHP: Hypertext Preprocessor" (o "PHP: Preprocesador de hipertexto"), pero aún sigue conociéndose en todo el mundo por su nombre original, "Personal Home Page" (o "Página de inicio personal"). Es el lenguaje de programación para servidor que ha tomado al mundo Web por asalto (PHP es, en gran medida, el lenguaje de programación más popular en uso para servidores Web). Ésa es la idea tras PHP: programar de manera sencilla en el servidor Web creando todo, desde bases de datos en línea hasta libros de visitantes; desde programadores de clientes hasta salas de chat; desde herramientas para cargar archivos hasta carritos de compra. Todo es posible con PHP.

¿De dónde vino PHP? Los usuarios de este lenguaje se sorprenden algunas veces al enterarse de que PHP ha estado entre nosotros desde hace buen tiempo; fue creado pro Rasmus Lerdorf en 1994 (Rasmus deseaba tener un recurso para registrar quién observaba su currículum en línea). PHP se hizo de tan buena reputación que, en 1995, estaba disponible para uso de otras personas y la revolución de PHP entró en curso.

En ese entonces, a PHP se le conocía como Personal Home Page o Personal Home Page Tools (herramientas para páginas de inicio personales). Por aquellas fechas, como sería de esperar, PHP era muy simple y se podía usar para crear contadores de visitas a páginas Web, libros de visitantes, etc. La versión de 1995 de PHP se denominó PHP/FI versión 2 (FI fue un paquete lector de formas codificadas en HTML, también escrito por Rasmus).

En su momento, Rasmus agregó soporte para interactuar con Mini SQL (mSQL) y PHP/FI comenzó a crecer a una velocidad impresionante, conforme más personas hicieron aportaciones

FIGURA 1-1 Página principal de la Universidad Estatal de Ohio

de código al lenguaje. Entonces existía una necesidad real de contar con un lenguaje de programación para servidores Web de fácil uso, mientras el número de páginas PHP simplemente continuaba en aumento. En 1996, PHP/FI ya era usado por 15 000 páginas Web. En 1997, ese número aumentó a más de 50 000.

A partir de ese punto, los eventos sucedieron con rapidez. En 1997, PHP/FI se convirtió simplemente en PHP y más personas se involucraron en el movimiento, conforme aparecieron equipos de trabajo PHP. Zeev Suraski y Andi Gutmans trabajaron mucho del código fuente, con la consiguiente aparición de PHP versión 3 (reescrito casi en su totalidad).

Hoy día, PHP está por doquier en la Web, con un estimado de 100 millones de páginas (es difícil obtener estadísticas precisas; si realiza una búsqueda de la palabra PHP en Google, por ejemplo, se obtiene una sorprendente aproximación de 9 790 000 000 coincidencias). PHP sigue fiel a su nombre original: Personal Home Page, ya que proporciona el medio más sencillo para que sus páginas Web cobren vida en el servidor. Pero PHP se ha vuelto también un lenguaje muy profesional, adecuado para sitios de muy altos vuelos.

Lo verá completo en este libro. Es aquí donde sus páginas y aplicaciones Web cobran vida.

La tapa de la caja se levanta cuando comienza a trabajar con PHP. Casi cualquier cosa realizable en la Web puede lograrse mediante PHP. Las páginas Web ya no tienen que ser estáticas e inmutables (podrá interactuar con los usuarios de manera segura, devolviendo páginas Web personalizadas según sus especificaciones). Y todo esto en tiempo real.

Puede manejar fácilmente clics en botones, selecciones de botones de radio y opciones en cuadros de lista con PHP. Es posible codificar aplicaciones Web simples, como libros de visitantes o comportamiento avanzado aplicable a la Web: diseño de aplicaciones de bases de datos,

FIGURA 1-2 Página principal de la Universidad de Minnesota

aplicaciones cliente/servidor y procesadores de datos multinivel; gráficos interactivos en el servidor y para su alcance desde el navegador; registrar estudiantes de su escuela en línea; crear salones de clase basados en Web y más.

Las páginas Web estáticas son, eh... estáticas. Pueden mostrar datos correctamente, además de texto e imágenes. Pero, en realidad, nada ocurre en ellas (el usuario puede interactuar con nada). PHP cambia todo eso haciendo que tales páginas Web cobren vida (las actividades se dan en el servidor). A diferencia de lenguajes como JavaScript, que trabajan en el navegador y sin efecto duradero (JavaScript no puede escribir archivos ni trabaja con datos almacenados en el servidor), los lenguajes en ejecución desde el servidor pueden usarse como base de aplicaciones Web reales. Los usuarios podrán abrir sus páginas, ver lo que esperarían obtener de una aplicación con todas las características, desde campos de texto en que pueden ingresar texto, hasta tablas repletas de datos creadas al instante; desde recuperar información de bases de datos, hasta crear gráficos con fluidez (cada detalle que usted querría ver en dicha aplicación Web interactiva profesional, está ahora a su alcance). Éste es el nombre del juego: PHP (responder al usuario de forma dinámica e instantánea).

Cómo obtener PHP

En este libro se usará PHP 5.2.0, cuyo sitio Web oficial es www.php.net (PHP perdió la oportunidad de conseguir el sitio php.com, que ahora pertenece a Parents Helping Parents, algo así como Padres de familia ayudándose unos a otros). Necesitará PHP para trabajar con este libro y ello significa acceder a un servidor Web que lo ejecute.

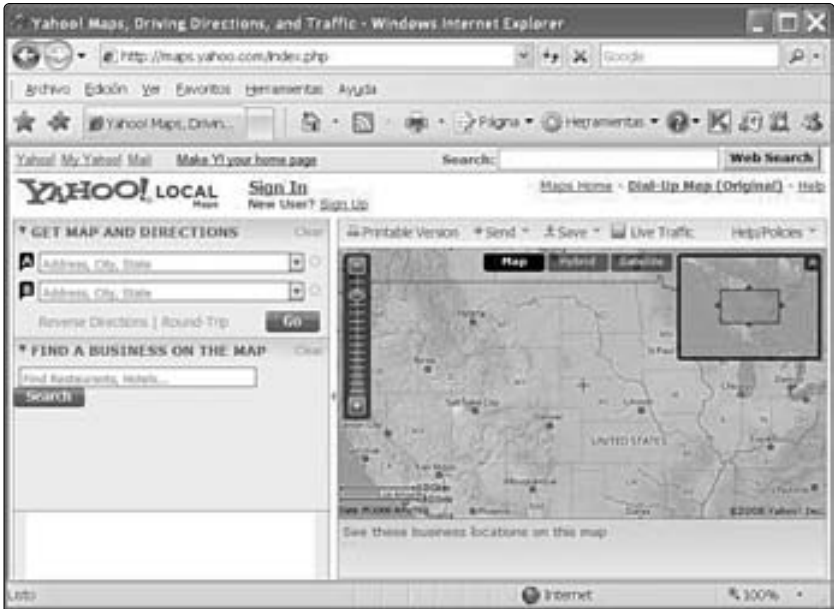

FIGURA 1-3 Página de mapas de Yahoo!

Mientras escribía el presente libro, PHP 6 estaba en proceso de desarrollo. Existen muchos cambios menores que aparecerán en PHP 6 —aunque ninguno de ellos debería impedir la ejecución de su código en PHP 5— y un cambio mayor: el soporte para Unicode. Unicode (www.unicode.org) es un conjunto de caracteres diseñado para abarcar lenguajes internacionales, a diferencia del PHP actual. Mientras escribo esto, el soporte para Unicode en PHP 6 es transparente —es decir, su código de PHP 5 se ejecutará sin problemas—. Exagerando, se le requerirá colocar una directiva al principio de sus scripts indicando si desea que el soporte para Unicode esté habilitado o deshabilitado. Después tendrá todo el conjunto de caracteres Unicode —cualquiera pasando por arábigo, cherokee a tibetano.

PHP en Internet

De hecho, es muy probable que su proveedor de servicios de Internet (PSI) ya cuente con soporte para PHP —puede preguntar a su personal de soporte o intentar la carga y ejecución de un archivo PHP del tipo que elaboraremos en breve.

También puede abrir un símbolo del sistema para su servidor y comprobar PHP de esa manera. Abriendo una ventana de símbolo del sistema conectada a su servidor, mediante diversas utilidades —Telnet, SSH o SSH2 (no necesitará estas aplicaciones en este libro, así que no se preocupe si no las tiene). Por ejemplo, Windows viene con una utilidad Telnet integrada (simplemente escriba C:\>telnet, seguido del nombre de su PSI (por ejemplo, phpbigserver.com) y pulse la tecla ENTRAR.

Una vez abierto el símbolo de sistema para su servidor, puede comprobar si PHP está instalado con la opción –v, que indicará la versión de PHP, si es que se encuentra (observe que voy a usar

% como símbolo genérico de sistema en este libro, pues representa el símbolo del sistema de Windows, Linux, etcétera):

```
%php –v
```

Si PHP está instalado y accesible, verá la versión y fecha de PHP de esta forma:

```
%php -v
PHP 5.2.5 (cli) (built: Nov 8 2007 23:18:51)
Copyright (c) 1997-2007 The PHP Group
Zend Engine v2.2.0, Copyright (c) 1998-2007 Zend Technologies
```

Otra forma de probar si tiene PHP instalado, consiste en cargar un script de PHP y ver si se ejecuta. Para ello, compruebe los scripts de muestra en este capítulo. Cargue uno de éstos en su PSI y vea si puede acceder a él desde su navegador.

Observe que en sistemas Unix, primero debe establecer de forma explícita el permiso del script como *ejecutable* y la mayoría de las utilidades FTP (utilidades File Transfer Protocol, o protocolo de transferencia de archivos, incluyendo la integrada en Windows y accesible tecleando **ftp** en el símbolo del sistema) le permitirán establecer ese permiso. Si tiene un script PHP en un servidor con Unix, debe establecer estos permisos a Owner (propietario): Execute (ejecución), Read (lectura) y Write (escritura); Group (grupo): Execute and Read y Public (público): Execute and Read. Numéricamente, eso da como resultado un parámetro de permiso 755 para scripts de PHP en servidores con Unix.

RECOMENDACIÓN *Si desea una lista de los PSI que ya utilizan PHP, dé un vistazo a www.php.net/links.php#hosts.*

PHP en su máquina local

Le conviene instalar PHP en su máquina local si desea realizar tareas de programación importantes con PHP (¿y quién no lo desea?). De esa forma examinará sus páginas PHP en un navegador Web de su máquina, inmediatamente después de editarlas localmente. Eso acelera los procesos considerablemente y reduce a la mitad el ciclo de desarrollo. Pero para lograrlo de este modo, deberá instalar PHP localmente.

Algunos sistemas operativos, como Linux y algunas versiones de Unix, vienen con PHP ya instalado. Puede probar con el comando –v, como se dijo en la sección anterior. Si obtiene una respuesta mostrando la fecha de creación de PHP, está listo para trabajar. Sin embargo, en Windows tendrá que instalar PHP usted mismo.

Existen versiones "binarias" preintegradas, listas para descargarse y ser instaladas de inmediato en varios sistemas operativos: Windows, Mac OS X, Novell NetWare, OS/2, RISC OS, SGI IRIX 6.5.x y AS/400. Puede encontrar el paquete de instalación binario para Windows en www.php.net/downloads.php, junto con vínculos a binarios para los sistemas operativos mencionados.

No debe ser complicada la instalación de PHP en su computadora con sólo utilizar el archivo binario apropiado. Por ejemplo, en Windows descargará un archivo instalador para Windows (.msi) y hará doble clic en él. Responda las preguntas que le haga y estará listo.

Antes de la instalación deberá decidir qué servidor Web desea utilizar. El instalador configurará Microsoft Internet Information Server (IIS), Apache, Xitami y Sambar Server; si utiliza un servidor Web diferente, necesitará configurarlo manualmente siguiendo las instrucciones que puede hallar en www.php.net/download-docs.php (descargue toda la documentación de PHP, ya que incluye una sección sobre instalación).

El servidor Web más popular para instalaciones locales de PHP es Apache Web, que se puede obtener en http://httpd.apache.org/. Sin embargo, en Windows es aún más fácil utilizar Microsoft Internet Information Server (IIS), ya integrado en la mayoría de versiones de Windows. Si desea comprobar que tiene IIS, busque el directorio C:\inetpub (si lo encuentra, entonces tiene IIS).

También le preguntará qué extensiones desea instalar (una extensión es un complemento de PHP, para agregar funciones). Considerando el manejo del material que se cubrirá en este libro, seleccione las extensiones MySQL, GD2 y SMTP.

RECOMENDACIÓN *Si se atora durante el proceso de instalación, puede descargar la documentación de PHP de www.php.net/download-docs.php (hay una sección especial de ayuda en el proceso de instalación y cubre una amplia gama de aspectos).*

En cualquier caso, descargar la documentación de PHP es una excelente idea (tratándose del manual oficial de PHP, puede proporcionarle respuestas en caso de necesitar métodos oficiales de soluciones PHP).

Creación de su entorno de desarrollo

Bueno, ya tiene acceso a PHP en un servidor en este momento. Para crear scripts en PHP necesitará un editor de texto —uno que le permita escribir PHP y guardarlo en archivos con extensión .php, la necesaria para generar scripts PHP (como shoppingcart.php). Existen muchos editores disponibles en diversos sistemas operativos que servirán para este propósito: vi, emacs, pico, BBEdit de Macintosh o SimpleText, Bloc de notas o WordPad de Windows.

Su editor de texto necesita guardar archivos en formato de texto simple (es decir, texto sin código de formato especial). En teoría, puede usar incluso procesadores de palabras como Microsoft Word, en tanto guarde sus páginas PHP con formato de texto simple. En Windows, muchas personas utilizan Microsoft WordPad, integrado a Windows. Si va a utilizar WordPad, asegúrese de seleccionar la opción Guardar como documento de texto cuando guarde su archivo, no como el tipo RTF (Rich Text Format o formato de texto enriquecido) predeterminado. Asimismo, WordPad tiene también el mal hábito de agregar el sufijo .txt a cualquier archivo cuya extensión no comprende y usted necesita corregir eso. Así que en caso de guardar un archivo como saladechat.php, por ejemplo, WordPad lo guardará como saladechat.php.txt, que es un problema. Para resolverlo, escriba el nombre del archivo que desea guardar entre comillas, como "saladechat.php". Esto indicará a WordPad que no toque la extensión —que resulta favorable, pues a menos que use la extensión .php para sus archivos, el servidor Web no entenderá que su PHP es en realidad un archivo PHP.

Por ejemplo, puede ver la forma en que Windows WordPad edita una página PHP en la Figura 1-4. Lo que ve es una combinación de HTML y PHP (PHP se especializa en permitirle insertar código HTML directamente en sus páginas PHP). Estamos a punto de ver en acción esta página, phphtml.php, un poco más adelante.

FIGURA 1-4 Windows WordPad

También puede utilizar un entorno de desarrollo integrado (IDE) PHP para crear páginas PHP. Los IDEs proporcionan todo tipo de herramientas que los editores de texto simples no tienen, como revisar automáticamente lo escrito por usted, para rectificar que es un archivo PHP válido; realce de sintaxis automático (cuya función es colorear elementos, como palabras clave de PHP, facilitando con ello distinguir el contenido a simple vista); así como implementación automática, con la que el IDE puede enviar las páginas PHP diseñadas a su PSI, apenas haga clic en un botón o seleccione un elemento del menú.

Ésta es una lista inicial de IDEs disponibles en línea que pueden manejar PHP. Sin embargo, observe que la mayoría de ellos representarán pagar dinero y, aunque tienen características que bien le vendría tener, no nos apoyaremos en ellas para este libro:

- **Komodo, www.activestate.com/Products/Komodo** Funciona en Linux y Windows.
- **Maguma, www.maguma.com** Funciona sólo en Windows.
- **PHPEdit, www.phpedit.com/products/PHPEdit** Gratuito, funciona sólo en Windows.
- **Zend Studio, www.zend.com/store/products/zend-studio.php** Funciona en Windows y Linux. Éste lo crean las mismas personas a cargo del desarrollo del "motor" de software Zend, en realidad ejecutado desde el núcleo de PHP mismo.

Si utiliza PHP con un PSI, también necesitará algún medio para cargar su PHP en el servidor Web. Puede emplear el mismo método que utiliza para cargar páginas HTML estándar con su PSI. Por ejemplo, a través de un programa FTP para cargar sus páginas PHP. Si utiliza un servidor basado en Unix, no olvide cambiar el permiso de su página PHP a ejecutable, como se

dijo antes en la sección "PHP en Internet". Si no tiene un programa FTP, pida recomendaciones al personal técnico de su PSI.

Creación de su primera página en PHP

Aquí es donde comienza la acción —crear una página PHP—. A partir de este punto inician sus aplicaciones Web —de libros de visitantes a aplicaciones profesionales de búsqueda en bases de datos; de juegos interactivos a carritos de compras.

Esta primera página PHP será simple. Podrá iniciar la página con marcas especiales indicando que está a punto de usar PHP y esas marcas lucen como éstas:

```
<?php
    .
    .
    .
```

Así es como se inicia una página PHP —con la marca <?php—. Esto da inicio a una sección de PHP, hasta donde compete al motor PHP del servidor.

Cuando el servidor devuelve una página PHP al navegador, el motor PHP comienza abriendo dicha página. Al encontrar la marca <?php, comienza la interpretación de que todo lo posterior a la marca es PHP. Al final de su página PHP deberá utilizar la marca de cierre, que es ?>:

```
<?php
    .
    .
    .
?>
```

Bueno, hasta ahora hemos indicado al motor PHP del servidor que deseamos insertar algo de código PHP en la página. Ahora hagamos precisamente eso, agregando esta línea:

```
<?php
  phpinfo()
?>
```

Ésta es una llamada a la *función* phpinfo( ) de PHP. Una función agrupa código PHP que puede ser ubicado mediante un nombre, como phpinfo( ). Estudiaremos las funciones de PHP en el capítulo 5; todo lo que necesita saber hasta ahora es que, cuando utiliza el nombre de la función, se ejecutará el código de esa función. En este caso, la phpinfo( ) creará una tabla de información acerca de su instalación PHP (qué hay instalado, cuándo se desarrolló, etc.) para mostrarla en la página Web que se devuelve al navegador.

Así, esta primera página PHP mostrará sólo información acerca de la instalación de PHP misma. De hecho, esta primera línea de código PHP no está del todo completa (también debe concluir toda instrucción de PHP con punto y coma (;). Entonces, agreguemos algo como esto:

```
<?php
  phpinfo();
?>
```

Bueno, ésa es nuestra primera página PHP. Guárdela como phpinfo.php y cárguela en su servidor. En tanto su servidor Web pueda manejar y ejecutar páginas Web, puede colocar esta

página en cualquier lugar donde colocaría una página HTML. Luego simplemente utilice la dirección URL de la página para verla; algo como http://www.supsi/sucuenta/phpinfo.php.

Usando un servidor Web local, como Windows IIS, significa colocar phpinfo.php en C:\Inetpub\wwwraiz o un subdirectorio de wwwraiz, para luego abrir su navegador con una dirección URL como http://localhost/phpinfo.php o http://localhost/*nombredesubdir*/phpinfo.php, si ha colocado phpinfo.php en un subdirectorio de wwwraiz.

El código acompañando este libro se almacena en las carpetas cap01 del capítulo 1, cap02 del capítulo 2 y así sucesivamente. De tal modo, encontrará phpinfo.php en la carpeta cap01.

Ejecución de su primera página en PHP

Asegúrese de que su servidor Web esté en ejecución (si utiliza IIS en Windows, éste se encuentra siempre en ejecución) y abra el archivo phpinfo.php en su navegador, como se muestra en la Figura 1-5. Felicitaciones, está ejecutando su primera página PHP.

Puede ver los resultados en esa figura —la llamada a phpinfo( ) devolvió una tabla HTML conteniendo información acerca de su instalación PHP. Lo que sucedió en realidad, es que la función phpinfo( ) devolvió el texto contenido en la tabla HTML y la tabla inserta en la página resultante, que se ve en la Figura 1-5.

Cómo corregir algunos problemas

¿Qué sucede si no funciona? ¿Qué sucede si no ve el contenido de la Figura 1-5? Por desgracia, hay muchas cosas que podrían salir mal, en especial con una instalación local de PHP. No se asuste —el problema tiene solución—. Sólo tomará un poco de tiempo.

FIGURA 1-5 Llamada a phpinfo( )

Primero, revise si su código PHP se ejecutó realmente seleccionando Ver | Código fuente en el navegador. Si ve su código PHP original allí, éste no se ejecutó por el servidor. Asegúrese de que no abrió phpinfo.php directamente en el navegador, sin haberlo ejecutado a través de su servidor Web. Por ejemplo, no dé doble clic en phpinfo.php directamente ni use las opciones de menú Archivo | Abrir del navegador, pues eso lo abriría directamente en el navegador, sin permitir que lo ejecute su servidor Web habilitado con PHP. Así, su navegador no tendrá la menor idea de cómo ejecutar una función phpinfo( ). De ese modo, escriba la dirección URL real de phpinfo.php en su navegador y ejecútela de esa manera.

Después, revise si PHP está en ejecución. Si ejecuta PHP localmente, es fácil comprobarlo: abra una ventana de símbolo del sistema y pruebe con el comando php –v. Si ve información de la versión de PHP, está en ejecución. Si su instalación de PHP no es local, utilice Telnet o SSH/SSH2 para comprobar el comando php –v en su servidor. Si no obtiene información de la versión, PHP podría estar inactivo, que sería el origen de cualquier problema con phpinfo.php.

El siguiente problema más común es que quizá PHP no se haya instalado correctamente en lo que respecta a su servidor. El inconveniente sería si obtiene una página en blanco y, al elegir el comando "ver código fuente" en el navegador Web, puede ver el código fuente de su script PHP. Esto significa que el servidor Web no pasó el script al motor PHP para su ejecución. Esta clase de remilgos son la razón por lo que las instrucciones de www.php.net resultan tan extensas. La mejor idea es repasar esas instrucciones, línea por línea, para asegurarse de haber hecho todo según se indica en ellas.

Después, asegúrese de que phpinfo.php se encuentra donde su servidor Web espera encontrarlo. En el servidor Web Apache, es el subdirectorio htdocs del directorio donde se ha instalado Apache. En el caso de IIS se trata de inetpub/wwwraiz. En Linux, puede ser /var/www/html. El directorio real puede ser diferente en servidores distintos; en un servidor PHP que manejo, el directorio correcto es /httpdocs/RAIZ; de ese modo, pregunte al soporte técnico de su PSI. Si ha cargado el archivo phpinfo.php en el directorio habitual de su PSI, donde guarda sus páginas HTML y no funciona, pregunte al soporte técnico de su PSI; a veces deben habilitar el soporte directorio por directorio. Para ello, algunos PSIs incluso exigen usar una extensión diferente para scripts de PHP 5, como .php5.

Por último, consulte la sección "Problems?" ("¿Problemas?") del manual de PHP para guía de solución de problemas. Las preguntas frecuentes (FAQ) sobre PHP en www.php.net/FAQ.php, manejan muchos de ellos. También lo hacen las preguntas frecuentes sobre instalación de PHP en www.php.net/manual/faq.installation; así que écheles un vistazo.

Combinación de HTML y PHP

Hasta ahora sólo ha creado una página Web simple que nada más incluye código PHP, específicamente, una llamada a la función phpinfo( ):

```
<?php
  phpinfo();
?>
```

Pero hay más en las páginas PHP que eso. Uno de los encantos de PHP es que puede combinar su código PHP con HTML. Eso es muy atractivo, ya que su navegador mostrará HTML y el código PHP se ejecutará en el servidor (en caso de que PHP genere algún HTML, este HTML se mostrará también en su navegador).

Por ejemplo, observe la página nueva, phphtml.php. Ésta comienza con una sección <head> estándar de HTML, como podría iniciar cualquier página HTML:

```
<html>
  <head>
    <title>
      Uso de PHP y HTML juntos
    </title>
  </head>
        .
        .
        .
```

Luego continúa con una sección <body>, conteniendo un encabezado <h1> y una porción de texto:

```
<html>
  <head>
    <title>
      Uso de PHP y HTML juntos
    </title>
  </head>

  <body>
    <h1>
      Uso de PHP y HTML juntos
    </h1>
    Aquí está su información PHP:
   <br>
        .
        .
        .
  </body>
</html>
```

Ahora, ésta es la clave (puede insertar PHP en cualquier parte de esta página y el motor de PHP del servidor Web lo ejecutará, en tanto lo contengan las marcas <?php...?>). Cuando se ejecute PHP, cualquier código HTML generado se insertará en la página, en el lugar de ese PHP. Así que, por ejemplo, si desea mostrar la tabla de configuración PHP que genera la función phpinfo( ), podría llamar a esa función de esta manera en la página Web:

```
<html>
  <head>
    <title>
      Uso de PHP y HTML juntos
    </title>
  </head>

  <body>
    <h1>
      Uso de PHP y HTML juntos
    </h1>
```

```
    Aquí está su información PHP:
   <br>
   <br>
    <?php
      phpinfo();
    ?>
  </body>
</html>
```

Ahora bien, cuando el motor PHO ejecute esta página en el servidor, el código HTML pasará por el navegador sin cambios —y el fragmento de PHP será ejecutado—. Cualquier código HTML que genere PHP se insertará también en la página enviada al navegador.

Dicha página se conoce como phphtml.php, que puede ver en la Figura 1-6.

Observe que el código HTML de la página aparece donde debe estar —así como el código HTML generado por PHP.

¿Desea dar a sus páginas un aspecto más profesional? Puede usar logotipos de PHP, disponibles en www.php.net/download-logos.php. El uso de estas imágenes descargables en sus páginas Web, es para conferirles el aspecto propio de PHP. Por ejemplo, así es como podría emplear la imagen php-med-trans-light.gif en una página llamada phpimage.php (coloque el archivo de imagen en el mismo directorio que phpimage.php en su servidor):

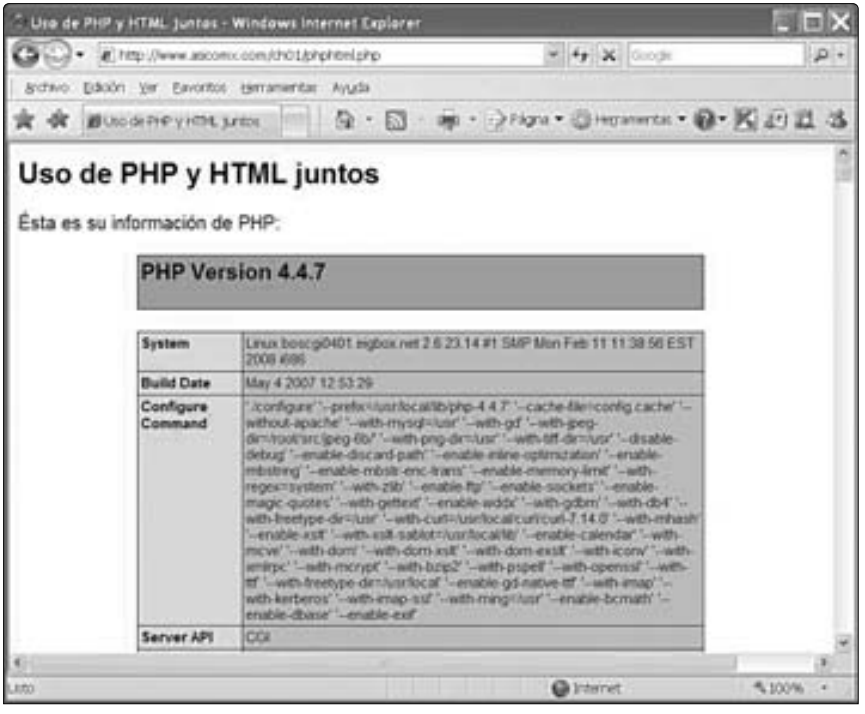

FIGURA 1-6 Combinación de PHP y HTML

```
<html>
  <head>
    <title>
      Uso de PHP y HTML juntos
    </title>
  </head>

  <body>
    <h1>
      Uso de PHP y HTML juntos
    </h1>
    Ésta es su información de PHP:
   <br>
   <br>
    <?php
      phpinfo();
    ?>
  <img src='php-med-trans-light .gif'>
  </body>
</html>
```

Y verá los resultados —el logotipo se encuentra en la esquina inferior derecha— en la Figura 1-7.

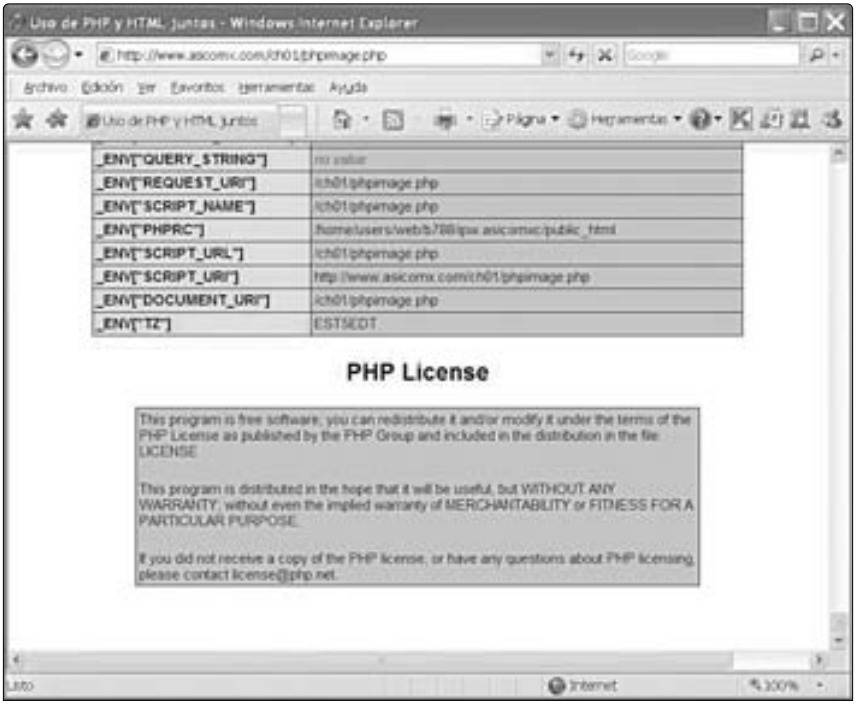

FIGURA 1-7 Uso de logotipos de PHP

Impresión de texto

De acuerdo, ¿qué tal si ahora imprimimos texto con PHP? Ya ha visto que puede colocar texto en sus archivos PHP de manera muy similar a como lo haría en una página HTML, como aquí, donde se coloca un encabezado <h1> y texto en una página:

```
<html>
  <head>
    <title>
      Uso de PHP y HTML juntos
    </title>
  </head>

  <body>
    <h1>
      Uso de PHP y HTML juntos
    </h1>
    Ésta es su información de PHP:
        .
        .
        .
```

Esto es correcto pero, evidentemente, también estático. El texto que se muestra no va a cambiar, sin importar lo que suceda en la parte de código de su página, y eso es claramente inaceptable. ¿Qué pasa si desea presentar resultados de una búsqueda en una base de datos o reservación de boletos?

Necesita insertar texto en la página manejando PHP y, para hacerlo, se emplea la instrucción echo de PHP. Por ejemplo, para "hacer eco" del texto "Bienvenido a PHP" en la página Web devuelta al navegador, se podría agregar esta línea de código a una nueva página, phpdisplaytext.php:

```
<html>
  <head>
    <title>
      Mostrando texto de PHP
    </title>
  </head>

  <body>
    <h1>
      Mostrando texto de PHP
    </h1>
    Esto es lo que PHP tiene que decir:
   <br>
   <br>
    <?php
      echo ("Bienvenido a PHP.");
    ?>
  </body>
</html>
```

Observe esta página, phpdisplaytext.php, en su navegador ahora, como se muestra en la Figura 1-8. Verá el texto que PHP insertó en la página de la figura (nada mal).

FIGURA 1-8 Cómo mostrar texto desde PHP

Puede pasar texto para mostrar la instrucción echo, empleando comillas sencillas o dobles:

```
echo "Bienvenido a PHP.";
echo 'Bienvenido a PHP.';
```

Hay una diferencia sutil entre estas dos formas de pasar texto, que verá en el siguiente capítulo. También puede pasar números directamente para "hacer eco" si lo desea, sin recurrir a comillas:

```
echo 1234.5678;
```

Y puede pasar texto entre paréntesis para "hacer eco" también:

```
echo ("Bienvenido a PHP.");
```

El paso de datos a funciones también se realiza de esta forma: coloque los datos donde desee que la función opere, entre paréntesis. Hay muchas funciones integradas en PHP; sin embargo, echo, técnicamente hablando, no es una función —es una característica integrada del lenguaje PHP.

Impresión de HTML

Nunca olvide que cuando trabaja con PHP en línea, interactúa con el usuario a través de un navegador. Eso significa que el texto enviado por usted de vuelta al navegador se interpretará como HTML, no un texto simple. Eso da también la posibilidad de utilizar HTML para dar formato a su texto, tal y como se indica aquí en una nueva muestra, phpdisplayhtml.php:

```
<html>
  <head>
    <title>
      Mostrando texto de PHP
    </title>
  </head>

  <body>
    <h1>
      Mostrando texto de PHP
    </h1>
    Esto es lo que PHP tiene que decir:
   <br>
   <br>
    <?php
      echo "<i>Bienvenido</i><br>";
      echo "<u>a</u><br>";
      echo "<b>PHP</b>.";
    ?>
  </body>
</html>
```

Puede ver esta página en la Figura 1-9 —incluyendo el formato en HTML.

FIGURA 1-9 Cómo mostrar HTML desde PHP

Así que cuando quiera pasar a la siguiente línea en su resultado, deberá insertar el comando HTML correcto, <br>, en el texto que se muestra.

Mayor poder de echo

También puede ejecutar PHP desde la línea de comandos, simplemente utilizando el comando php. Esto es lo que vería ejecutando el archivo phpdisplayhtml.php desde la línea de comandos (recuerde que % representa un símbolo de sistema genérico en este libro):

```
%php phpdisplayhtml.php
<html>
  <head>
    <title>
      Mostrando texto de PHP
    </title>
  </head>

  <body>
    <h1>
      Mostrando texto de PHP
    </h1>
    Esto es lo que PHP tiene que decir:
   <br>
   <br>
   <i>Bienvenido</i><br><u>a</u><br><b>PHP</b>.   </body>
</html>
```

Observe que el código HTML no se interpretó como HTML; se imprimió como texto simple. Si, en este caso, desea pasar a la línea siguiente, debe utilizar el *carácter de control* \n, que PHP interpretará como carácter de línea nueva (éste mostrará las tres palabras en sus propias líneas):

```
echo "Bienvenido\n";
echo "a\n";
echo "PHP.";
```

Éste es un ejemplo de los caracteres de control disponibles en PHP:

- **\n** Carácter de línea nueva
- **\r** Cambio de línea
- **\t** Tabulador
- **** Muestra una \
- **\$** Muestra un signo $
- **\"** Muestra una"
- **\0 a \777** Muestra un carácter correspondiente a un código hexadecimal (de base 8)
- **\x0 a \xFF** Muestra un carácter correspondiente a un código hexadecimal (de base 16)

Puede imprimir un carácter sensible como una comilla ("), sin decir a PHP que ha llegado al final de su texto (lo que una marca " haría en su defecto). Para esto, use \" en su lugar de esta forma:

```
echo "Él dijo, \"Me gusta el helado.\"";
```

A esto se le conoce como escapar de la comilla, para que PHP la muestre en lugar de tratarla como marca al final de una cadena de texto.

Si lo desea, puede dividir una cadena entrecomillada larga en varias líneas en su escrito, para conservar los cambios de línea —si imprime desde la línea de comando—. (Si imprime en una página Web, los cambios de línea serán pasados por alto):

```
<?php
echo "Este texto
se extiende a
múltiples
líneas
cuando
se imprime
desde
la
línea de
comando.";
?>
```

También puede separar los elementos que desea imprimir con comas, de esta forma:

```
echo "Bienvenido", "a", "PHP.";
```

Todos los elementos que imprima de esta forma aparecerán en la impresión, uno detrás del otro:

```
BienvenidoaPHP.
```

Si desea espacios entre las palabras, haga algo como esto:

```
echo "Bienvenido ", "a ", "PHP.";
```

Esto produciría

```
Bienvenido a PHP.
```

También puede ensamblar cadenas de texto en una sola utilizando un punto (.). Éste es un ejemplo:

```
echo "Bienvenido " . "a " . "PHP.";
```

En este caso, PHP toma su expresión "Bienvenido" . "a " . "PHP." y la ensambla junta (a esto se le llama concatenación) en una cadena, "Bienvenido a PHP."; luego pasa esa cadena a la instrucción echo.

Además, existe otra forma de mostrar texto: puede usar también la instrucción print de PHP con la misma sintaxis, así: print"Bienvenido a PHP.";. ¿Cuál es la diferencia entre print y echo? No mucho; print es más una función de PHP (vea el capítulo 5), de modo que produce un

valor siempre iniciado en 1. Para la mayoría de los fines, echo y print funcionan igual en PHP, de modo que el uso de cada una dependerá de usted.

Uso de documentos "here" de PHP

De hecho, hay otra forma de mostrar texto que debe conocer y consiste en usar documentos "here" de PHP. Un documento *here* es tan sólo texto inserto directamente en una página PHP, entre dos instancias del mismo token; ese token es una palabra, como END. Luego puede mostrar el texto en un documento here utilizando la sintaxis echo <<<*TOKEN*, donde *TOKEN* es la palabra que inicia y termina el documento here.

Éste es un ejemplo, phphere.php:

```
<html>
  <head>
    <title>
      Mostrando texto de PHP
    </title>
  </head>

  <body>
    <h1>
      Mostrando texto de PHP
    </h1>
    Esto es lo que PHP tiene que decir:
   <br>
   <br>
    <?php
echo <<<END
Este ejemplo usa
sintaxis del "documento here" para mostrar todo
el texto hasta llegar al token final.
END;
    ?>
  </body>
</html>
```

Y puede ver lo que esto produce en un navegador en la Figura 1-10. Como lo muestra la figura, el texto del documento here se mostró en el navegador.

De ninguna manera es necesario usar el token END; éste es el mismo documento here con el token FINISH:

```
echo <<<FINISH
Este ejemplo usa
sintaxis del "documento here" para mostrar todo
el texto hasta llegar al token final.
FINISH;
```

FIGURA 1-10 Cómo mostrar un documento here en PHP

PHP en la línea de comandos

Este libro se centra en el uso de PHP en servidores Web y visualizar los resultados en navegadores Web, pero también se puede utilizar PHP desde la línea de comandos. Esta vertiente puede ser muy buena para desarrollo Web —el uso de PHP desde la línea de comandos puede ubicar errores ocasionando que los servidores Web se detengan y también ahorra el tiempo empleado para cargar su código PHP en el servidor. Cuando ejecute instrucciones PHP específicas para Web en la línea de comandos, podrá ver el código HTML de su página y entonces depurarla con mayor facilidad.

PHP es un lenguaje interpretado; el motor PHP lee el texto de una página PHP y lo interpreta, ejecutando esa secuencia PHP de inmediato, línea por línea. La versión de PHP en la línea de comandos se conoce como intérprete de la línea de comandos o CLI, que simplemente asigna el nombre php. El motor de PHP en línea suele llamarse php-cgi ("cgi" corresponde a Common Gateway Interface, interfaz de puerta de acceso común), término estándar para designar programas en línea interactuando con servidores Web). De hecho, al ejecutar el comando **php –v** en la línea de comandos, puede ver que ejecuta el CLI, porque así lo dice:

```
%php -v
PHP 5.2.5 (cli) (built: Nov  8 2007 23:18:51)
Copyright (c) 1997-2007 The PHP Group
Zend Engine v2.2.0, Copyright (c) 1998-2007 Zend Technologies
```

Es fácil ejecutar archivos PHP desde la línea de comandos. Por ejemplo, suponga que tiene un archivo PHP llamado echoer.php:

```
<?php
      echo "Hola, aquí PHP.";
?>
```

La secuencia <?php...?> sigue siendo obligatoria, aun cuando sólo se utiliza PHP directo. Puede ejecutar este archivo desde la línea de comandos de esta forma:

```
%php echoer.php
Hola, aquí PHP.";
%
```

Y verá el resultado —el CLI imprime el mensaje enviado por el archivo. Cualquier texto distinto de PHP fuera de la sección <?php...?> se interpreta como HTML y simplemente se reproduce en la presentación final.

Ejecutar PHP de esta forma supone que CLI está en la ruta de acceso de su computadora, que así debe ser mientras tenga PHP instalado. Si no funciona, puede especificar la ubicación exacta de php. En Unix o Linux podría tener un aspecto como éste:

```
$/user/local/bin/php echoer.php
```

Y algo como esto en Windows:

```
C:\>C:\php\php echoer.php
```

El CLI tiene muchas opciones para la línea de comandos, que usted puede usar para personalizar su operación. De hecho, php le da información de todas estas opciones escribiendo **php –h** para desplegar esta lista:

```
%php -h
Usage: php [options] [-f] <file> [--] [args...]
       php [options] -r <code> [--] [args...]
       php [options] [-B <begin_code>] -R <code> [-E <end_code>] [--] [args...]
       php [options] [-B <begin_code>] -F <file> [-E <end_code>] [--] [args...]
       php [options] -- [args...]
       php [options] -a

  -a               Run interactively
  -c <path>|<file> Look for php.ini file in this directory
  -n               No php.ini file will be used
  -d foo[=bar]     Define INI entry foo with value 'bar'
  -e               Generate extended information for debugger/profiler
  -f <file>        Parse and execute <file>.
  -h               This help
  -i               PHP information
  -l               Syntax check only (lint)
  -m               Show compiled in modules
  -r <code>        Run PHP <code> without using script tags <?..?>
  -B <begin_code>  Run PHP <begin_code> before processing input lines
  -R <code>        Run PHP <code> for every input line
  -F <file>        Parse and execute <file> for every input line
  -E <end_code>    Run PHP <end_code> after processing all input lines
```

```
  -H              Hide any passed arguments from external tools.
  -s              Display colour syntax highlighted source.
  -v              Version number
  -w              Display source with stripped comments and whitespace.
  -z <file>       Load Zend extension <file>.

  args...         Arguments passed to script. Use -- args when first argument
                  starts with - or script is read from stdin

  --ini           Show configuration file names

  --rf <name>     Show information about function <name>.
  --rc <name>     Show information about class <name>.
  --re <name>     Show information about extension <name>.
  --ri <name>     Show configuration for extension <name>.
```

Muchas de estas opciones son útiles. Por ejemplo, php –a permite ejecutar el CLI en modo interactivo, haciendo posible la ejecución de PHP con sólo teclearlo. Éste es un ejemplo —la respuesta del CLI aparece en negritas:

```
%php –a
Interactive mode enabled

<?php
echo "Hola de PHP.";
Hola de PHP.
```

Podría continuar escribiendo en PHP y el CLI se mantendría en ejecución:

```
%php -a
Interactive mode enabled

<?php
echo "Hello from PHP.";
Hello from PHP.
echo "Hola de nuevo.";
Hola de nuevo.
```

El comando php –i brinda mucha información de phpinfo( ) acerca de la forma en que se instaló PHP en la computadora:

```
%php –i
phpinfo()
PHP Version => 5.2.0

System => Windows NT DM8400 5.1 build 2600
Build Date => Nov  2 2006 11:50:55
Configure Command => cscript /nologo configure.js  "--enable-snapshot-build" "--
with-gd=shared"
Server API => Command Line Interface
Virtual Directory Support => enabled
Configuration File (php.ini) Path => C:\Program Files\PHP\php.ini
PHP API => 20041225
PHP Extension => 20060613
Zend Extension => 220060519
Debug Build => no
```

```
Thread Safety => enabled
Zend Memory Manager => enabled
IPv6 Support => enabled
Registered PHP Streams => php, file, data, http, ftp, compress.zlib
Registered Stream Socket Transports => tcp, udp
Registered Stream Filters => convert.iconv.*, string.rot13, string.toupper, stri
ng.tolower, string.strip_tags, convert.*, consumed, zlib.*

This program makes use of the Zend Scripting Language Engine:
        .
        .
        .
```

¿Desea ver el código fuente de su archivo PHP utilizando realce de sintaxis, donde las palabras clave de PHP son color verde y los elementos datos color rojo? Simplemente utilice php –s. Por ejemplo, si desea obtener una versión con sintaxis realzada de echoer.php y almacenarla en un documento HTML listo para verlo en su navegador, echoer.html, podría ejecutar este comando:

```
%php –s echoer.php > echoer.html
```

Éste sería el contenido final del archivo echoer.html:

```
<code><span style="color: #000000">
<span style="color: #0000BB">&lt;?php
<br /></span><span style="color: #007700">echo </span><span style="color: #DD00
00">"Hello, PHP here."</span><span style="color: #007700">;
<br /></span><span style="color: #0000BB">?&gt;
<br /></span>
</span>
</code>
```

También puede utilizar el CLI para revisar la sintaxis de un archivo PHP, para ver si tiene errores de PHP. Por ejemplo, suponga que omitiera la comilla de apertura en esta instrucción en echoer.php:

```
<?php
    echo Hola, aquí PHP.";
?>
```

Esto es lo que obtendría al ejecutar la instrucción a través del CLI con la opción –l:

```
%php -l echoer.php
PHP Parse error:  syntax error, unexpected T_STRING, expecting ',' or ';' in echoer.
php on line 2
Errors parsing echoer.php
```

Es el mismo error que obtendría de PHP en línea (php-cgi).

Debe saber que el resultado producido por CLI difiere de php-cgi; por ejemplo, en CLI no se incluyen encabezados HTTP estándar que imprime php-cgi al devolver el resultado al navegador. Y el CLI imprime los mensajes de error en texto simple, no en código HTML que dará php-cgi.

También vale la pena observar que en Linux y Unix, pueden ejecutarse scripts PHP con sólo escribir el nombre del script en la línea de comandos, si indica dónde encontrar PHP con una línea que inicia con los caracteres #!:

```
#! /usr/bin/php
<?php
    echo "Hola de PHP.";
?>
```

Adición de comentarios a código de PHP

Al igual que virtualmente todos los lenguajes de programación, se pueden agregar comentarios al código PHP. Los comentarios son anotaciones agregadas por usted para hacer su código legible y omitido por PHP. Es importante agregar comentarios si el código es extenso, ya que es posible que usted —o alguien más— revise el código tiempo después y no tenga la menor idea de para qué sirve. Puede reconstruir laboriosamente lo que sucede, pero ¿por qué no agregar comentarios que faciliten el proceso? De esa forma puede descifrar su viejo código (o de alguien más) con facilidad.

Existen tres tipos de comentarios en PHP. El primer tipo de comentario permite crear comentarios en diferentes líneas, que comiencen con /* y terminen con */, de esta forma:

```
<?php
/* Comience mostrando un
   mensaje que explique al
   usuario lo que hacemos */

    echo “Bienvenido a PHP.”;
?>
```

Eso es genial si tiene un comentario extenso que ocupará múltiples líneas para mostrarse. Sin embargo, algo que vuelve loco a PHP es que el usuario anide comentarios, que comienzan con /* y terminan con */, de esta forma:

```
<?php
/* Comience mostrando un mensaje
   /* en español */
   mensaje que explique al
   usuario lo que hacemos */

    echo "Bienvenido a PHP.";
?>
```

PHP busca */ para marcar el final del comentario y si anida comentarios dentro de otros, encontrará */ antes del final real del comentario.

Los otros dos tipos de comentarios son de una línea. Este tipo de comentarios sólo se extienden a una línea, no como los del tipo multilíneas /*...*/. El primer tipo de comentario de una línea comienza con //:

```
<?php
// Muestre un mensaje de bienvenida.

    echo “Bienvenido a PHP.”;
?>
```

Y el siguiente tipo de comentario en una línea funciona de la misma forma, salvo que comienza con el signo #:

```
<?php
// Muestre un mensaje de bienvenida.
# Muestre un mensaje de bienvenida.

    echo "Bienvenido a PHP.";
?>
```

De hecho, PHP pasa por alto todo después de la doble barra // o el signo #; de ese modo también puede colocar comentarios de un renglón al final de las líneas, después de código PHP válido como éste:

```
<?php
// Muestre un mensaje de bienvenida.
# Muestre un mensaje de bienvenida.

    echo "Bienvenido a PHP.";  // Muestre un mensaje de bienvenida.
    echo "¡Bienvenido nuevamente!";  # Muestre otro mensaje de bienvenida.
?>
```

Puede utilizar, incluso, comentarios en una línea para hacer bloques de comentarios en su código PHP, similares a comentarios multilíneas, como éste:

```
<?php
// Comience mostrando un
// mensaje que explique al
// usuario lo que hacemos

    echo "Bienvenido a PHP.";
?>
```

¿Desea algo que destaque realmente, en términos de comentarios? Pruebe algo como esto:

```
<?php
####################################
#      Comience mostrando un       #
#      mensaje que explique al     #
#      usuario lo que hacemos      #
####################################

    echo "Bienvenido a PHP.";
?>
```

No es necesariamente bonito, pero capta su atención.

Probablemente descubrirá que utiliza más comentarios en una línea que en múltiples líneas. Los comentarios en una línea son más fáciles de escribir, pues no tiene que llevar el control del punto final del comentario. Sin embargo, los comentarios en múltiples líneas son útiles también en PHP, y los verá con frecuencia en la definición de funciones de PHP, explicando qué datos pasa a la función y cuáles recibe de vuelta.

Trabajo con variables

Hasta este punto, el texto producido por nuestros scripts ha sido estático. En este caso, por ejemplo, el script muestra sólo el texto"Bienvenido a PHP.":

```
<?php
    echo "Bienvenido a PHP.";
?>
```

Eso es texto bastante estático. Puede hacer añadiduras con el operador de adición de PHP, +, para agregar números de esta forma:

```
<html>
 <head>
    <title>Ésta es la respuesta</title>
  </head>
  <body>
    <h1>
      La respuesta
    <h1>
    <?php>
      echo "Ésta es la respuesta: " , 1 + 5 + 8 , ".";
   ?>
  <body>
</html>
```

La primera vez que ejecuta esto se muestra el mensaje"Ésta es la respuesta: 14." Y la segunda vez que se ejecuta, vuelve a mostrar el mensaje"Ésta es la respuesta: 14."Así que es lo mismo —se vuelve a obtener texto estático.

Esto está bien hasta donde llega, pero no muy lejos. Si sólo deseara texto estático, podría quedarse con HTML. Es ahí donde las variables entran en acción. Como en cualquier otro lenguaje de programación, las variables de PHP son ubicaciones en la memoria con nombres asignados alojando datos. Suponga por ejemplo que deseara llevar el control del número de usuarios de su sitio en tres ciudades —París, Londres y Acapulco—, pero el número de usuarios en cada lugar no se conoce hasta ejecutar el programa. Las variables llegan al rescate.

Con las variables, usted podría almacenar el número de usuarios en la variable de cada ciudad al momento de la ejecución. En PHP, los nombres de variables comienzan con un signo $, seguido de un nombre —y ese nombre debe comenzar con una letra o línea de subrayado, no con un número. Por ejemplo, para almacenar el número de usuarios en París, podría tener una variable llamada $parís; el número de usuarios en Londres y Acapulco podría estar en las variables $londres y $acapulco. Podría colocar datos en esas variables al momento de la ejecución y sumar los tres elementos de datos, también al momento de la ejecución. La suma se vería así:

```
<html>
 <head>
    <title>Ésta es la respuesta</title>
  </head>
  <body>
    <h1>
      La respuesta
    <h1>
```

```
    <?php>
      echo "Ésta es la respuesta: " , $parís + $londres + $acapulco , ".";
   ?>
  <body>
</html>
```

Al momento de la ejecución, el valor dentro de cada variable se sustituye por esa variable; de modo que si $parís = 1, $londres = 5 y $acapulco = 8, PHP podría ver la línea anterior de código PHP de esta forma:

```
<?php
  echo "Ésta es la respuesta: " , 1 + 5 + 8 , ".";
?>
```

Y entonces obtiene "Ésta es la respuesta: 14." (la misma que antes). Pero ahora supongamos que las tres variables alojaran 2, 4 y 6 al momento de la ejecución, con lo que obtendría

```
<?php
  echo "Ésta es la respuesta: " , 2 + 4 + 6 , ".";
?>
```

Entonces obtiene "Ésta es la respuesta: 12."—lo que significa que nuestro mensaje ya no es estático—. Ésa es la razón por la que se llaman variables —los datos que contienen pueden variar—. Ése es el primer paso para trabajar con datos en PHP: usar variables para almacenar datos. Así que, ¿cómo almacenar exactamente datos en variables?

Almacenaje de datos en variables

En PHP, como otros lenguajes, puede asignar datos a variables. PHP le permite almacenar números y texto en dichas variables, como en estos ejemplos:

```
$pi = 3.1415926535;
$número_de_Saturnos = 1;
$nombre = "Klostix Díaz";
$pez = "bacalao";
```

A diferencia de otros lenguajes en línea, como Java, PHP no insiste en que usted cree diferentes tipos de variables para cada dato. Ésta es la forma en que tales variables podrían crearse en Java:

```
double pi = 3.1415926535;
int número_de_Saturnos = 1;
String nombre = "Klostix Díaz";
String pez = "bacalao";
```

PHP puede almacenar todo tipo de datos en variables y no tiene que crear diferentes tipos de variables para almacenar esos datos, que facilita mucho las cosas. Por otra parte, en virtud de la arquitectura interna de las computadoras, PHP almacena internamente sus datos en diferentes formatos y a veces es importante saber cuáles son esos formatos internos (hablaremos más de esto después).

Observe el uso del signo =, se utiliza para asignar datos a variables. El signo = es el *operador de asignación* en PHP, que le permite encuadrar datos a variables. (De hecho, existen otros operadores de asignación, como se verá en el siguiente capítulo; pero el signo = es el principal.) Por ejemplo, para inicializar el número de hamburguesas con queso de su código en 1, usted haría esto:

```
<?php
    echo "Inicializar el número de hamburguesas con queso en 1.<br>";
    $hamburguesasconqueso = 1;
    .
    .
    .
?>
```

Luego podría transmitir el número actual de hamburguesas con queso al navegador:

```
<?php
    echo "Inicializar el número de hamburguesas con queso en 1.<br>";
    $hamburguesasconqueso = 1;
    echo "Número actual de hamburguesas con queso: ", $hamburguesasconqueso, "<br>";
    .
    .
    .
?>
```

Ahora podría cambiar el número almacenado en la variable $hamburguesasconqueso conforme se sumaran más hamburguesas en línea. Por ejemplo, si tiene tres hamburguesas más, podría utilizar el operador + de PHP para sumarlas al número actual almacenado en $hamburguesasconqueso de esta forma:

```
<?php
    echo "Inicializar el número de hamburguesas con queso en 1.<br>";
    $hamburguesasconqueso = 1;
    echo "Número actual de hamburguesas con queso: ", $hamburguesasconqueso, "<br>";
    echo "Se suman 3 hamburguesas con queso más.<br>";
    $hamburguesasconqueso = $hamburguesasconqueso + 3;
    .
    .
    .
?>
```

Ahora puede mostrar el nuevo número de hamburguesas con queso en este ejemplo, phpvariables.php, de la siguiente forma:

```
<html>
    <head>
        <title>
            Almacenar datos en variables
        </title>
    </head>
    <body>
        <h1>
            Almacenar datos en variables
        </h1>
```

```
        <?php
            echo "Inicializar el número de hamburguesas con queso a 1.<br>";
            $hamburguesasconqueso = 1;
            echo "Número actual de hamburguesas con queso: ", $hamburguesasconqueso, "<br>";
            echo "Se suman 3 hamburguesas con queso más.<br>";
            $hamburguesasconqueso = $hamburguesasconqueso + 3;
            echo "Número de hamburguesas con queso ahora: ", $hamburguesasconqueso, "<br>";
        ?>
    </body>
</html>
```

Puede ver phpvariables.php en la Figura 1-11 donde, como apreciará, el valor almacenado en $hamburguesasconqueso cambió exitosamente.

En algunos lenguajes, como Java, tienen que declararse variables, dando su tipo y nombre, antes de usarlas, como

```
double pi;
int número_de_Saturnos = 1;
```

En PHP no tiene que declarar variables antes de usarlas —otra forma en que PHP ahorra tiempo y esfuerzo—. Todo lo que debe hacer antes de usar una variable en PHP, consiste en asignar valor —cuando lo haga, PHP creará esa variable por usted y almacenará sus datos

FIGURA 1-11 Cómo poner las variables a trabajar en PHP

en ella—. Sin embargo, eso significa que debe asignar datos a una variable antes de leer el valor de la misma. Por ejemplo, si intentara leer el número de hamburguesas con queso sin antes inicializarlo,

```
<?php
    echo "Número actual de hamburguesas con queso: ", $hamburguesasconqueso, "<br>";
    echo "Se suman 3 hamburguesas con queso más.<br>";
    $hamburguesasconqueso = $hamburguesasconqueso + 3;
    echo "Número de hamburguesas con queso ahora: ", $hamburguesasconqueso, "<br>";
?>
```

obtendría el error en la Figura 1-12.

Éste es el mensaje de error:

```
PHP Notice: Undefined variable: hamburguesasconqueso in C:\Inetpub\wwwroot\ch01\
phpvariables.php on line 14 PHP Notice: Undefined variable: hamburguesasconqueso in
C:\Inetpub\wwwroot\ch01\phpvariables.php on line 16
```

¿Desea anular una variable? Resulta difícil pensar en aquellas ocasiones que usted podría ya no desear el uso de una variable, pero en realidad puede anular variables en PHP. Tan sólo use un código como éste:

```
unset ($hamburguesasconqueso);
```

Después de ejecutar esta instrucción de PHP, la variable $hamburguesasconqueso ya no existirá ni obtendría un error, si intentara leer el valor que ésta contenía.

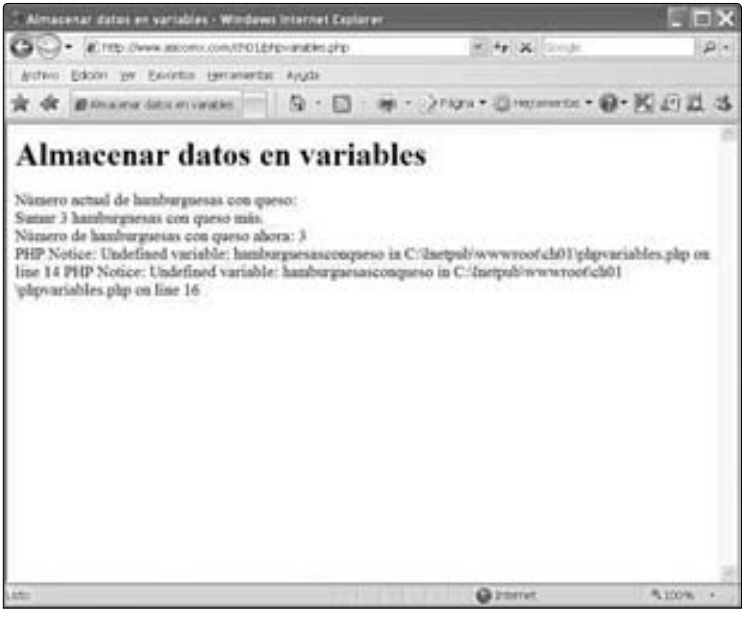

FIGURA 1-12 Cuando olvida inicializar una variable en PHP

Interpolación de cadenas

Puede mostrar el valor de una variable como ésta, desde luego:

```
<?php>
    $hamburguesasconqueso = 1;
    echo "Número actual de hamburguesas con queso: ", $hamburguesasconqueso, "<br>";
?>
```

En este caso, usted ha colocado explícitamente $hamburguesasconqueso en la lista de elementos cuyos valores desea mostrar. Por otra parte, hay una forma abreviada de hacer esto en PHP que debe conocer, llamada *interpolación de cadenas*.

Cuando usted usa la interpolación, sólo tiene que colocar la variable cuyo valor desea insertar, dentro de una cadena de texto con comillas dobles (no con comillas sencillas). Por ejemplo, podría convertir el ejemplo phpvariables.php anterior para usar la interpolación de cadenas, en un nuevo ejemplo llamado phpinterpolation.php, de esta forma:

```
<html>
    <head>
        <title>
            Uso de la interpolación de cadenas
        </title>
    </head>
    <body>
        <h1>
            Uso de la interpolación de cadenas
        </h1>
        <?php
            echo "Inicializar el número de hamburguesas con queso en 1.<br>";
            $hamburguesasconqueso = 1;
            echo "Número actual de hamburguesas con queso: $hamburguesasconqueso <br>";
            echo " Se suman 3 hamburguesas con queso más.<br>";
            $hamburguesasconqueso = $hamburguesasconqueso + 3;
            echo "Número de hamburguesas con queso ahora: $hamburguesasconqueso <br>";
        ?>
    </body>
</html>
```

Observe cómo funciona esto: PHP ve el nombre de una variable, $hamburguesasconqueso, dentro de una cadena de texto con comillas dobles:

```
echo "Número actual de hamburguesas con queso: $hamburguesasconqueso <br>";
```

Eso significa que PHP colocará de inmediato el valor alojado en $hamburguesasconqueso en la cadena, de esta forma:

```
echo "Número actual de hamburguesas con queso: 1 <br>";
```

Es así como funciona la interpolación de cadenas; PHP sustituirá el valor de una variable por aquella en una cadena de texto con comillas dobles. Puede ver phpinterpolation.php en acción en la Figura 1-13.

La interpolación de cadenas es un atajo rápido que permite colocar el valor de una variable en una cadena con comillas dobles (no con comillas sencillas), pero hay algo que debe saber

FIGURA 1-13 Uso de la interpolación de cadenas en PHP

aquí: necesita rodear el nombre de la variable con espacios o puntuación simple. Si coloca el nombre de la variable directamente junto al texto, podría confundir a PHP. Por ejemplo, si la variable $tipo aloja el texto "balon" y desea insertar el texto "baloncesto" en texto con comillas dobles, podría verse tentado a utilizar la expresión $tipocesto. Y eso no funcionará; verá el aviso

```
PHP Notice:  Undefined variable:  tipocesto en C:\php\wrong.php on line 8
```

La solución consiste en encerrar el nombre de la variable entre llaves, como éstos: {$tipo}, creando la expresión ${tipo}cesto, como se puede apreciar en phpadjoininginterpolation.php:

```
<html>
    <head>
        <title>
            Uso de interpolación de variables con palabras adjuntas
        </title>
    </head>
    <body>
        <h1>
            Uso de interpolación de variables con palabras adjuntas
        </h1>
        <?php
            $tipo = "balon";
            echo "El nombre del juego es {$tipo}cesto.<br>";
        ?>
    </body>
</html>
```

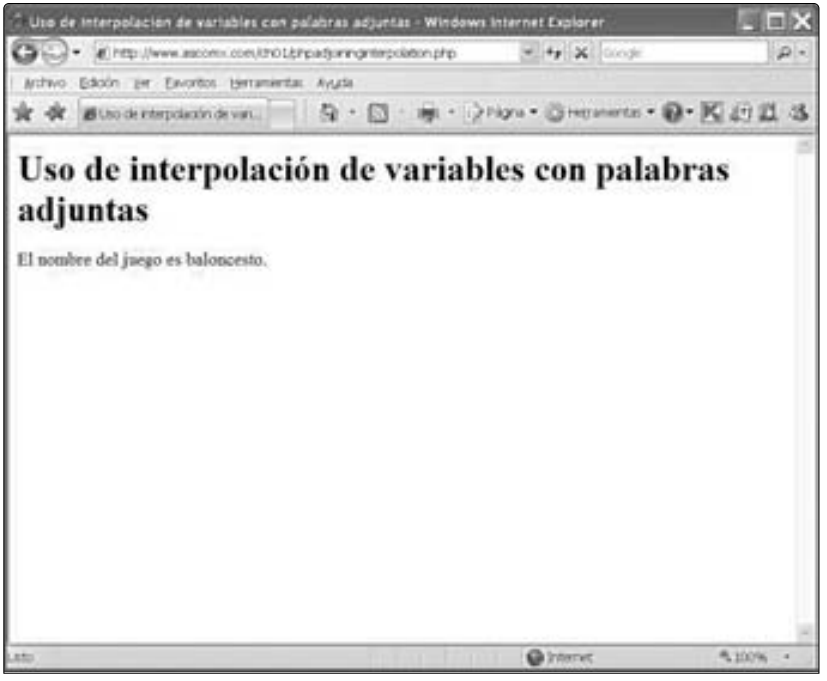

FIGURA 1-14 Uso de la interpolación de cadenas con texto adjunto en PHP

Ahora funciona, como se ve en la Figura 1-14.

La interpolación de cadenas es útil, pero podría no utilizarla todo el tiempo. Por ejemplo, ¿qué sucede si tiene la cadena de texto:"Conseguimos muy buen precio en la comida —fue $barato incluso para Jessica."?). Así como se plantea, confundirá a PHP, que intentará hallar una variable llamada $barato y dará un error cuando no pueda encontrarla.

Existen dos soluciones a este problema; primero, podría usar comillas sencillas en vez de dobles, así:'Conseguimos muy buen precio en la comida —fue $barato incluso para Jessica.'. Cuando se utilizan comillas sencillas, PHP no aplica la interpolación de cadenas.

La segunda forma de evitar interpolación cuando no la desea —pero aún pretende usar comillas dobles— consiste en anular el nombre de la variable colocando antes del signo $ una diagonal inversa (\), que da como resultado:"Conseguimos muy buen precio en la comida —fue \$barato incluso para Jessica.". PHP verá el símbolo"\$"y lo convertirá en un símbolo $ inocuo, como debe ser aquí, no el inicio de un nombre de variable.

Eso es interpolación de cadenas —si desea colocar el valor de una variable en una cadena con comillas dobles, ésta es la herramienta para usted.

Creación de variables alternas

Mientras analizamos las variables, también vale la pena analizar variables alternas. PHP permite éstas para alojar el *nombre* de otra variable.

Así es como funciona. Podría tener una variable llamada $hamburguesasconqueso, a la que podría dar un valor de 1:

```
<?php
  $hamburguesasconqueso = 1;
        .
        .
        .
?>
```

Luego podría crear una nueva variable, $tipodehamburguesa, que aloje el nombre de la primera variable,"hamburguesasconqueso":

```
<?php
  $hamburguesasconqueso = 1;
  $tipo de hamburguesa = "hamburguesasconqueso";
        .
        .
        .
?>
```

Sabe que puede mostrar el valor en $hamburguesasconqueso de esta forma:

```
<?php
  $hamburguesasconqueso = 1;
  $tipo de hamburguesa = "hamburguesasconqueso";
  echo "Número de hamburguesas con queso: ", $hamburguesasconqueso, "<br>";
        .
        .
        .
?>
```

Resulta que también puede tener acceso al valor en $hamburguesasconqueso mediante la variable alterna, $tipodehamburguesa, de esta forma:

```
<?php
  $hamburguesasconqueso = 1;
  $tipo de hamburguesa = "hamburguesasconqueso";
  echo "Número de hamburguesas con queso: ", $hamburguesasconqueso, "<br>";
  echo "Ese número de nuevo: ", $$tipodehamburguesa, "<br>";
        .
        .
        .
?>
```

Observe la sintaxis —como $tipodehamburguesa contiene el texto"hamburguesasconqueso", la expresión $$tipodehamburguesa es equivalente a $hamburguesasconqueso.

¿Desea utilizar variables alternas con interpolación de cadenas? PHP tendrá problemas con una expresión como $$tipodehamburguesa entre comillas dobles; de modo que la forma de corregir eso consiste en utilizar phpvariablevariables.php:

```
<html>
    <head>
        <title>
            Uso de variables alternas
        </title>
    </head>
    <body>
        <h1>
            Uso de variables alternas
        </h1>
        <?php
         $hamburguesasconqueso = 1;
         $tipodehamburguesa = "hamburguesasconqueso";
         echo "Número de hamburguesas con queso: ", $hamburguesasconqueso , "<br>";
         echo "Ese número de nuevo: ", $$tipodehamburguesa, "<br>";
         echo "Una vez más: ${$tipodehamburguesa} <br>";
        ?>
    </body>
</html>
```

Si no hubiera usado los llaves en este ejemplo, habría obtenido el siguiente resultado:

```
Número de hamburguesas con queso: 1
Ese número de nuevo: 1
Una vez más: $hamburguesasconqueso
```

Ahora mismo, las variables alternas podrían parecer poco más que una curiosidad, pero tienen sus usos cuando se trabaja con ciclos y matrices, que se analizan en el capítulo 3.

Creación de constantes

A veces no desea que un elemento de datos sea variable. Por ejemplo, el valor de pi, 3.1415926535, no debe cambiar. Si creara una variable llamada $pi, algo en su código podría asignar un nuevo valor a $pi por error. Lo que debe hacer es crear una *constante*, cuyo valor no se pueda modificar.

En PHP, una constante se crea con la función define, pasando a define el nombre de la constante que desea crear y el valor que quiere asignarle, como define ("PI", 3.1415926535). Eso crea una constante llamada PI —distinguiendo entre mayúsculas y minúsculas—: PI no es lo mismo que pi (si quiere que una constante no distinga entre mayúsculas y minúsculas, puede pasar un valor TRUE como éste: define ("PI", 3.1415926535, TRUE)). Observe que esto simplemente crea una constante llamada pi —no se utiliza un signo $ frente al nombre, porque la convertiría en variable.

Así es como funciona la creación de una constante en un archivo de ejemplo phpconstant.php:

```
<html>
    <head>
        <title>
            Definición de constantes
        </title>
    </head>
```

```
    <body>
        <h1>
            Definición de constantes
        </h1>
        <?php
         define ("PI", 3.1415926535);
         echo "El valor de pi es ", PI, "<br>";
        ?>
    </body>
</html>
```

Y puede ver los resultados de la Figura 1-15, donde la constante se creó en realidad.

Así, ésa es la idea tras las constantes (si intenta alterar el valor de esta constante, pi (por ejemplo, pi = 3.14), PHP no lo aceptará, ni siquiera iniciará el script).

Esto es algo de tomar en cuenta: como a las constantes no se les antepone un prefijo $, PHP se puede confundir si utiliza una constante con el mismo nombre de una de las palabras clave reservadas a PHP. Estas palabras clave aparecen en la Tabla 1-1.

Asimismo, existen varias constantes predefinidas disponibles para sus scripts. Utilizaremos estas constantes conforme las necesitemos; éste es un ejemplo:

- **_LINE_** El número de línea actual del archivo
- **_FILE_** La ruta completa y el nombre del archivo
- **_FUNCTION_** El nombre de la función
- **_CLASS_** El nombre de la clase
- **_METHOD_** Nombre del método de clase
- **PHP_VERSION** Versión de PHP

CLASS	_FILE_	_FUNCTION_	_LINE_
METHOD	and	array	as
break	case	cfunction	class
const	continue	declare	default
die	do	echo	else
elseif	empty	enddeclare	endfor
endforeach	endif	endswitch	endwhile
eval	exception	exit	extends
for	foreach	function	global
if	include	include_once	isset
list	new	old_function	or
php_user_filter	print	require	require_once
return	static	switch	unset
use	var	while	xor

TABLA 1-1 Palabras clave de PHP

FIGURA 1-15 Definición de constantes

- **PHP_OS** El sistema operativo
- **DEFAULT_INCLUDE_PATH** Donde PHP buscará lo que necesita

Por ejemplo, el uso de echo_LINE_ en la ubicación específica de un script mostrará la línea actual en ejecución.

Entendamos los tipos de datos internos de PHP

PHP le hace un gran favor al permitirle almacenar datos sin especificar el tipo. En otros lenguajes necesita establecer el formato exacto de los datos de cada variable, pero PHP se encarga de eso por usted.

Sin embargo, a veces tiene que conocer los tipos de datos internos usados por PHP, como verá a continuación. Éstos son esos tipos internos:

- **boolean** Aloja valores verdaderos/falsos
- **integer** Aloja números como -1, 0, 5, etcétera.
- **float** Aloja números de punto flotante (“dobles”), como 3.14159 o 2.7128
- **string** Aloja texto como“Bienvenido a PHP.”
- **array** Aloja matrices de elementos de datos
- **object** Aloja objetos de programación
- **resource** Aloja un recurso de datos
- **NULL** Aloja un valor NULL (nulo)

PHP suele manejar los tipos de sus datos automáticamente. Por ejemplo, esta instrucción crea una variable alojando una cadena:

```
$datos = "Son muchas hamburguesas con queso las que tiene ahí.";
```

Esta instrucción crea una variable que aloja un número flotante internamente:

```
$datos = 123.456;
```

Éste es un ejemplo que crea una variable booleana (TRUE/FALSE, VERDADERO/FALSO):

```
$datos = TRUE;
```

Todo eso está bien; el problema se presenta cuando mezcla tipos de datos. Por ejemplo, si comienza con una variable simple inicializada en "0"

```
<?php
  $variable = 0;                  // $variable es una cadena inicializada en "0"
        .
        .
        .
?>
```

entonces eso crea una variable que contiene la cadena "0". Ahora bien, ¿qué sucede si suma 1 al valor de esa variable?:

```
<?php
  $variable = 0;                  // $variable es una cadena inicializada a "0"
  $variable = $variable + 1;      // $variable es ahora un entero inicializado a 1
        .
        .
        .
?>
```

En este caso, PHP hace lo mejor que puede; sumar el número 1 a la cadena "0" deja $variable alojando el entero 1. ¿Qué sucede si suma un valor de punto flotante?

```
<?php
  $variable = 0;               // $variable es una cadena inicializada a "0"
  $variable = $variable + 1;  // $variable es ahora un entero inicializado a 1
  $variable = $variable + 1.2; // $variable es ahora un número flotante inicializado a 2.2
        .
        .
        .
?>
```

Ahora $variable aloja un valor flotante de 2.2. ¿Qué pasa si suma 3 + "8 hamburguesas con queso"? De nueva cuenta, PHP hará su mejor esfuerzo, sumando 3 + 8 y dejando el resultado, un valor entero de 11, como resultado:

```
<?php
  $variable = 0;                          // $variable es una cadena inicializada a "0"
  $variable = $variable + 1;              // $variable es ahora un entero inicializado a 1
  $variable = $variable + 1.2;            // $variable es ahora un número flotante
                                             inicializado a 2.2
  $variable = 3 + "8 hamburguesas con queso"; // $variable es un entero inicializado a 11
?>
```

Debe evitar la confianza en estas reglas. Están integradas en PHP, pero confiar en ellas podría darle un resultado equivocado. Si desea convertir un valor de un tipo de datos a otro, puede usar explícitamente un tipo de datos *combinado*. Los tipos de datos combinados se ponen entre paréntesis; por ejemplo:

```
$entero = (integer) $datos;
```

convierte explícitamente el valor de $datos a entero. El siguiente convierte el valor en $datos en un valor $flotante:

```
$flotante = (float) $datos;
```

Cuando se hace la conversión al tipo booleano, estos valores se consideran FALSE (falsos):

- El booleano FALSE
- El entero 0
- El flotante 0
- La cadena vacía y la cadena "0"
- Una matriz con cero elementos
- Un objeto sin variables miembros
- Un tipo especial NULL (incluyendo valores no inicializados)

Cualquier otro valor se considera TRUE (verdadero).

Cuando se convierte al tipo entero, éstas son las reglas:

- Un booleano FALSE producirá 0 (cero) y el booleano TRUE producirá 1 (uno).
- Los valores de tipo flotante se redondearán a cero.

Cuando se convierten datos al tipo flotante, PHP convierte primero los datos a entero y luego a flotante. También puede convertir el tipo de cadena a tipos numéricos, pero es un poco enredado —vea el capítulo 3 para conocer los detalles.

Asimismo, PHP incluye funciones especiales que le permiten comprobar el formato interno de los datos —is_int( ), is_float( ), is_array( ), etcétera—; por ejemplo, si pasa una variable almacenada por PHP como entero internamente en is_int( ), esa función producirá un valor TRUE —en el siguiente capítulo ahondaremos en este tema.

CAPÍTULO 2

Operadores y control de flujo

El capítulo 1 fue una introducción a PHP y este capítulo profundiza más en el tema. Todo lo que verá en este capítulo es una habilidad esencial para el resto del libro.

En este capítulo se analizan *operadores* de PHP que usted empleará para manipular datos; ésta es parte esencial de los fundamentos de PHP que necesitará. Por ejemplo, la expresión $variable + 8, suma 8 al valor en $variable. La expresión $variable * 4, multiplica el valor de $variable por 4. Incluso si ha visto operadores en otros lenguajes, al menos revise este material, pues encontrará contenido exclusivo de PHP.

También verá todo sobre *control del flujo* en este capítulo. Éste le permite tomar decisiones en su código. ¿Va a tener un día de campo? Puede decidir basándose en la temperatura ambiente actual, según código PHP. ¿Tiene suficiente inventario para surtir pedidos de sus clientes? De nuevo, es cuestión de control del flujo. En PHP, como otros lenguajes, la instrucción principal de control del flujo es la instrucción if, que permite tomar una decisión y ejecutar cierto código, si esa decisión se orienta hacia un lado y código alternativo si la decisión apunta hacia el otro.

Además de instrucciones como if, también analizaremos ciclos. Los ciclos son fundamentales para PHP, como para muchos lenguajes de programación; permiten manipular grandes conjuntos de datos recorriéndolos en ciclos, un elemento de datos a la vez. Las computadoras fueron creadas para realizar tareas repetitivas como éstas, de modo que los ciclos tienen un lugar preponderante en PHP.

Operadores matemáticos de PHP

Comenzaremos con los operadores más básicos (los matemáticos), que son éstos:

- + Suma dos números
- – Resta un número de otro
- * Multiplica dos números
- / Divide un número entre otro
- % Devuelve el residuo, cuando un número se divide entre otro (módulo)

Estos operadores funcionan como sería de esperarse: para sumar dos valores se utiliza el operador +, de esta forma: a + b. Para restar b de a, es a – b. Bueno, es momento de escribir algo de código. Éste es un ejemplo, phpmathoperators.php, que pone a trabajar los operadores matemáticos:

```
<html>
    <head>
        <title>
            Uso de los operadores matemáticos
        </title>
    </head>
    <body>
        <h1>
            Uso de los operadores matemáticos
        </h1>
        <?php
            echo "7 + 2 = ", 7 + 2, "<br>";
            echo "7 - 2 = ", 7 - 2, "<br>";
            echo "7 * 2 = ", 7 * 2, "<br>";
            echo "7 / 2 = ", 7 / 2, "<br>";
            echo "7 % 2 = ", 7 % 2, "<br>";
        ?>
    </body>
</html>
```

Puede ver resultados en la Figura 2-1, donde los operadores matemáticos ejecutan sus funciones.

FIGURA 2-1 Operadores matemáticos de PHP en acción

Este ejemplo demuestra el uso de operadores matemáticos con dos números, 7 y 2, pero desde luego puede usar variables también:

```
$resultado = $operando1 + $operando2;
$resultado = $operando1 - $operando2;
$resultado = $operando1 * $operando2;
$resultado = $operando1 / $operando2;
$resultado = $operando1 % $operando2;
```

Además de los operadores matemáticos integrados, PHP también cuenta con varias funciones matemáticas y, mientras analizamos los operadores matemáticos, bien vale la pena dar un vistazo a las funciones matemáticas. Éstas son:

- **abs** Valor absoluto
- **acos** Arco coseno
- **acosh** Coseno hiperbólico inverso
- **asin** Arco seno
- **asinh** Seno hiperbólico inverso
- **atan2** Arco tangente de dos variables
- **atan** Arco tangente
- **atanh** Tangente hiperbólica inversa
- **base_convert** Convierte un número entre bases
- **bindec** Convierte de binario a decimal
- **ceil** Redondea fracciones hacia arriba
- **cos** Coseno
- **decbin** Convierte de decimal a binario
- **dechex** Convierte de decimal a hexadecimal
- **decoct** Convierte de decimal a octal
- **deg2rad** Convierte el número en grandes al equivalente en radianes
- **exp** Calcula el exponente de e
- **expm1** Devuelve exp(número) – 1
- **floor** Redondea fracciones hacia abajo
- **fmod** Devuelve el residuo de punto flotante de la división de los argumentos
- **getrandmax** Muestra el mayor valor aleatorio posible
- **hexdec** Convierte de hexadecimal a decimal
- **hypot** Devuelve sqrt(num1*num1 + num2+num2)
- **is_finite** Determina si un valor es un número finito válido
- **is_infinite** Determina si un valor es infinito
- **is_nan** Determina si un valor no es número

- **lcg_value** Generador de congruencia lineal combinado
- **log10** Logaritmo de base 10
- **log1p** Devuelve log(1 + número)
- **log** Devuelve el logaritmo natural
- **max** Encuentra el valor más alto
- **min** Encuentra el valor más bajo
- **mt_getrandmax** Muestra el mayor valor aleatorio posible
- **mt_rand** Genera un mejor valor aleatorio
- **mt_srand** Siembra el mejor generador de números aleatorios
- **octdec** Convierte de octal a decimal
- **pi** Obtiene el valor de pi
- **pow** Expresión exponencial
- **rad2deg** Convierte el número de radianes a un equivalente en grados
- **rand** Genera un entero aleatorio
- **round** Redondea un número flotante
- **sin** Seno
- **sinh** Seno hiperbólico
- **sqrt** Raíz cuadrada
- **srand** Siembra el generador de números aleatorios
- **tan** Tangente
- **tanh** Tangente hiperbólica

Éste es un ejemplo, phpmathfunctions.php, que pone a trabajar algunas de estas funciones. Por ejemplo, usted podría querer la tangente de 45º y utilizaría la función *tan* para ello. La función *tan* espera que su operando esté en radianes; de modo que primero debe convertir 45º a radianes, que puede hacer con la función deg2rad:

```
<?php
    echo "tan(deg2rad(45)) = ", tan(deg2rad(45)), "<br>";
    .
    .
    .
?>
```

¿Qué tal si calculamos ahora un exponente? Puede calcular 4 elevado a la potencia 3, es decir 4^3, utilizando la función *pow*:

```
<?php
    echo "tan(deg2rad(45)) = ", tan(deg2rad(45)), "<br>";
    echo "pow(4, 3) = ", pow(4, 3), "<br>";
    .
    .
    .
?>
```

Puede redondear números hacia abajo utilizando la función *floor* o bien, usando la función *ceil*. En este ejemplo, redondeamos pi hacia abajo:

```
<?php
    echo "tan(deg2rad(45)) = ", tan(deg2rad(45)), "<br>";
    echo "pow(4, 3) = ", pow(4, 3), "<br>";
    echo "floor(3.14159) = ", floor(3.14159), "<br>";
    .
    .
    .
?>
```

¿Qué tal un poco de matemáticas hexadecimales? La función *dechex* convierte valores decimales a hexadecimales de esta forma:

```
<?php
    echo "tan(deg2rad(45)) = ", tan(deg2rad(45)), "<br>";
    echo "pow(4, 3) = ", pow(4, 3), "<br>";
    echo "floor(3.14159) = ", floor(3.14159), "<br>";
    echo "dechex(16) = ", dechex(16), "<br>";
    .
    .
    .
?>
```

Y puede obtener el arco tangente de 1 con la función *atan,* convirtiendo la respuesta en radianes a grados en phpmathfunctions.php de esta forma:

```
<html>
    <head>
        <title>
            Uso de las funciones matemáticas
        </title>
    </head>
    <body>
        <h1>
            Uso de las funciones matemáticas
        </h1>
        <?php
            echo "tan(deg2rad(45)) = ", tan(deg2rad(45)), "<br>";
            echo "pow(4, 3) = ", pow(4, 3), "<br>";
            echo "floor(3.14159) = ", floor(3.14159), "<br>";
            echo "dechex(16) = ", dechex(16), "<br>";
            echo "rad2deg(atan(1)) = ", rad2deg(atan(1)), "<br>";
        ?>
    </body>
</html>
```

Puede ver phpmathfunctions.php en la Figura 2-2, funcionando según lo planeado.

FIGURA 2-2 Funciones matemáticas de PHP en acción

Trabajando con los operadores de asignación

El operador de asignación principal es =, que simplemente asigna un valor, como éste, almacenando el valor 99 en $botellas_de_cerveza_en_la_pared:

```
$botellas_de_cerveza_en_la_pared = 99;
```

Esto es algo más que debe saber acerca del operador de asignación: puede hacer múltiples asignaciones en la misma línea, de esta forma:

```
<?php
  $a = $b = $c = $d = 3;
  .
  .
  .
?>
```

Esta línea de código asigna el valor 3 a cada una de las variables $a, $b, $c y $d. Es bastante útil.

Ahora dé un vistazo a estas líneas de código:

```
<?php
  $a = 3;
  $a = $a + 6;
  .
  .
  .
?>
```

La idea aquí es que asignamos un valor de 3 a la variable $a y luego sumamos 6 a ese valor, para terminar con 9 en $a. PHP proporciona un conjunto de operadores de asignación de combinación; de modo que si desea contraer una línea como $a = $a = 6, puede usar el operador de asignación de combinación +=. El operador =+ combina los operadores + e = de tal modo, que esta línea:

```
$a = $a + 6;
```

funciona igual que esta otra:

```
<?php
  $a = 3;
  $a += 6;
  .
  .
  .
?>
```

Así, los operadores de asignación de combinación proporcionan una forma abreviada de realizar dos operaciones, una de ellas como asignación; $a = $a + 6 se convierte en $a += 6;.

De manera similar, el operador de asignación -= combina una resta con una asignación; este código deja 1 en $a:

```
<?php
  $a = 3;
  $a -= 2;
  .
  .
  .
?>
```

Éstos son los operadores de asignación combinados de PHP. Algunos de los que aquí se muestran podrían no serle familiares aún, pero pronto los analizaremos en este capítulo:

- +=
- –=
- *=
- /=
- .=
- %=
- &=
- |=
- ^=
- <<=
- >>=

Éste es otro ejemplo, que usa el operador de concatenación de cadenas de PHP (.), de esta forma:

```
<?php
  $a = "Sin ";
  $a .= "preocupaciones.";
  .
  .
  .
?>
```

Esto deja el texto"Sin preocupaciones."en $a.

Incremento y disminución de valores

Otra cosa hecha con frecuencia en PHP, es incrementar (sumar 1) o disminuir (restar 1) valores. Por ejemplo, podría utilizar código como éste para aumentar el valor en $a:

```
<?php
  $a = 1;
  $a = $a + 1;
  .
  .
  .
?>
```

De hecho, incluso podría usar el operador de asignación abreviado += de esta forma:

```
<?php
  $a = 1;
  $a = $a += 1;
  .
  .
  .
?>
```

Pero PHP tiene un operador más sencillo, específicamente para incrementar valores: ++, usado de esta forma:

```
<?php
  $a = 1;
  $a++;
  .
  .
  .
?>
```

Después de la ejecución de este código, $a queda con un valor de 2. De forma similar, el operador – disminuye valores.

Esto es clave para entender los operadores ++ y --. Si utiliza ++ o -- después de una variable como en $a++, el valor de la variable se incrementa luego de ejecutarse el resto de la instrucción. Así que, por ejemplo, esta línea de código calcula y muestra la raíz cuadrada de 4, mostrando un valor de 2 y, sólo entonces, incrementa el valor de $a:

```
<?php
  $a = 4;
  echo sqrt($a++);
  .
  .
  .
?>
```

Por otra parte, este código, donde el operador ++ va primero, incrementa primero el valor en $a y luego muestra la raíz cuadrada de 5:

```
<?php
  $a = 4;
  echo sqrt(++$a);
  .
  .
  .
?>
```

Es importante tener eso presente (si coloca ++ o -- enfrente de una variable, el valor de esta se incrementa o disminuye *antes* de que se ejecute el resto de la instrucción; si coloca el operador ++ o -- después de la variable, el valor de esa variable se incrementa o disminuye *después* de ejecutarse el resto de la instrucción.

Ésta es una página PHP de ejemplo, que muestra cómo funcionan los incrementos en PHP, phpincrement.php:

```
<html>
    <head>
       <title>
           Incremento y disminución
       </title>
    </head>

    <body>
      <h1>
          Incremento y disminución
      </h1>
      <?php
        $a = $b = $c = $d = 1;

        echo "\$a = \$b = \$c = \$d = 1 <br>";
        echo "\$a++ da ", $a++, "<br>";
        echo "Ahora \$a = ", $a, "<br>";
        echo "++\$b da ", ++$b, "<br>";
        echo "\$c-- da ", $c--, "<br>";
        echo "Ahora \$c = ", $c, "<br>";
        echo "--\$d da ", --$d, "<br>";
      ?>
    </body>
</html>
```

Puede ver esta página, phpincrement.php, en la Figura 2.3.

FIGURA 2-3 Incremento y disminución con PHP

Operadores de cadena de PHP

Quizás haya escuchado que PHP logra la excelencia al trabajar con cadenas de texto y es verdad. Así que tal vez se sorprenda al enterarse de que PHP sólo tiene dos operadores para funcionar con cadenas: el operador de concatenación, ., y el operador de asignación de concatenación combinado, .=. El verdadero poder de encadenamiento PHP está en sus funciones de cadena, que verá en el capítulo 3.

Éste es un ejemplo del uso de operadores de cadena, phpstringoperators.php:

```
<html>
  <head>
   <title>Operadores de cadena</title>
  </head>
  <body>
    <h1>Operadores de cadena</h1>
    <?php
      $a = "Sin ";
      echo "\$a = ", $a, "<br>";
      echo "\$b = \$a . \"preocupaciones \"<br>";
      $b = $a . "preocupaciones ";
      echo "Ahora \$b = ", $b, "<br>";
      echo "\$b .= \"en absoluto.\"<br>";
      $b .= "en absoluto.";
      echo "Ahora \$b = ", $b, "<br>";
    ?>
  </body>
</html>
```

FIGURA 2-4 Operadores de cadena en PHP

Puede ver phpstringoperators.php en la Figura 2-4.

Operadores orientados a bits

PHP brinda también un conjunto de operadores orientados a trabajar con bits individuales de números. En general, tiene poco sentido utilizar estos operadores a menos que sepa lo que hace con lujo de detalle. Si no sabe de los bits contenidos en bytes, no se preocupe por ello. Aunque normalmente se usan para enteros y similares, de hecho pueden usarse también con cadenas, en cuyo caso trabajará con el código ASCII numérico de cada carácter.

NOTA *Estos operadores están diseñados para trabajar en los bits individuales contenidos en sus operandos; si busca los operadores booleanos que funcionan con valores TRUE/FALSE, esto lo veremos más adelante en este capítulo.*

Por ejemplo, el operador Or, |, trabaja con dos operandos como éste: $a | $b. En el resultado, los bits que se inicializan (es decir, igual a 1, no a 0) en $a o $b se inicializan en el resultado. Así, por ejemplo, si $a es igual a 1 (que tiene el bit 0 inicializado) y $b = 2 (con el primer bit inicializado), entonces $a | $b será igual a 1 | 2; los bits 0 y 1 se inicializan en el resultado, de modo que es igual a 3.

Puede observar los operadores orientados a bits en la Tabla 2-1.

Operador	Operación	Ejemplo	Resultado
$a & $b	And	$a & $b	Se inicializan los bits en $a y $b.
$a | $b	Or	$a | $b	Se inicializan los bits en $a o en $b.
$a ^ $b	Xor	$a ^ $b	Se inicializan los bits en $a o $b pero no en ambos.
~ $a	Not	~ $a	Los bits en $a no se inicializan y viceversa.
$a << $b	Desplazar a la izquierda	$a << $b	Desplaza los bits de $a $b pasos a la izquierda (cada paso significa "multiplicar por dos").
$a >> $b	Desplazar a la derecha	$a >> $b	Desplaza los bits de $a $b pasos a la derecha (cada paso significa "dividir entre dos").

TABLA 2-1 Operadores orientados a bits

Aquí podría observar también los operadores de desplazamiento, << y >>. Éstos le permiten desplazar los bits contenidos en sus operandos a la izquierda (<<) o a la derecha (>>). Usted indica el número que desea desplazar y el número de espacios que desea se desplacen los bits de ese número. Por ejemplo, 8 << 1 desplaza los bits con valor 8 un espacio a la izquierda, ello multiplica 8 por 2 para darle 16. O bien, 8 >> 2 desplaza los bits de valor 8 dos espacios a la derecha, lo mismo que dividir entre 4, de modo que 8 >> 2 = 2.

Operador de ejecución

El operador de ejecución de PHP es formidable (le permite ejecutar comandos del sistema, como fecha o dir (de directorio) de Windows. Todo lo que debe hacer para ejecutar un comando del sistema es encerrarlo entre comillas individuales inversas (`).

Éste es un ejemplo. El código ejecutará el comando de Windows "dir c:\Inetpub\wwwroot\ch02" y mostrará los resultados (observe que la diagonal inversa es un carácter sensible, de modo que debe cerrarlo como \\):

```
<?php
    $resultado = 'dir c:\\Inetpub\\wwwroot\\ch02';
    echo $resultado;
?>
```

Esto es lo que se obtiene al ejecutar este ejemplo, phpdir.php:

```
%php phpdir.php
 El volumen de la unidad K es AsicomDBaseII

 Directorio de K:\Asicom\Clientes\McGrawHill\PHP\Traducidos\Capítulos\Cap 02\Código\
ch02

07/03/2008  09:39 p.m.    <DIR>          .
07/03/2008  09:39 p.m.    <DIR>          ..
03/03/2008  11:06 a.m.               457 phpbreak.php
```

```
03/03/2008  11:06 a.m.               375 phpcontinue.php
07/03/2008  09:39 p.m.                79 phpdir.php
03/03/2008  11:06 a.m.               554 phpdoornot.php
03/03/2008  11:06 a.m.               434 phpdowhile.php
03/03/2008  11:06 a.m.               406 phpelse.php
03/03/2008  11:06 a.m.               642 phpelseif.php
03/03/2008  11:06 a.m.               358 phpequality.php
03/03/2008  11:06 a.m.               384 phpfor.php
03/03/2008  11:06 a.m.               323 phpforeach.php
03/03/2008  11:06 a.m.               359 phpif.php
07/03/2008  06:17 p.m.               549 phpincrement.php
03/03/2008  11:06 a.m.               344 phpisfloat.php
03/03/2008  11:06 a.m.               347 phplogical.php
07/03/2008  01:40 p.m.               555 phpmathfunctions.php
06/03/2008  07:43 p.m.               475 phpmathoperators.php
03/03/2008  11:06 a.m.                30 phpprecedence.php
03/03/2008  11:06 a.m.               267 phpsettingprecedence.php
07/03/2008  06:32 p.m.               454 phpstringoperators.php
03/03/2008  11:06 a.m.             1,150 phpstrings.php
03/03/2008  11:06 a.m.               897 phpswitch.php
03/03/2008  11:06 a.m.               405 phpwhile.php
              22 archivos          9,844 bytes
               2 dirs  28,034,473,984 bytes libres
```

Aquí hay otro ejemplo; éste ejecuta el comando date:

```
<?php
$resultado = 'date';
echo $resultado;
?>
```

Éste es el tipo de resultado que podría ver con Unix, utilizando el shell bash:

```
-bash-2.05$b php phpdate.php
Fri April 06 11:24:45 PDT 2007
```

Como date es también un comando de DOS, esto es lo que podría ver en una ventana DOS:

```
C:\php>php phpdate.php
La fecha actual es: 07/03/2008
Escriba la nueva fecha: (dd-mm-aa)
```

Precedencia de operadores de PHP

La mayoría de operadores adicionales de PHP que verá en este libro son para uso en instrucciones if y ciclos, es los que veremos a continuación. Sin embargo, primero hay una cosa que decir acerca de los operadores y es la precedencia (el orden en que se ejecutan los operadores).

Por ejemplo, observe esta expresión:

```
4 + 3 * 9
```

¿Qué pasará aquí? ¿Se hará la suma 4 + 3 y el resultado se multiplicará por 9 para producir 63? O bien, ¿el 3 se multiplicará por 9 para dar 27 y a ese número se sumará 4 para producir 31? Resulta que PHP procesa las multiplicaciones antes que las sumas; así que aquí obtendrá 31, como se puede verificar con este script, phpprecedence.php:

```
<?php
  echo 4 + 3 * 9;
?>
```

Cuando lo ejecute obtendrá

```
%php phpprecedence.php
31
```

¿Cómo puede averiguar la precedencia de diferentes operadores? Observe la Tabla 2-2, mostrando la precedencia de diversos operadores, de un valor alto en la cima de la tabla, a un valor bajo en la base de la misma.

¿Desea establecer la precedencia usted mismo? Puede indicar a PHP qué operaciones ejecutar primero, encerrándolas entre paréntesis. Éste es un ejemplo, phpsettingprecedence.php:

```
<html>
  <head>
   <title>Establecer precedencia de operadores</title>
  </head>

  <body>
    <h1>Establecer precedencia de operadores</h1>
    <?php
      echo "4 + 3 * 9 = ", 4 + 3 * 9, "<br>";
      echo "(4 + 3) * 9 = ", (4 + 3) * 9, "<br>";
    ?>
  </body>
</html>
```

TABLA 2-2 Operador

Operadores
new
[
! ~ ++ - (int) (float) (string) (array) (object)
@
*/ %
+ - .
<< >>
< <= > >=
== != === !==
&
^
|
&&
||
? :
= += -= *= /= .= %= &= |= ^= <<= >>=
print
and
xor
or
,

FIGURA 2-5 Cómo establecer la precedencia en PHP

Puede ver esta página en un navegador en la Figura 2-5. Como puede apreciar, el uso de paréntesis determina la precedencia de la ejecución de los operadores.

Uso de la instrucción if

Éste es el punto donde usted comienza a tomar decisiones en su código y a ejecutar otro, dependiendo de los resultados de esa decisión. Ésta es la instrucción if, la instrucción principal para tomar decisiones en PHP. Así es como se ve formalmente:

```
if(expresión)
  instrucción
```

Aquí, una *expresión* de PHP se evalúa de acuerdo con un valor TRUE o FALSE. Al igual que otros lenguajes a los que haya tenido alcance, si la *expresión* es TRUE (verdadera), se ejecuta la *instrucción* siguiente; si es FALSE (falsa), la *instrucción* no se ejecuta. Se usan operadores condicionales y lógicos, que veremos a continuación, para crear expresiones del tipo que pueden evaluar instrucciones if. Por ejemplo, puede utilizar el operador mayor que, >, para formar una expresión como 4 > 1, que es TRUE, ya que 4 es realmente mayor que 1.

También cabe observar que aunque la instrucción pueda ser una línea de código, también es posible utilizar una instrucción compuesta de PHP, integrada por múltiples instrucciones individuales encerradas entre llaves { y }. Ésta es una instrucción individual:

```
echo "Ésta es la respuesta.";
```

Ésta es una instrucción compuesta:

```
{
  echo "Ésta";
  echo " es";
  echo " la";
  echo " respuesta.";
}
```

La instrucción if es formidable porque le permite hacer elecciones al instante y ejecutar código alternativo, dependiendo de los resultados de esa elección. Por ejemplo, podría verificar el valor de una contraseña para asegurarse de que es correcta o comprobar la respuesta del usuario a preguntas sí/no ("¿Desea papas fritas con su platillo?").

Por ejemplo, podría mostrar texto si la temperatura exterior es superior a 18 grados centígrados, lo que se podría hacer de esta forma:

```
<?php
  $temperatura = 19
  if ($temperatura > 18)
    echo "Está agradable afuera.";
?>
```

Aunque esta forma de hacer las cosas, con una sola instrucción después de la instrucción if, funciona, es más usual emplear los llaves con instrucciones compuestas como ésta:

```
<?php
  $temperatura = 19
  if ($temperatura > 18) {
    echo "Está agradable afuera.";
  }
?>
```

Como ésta es la forma en que generalmente verá que se hacen las cosas (incluso con instrucciones individuales), así verá la instrucción if en este libro (con las llaves). Desde luego, también puede usar instrucciones compuestas aquí:

```
<?php
  $temperatura = 19
  if ($temperatura > 18)
    echo "Está ";
    echo "agradable ";
    echo "afuera.";
  }
?>
```

Éste ejemplo, phpif.php, comprueba cuántos minutos ha estado alguien en la piscina (si son más de 30, es tiempo de salir):

```
<html>
    <head>
        <title>Uso de la instrucción if</title>
    </head>
```

```
    <body>
        <h1> Uso de la instrucción if</h1>
        <?php
            $minutos = 31;
            if($minutos > 30) {
                .
                .
                .
            }
        ?>
    </body>
</html>
```

Si es tiempo de salir de la piscina, puede mostrar un mensaje:

```
<html>
    <head>
        <title>Uso de la instrucción if</title>
    </head>

    <body>
        <h1> Uso de la instrucción if</h1>
        <?php
            $minutos = 31;
            if($minutos > 30) {
                echo "¡Se ha terminado tu tiempo!<br>";
                echo "Por favor sal de la piscina.";
            }
        ?>
    </body>
</html>
```

Puede ver phpif.php en acción en la Figura 2-6.

FIGURA 2-6 Uso de la instrucción if en PHP

Éste es otro ejemplo, phpisfloat.php. Como se dijo en el capítulo anterior, PHP incluye funciones especiales que permiten determinar el formato de almacenaje interno seleccionado por PHP para las variables is_int, is_float, etcétera. Este ejemplo comprueba si una variable está almacenada como número de punto flotante, luego le suma 4.5 y utiliza la función var_dump de PHP para "descargar" (es decir, mostrar) el valor de la variable en el navegador:

```
<html>
    <head>
        <title>Uso de la función is_float</title>
    </head>

    <body>
        <h1> Uso de la función is_float</h1>
        <?php
         $variable = 10.7;
         if (is_float($variable)) {
           $variable = $variable + 4.5;
           var_dump($variable);
         }
       ?>
    </body>
</html>
```

Puede ver phpisfloat.php en acción en la Figura 2-7. Como aparece ahí, la función vardump da aquí la respuesta float(15.2), indicando valor y tipo interno de la variable.

Hablaremos más de la instrucción if a continuación. Por ejemplo, el operador mayor que (>), es tan sólo uno de entre un conjunto de operadores de comparación de PHP, que se analizarán a continuación.

FIGURA 2-7 Uso de la función is_float en PHP

Operadores de comparación de PHP

Ya ha visto uno de los operadores de comparación de PHP, el operador mayor que:

```
<?php
  $temperatura = 19
  if ($temperatura > 18) {
    echo "Está agradable afuera.";
  }
?>
```

Existen muchos otros operadores de comparación, como menor que o igual a (<=), en acción aquí:

```
<?php
  $temperatura = 19
  if ($temperatura > 18) {
    echo "Hace frío afuera.";
  }
?>
```

Puede ver todos los operadores de comparación de PHP en la Tabla 2-3.

Por ejemplo, para saber cómo usar el operador de igualdad == en phpequality.php, que comprueba si alguien ha estado en la piscina exactamente 30 minutos:

```
<html>
    <head>
        <title>Uso del operador ==</title>
    </head>
```

Operador	Operación	Ejemplo	Resultado
==	Igual	$a == $b	TRUE si $a es igual a $b
===	Idéntico	$a === $b	TRUE si $a es igual a $b y son del mismo tipo
!=	No igual	$a != $b	TRUE si $a no es igual a $b
<>	No igual	$a <> $b	TRUE si $a no es igual a $b
!==	No idéntico	$a !== $b	TRUE si $a no es igual a $b o no son del mismo tipo
<	Menor que	$a < $b	TRUE si $a es estrictamente menor que $b
>	Mayor que	$a > $b	TRUE si $a es estrictamente mayor que $b
<=	Menor que o igual a	$a <= $b	TRUE si $a es menor que o igual a $b
>=	Mayor que o igual a	$a >= $b	TRUE si $a es mayor que o igual a $b

TABLA 2-3 Operadores de comparación

```
    <body>
        <h1> Uso del operador ==</h1>
        <?php
            $minutos = 30;
            if($minutos == 30) {
                echo "Advertencia:<br>";
                echo "Tu tiempo en la piscina casi se ha agotado.";
            }
        ?>
    </body>
</html>
```

Puede ver phpequality.php en la Figura 2-8; asegúrese de no confundir el operador de igualdad, ==, y el operador de asignación, =.

De forma similar, en este ejemplo se usa el operador no igual, !=, para probar si la temperatura no es igual a 20 grados:

```
<?php
  $temperatura = 30;
  if ($temperatura != 20 {
     echo "La temperatura no es de 20 grados.";
  }
?>
```

Esto se obtiene del script:

```
La temperatura no es de 20 grados.
```

Además de operadores de comparación, existe también un conjunto de operadores lógicos que puede manejar con instrucciones de toma de decisiones, como la instrucción if.

FIGURA 2-8 Uso del operador de igualdad en PHP

Operadores lógicos de PHP

Ya ha visto cómo comprobar si la temperatura ambiente es mayor que 18 grados:

```
<?php
  $temperatura = 19
  if ($temperatura > 18) {
    echo "Está agradable afuera.";
  }
?>
```

¿Pero qué sucedería si deseara comprobar la temperatura es mayor que 18 grados *y* menor que 24 grados? En ese caso, puede utilizar el operador and lógico, &&, para conectar dos cláusulas condicionales, como en phplogical.php:

```
<html>
    <head>
        <title>Uso de los operadores lógicos</title>
    </head>

    <body>
        <h1>Uso de los operadores lógicos</h1>
        <?php
          $temperatura = 19;
          if ($temperatura > 18 && $temperatura < 24) {
            echo "Está entre 18 y 24 grados allá afuera.";
          }
        ?>
    </body>
</html>
```

Observe la línea

```
if ($temperatura > 18 && $temperatura < 24) {
  echo "Está entre 18 y 24 grados allá afuera.";
}
```

Puede ver phplogical.php en la Figura 2-9.

Es posible usar operadores lógicos como &&, para conectar cláusulas en una instrucción if —en este caso, la expresión condicional de la instrucción if es verdadera, sólo si el valor $temperatura es mayor que 18 y menor que 24.

Por otra parte, suponga que deseara comprobar si la temperatura es menor que 0 grados *o* mayor a 38 grados. Podría hacer eso con el operador or lógico, ||, de esta forma:

```
<html>
    <head>
        <title>Uso de los operadores lógicos</title>
    </head>

    <body>
        <h1>Uso de los operadores lógicos</h1>
        <?php
```

FIGURA 2-9 Uso del operador and lógico en PHP

```
        $temperatura = 19;
        if ($temperatura < 0 || $temperatura > 38) {
          echo "Mejor quédese en casa hoy.";
        }
      ?>
    </body>
</html>
```

Puede hallar todos los operadores lógicos de PHP en la Tabla 2-4.

Es de llamar la atención que existan dos operadores and lógicos (“and”y &&) y dos operadores or lógicos (“or”y ||). La respuesta es que los operadores && y || tienen una alta precedencia y hay ocasiones en que podría desear operadores lógicos de baja precedencia; de ese modo, no tiene que usar específicamente paréntesis en sus expresiones para obtener resultados correctos.

Operador	Operación	Ejemplo	Resultado
and	And	$a and $b	TRUE si $a y $b son TRUE
or	Or	$a or $b	TRUE si $a o $b es TRUE
xor	Xor	$a xor $b	TRUE si $a o $b es TRUE, pero no ambos
!	Not	! $a	TRUE si $a no es TRUE
&&	And	$a && $b	TRUE si $a y $b son TRUE
||	Or	$a || $b	TRUE si $a o $b es TRUE

TABLA 2-4 Operadores lógicos

Instrucción else

Hay más en relación con la instrucción if. Por ejemplo, ¿qué sucede si la condición de la instrucción if resulta falsa? ¿Hay aún una forma de ejecutar código? Sí existe; simplemente use una instrucción else. Así es como se ve la instrucción else formalmente:

```
if expresión
  instrucción1
else
  expresión2
```

Aquí, si la *expresión* es verdadera, se ejecutará la *instrucción1*. Por otra parte, si la *expresión* es falsa, se ejecutará la *expresión2*.

Éste es un ejemplo: phpelse.php. Suponga que desea imprimir un mensaje si la temperatura ambiente está fuera del rango de 0 a 38 grados y otro si la temperatura está dentro de ese rango. Puede comenzar determinando si la temperatura está fuera de ese rango y mostrando un mensaje si lo está:

```
<html>
    <head>
        <title>Uso de la instrucción else</title>
    </head>

    <body>
        <h1>Uso de la instrucción else</h1>
        <?php
          $temperatura = 19;
          if ($temperatura < 0 || $temperatura > 38) {
            echo "Mejor quédese en casa hoy.";
          }
          else {
            echo "Hace un bonito día.";
          }
        ?>
    </body>
</html>
```

En su defecto, si la temperatura está dentro del rango de 0 a 38 grados, puede mostrar un mensaje alternativo,"Hace un bonito día.":

```
<html>
    <head>
        <title>Uso de la instrucción else</title>
    </head>

    <body>
        <h1>Uso de la instrucción else</h1>
        <?php
          $temperatura = 19;
          if ($temperatura < 0 || $temperatura > 38) {
            echo "Mejor quédese en casa hoy.";
          }
```

```
          else {
            echo "Hace un bonito día.";
          }
        ?>
    </body>
</html>
```

Puede ver los resultados en la Figura 2-10 (Hace un bonito día).

Éste es otro ejemplo que prueba si su avión tiene o no combustible suficiente para llegar al aeropuerto:

```
<html>
    <head>
        <title>Uso de la instrucción else</title>
    </head>

    <body>
        <h1>Uso de la instrucción else</h1>
        <?php
          $galones = 6;
          if ($galones < 5) {
            echo "¿Tienes un paracaídas?";
          }
          else {
            echo "Vas a estar bien.";
          }
        ?>
    </body>
</html>
```

Éste es el resultado del script:

```
Vas a estar bien.
```

FIGURA 2-10 Uso de la instrucción else en PHP

Instrucción elseif

PHP también tiene una instrucción elseif, que puede usar para comprobar condiciones if alternativas (si la expresión condicional de una instrucción if es falsa, puede realizar pruebas adicionales con elseif). Así es como se ve la instrucción if completa, incluyendo instrucciones elseif:

```
if expresión1
  instrucción1
elseif expresión2
  expresión2
elseif expresión3
  expresión3
elseif expresión4
  expresión4
elseif expresión5
  expresión5
        .
        .
        .
else
  expresión6
```

Este ejemplo, phpelseif.php, muestra diferentes mensajes dependiendo de la temperatura. Si es menor a 0 grados, este script muestra el mensaje"Hace demasiado frío.". Si la temperatura está entre 0 y 16 grados, muestra el mensaje"Hace bastante frío.". Si la temperatura está entre 17 y 21 grados, muestra el mensaje"Está bastante agradable allá afuera.", y así sucesivamente:

```
<html>
    <head>
        <title>Uso de la instrucción elseif</title>
    </head>

    <body>
        <h1>Uso de la instrucción elseif</h1>
        <?php
          $temperatura = 19;
          if ($temperatura < 0) {
            echo "Hace demasiado frío.";
          }
          elseif ($temperatura < 16) {
            echo "Hace bastante frío.";
          }
          elseif ($temperatura < 21) {
            echo "Está bastante agradable allá afuera.";
          }
          elseif ($temperatura < 27) {
            echo "Hace bastante calor allá afuera.";
          }
          else {
            echo "Hace demasiado calor.";
          }
        ?>
    </body>
</html>
```

FIGURA 2-11 Uso de la instrucción elseif en PHP

Puede ver cómo funciona esta instrucción. Si la expresión condicional de la instrucción if es falsa, PHP comprueba la expresión condicional de la primera instrucción elseif, si es verdadera, se ejecuta el código de la instrucción elseif. Si la expresión condicional de la primera instrucción elseif es falsa, PHP pasa a la siguiente instrucción elseif y así sucesivamente. Al final de la cadena hay una instrucción else, cuyo código se ejecuta si no se ha ejecutado ningún otro en la instrucción if hasta este punto. Todo el proceso se muestra en la Figura 2-11.

Y con la adición de las cláusulas elseif, ahora ha logrado dominar la instrucción if (ya sabe todo lo que debe saber al respecto).

Operador ternario

En realidad existe un operador integrado, actuando como instrucción if en PHP: el operador ternario, que PHP comparte con otros lenguajes. Este operador tiene una forma inusual:

```
$resultado = condición ? expresión1 : expresión2;
```

Bueno, ¿qué está pasando aquí? Si la condición es verdadera, el operador ternario (compuesto por los caracteres ?:) devuelve la expresión1. En su defecto, devuelve expresión2. Así que, como puede ver, este operador le permite hacer elecciones en tiempo real.

Éste es un ejemplo. Este código muestra "Hace un bonito día." si la temperatura ambiente está entre 0 y 38 grados y "Mejor quédese en casa hoy." en caso contrario:

```
<?php
  $temperatura = 19;
  if ($temperatura < 0 || $temperatura > 38) {
        echo "Mejor quédese en casa hoy.";
  }
  else {
    echo "Hace un bonito día.";
  }
?>
```

Éste es el mismo código manejando el operador ternario (observe lo compacto que es):

```
<?php
  $temperatura = 19;
  if ($temperatura < 0 || $temperatura > 38) ? "Mejor quédese en casa hoy." :
"Hace un bonito día.";
  ?>
```

Otro ejemplo calcula el valor absoluto de los números —todo lo que hace es utilizar el operador de negación, –, para cambiar el signo de un valor si es negativo:

```
<?php
    $valor = -3;
    $valor_abs = $valor < 0 ? -$valor : $valor;
    echo $valor_abs;
?>
```

Este ejemplo muestra el valor 3.

Éste es otro ejemplo —ligeramente más comprometido—. Convierte dígitos decimales en hexadecimales (en tanto el resultado sea un dígito hexadecimal, 0-F). Éste es el código:

```
<?php
    $valor = 15;
    $hex = $valor < 10 ? "0x" . $valor : "0x" . chr($valor - 10 + 65);;
    echo "En hexadecimal, $valor = $hex;
?>
```

Este script mostrará: En hexadecimal, 15 = 0xF (en PHP, a los valores hexadecimales se les antepone 0x para indicar a PHP que se trata de un valor hexadecimal). Bastante conveniente, ya que la conversión se realiza en una instrucción (por supuesto, la función dechex de PHP puede dar mejores resultados, convirtiendo valores de múltiples dígitos).

Instrucción switch

La instrucción switch de PHP permite reemplazar secuencias largas if-elseif-else para comprobar condiciones con una instrucción sencilla. Se da un valor a la instrucción switch y ésta prueba ese valor contra instrucciones case listadas por usted, ejecutando el código de cualquier instrucción case, cuyo valor de prueba coincida con el que está comprobando. Así es como se ve la instrucción switch formalmente (observe que los valores entre corchetes [ ] son opcionales):

```
switch (valordeprueba) {
  case expresión1:
    instrucción1
    [break;]
  case expresión2:
    instrucción2
    [break;]
  case expresión3:
    instrucción3
    [break;]
  case expresión4:
    instrucción4
```

```
            [break;]
            .
            .
            .
          [default:
            instrucción_predeterminada]
        }
```

Así es como funciona esto: se pasa un valor (números o cadenas enteros o de puntos flotante) a la instrucción switch y ésta lo comprueba contra todas las instrucciones case listadas por usted; si el valor que comprueba coincide con el de una instrucción case, se ejecuta el código de esa instrucción case. Si ninguna instrucción case coincide con el valor que está comprobando, se ejecuta el código de la instrucción predeterminada, en caso de existir una. La instrucción break pone fin a la ejecución de la instrucción switch —si omite la instrucción break, la ejecución continuará automáticamente con el código de la siguiente instrucción case.

En este ejemplo, phpswitch.php, se muestran diferentes mensajes, dependiendo de la temperatura ambiente actual:

```
<html>
    <head>
        <title>
            Uso de la instrucción switch
        </title>
    </head>

    <body>
        <h1>
            Uso de la instrucción switch
        </h1>
        <?php
            $temperatura = 21;
            switch ($temperatura) {
                case 21:
                case 22:
                case 23:
                    echo "Hace un bonito día.";
                    break;
                case 24:
                case 25:
                case 26:
                    echo "Bueno, pero un poco caluroso.";
                    break;
                case 27:
                case 28:
                case 29:
                    echo "Un poco más caluroso.";
                    break;
                default:
                  echo "Temperatura fuera del rango que esta instrucción puede manejar.";
            }
        ?>
    </body>
</html>
```

Puede ver los resultados de phpswitch.php en la Figura 2-12.

FIGURA 2-12 Uso de la instrucción switch en PHP

Uso de ciclos for

Las computadoras son formidables para realizar tareas repetitivas (ése es su punto fuerte). Por esa razón, PHP viene equipado con ciclos estándar de otros lenguajes de programación, como el ciclo for, que ahora verá. Así es como funciona formalmente:

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

El ciclo for ejecuta *expresión1* antes de iniciar; luego comprueba el valor de *expresión2* (si es verdadero, el ciclo ejecuta la *instrucción* una vez). Luego el ciclo ejecuta *expresión3* (que a menudo incrementa el valor de una variable contadora de ciclos) y, después de eso, comprueba el valor de *expresión2* de nueva cuenta (tentativamente, podría comprobar el valor de la variable contadora de ciclos). Si *expresión2* sigue siendo verdadera, el ciclo ejecuta la *instrucción* una vez más. Luego el ciclo ejecuta *expresión3* de nueva cuenta y el proceso continúa hasta que *expresión2* se evalúa falsa, una vez concluido el ciclo. Observe que *expresión2* se comprueba antes de ejecutarse el ciclo y *expresión3* se ejecuta después de cada ocasión que se repite el ciclo.

En phpfor.php, se muestra un mensaje seis veces. Puede comenzar inicializando una variable contadora de ciclos a 0:

```
<?php
    for ($contador_ciclos = 0;...){
    .
    .
    .
    }
?>
```

Cada vez que se recorre el ciclo, puede comprobar si el contador de ciclos es mayor que 6, en cuyo caso debe desistir:

```
<?php
    for ($contador_ciclos = 0; $contador_ciclos < 6;...){
    .
    .
    .
    }
?>
```

Y después de cada ejecución del ciclo, puede incrementar el contador de ciclos:

```
<?php
    for ($contador_ciclos = 0; $contador_ciclos < 6; $contador_ciclos++){
    .
    .
    .
    }
?>
```

Eso conforma el ciclo, que se ejecutará seis veces. Todo lo que queda es el cuerpo del ciclo (la instrucción, que puede ser compuesta, que desea ejecutar cada vez que se repite el ciclo). En este caso, el ejemplo mostrará tan sólo otra línea de texto, como se aprecia en phpfor.php:

```
<html>
    <head>
        <title>
            Uso del ciclo for
        </title>
    </head>

    <body>
        <h1>
            Uso del ciclo for
        </h1>
        <?php
            for ($contador_ciclos = 0; $contador_ciclos < 6; $contador_ciclos++){
                echo "Verá este mensaje seis veces.<br>";
            }
        ?>
    </body>
</html>
```

Puede ver los resultados en la Figura 2-13 donde, como notará, el mensaje aparece seis veces, una vez por cada iteración del ciclo for. Muy conveniente.

FIGURA 2-13 Uso del ciclo for en PHP

También hay más en los ciclos for; pueden manejar múltiples contadores de ciclos si lo desea, en tanto los separe con el operador coma. Éste es un ejemplo que utiliza dos contadores de ciclos:

```
<?php
for ($ciclo1 = 2, $ciclo2 = 2; $ciclo1 < 6 && $ciclo2 < 6; $ciclo1++, $ciclo2++) {
    echo "$ciclo1 x $ciclo2 = ", $ciclo1 + $ciclo2, "<br>";
}
?>
```

Y así es como se ve el resultado en un navegador:

```
2 x 2 = 4
3 x 3 = 9
4 x 4 = 16
5 x 5 = 25
```

De hecho, no tiene que usar contadores de ciclos en absoluto en un ciclo for. Éste es un ejemplo que comienza inicializando una conexión con una base de datos; obtiene un nuevo registro cada vez que se recorre el ciclo y procesa el nuevo registro hasta llegar al final de los datos, donde la función process_record devuelve -1:

```
for (initialize_connection(); process_record() != -1; get_next_record() ) {
}
```

También puede anidar ciclos for, uno dentro de otro, como verá cuando trabajemos con matrices. Éste es un ejemplo rápido:

```
<?php
  for ($contador_ciclos1 = 0; $contador_ciclos1 < 6; $contador_ciclos1++){
    for ($contador_ciclos2 = 0; $contador_ciclos2 < 2; $contador_ciclos2++){
```

```
            echo "Verá este mensaje doce veces.<br>";
          }
        }
    ?>
```

Uso de ciclos while

PHP tiene otro tipo de ciclo: while. Éste es realmente sencillo y se ve así:

```
while (expresión)
  instrucción
```

Mientras la *expresión* sea verdadera, el ciclo ejecuta la *instrucción*. Eso es todo. Cuando la *expresión* es falsa, el ciclo termina (lo que significa, tome nota por favor, que es posible que la *instrucción* nunca se ejecute). En otras palabras, el ciclo while se mantiene en ejecución mientras su expresión de prueba sea verdadera.

Éste es un ejemplo, phpwhile.php continúa incrementando una variable hasta que su valor sea mayor que o igual a 10, momento en el que se detiene:

```
<html>
    <head>
        <title>
            Uso del ciclo while
        </title>
    </head>

    <body>
        <h1>
            Uso del ciclo while
        </h1>
        <?php
            $variable = 1;

            while ($variable < 10){
                echo "Ahora \$variable contiene: ", $variable, "<br>";
                $variable++;
            }
        ?>
    </body>
</html>
```

Y puede ver los resultados en la Figura 2-14 (el ciclo while continuó en ejecución mientras el valor de $variable era menor que 10 y cuando esa condición ya no fue cierta, se detuvo).

Incluso puede convertir un ciclo while en un ciclo for, usando un contador de ciclos explícito, como en este ejemplo:

```
<?php
    $contador_ciclos = 0;
    while ($contador_ciclos < 6) {
        .
        .
        .
    }
?>
```

FIGURA 2-14 Uso del ciclo while en PHP

Luego se incrementa y usa el contador de ciclos en el cuerpo del ciclo:

```
<?php
  $contador_ciclos = 0;
  while ($contador_ciclos < 6) {
      echo "Verá este mensaje seis veces. <br>";
      $contador_ciclos++;
  }
?>
```

A menudo se utiliza el ciclo while cuando tiene una situación de repetición, para la que en realidad no necesita un contador de ciclos. Por ejemplo, los ciclos while se usan a menudo para leer datos de archivos; usted podría abrir un archivo y luego utilizar un ciclo while para comprobar si ha llegado al final del archivo:

```
<?php
    open_file();
    while (not_at_end_of_file()){
        .
        .
        .
    }
?>
```

Si no ha llegado al final del archivo, puede leer datos del archivo y reproducirlos con echo:

```
<?php
    open_file();
    while (not_at_end_of_file()){
```

```
        $datos = rad_data_from_file();
        echo $datos;
    }
?>
```

Uso de ciclos do...while

Existe otro tipo de ciclo while (do...while). Este tipo de ciclo es como while con una excepción: verifica su expresión de prueba al final del ciclo, no al principio. Así es como se ve do...while:

```
do
    instrucción
while (expresión)
```

En este caso, el ciclo do...while continúa ejecutando la *instrucción* mientras la *expresión* sea verdadera —y observe en particular que como *expresión* se prueba al final del ciclo, podría ser falsa y el cuerpo del ciclo, pese a ello, se ejecutaría al menos una vez—. Eso es útil cuando se tiene una situación en que la condición de prueba se establece dentro del cuerpo del ciclo y, por tanto, no se puede probar hasta el final del ciclo.

Éste es un ejemplo, phpdowhile.php, que convierte el anterior de while a la forma do... while:

```
<html>
    <head>
        <title>
            Uso del ciclo do...while
        </title>
    </head>

    <body>
        <h1>
            Uso del ciclo do...while
        </h1>
        <?php
            $variable = 1;

            do {
                echo "Ahora \$variable contiene: ", $variable, "<br>";
                $variable++;
            }
            while ($variable < 10)
        ?>
    </body>
</html>
```

Puede ver los resultados en la Figura 2-15, donde el ciclo do...while ha hecho lo suyo.

FIGURA 2-15 Uso del ciclo do...while en PHP

Éste es otro ejemplo que señala la diferencia entre el ciclo while y do...while, phpdoornot.php. Aquí, aunque la condición se evalúa falsa, verá que el ciclo do...while ejecuta su contenido una vez (pero el ciclo while no lo hace):

```
<html>
    <head>
        <title>
            Uso del ciclo do...while
        </title>
    </head>

    <body>
        <h1>
            Uso del ciclo do...while
        </h1>
        <?php
            $variable = 20;

            do {
                echo "El ciclo do...while dice \$variable = ", $variable, "<br>";
            }
            while ($variable < 10);

            while ($variable < 10){
                echo "El ciclo do...while dice \$variable = ", $variable, "<br>";
            }
        ?>
    </body>
</html>
```

FIGURA 2-16 Ciclo do...while *versus* while en PHP

Puede ver los resultados en la Figura 2-16 (el ciclo do...while se ejecutó una vez, pero while no).

Uso del ciclo foreach

Existe un ciclo especial diseñado para trabajar con colecciones de PHP, como las matrices que verá en el capítulo siguiente. Las colecciones se componen de múltiples elementos y el ciclo foreach está diseñado para manipular colecciones. Es útil tener un ciclo como foreach, pues procesa automáticamente todos los elementos de la colección (en particular, no tiene que preocuparse por tener los contadores de ciclos funcionando correctamente). Así es como se ve cada ciclo foreach:

```
foreach (expresión_colección as $valor) instrucción
foreach (expresión_colección as $clave => $valor) instrucción
```

Y éste es un ejemplo, phpforeach.php, que pone a trabajar el ciclo foreach. Este ejemplo recorre todos los elementos de una matriz, tipos de emparedados (vea el siguiente capítulo para más información acerca de matrices) y muestra cada elemento:

```
<html>
  <head>
    <title>
      Uso del ciclo foreach
    </title>
  </head>
</html>
```

FIGURA 2-17 Ciclo foreach en PHP

```
  <body>
    <h1>Uso del ciclo foreach</h1>
      <?php
        $matriz = array("pavo", "jamón", "res");
        foreach ($matriz as $valor) {
          echo "Emparedado actual: $valor<br>";
        }
      ?>
  </body>
</html>
```

Puede ver los resultados en la Figura 2-17, donde el ciclo foreach ha recorrido la matriz. Muy bien.

Terminación prematura de ciclos

Puede detener un ciclo o instrucción switch en cualquier momento con la instrucción break. Ya la ha visto dentro de la instrucción switch:

```
switch ($temperatura) {
    case 21:
    case 22:
    case 23:
        echo "Hace un bonito día.";
        break;
        .
        .
        .
```

FIGURA 2-18 Uso de la instrucción break en PHP

Pero también puede utilizar la instrucción break en ciclos. Con phpbreak.php, se detiene un ciclo cuando el contador de ciclos alcanza un valor de 5:

```
<html>
  <head>
    <title>
      Uso de la instrucción break
    </title>
  </head>

  <body>
    <h1>Uso de la instrucción break</h1>
    <?php
      for ($contador_ciclos = 0; $contador_ciclos < 1000; $contador_ciclos++){
        echo "¡Voy a hacer esto mil veces a menos que me detengas!<BR>";
        if ($contador_ciclos == 5) {
            echo "Está bien, voy a parar.<BR>";
            break;
        }
      }
    ?>
  </body>
</html>
```

Y puede ver este ejemplo en acción en la Figura 2-18.

Omisión de iteraciones

A veces quizás quiera omitir la iteración de un ciclo para evitar alguna clase de problema, como una división entre cero —y puede hacerlo con la instrucción continue, que hace pasar un ciclo a la siguiente iteración.

Este ejemplo, phpcpontinue.php, busca los recíprocos (es decir, 1 dividido entre el número) de diferentes enteros —observe que comprueba si se le pide dividir entre cero, y si esto sucede, pasa a la siguiente iteración:

```
<html>
  <head>
    <title>
      Uso de la instrucción continue
    </title>
  </head>

  <body>
    <h1>Uso de la instrucción continue</h1>
    <?php
      $número =1;
      for ($número = -3; $número < 4; $número++){
        if($número == 0){
          continue;
        }
        echo "1/$número = ", 1 / $número, "<br>";
      }
    ?>
  </body>
</html>
```

Y puede ver los resultados en la Figura 2-19 donde, como se aprecia, el script pasó por alto la etapa en que se suponía debía dividir entre cero. Genial.

FIGURA 2-19 Uso de la instrucción continue en PHP

Sintaxis alterna de PHP

PHP admite también una sintaxis alternativa para instrucciones if, while, for, foreach y switch. En cada caso, la forma de la sintaxis alterna cambia la llave de apertura por dos puntos (:) y la llave de cierre por endif;, endwhile;, endfor;, endforeach; o endswitch;, respectivamente.

Este ejemplo muestra una instrucción if con sintaxis alterna:

```
<?php
  $temperatura = 16
  if ($temperatura == 15):
    echo "La temperatura es 15";
  elseif ($temperatura == 21):
    echo "La temperatura es 21";
  else:
    echo "La temperatura no es 15 o 21";
  endif;
?>
```

Este ejemplo reproduce:"La temperatura no es 15 o 21".

En este otro se usa un ciclo for y sintaxis alterna:

```
<?php
    for ($contador_ciclos = 0; $contador_ciclos < 6; $contador_ciclos++) :
        echo "Verá este mensaje seis veces.<BR>";
    endfor;
?>
```

Ésta es una instrucción switch que utiliza sintaxis alterna:

```
<?php
  $temperatura = 18;
  switch ($temperatura) :
    case 15:
      echo "La temperatura es 15.";
      break;
    case 21:
      echo "La temperatura es 21.";
      break;
    case 27:
      echo "La temperatura es 27.";
      break;
    default:
      echo "La temperatura no es 15, 21 o 27.";
endswitch;
?>
```

Bueno, ha obtenido cimientos consistentes para el trabajo con operadores y estructuras de control de PHP, como instrucciones if y ciclos for. En el capítulo que sigue hablaremos del manejo de matrices y cadenas.

CAPÍTULO 3

Cadenas y matrices

Este capítulo representa el siguiente paso ascendente en el manejo de datos en PHP: cadenas de texto y matrices. Ya ha visto las cadenas hasta cierto punto, pero hay mucho más que añadir. Los datos en Internet suelen estar en formato de texto y PHP es excelente en el manejo de cadenas —existe una enorme biblioteca de funciones de cadena preconstruidas en PHP. Desde la clasificación de cadenas hasta su búsqueda; desde recortar espacios extra hasta la obtención de la longitud de una cadena. En este capítulo dominará las funciones de cadena de PHP.

Además de las cadenas, también aprenderá a trabajar con matrices en PHP en este capítulo. Las matrices representan nuestro primer contacto con colecciones en PHP —el almacenaje de más de un elemento de datos con un nombre—. Las matrices son parte importante de PHP porque, entre otras cosas, la notación de matriz se usa para leer datos enviados por el usuario.

Ya ha visto variables simples, conteniendo elementos de datos simples como cadenas o números. Las matrices almacenan conjuntos completos de elementos de datos de la misma forma y usted puede tener acceso a esos elementos de datos individuales, por ejemplo, con un índice de matrices numéricas. Eso es formidable desde el punto de vista de la programación, ya que puede usar un ciclo para incrementar de forma sostenida ese índice de matrices, lo que permitirá recorrer y manejar todo elemento contenido en una matriz.

Suponga que tiene 12 000 elementos de datos numéricos y desea calcular su valor promedio. Sería una tarea considerable escribir código para el manejo de todos esos elementos de datos de forma individual. Pero puede agruparlos en una matriz y luego recorrerla toda, manejando elementos de datos uno por uno, con un simple ciclo en un par de líneas. No hay problema.

Bueno, estamos listos para comenzar. Empezaremos dando un vistazo a las funciones de cadena integradas de PHP.

Funciones de cadena

PHP se orienta a cadenas e integra muchas funciones para las mismas. A menudo, los programas PHP contienen grandes cantidades de texto —los datos obtenidos del usuario están en forma de texto, por ejemplo, y usted podría anular espacios extra del principio y final de éste—. O podría buscar comandos de texto para que el usuario pueda manejarlos. Hasta podría capitalizarlo o cualquiera de entre una centena de cosas.

Puede realizar todas esas operaciones con funciones de cadena de PHP. Para comenzar a trabajar con funciones de cadena y darle una indicación de lo que está disponible, puede encontrar todas las funciones de cadena en la Tabla 3-1.

Función	Hace esto
addcslashes	Encierra una cadena con diagonales (al estilo C)
addslashes	Encierra una cadena con diagonales
bin2hex	Convierte datos binarios a una representación hexadecimal
chop	Alias de la función rtrim
chr	Devuelve un carácter específico dado su código ASCII
chunk_split	Divide una cadena en segmentos de menor tamaño
convert_cyr_string	Convierte de un conjunto de caracteres cirílicos a otro
count_chars	Devuelve información acerca de caracteres en una cadena
crc32	Calcula el polinomio crc32 de una cadena
crypt	Permite el cifrado de cadenas en un sentido (combinación)
echo	Muestra una o más cadenas
explode	Divide una cadena en una subcadena
fprintf	Escribe una cadena con formato en una secuencia
get_html_translation_table	Devuelve una tabla de traducción
hebrev	Convierte texto en hebreo en texto visual
hebrevc	Convierte texto lógico en hebreo en texto visual
html_entity_decode	Convierte todas las entidades HTML en sus caracteres aplicables
htmlentities	Convierte todos los caracteres aplicables en entidades HTML
htmlspecialchars	Convierte caracteres especiales en entidades html
implode	Une elementos de matriz con una cadena
join	Alias de la función implode
levenshtein	Calcula la distancia Levenshtein entre dos cadenas
localeconv	Obtiene la información de formato numérico
ltrim	Recorta espacio en blanco del inicio de una cadena
md5_file	Calcula la combinación md5 de un nombre de archivo determinado
md5	Calcula la combinación md5 de una cadena
metaphone	Calcula la clave metaphone de una cadena
money_format	Formatea un número como cadena de moneda
nl_langinfo	Consulta información de idioma y configuración regional
nl2br	Inserta cambios de línea HTML antes de todas las líneas nuevas en una cadena
number_format	Formatea un número con separadores de unidades de millar agrupados
ord	Devuelve el valor ASCII de un carácter

TABLA 3-1 Funciones de cadena

Función	Hace esto
parse_str	Clasifica la cadena en variables
print	Muestra una cadena
printf	Muestra una cadena con formato
quoted_printable_decode	Convierte una cadena imprimible entrecomillada en una cadena de 8 bits
quotemeta	Coloca metacaracteres entre comillas
rtrim	Recorta espacio en blanco del final de una cadena
setlocale	Establece información de configuración regional
sha1_file	Calcula la combinación sha1 de un archivo
sha1	Calcula la combinación sha1 de una cadena
similar_text	Calcula la similitud entre dos cadenas
soundex	Calcula la clave soundex de una cadena
sprintf	Devuelve una cadena con formato
sscanf	Clasifica la entrada de una cadena según un formato
str_ireplace	Versión que distingue mayúsculas y minúsculas de la función str_replace
str_pad	Rellena una cadena con otra cadena
str_repeat	Repite una cadena
str_replace	Reemplaza todas las incidencias de la cadena de búsqueda con la cadena de reemplazo
str_rot13	Realiza la transformación rot13 en una cadena
str_shuffle	Mezcla una cadena al azar
str_split	Convierte una cadena en una matriz
str_word_count	Devuelve información acerca de palabras que se usan en una cadena
strcasecmp	Comparación de cadenas binarias que distinguen entre mayúsculas y minúsculas
strchr	Alias de la función strstr
strcmp	Comparación de cadenas binarias seguras
strcoll	Comparación de cadenas basada en la configuración regional
strcspn	Calcula la longitud del segmento inicial que no coincide con una máscara
strip_tags	Recorta etiquetas HTML y PHP de una cadena
stripcslashes	Retira addcslashes( ) de una cadena
stripos	Calcula la posición de la primera incidencia de una cadena que distingue mayúsculas y minúsculas
stripslashes	Retira addslashes( ) de una cadena
stristr	Versión que distingue mayúsculas y minúsculas de la función strstr
strlen	Calcula la longitud de una cadena

TABLA 3-1 Funciones de cadena (*continuación*)

Función	Hace esto
strnatcasecmp	Comparaciones de cadenas que distinguen mayúsculas y minúsculas
strnatcmp	Comparaciones de cadenas utilizando un algoritmo de "orden natural"
strncasecmp	Comparación de cadenas binarias que distinguen mayúsculas y minúsculas de los primeros *n* caracteres
strncmp	Comparación de cadenas binarias seguras de los primeros *n* caracteres
strpos	Determina la posición de la primera incidencia de una cadena
strrchr	Determina la última incidencia de un carácter en una cadena
strrev	Invierta una cadena
strripos	Determina la posición de la última incidencia de una cadena que distingue mayúsculas y minúsculas
strrpos	Calcula la posición de la última incidencia de un carácter en una cadena
strspn	Determina la longitud del segmento inicial que coincide con la máscara
strstr	Determina la primera incidencia de una cadena
strtok	"Tokeniza" una cadena
strtolower	Convierte una cadena a minúsculas
strtoupper	Convierte una cadena a mayúsculas
strtr	Traduce ciertos caracteres
substr_compare	Comparación binaria segura (con distinción opcional de mayúsculas y minúsculas) de dos cadenas desde una bifurcación
substr_count	Cuenta el número de incidencias de subcadenas
substr_replace	Reemplaza texto en parte de una cadena
substr	Devuelve parte de una cadena
trim	Recorta espacio en blanco del principio y final de una cadena
ucfirst	Cambia a mayúscula el primer carácter de una cadena
ucwords	Pone en mayúscula el primer carácter de cada palabra en una cadena
vprintf	Imprime una cadena con formato
vsprintf	Devuelve una cadena con formato
wordwrap	Recorta una cadena a un número dado de caracteres

TABLA 3-1 Funciones de cadena (*continuación*)

Como notará, existen muchas funciones de cadena. Pongamos a trabajar algunas de ellas en un ejemplo llamado phpstrings.php. Podría comenzar con la cadena"No hay problema":

```
echo "La cadena de prueba es 'No hay problema'.<br>";
```

Puede usar la función strlen para calcular la longitud de la cadena:

```
echo "'No hay problema' tiene ", strlen("No hay problema"), " caracteres de largo<br>;
```

Y puede extraer subcadenas con substr:

```
echo "La subcadena substr('No hay problema', 3, 7) tiene '", substr("No hay pro-
blema", 3, 7), "'<br>;
```

Puede buscar cadenas con strpos:

```
echo "La palabra 'problema' está en la posición ", strpos("No hay problema", "pro-
blema"), "<br>";
```

Y reemplazar subcadenas con substr_replace:

```
echo "Reemplazar 'problema' con 'problemas' da: ", substr_replace("No hay problema",
"problemas", 3, 9), "<br>";
```

La función chr convierte códigos numéricos ASCII en sus caracteres correspondientes (aquí es ABC en ASCII):

```
echo "Uso de códigos ASCII: ", chr(65), chr(66), chr(67), "<br>";
```

Puede cambiar a mayúscula la primera letra de la cadena con ucfirst:

```
echo "Poner en mayúscula la primera letra le da: ", ucfirst("no hay problema"),
"<br>";
```

O bien pasar a mayúsculas toda la cadena con strtoupper:

```
echo "En mayúsculas: ", strtoupper("No hay problema"), "<br>";
```

Y en minúsculas toda la cadena con strtolower:

```
echo "En minúsculas: ", strtolower("No hay problema"), "<br>";
```

Es posible recortar espacios excesivos con la función trim (los códigos   que ve aquí, sirven para crear espacios no divisorios en el navegador, para que éste no contraiga los cuatro espacios aquí en un solo espacio):

```
echo "'    No hay problema' recortado es: '" trim("    No hay
problema"), "'<br>";
```

Puede invertir cadenas con strrev:

```
echo "Invertido: ", strrev("No hay problema"), "<br>";
```

Así como contar el número de incidencias de una subcadena con la función substr_count de esta forma en phpstrings.php:

```
<html>
  <head>
    <title>
      Uso de funciones de cadena
    </title>
  </head>
```

```
  <body>
    <h1>
      Uso de funciones de cadena
    </h1>
    <?php
      echo "La cadena de prueba es 'No hay problema'.<br>";
      echo "'No hay problema' tiene ", strlen("No hay problema"), " caracteres de
largo<br>";
      echo "La subcadena substr('No hay problema', 3, 7) es '", substr("No hay pro-
blema", 3, 7), "'<br>";
      echo "La palabra 'problema' está en la posición ", strpos("No hay problema",
"problema"), "<br>";
      echo "Reemplazar 'problema' con 'problemas' da: ", substr_replace("No hay pro-
blema", "problemas", 3, 9), "<br>";
      echo "Uso de códigos ASCII: ", chr(65), chr(66), chr(67), "<br>";
      echo "Poner en mayúscula la primera letra le da: ", ucfirst("no hay pro-
blema"), "<br>";
      echo "En mayúsculas: ", strtoupper("No hay problema"), "<br>";
      echo "En minúsculas: ", strtolower("No hay problema"), "<br>";
      echo "'   No hay problema' recortado es: '", trim("    No
hay problema"), "'<br>";
      echo "Invertido: ", strrev("No hay problema"), "<br>";
      echo "Hay ", substr_count("No hay problema", "o"), " letras o en 'No hay pro-
blema'.";
    ?>
  </body>
</html>
```

¿Cómo se verá todo esto en un navegador? Observe la Figura 3-1, donde aparecen los resultados. Como puede ver, PHP tiene mucho poder para manejar cadenas.

FIGURA 3-1 Uso de funciones de cadena

Conversión a y desde cadenas

Como los datos son enviados a usted en formato de cadena y deberá mostrar sus datos también bajo las mismas características en el navegador del usuario, la conversión entre cadenas y números es una de las tareas de interfaz más importantes en PHP.

Éste es el ejemplo, phpconvert.php. Para generar conversiones a cadena, puede utilizar la función *cast* (cadena) o strval, que devuelve el valor de cadena del elemento que usted le pasa (técnicamente es innecesario usar las funciones cast y strval, ya que echo también convierte números en cadenas):

```
<?php
    $flotante = 3.1415;
    echo (cadena) $flotante, "<br>";
    echo strval($flotante), "<br>";
?>
```

¿Cómo trabajan las funciones (cadena) cast y strval cuando se convierten otros valores? Un valor TRUE booleano se convierte en la cadena"1"; el valor FALSE se representa como ""(cadena vacía). Un número entero o de punto flotante (flotante) se convierte en una cadena que representa el número con sus dígitos (incluyendo la parte del exponente de números de punto flotante). El valor NULL se convierte siempre en una cadena vacía.

También puede ir en la dirección opuesta y convertir una cadena en número. La cadena será tratada como número flotante si contiene alguno de los caracteres'.','e' o'E'. En su defecto, será tratada como entero. El valor numérico de una cadena está dado por la *parte inicial* de la cadena. Si ésta comienza con datos numéricos, ése será el valor empleado. En caso contrario, el valor será 0 (cero). Los datos numéricos válidos constan de un signo opcional (+ o –), seguido de uno o más dígitos (incluido, en caso de usarlo, un punto decimal), seguido de un exponente opcional (la parte del exponente es una'e' o'E', seguida de uno o más dígitos).

Por ejemplo, puede ingresar un número de manera normal y entrecomillado (es decir, un número en una cadena), PHP hará lo que sea preciso:

```
$valor = 1 + "19.2";
echo "$valor <br>";
```

Éste es otro ejemplo, muestra un número con una potencia de 10:

```
$valor = 1 + "2.5e4";
echo "$valor <br>";
```

También puede convertir explícitamente texto en números flotantes con la función (flotante) cast, por ejemplo:

```
$texto = "3.0";
$valor = (flotante) $texto;
echo $valor / 2.0, "<br>";
```

Así es como se ve phpconvert.php, reuniendo todos estos elementos:

```
<html>
  <head>
    <title>
      Conversión desde cadenas y números
```

FIGURA 3-2 Conversión entre cadenas y números

```
    </title>
  </head>
  <body>
    <h1>
      Conversión desde cadenas y números
    </h1>
    <?php
    $flotante = 3.1415;
    echo (string) $flotante, "<br>";
    echo strval($flotante), "<br>";
    $valor = 1 + "19.2";
    echo "$valor <br>";
    $valor = 1 + "2.5e4";
    echo "$valor <br>";
    $texto = "3.0";
    $valor = (float) $texto;
    echo $valor / 2.0, "<br>";
?>
  </body>
</html>
```

Puede ver phpconvert.php en acción en la Figura 3-2.

Formateo de cadenas de texto

Debido a que todos los datos enviados a sus scripts de PHP y aquellos devueltos por usted al navegador están en formato de texto, el formateo de tales datos es uno de los aspectos más importantes que realizará en PHP. Por ejemplo, ¿cómo cerciorarse de que un precio mostrado

tiene exactamente dos espacios después del punto decimal? Resulta que hay dos funciones de PHP para manejar específicamente el formateo de cadenas de texto, son printf y sprintf:

```
printf (formato [, args])
sprintf (formato [, args])
```

La función printf imprime una cadena (de manera muy similar a echo) y la función sprintf también "imprime" sus datos, pero en este caso, el resultado es una cadena —es decir, devuelve una cadena.

La cadena de formato aquí es un poco compleja. Se compone de cero o más directivas, en realidad, caracteres que se copian en el resultado y especificadores de conversión, compuestos por un signo de porcentaje (%), seguido de uno o más de estos elementos (en este orden):

- Un *especificador de relleno* opcional, indicando qué carácter debe usarse para rellenar los resultados hasta alcanzar el tamaño de cadena correcto. Éste puede ser un carácter de espacio o un 0 (carácter cero). La opción predeterminada es rellenar con espacios.
- Un *especificador de alineación* opcional, que indica si los resultados deben justificarse a la izquierda o derecha. La opción predeterminada es la de justificar a la derecha (aquí, un carácter – lo justificará a la izquierda).
- Un número opcional, el *especificador de ancho*, indica cuántos caracteres (como mínimo) debe producir esta conversión.
- Un *especificador de precisión* opcional señalando cuántos dígitos decimales deben mostrarse para números de punto flotante. (No hay efecto para otros tipos más que el flotante.)
- Un *especificador de tipo*, indicando de qué tipo deben tratarse los datos de los argumentos.

Éstos son los posibles especificadores de tipo:

- % Un carácter de porcentaje literal. No se requiere argumento.
- **b** El argumento se trata como entero y se presenta como número binario.
- **c** El argumento es tratado como entero y presenta como carácter con valor ASCII.
- **d** El argumento se trata como entero y presenta como número decimal (con signo).
- **u** El argumento se trata como entero y presenta como número decimal sin signo.
- **f** El argumento es tratado como flotante y presenta como número de punto flotante.
- **–** El argumento es tratado como entero y presenta como número octal.
- **s** El argumento se trata y presenta como cadena.
- **x** El argumento es tratado como un entero y presenta como número hexadecimal (con letras minúsculas).
- **X** El argumento se trata como un entero y presenta como número hexadecimal (con letras mayúsculas).

Lograr que los formatos funcionen puede resultar complicado. Éste es un ejemplo, phpformat.php. Podría comenzar con el formato %s, que simplemente inserta una cadena, con printf:

```
printf("Tengo %s pantalones y %s camisas.<br><br>", 4, 12);
```

Esto mostrará "Tengo 4 pantalones y 12 camisas.". Así es como funciona (se pone el formato en una plantilla de cadena y luego siguen datos a los que se dará formato).

Así es como se utiliza sprintf, que devuelve una cadena:

```
$cadena = sprintf("Después de la venta tengo %s pantalones y %s camisas.<br>", 6, 21);
echo $cadena, "<br>";
```

Éste es un uso más intensivo del formato, empleando %01.2f para producir dos espacios después del punto decimal:

```
$precio = 2789.992;
echo "El precio es ";
printf("\$%01.2f<br><br>", $precio);
```

Esto mostrará "El precio es 2789.99". Aquí hay otros usos de las cadenas de formato, para modificar números de punto flotante:

```
echo "printf(\"%6.2f\", 3.1) le da ";
printf("%6.2f<br>", 3.1);
echo "printf(\"%6.2f\", 30.1) le da ";
printf("%6.2f<br>", 30.1);
echo "printf(\"%6.3f\", 300.1) le da ";
printf("%6.2f<br><br>", 300.1);
```

Éste es un ejemplo que utiliza el especificador de formato d, para formatear enteros:

```
$año = 2007;
$mes = 9;
$día = 2;
echo "La fecha es ";
printf("%04d-%02d<br>", $año, $mes, $día);
```

A continuación tenemos todo el contenido de phpformat.php:

```
<html>
  <head>
    <title>
      Uso del formato de cadenas
    </title>
  </head>
  <body>
    <h1>
      Uso del formato de cadenas
    </h1>
    <?php
      printf("Tengo %s pantalones y %s camisas.<br><br>", 4, 12);

      $cadena = sprintf("Después de la venta tengo %s pantalones y %s camisas.<br>", 6, 21);
      echo $cadena, "<br>";
```

```
    $precio = 2789.992;
    echo "El precio es ";
    printf("\$%01.2f<br><br>", $precio);

    echo "printf(\"%6.2f\", 3.1) le da ";
    printf("%6.2f<br>", 3.1);
    echo "printf(\"%6.2f\", 30.1) le da ";
    printf("%6.2f<br>", 30.1);
    echo "printf(\"%6.3f\", 300.1) le da ";
    printf("%6.2f<br><br>", 300.1);

    $año = 2007;
    $mes = 9;
    $día = 2;
    echo "La fecha es ";
    printf("%04d-%02d-%02d<br>", $año, $mes, $día);

  ?>
 </body>
</html>
```

Y puede ver phpformat.php en la Figura 3-3.

RECOMENDACIÓN *Casualmente, hay otra función para dar formato a números en PHP: number_format( ).*

FIGURA 3-3 Formateo de cadenas en PHP

Construya sus propias matrices

Ahora abordaremos un nuevo tema: las matrices. Éstas son parte integral de PHP, más que en otros lenguajes (por ejemplo, usted se comunica con el usuario de su aplicación Web mediante matrices en PHP). También las usa cuando tiene un conjunto de datos (múltiples elementos de datos).

Las matrices son colecciones de elementos de datos almacenados con un nombre. Podría llevar registro de las calificaciones de 500 alumnos de la clase de PHP impartida por usted, por ejemplo; las matrices son perfectas para eso.

Puede crear matrices de la misma forma que variables (tan sólo asigne valores a un nombre de variable). Por ejemplo, podría crear una matriz llamada $actores y asignar el nombre "Cary Grant" al primer espacio en esa matriz, el elemento 0:

```
$actores[0] = "Cary Grant";
```

Esta instrucción crea una nueva matriz llamada $actores y almacena "Cary Grant" en el primer elemento, el 0. Como sucede en otros lenguajes de programación, las matrices comienzan con el elemento 0 en PHP. Ahora, cuando reproduzca con echo la matriz $actores:

```
echo $actores[0];
verá "Cary Grant".
```

También puede agregar otros actores a la matriz $actores, como en phpactors.php:

```
<html>
  <head>
    <title>
      Creación de una matriz
    </title>
  </head>
  <body>
    <h1>
      Creación de una matriz
    </h1>
    <?php
     $actores[0] = "Cary Grant";
     $actores[1] = "Myrna Loy";
     $actores[2] = "Lorne Green";

      echo "\$actores[0] = ", $actores[0], "<br>";
      echo "\$actores[1] = ", $actores[1], "<br>";
      echo "\$actores[2] = ", $actores[2], "<br>";
    ?>
  </body>
</html>
```

Así, $actores contiene "Cary Grant", en $actores[1] está "Myrna Loy" y $actores[2] tiene "Lorne Green", como se ve en la Figura 3-4.

Hasta aquí, estos ejemplos de matrices han manejado índices numéricos, que es útil cuando se desea recorrer una matriz en repetidas ocasiones (cosa que veremos pronto), ya que puede usar el contador de ciclos como índice de la matriz. Sin embargo, PHP le permite usar

FIGURA 3-4 Creación de matrices en PHP

también un índice de matriz de cadena y ése puede ser también de utilidad, pues podría olvidar que el dinero en deuda de su prima Jimena se almacena en $deudas[342], pero no olvidará $deudas[“Jimena”]:

```
$deudas["Jimena"] = 2493.77;
```

Ahora $deudas[“Jimena”] contiene 2493.77.Y puede almacenar otras deudas en esa matriz también:

```
$deudas[“Jimena”] = 2493.77;
$deudas[“Roberto”] = 4930.33;
$deudas[“Samuel”] = 5493.22;
```

De hecho, aunque use índices de cadena para estos elementos de matriz, resulta que puede recorrerla mediante PHP (hablaremos sobre eso más adelante).

Existe un método abreviado para crear matrices (puede utilizar simplemente el nombre de la matriz, seguido de corchetes vacíos como éstos y PHP completará los números):

```
$actores[] = "Cary Grant";
$actores[] = "Myrna Loy";
$actores[] = "Lorne Green";
```

Después de ejecutar este código, $actores[0] contendrá a“Cary Grant”, $actores[1] contendrá a “Myrna Loy”y $actores[2] contendrá a“Lorne Green”.

De hecho, existe un atajo aún más corto en PHP para crear una matriz: puede utilizar la función matriz, que justo le devuelve una. Así se ve:

```
$actores = array("Cary Grant", "Myrna Loy", "Lorne Green");
```

Esto crea también una matriz de tal modo que $actores[0] contendrá a“Cary Grant”, $actores[1] contendrá a“Myrna Loy”y $actores[2] contendrá a“Lorne Green”.

De forma predeterminada, las matrices comienzan con el índice 0. ¿Qué sucedería si deseara comenzar con el índice 1 en su lugar? Podría usar el operador => de PHP de esta forma:

```
$actores = array(1 => "Cary Grant", "Myrna Loy", "Lorne Green");
```

Esto lo dejaría con:

```
actores[1] "Cary Grant";
actores[2] "Myrna Loy";
actores[3] "Lorne Green";
```

El operador => le permite crear pares clave/valor en matrices (el elemento a la izquierda del operador => es la clave y el de la derecha es el valor). Por ejemplo, podría hacer esto en phparrayfunction.php:

```
<html>
  <head>
    <title>
      Uso de la función array
    </title>
  </head>
  <body>
    <h1>
      Uso de la función array
    </h1>
    <?php
      $deudas = array("Jimena" => 2493.77, "Roberto" => 4930.33, "Samuel" => 5493.22);

      echo "\$deudas['Jimena'] = ", $deudas["Jimena"], "<br>";
      echo "\$deudas['Roberto'] = ", $deudas["Roberto"], "<br>";
      echo "\$deudas['Samuel'] = ", $deudas["Samuel"], "<br>";
    ?>
  </body>
</html>
```

Como puede ver en la Figura 3-5, esto lo deja con

```
$deudas["Jimena"] = 2493.77;
$deudas["Roberto"] = 4930.33;
$deudas["Samuel"] = 5493.22;
```

Como puede observar, el operador => es de utilidad. Por ejemplo, serviría en caso de que no desee rellenar valores consecutivos, de la siguiente forma:

```
$actores = array(1 => "Cary Grant", 5 => "Myrna Loy", 7 => "Lorne Green");
```

Esto lo dejaría con

```
actores[1] "Cary Grant";
actores[5] "Myrna Loy";
actores[7] "Lorne Green";
```

FIGURA 3-5 Creación de matrices utilizando la función array en PHP

Éste es un método abreviado más: si tiene un rango de datos específico, como los números 1 a 10 o los caracteres *a* a *z*, puede usar la función range de PHP para crear matrices. Éste es un ejemplo:

```
$datos = range(1, 4);
```

Esto da como resultado

```
$datos[0] = 1
$datos[1] = 2
$datos[2] = 3
$datos[3] = 4
```

Modificación de datos en matrices

Ahora que ha visto cómo crear matrices, ¿qué tal si modifica los datos contenidos en ellas? No hay de qué preocuparse (puede modificar los valores contenidos en una matriz, tan fácil como el valor alojado en una variable).

Por ejemplo, suponga que ha creado esta matriz:

```
<?php
  $actores[0] = "Cary Grant";
  $actores[1] = "Myrna Loy";
  $actores[2] = "Lorne Green";
?>
```

Ahora suponga que desearía cambiar de $actores[2] "Lorne Green" por "Jimmy Stewart". No hay problema, simplemente haga esto:

```
<?php
  $actores[0] = "Cary Grant";
```

```
  $actores[1] = "Myrna Loy";
  $actores[2] = "Lorne Green";

  $actores[2] = "Jimmy Stewart ";
?>
```

Asimismo, imagine que quiere agregar un nuevo actor, "Julie Andrews", a esta lista. Tampoco hay problema, puede hacerlo en phpmodify.php:

```
<html>
  <head>
    <title>
      Modificación de matrices
    </title>
  </head>
  <body>
    <h1>
      Modificación de matrices
    </h1>
   <?php
      $actores[0] = "Cary Grant";
      $actores[1] = "Myrna Loy";
      $actores[2] = "Lorne Green";

      $actores[2] = "Jimmy Stewart";

      $actores[] = "Julie Andrews";

      echo "\$actores[0] = ", $actores[0], "<br>";
      echo "\$actores[1] = ", $actores[1], "<br>";
      echo "\$actores[2] = ", $actores[2], "<br>";
      echo "\$actores[3] = ", $actores[3], "<br>";
    ?>
  </body>
</html>
```

Puede ver los resultados en la Figura 3-6, donde ha modificado con éxito los valores contenidos en la matriz.

Es posible copiar matrices completas con una instrucción de asignación —por ejemplo, copiar la matriz $actores en una nueva, llamada $personal_creativo, de la siguiente manera— observe que se refiere a la matriz tan sólo por su nombre; puede prescindir de corchetes:

```
<?php
   $actores[0] = "Cary Grant";
   $actores[1] = "Myrna Loy";
   $actores[2] = "Lorne Green";

   $actores[2] = "Jimmy Stewart";

   $actores[] = "Julie Andrews";
```

FIGURA 3-6 Modificación de matrices en PHP

```
   echo "\$actores[0] = ", $actores[0], "<br>";
   echo "\$actores[1] = ", $actores[1], "<br>";
   echo "\$actores[2] = ", $actores[2], "<br>";
   echo "\$actores[3] = ", $actores[3], "<br>";
?>
```

Al final de este script, la instrucción echo $personal_creativo[1]; mostrará "Myrna Loy". No está mal.

Eliminación de elementos de una matriz

Puede eliminar elementos de matrices si lo hace correctamente. Tal vez se sienta tentado a hacer algo como esto, colocar simplemente un elemento de matriz en una cadena vacía:

```
<?php
   $actores[0] = "Cary Grant";
   $actores[1] = "Myrna Loy";
   $actores[2] = "Lorne Green";

   $actores[1] = "";
?>
```

Pero eso no elimina el elemento —simplemente lo sitúa en una cadena vacía; de tal modo, si muestra los elementos de matriz:

```
<?php
   $actores[0] = "Cary Grant";
   $actores[1] = "Myrna Loy";
   $actores[2] = "Lorne Green";

   $actores[1] = "";

   echo "\$actores[0] = ", $actores[0], "<br>";
   echo "\$actores[1] = ", $actores[1], "<br>";
   echo "\$actores[2] = ", $actores[2], "<br>";
?>
```

obtendrá esto en pantalla:

```
   $actores[0] = "Cary Grant";
   $actores[1] = "";
   $actores[2] = "Lorne Green";
```

Así que, ¿cómo elimina en realidad un elemento de una matriz? Utiliza la función unset de la siguiente manera, como en phpunset.php:

```
<html>
  <head>
    <title>
      Eliminación de un elemento de una matriz
    </title>
  </head>
  <body>
    <h1>
      Eliminación de un elemento de una matriz
    </h1>
   <?php
      $actores[0] = "Cary Grant";
      $actores[1] = "Myrna Loy";
      $actores[2] = "Lorne Green";

      unset($actores[1]);

      echo "\$actores[0] = ", $actores[0], "<br>";
      echo "\$actores[1] = ", $actores[1], "<br>";
      echo "\$actores[2] = ", $actores[2], "<br>";
    ?>
  </body>
</html>
```

Puede ver los resultados en la Figura 3-7.

Observe que aquí recibe un aviso porque $actores[1] no está definido:

```
PHP Notice: Undefined offset: 1 in C:\Inetpub\wwwroot\ch03\phpunset.php on line 19
```

FIGURA 3-7 Eliminación de un elemento de una matriz en PHP

Manejo de matrices con ciclos

Como ya hemos mencionado, matrices y ciclos son una combinación natural —las matrices se indizan utilizando un índice de matriz y los ciclos utilizan un contador de ciclos—, mediante el contador de ciclos como índice de matriz, puede incrementar toda una matriz en unas cuantas líneas. Por ejemplo, podría almacenar las calificaciones de estudiantes en una matriz y recuperar la calificación promedio —a su vez, los ciclos ofrecen una forma sensacional de recorrer toda la matriz, ya que permiten sumar todas las calificaciones y luego dividir entre el número total de elementos, para obtener la calificación promedio.

Ciclo for

Una función útil que puede emplear al utilizar ciclos con matrices es count. Esta función devuelve el número de elementos que hay en una matriz y eso es de utilidad cuando desea establecer, por ejemplo, el número de veces que se recorre un ciclo for como éste, en phparrayfor.php:

```
<html>
  <head>
    <title>
      Uso de un ciclo for para recorrer una matriz
    </title>
  </head>
  <body>
    <h1>
      Uso de un ciclo for para recorrer una matriz
    </h1>
   <?php
      $actores[0] = "Cary Grant";
      $actores[1] = "Myrna Loy";
      $actores[2] = "Lorne Green";
```

```
      for ($índice_ciclo = 0; $índice_ciclo < count($actores); $índice_ciclo++){
        echo "\$actores[$índice_ciclo] = ", $actores[$índice_ciclo], "<br>";
      }
    ?>
  </body>
</html>
```

Puede ver los resultados en la Figura 3-8. Como notará, el ciclo for recorrió todos los elementos de la matriz. Precioso.

Función print_r

De hecho, existe una función simple para mostrar el contenido de una matriz (la función print_r):

```
print_r ($matriz [, bool return])
```

Por ejemplo, si deseara imprimir la matriz $actores, podría usar la función print_r de esta forma:

```
<?php
   $actores[0] = "Cary Grant";
   $actores[1] = "Myrna Loy";
   $actores[2] = "Lorne Green";

 print_r($actores);
?>
```

Esta función utiliza líneas nuevas para imprimir, no elementos <br> de HTML; así que ejecute esto en la línea de comandos para obtener

```
%php phpprint_r.php
   Array
(
    [0] => Cary Grant
    [1] => Myrna Loy
    [2] => Lorne Green
```

FIGURA 3-8 Recorrido por un elemento de una matriz en PHP

Observe que esto da el resultado en forma de clave => valor. Pruebe con esto:

```
<?php
  $deudas["Jimena"] = 2493.77;
  $deudas["Roberto"] = 4930.33;
  $deudas["Samuel"] = 5493.22;

  print_r($deudas);
?>
```

La ejecución de esta nueva versión muestra cómo print_r produce pares clave/valor:

```
%php phpprint_r.php
   Array
(
    ["Jimena"] = 2493.77
    ["Roberto"] = 4930.33
    ["Samuel"] = 5493.22
)
```

También puede pasar un valor true (verdadero) a la función print_r para hacer que devuelva su valor como cadena, para asignar a una variable:

```
$resultado = print_r($deudas, true);
```

Ciclo foreach

El ciclo foreach está diseñado para trabajar con colecciones como matrices y es de utilidad porque no debe establecer condiciones de inicialización y terminación, como se hace con los ciclos for. Así es como se utiliza el ciclo foreach de manera formal:

```
foreach (array as $valor) instrucción
foreach (array as $clave => $valor) instrucción
```

En el ejemplo siguiente, phpforeach1.php, pone a trabajar el ciclo foreach, recorriendo la matriz $actores de esta forma:

```
<html>
  <head>
    <title>
      Uso de un ciclo foreach para recorrer una matriz
    </title>
  </head>
  <body>
    <h1>
      Uso de un ciclo foreach para recorrer una matriz
    </h1>
   <?php
      $actores[0] = "Cary Grant";
      $actores[1] = "Myrna Loy";
      $actores[2] = "Lorne Green";
```

```
      foreach ($actores as $valor) {
        echo "Valor: $valor <br>";
      }    ?>
  </body>
</html>
```

Aquí, el ciclo foreach completa la variable $valor, con un nuevo elemento de la matriz $actores cada vez que la recorre. Puede ver los resultados en la Figura 3-9.

Puede utilizar la segunda forma del ciclo foreach (la versión que le permite trabajar con claves y también con valores) en un ejemplo llamado phpforeach2.php. Ésta tiene este aspecto:

```
<html>
  <head>
    <title>
      Uso de un ciclo foreach con claves y valores en una matriz
    </title>
  </head>
  <body>
    <h1>
      Uso de un ciclo foreach con claves y valores en una matriz
    </h1>
   <?php
     $deudas["Jimena"] = 2493.77;
     $deudas["Roberto"] = 4930.33;
     $deudas["Samuel"] = 5493.22;

      foreach ($deudas as $clave => $valor) {
        echo "Clave: $clave; Valor: $valor <br>";
      }
    ?>
  </body>
</html>
```

Puede ver los resultados en la Figura 3-10.

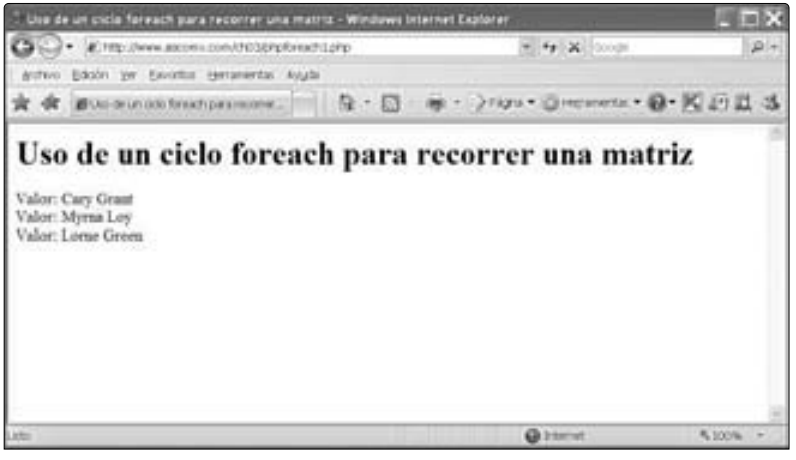

FIGURA 3-9 Uso del ciclo foreach en una matriz en PHP

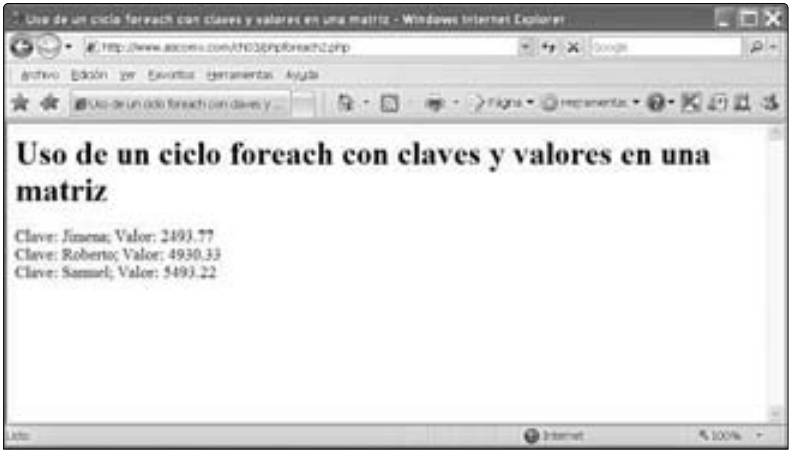

FIGURA 3-10 Uso del ciclo foreach con claves y valores en una matriz en PHP

Ciclo while

Puede utilizar también un ciclo while para realizar iteraciones en una matriz en PHP, si emplea una nueva función *each*. Ésta fue diseñada para utilizarse en ciclos por colecciones como matrices; cada vez que recorre la matriz devuelve la clave y valor del elemento actual. Para manejar un valor de retorno de múltiples elementos desde la función each, puede utilizar la función *list* de PHP, que asignará los dos valores de retorno de cada uno a variables independientes. Así es como se ve en phparraywhile.php:

```
<html>
  <head>
    <title>
      Uso de un ciclo while con claves y valores en una matriz
    </title>
  </head>
  <body>
    <h1>
      Uso de un ciclo while con claves y valores en una matriz
    </h1>
   <?php
     $deudas["Jimena"] = 2493.77;
     $deudas["Roberto"] = 4930.33;
     $deudas["Samuel"] = 5493.22;

      while (list($clave, $valor) = each ($deudas)) {
        echo "Clave: $clave; Valor: $valor <br>";
      }
    ?>
  </body>
</html>
```

Los resultados aparecen en la Figura 3-11.

FIGURA 3-11 Uso del ciclo while con claves y valores en una matriz en PHP

Funciones de matriz de PHP

PHP ofrece mucho soporte para el trabajo con matrices. Existen muchas funciones de cadena integradas en PHP —y también múltiples funciones de matriz—. Puede verlas en la Tabla 3-2.

Función	Hace esto
array_change_key_case	Devuelve una matriz con todas las claves de cadena convertidas a minúsculas o mayúsculas
array_chunk	Divide una matriz en segmentos
array_combine	Crea una matriz a partir de dos matrices, una para claves y otra para valores
array_count_values	Cuenta los valores en una matriz
array_diff_assoc	Calcula la diferencia de dos matrices con verificación de índice adicional
array_diff_uassoc	Calcula la diferencia de dos matrices con verificación de índice adicional, realizada por una función de devolución de llamada que proporciona el usuario
array_diff	Calcula la diferencia de matrices
array_fill	Rellena una matriz con valores
array_filter	Filtra elementos de una matriz utilizando una función de devolución de llamada
array_flip	Intercambia todas las claves con sus valores asociados en una matriz
array_intersect_assoc	Calcula la intersección de matrices con verificación de índice adicional
array_intersect	Calcula la intersección de matrices

TABLA 3-2 Funciones de matriz

Función	Hace esto
array_key_exists	Comprueba si la clave o índice dado existe en la matriz
array_keys	Devuelve las claves contenidas en una matriz
array_map	Aplica la devolución de la llamada a los elementos de las matrices dadas
array_merge_recursive	Combina dos o más matrices de forma recursiva
array_merge	Combina dos o más matrices
array_multisort	Ordena matrices múltiples o multidimensionales
array_pad	Rellena la matriz hasta la longitud especificada con un valor
array_pop	Retira el elemento del final de la matriz
array_push	Desplaza uno o más elementos hacia el final de la matriz
array_rand	Toma uno o más elementos al azar de una matriz
array_reduce	Reduce la matriz a un valor con una función de devolución de llamada
array_reverse	Devuelve una matriz con elementos en orden inverso
array_search	Busca un valor dado en la matriz y devuelve la clave correspondiente
array_shift	Desplaza un elemento del inicio de una matriz
array_slice	Extrae una porción de la matriz
array_splice	Elimina parte de la matriz y la reemplaza con algo más
array_sum	Calcula la suma de valores en una matriz
array_udiff_assoc	Calcula la diferencia de matrices con una verificación de índice adicional. Aquí, los datos se comparan mediante una función de devolución de llamada
array_udiff_uassoc	Calcula la diferencia de matrices con una verificación de índice adicional. Aquí, los datos se comparan utilizando una función de devolución de llamada y la verificación del índice es realizada también por la devolución de llamada
array_udiff	Calcula la diferencia entre matrices utilizando una función de devolución de llamada
array_unique	Elimina elementos duplicados de una matriz
array_unshift	Agrega uno o más elementos al inicio de una matriz
array_values	Devuelve todos los valores de una matriz
array_walk	Llama a una función provista por el usuario en todos los miembros de una matriz
array	Crea una matriz
arsort	Clasifica una matriz en orden invertido, preservando la asociación del índice
compact	Crea una matriz conteniendo variables y sus valores
count	Cuenta los elementos contenidos en una matriz
current	Devuelve el elemento actual en una matriz

Tabla 3-2 Funciones de matriz (*continuación*)

Función	Hace esto
each	Devuelve el par de clave y valor actual de una matriz y avanza el cursor en la matriz
end	Fija el puntero de la matriz en el último elemento
extract	Importa variables a la tabla de símbolos actual desde una matriz
in_array	Verifica si existe un valor en una matriz
key	Obtiene una clave de una matriz asociada
krsort	Clasifica una matriz por clave en orden inverso
ksort	Clasifica una matriz por clave
list	Asigna variables como si fueran una matriz
natcasesort	Clasifica una matriz utilizando un algoritmo de "orden natural" que distingue mayúsculas y minúsculas
natsort	Clasifica una matriz mediante un algoritmo de "orden natural"
next	Avanza el puntero de una matriz
pos	Alias de la función current
prev	Devuelve el puntero de la matriz al elemento
range	Crea una matriz conteniendo un rango de elementos
reset	Fija el puntero de una matriz a su primer elemento
rsort	Clasifica una matriz en orden invertido
shuffle	Mezcla los elementos de una matriz
sizeof	Alias de la función count
sort	Ordena una matriz
uasort	Ordena una matriz con una función de comparación definida por el usuario, manteniendo índices
uksort	Ordena una matriz por claves; utiliza una función de comparación definida por el usuario
usort	Ordena una matriz por valores; utiliza una función de comparación definida por el usuario

TABLA 3-2 Funciones de matriz (*continuación*)

Existen muchas funciones de matriz aquí (eso es mucho poder de matrices). En lo que resta del capítulo le mostramos los tipos de cosas que pueden hacer estas funciones, como la "implosión" y "explosión" de matrices.

Conversión entre cadenas y matrices utilizando implode y explode

La función implode contrae una matriz en una cadena y la función explode expande una cadena en una matriz. Eso es útil si ha guardado sus datos de cadenas en archivos y desea convertir esas cadenas en elementos de una matriz, al ejecutar su aplicación Web.

En este ejemplo se "implota" y muestra una matriz en una cadena de texto:

```
<?php
    $helado[0] = "chocolate";
    $helado[1] = "nuez";
    $helado[2] = "fresa";

    $texto = implode(", ", $helado);
    echo $texto;
?>
```

Esto da como resultado

```
chocolate, nuez, fresa
```

Como puede ver, usted mismo especifica la cadena que separa los elementos de la matriz (en este caso se trata de“,”).

¿Qué tal si explosionamos una cadena en una matriz? Para hacerlo, usted indica el texto en el que desea dividir la cadena, como“,”, y lo pasa a la función explode. Éste es un ejemplo:

```
<?php
    $texto = "chocolate, nuez, fresa";
    $helado = explode(", ", $texto);
    print_r($helado);
?>
```

Y éstos son los resultados (explotó la cadena en una matriz):

```
Array
(
    [0] => chocolate
    [1] => nuez
    [2] => fresa
)
```

Extracción de datos de matrices

Puede usar la función extract para extraer datos de matrices y almacenarlos en variables. Suponga que tiene esta matriz:

```
<?php
    $helado["bueno”] = "naranja";
    $helado["mejor”] = "vainilla";
    $helado["el mejor”] = "ron con pasas";
?>
```

Ahora puede utilizar la función extract para crear variables cuyos nombres serán tomados de las claves de la matriz y a esas variables se les asignarán los valores contenidos en la matriz. Así es como esto funciona en phpextract.php:

```
<html>
  <head>
    <title>
      Extracción de variables de matrices
    </title>
  </head>
```

```
  <body>
    <h1>Extracción de variables de matrices</h1>
    <?php
      $helado["bueno"] = "naranja";
      $helado["mejor"] = "vainilla";
      $helado["el_mejor"] = "ron con pasas";

      extract($helado);

      echo "\$bueno = $bueno<BR>";
      echo "\$mejor = $mejor<BR>";
      echo "\$el_mejor = $el_mejor<BR>";
    ?>
  </body>
</html>
```

Puede ver esta página en acción en la Figura 3-12.

De hecho, puede usar también la función list para extraer variables de una matriz y asignar los nombres que guste a esas variables. Éste es un ejemplo:

```
<?php
    $helado[0] = "chocolate";
    $helado[1] = "nuez";
    $helado[2] = "fresa";
    list($uno, $dos) = $helado;
    echo $uno, "<br>";
    echo $dos;
?>
```

Esto es lo que obtiene:

```
chocolate
nuez
fresa
```

FIGURA 3-12 Uso de la función extract con matrices en PHP

Clasificación de matrices

Hay mucho poder para clasificar matrices encerrado en PHP, comenzando con la función sort, que clasifica matrices en orden ascendente. Éste es un ejemplo, phpsortarray.php:

```
<?php
    $helado[0] = "naranja";
    $helado[1] = "vainilla";
    $helado[2] = "ron con pasas";

    print_r($helado);

    sort($helado);

    print_r($helado);
?>
```

Éste es el resultado de phpsortarray.php:

```
%php phpsortarray.php
Array
(
    [0] => naranja
    [1] => vainilla
    [2] => ron con pasas
)
Array
(
    [0] => naranja
    [1] => ron con pasas
    [2] => vainilla
)
```

¿Qué tal una clasificación invertida? Puede hacerlo con rsort:

```
<?php
    $helado = "naranja";
    $helado = "vainilla";
    $helado = "ron con pasas";

    print_r($helado);

    rsort($helado);

    print_r($helado);
?>
```

que da como resultado

```
Array
(
    [0] => naranja
    [1] => vainilla
    [2] => ron con pasas
)
```

```
Array
(
    [0] => vainilla
    [1] => ron con pasas
    [2] => naranja
)
```

Esto no funciona del todo bien si utiliza claves de cadenas de texto, en vez de valores de índices numéricos —las claves de las cadenas de texto serán reemplazadas con números—. Utilice asort en su lugar, como en phpasortarray.php:

```
<?php
  $helado["bueno"] = "naranja";
  $helado["mejor"] = "vainilla";
  $helado["el_mejor"] = "ron con pasas";

  print_r($helado);

  asort($helado);

  print_r($helado);
?>
```

Esto es lo que obtiene:

```
%php phpasortarray.php
Array
(
    [bueno] => naranja
    [mejor] => vainilla
    [el mejor] => ron con pasas
)
Array
(
    [bueno] => naranja
    [el mejor] => ron con pasas
    [mejor] => vainilla
)
```

Y puede usar asort para clasificar matrices en orden invertido.

Podría preguntar: ¿qué sucedería en caso de clasificar una matriz basándose en claves y no valores? Utilice ksort en su lugar. Para clasificar en orden invertido por claves, use krsort. Incluso puede definir sus propias operaciones de clasificación con una función de clasificación personalizada que recurre a la función usort de PHP. Asimismo, natsort realiza clasificaciones "naturales" con datos mixtos de cadenas y números. Por ejemplo, natsort clasificaría "opción1","opción2", "opción15" como "opción1","opción2","opción15" (no como "opción1","opción15","opción2", que sería el resultado de sort).

Uso de operadores de matriz de PHP

PHP le permite operar también en matrices con operadores, que puede ser muy útil. Éstos son los operadores de manejo de matrices:

- $a + $b produce la unión de $a y $b.
- $a == $b produce TRUE si $a y $b tienen los mismos elementos.
- $a === $b produce TRUE si $a y $b tienen los mismos elementos en el mismo orden.
- $a != $b produce TRUE si $a no es igual a $b.
- $a <> $b produce TRUE si $a no es igual a $b.
- $a !== $b produce TRUE si $a no es idéntico a $b.

Éste es un ejemplo, phparrayoperators.php, que pone a trabajar los operadores de adición e igualdad:

```
<?php
  $budín["frambuesa"] = 333;
  $budín["durazno"] = 222;
  $helado["nuez"] = 111;
  $helado["chocolate"] = 999;
  echo "\$helado: ";
  print_r($helado);
  echo "\n";
  echo "\$budín: ";
  print_r($budín);
  echo "\n";
  $postres = $budín + $helado;
  echo "\$postres: ";
  print_r($postres);
  echo "\n";
  if ($budín == $helado){
    echo "\$budín tiene los mismos elementos que \$helado";
  }
  else {
    echo "\$budín no tiene los mismos elementos que \$helado";
  }
?>
```

Esto es lo que obtiene cuando ejecuta este código:

```
%php phparrayoperators.php
  $helado: Array
(
    [nuez] => 111
    [chocolate] => 999
)

$budín: Array
(
    [frambuesa] => 333
    [durazno] => 222
)
```

```
$postres: Array
(
    [frambuesa] => 333
    [durazno] => 222
    [nuez] => 111
    [chocolate] => 999
)

$budín no tiene los mismos elementos que $helado.
```

Comparación de unas matrices con otras

Puede hallar la diferencia entre dos matrices utilizando la función array_diff; éste es un ejemplo, phparraydiff.php:

```
<?php
  $helado = array("vainilla", "chocolate", "fresa");
  $helado2 = array("vainilla", "chocolate", "papaya");

  $diferencia = array_diff($helado, $helado2);

  foreach ($diferencia as $clave => $valor) {
      echo "Clave: $clave; Valor: $valor\n";
  }
?>
```

Este ejemplo compara dos matrices y muestra los elementos diferentes. Esto es lo que se obtiene:

```
%php phparraydiff.php
Clave: 2; Valor: fresa
```

Si utiliza claves de cadenas de texto, utilice la función array_diff_assoc (matrices con claves de cadenas de texto, también son conocidas como matrices asociativas). Si desea hallar todos los elementos que dos matrices tienen en común, simplemente use array_intersect o array_intersect_assoc.

Manejo de matrices multidimensionales

PHP puede trabajar también con matrices multidimensionales. Por ejemplo, suponga que lleva el registro de calificaciones de estudiantes en un examen:

```
$calificaciones["Samuel"] = 79;
$calificaciones["Elena"] = 69;
```

Ahora suponga que aplicará un segundo examen. Podría simplemente agregar otro índice (del número de examen) a cada estudiante:

```
<?php
    $calificaciones["Samuel"] [1] = 79;
    $calificaciones["Samuel"] [2] = 74;
```

```
    $calificaciones["Elena"] [1] = 69;
    $calificaciones["Elena"] [2] = 84;
    print_r($calificaciones);
?>
```

En este punto, $calificaciones[“Samuel”][1] es la calificación de Samuel en el primer examen; $calificaciones[“Samuel”][2] es su calificación en el segundo examen, etc. Este script muestra la nueva matriz multidimensional con print_r, que produce:

```
[Samuel] => Array
       (
           [1] => 79
           [2] => 74
        )

[Elena] => Array
       (
           [1] => 69
           [2] => 84
        )
```

Además, puede tener acceso a elementos utilizando ambos índices, de la siguiente forma:

```
echo "La calificación de Samuel en el primer examen fue ", $calificaciones["Samuel"]
[1], "\n";
```

Si desea interpolar un elemento de una matriz entre comillas dobles, se requiere un poco más de trabajo (tiene que encerrarlo entre llaves):

```
echo "La calificación de Samuel en el primer examen fue {$calificaciones["Samuel"]
[1]}\n";
```

Como se esperaría, también puede crear matrices multidimensionales de esta forma (observe que esto iniciará las matrices con un valor índice de 0):

```
<?php
    $calificaciones["Samuel"] [] = 79;
    $calificaciones["Samuel"] [] = 74;
    $calificaciones["Elena"] [] = 69;
    $calificaciones["Elena"] [] = 84;
    print_r($calificaciones);
?>
```

También puede considerar las matrices multidimensionales como matrices de matrices. Por ejemplo, puede considerar una matriz bidimensional como matriz unidimensional (las filas de la matriz bidimensional) de matrices unidimensionales (las columnas de la matriz bidimensional). Esto significa que la siguiente sintaxis es válida:

```
<?php
    $calificaciones = array("Samuel" => array(79, 74), "Elena" => array(69, 84));
    print_r($calificaciones);
?>
```

Esto es lo que se obtiene de este código:

```
[Samuel] => Array
        (
            [0] => 79
            [1] => 74
         )

[Elena] => Array
        (
            [0] => 69
            [1] => 84
         )
```

Podría incluso iniciar los índices de la matriz en 1 y no en 0 de esta forma:

```
<?php
    $calificaciones = array("Samuel" => array(1 => 79, 2 => 74), "Elena" => array(1
       => 69, 2 => 84));
    print_r($calificaciones);
?>
```

Y esto es lo que se obtiene:

```
[Samuel] => Array
        (
            [1] => 79
            [2] => 74
         )

[Elena] => Array
        (
            [1] => 69
            [2] => 84
         )
```

Uso de matrices multidimensionales en ciclos

El trabajo con matrices multidimensionales en ciclos requiere un poco de reflexión extra. Por ejemplo, suponga que tiene una matriz bidimensional llamada $calificaciones y quiere recorrerla, mostrando cada elemento contenido en ella. Podría comenzar con un ciclo por el índice exterior, que va de 0 a 1:

```
      <?php
        $calificaciones[0] [] = 79;
        $calificaciones[0] [] = 74;
        $calificaciones[1] [] = 69;
        $calificaciones[1] [] = 84;
        for $exterior = 0; $exterior < count($calificaciones); $exterior++){
          .
          .
          .
        }
?>
```

Ahora puede realizar un ciclo por el índice interior de esta forma en phpmultiloop.php:

```
<html>
  <head>
    <title>
      Recorrido por matrices multidimensionales
    </title>
  </head>

  <body>
    <h1>
      Recorrido por matrices multidimensionales
    </h1>
    <?php
      $calificaciones[0][] = 79;
      $calificaciones[0][] = 74;
      $calificaciones[1][] = 69;
      $calificaciones[1][] = 84;
      for ($exterior = 0; $exterior < count($calificaciones); $exterior++){
        for($interior = 0; $interior < count($calificaciones[$exterior]); $interior++){
          echo "\$calificaciones[$exterior][$interior] = ",
            $calificaciones[$exterior][$interior], "<br>";
        }
      }
    ?>
  </body>
</html>
```

Puede ver los resultados, donde, en la práctica, la matriz bidimensional se ha recorrido, en la Figura 3-13.

FIGURA 3-13 Recorrido de una matriz multidimensional en PHP

Desplazamiento por matrices

PHP da soporte a varias funciones que puede utilizar para recorrer matrices. Esta navegación se realiza con un *puntero de matriz*, que lleva el registro de su ubicación en la matriz, entre llamadas a las funciones de navegación. Suponga que tiene esta matriz:

```
$helado[0] = "chocolate";
$helado[1] = "nuez";
$helado[2] = "fresa";
```

Puede acceder al elemento actual en la matriz con la función current:

```
echo "Current: ", current($helado), "<BR>;
```

Puede mover el puntero de la matriz al siguiente elemento con la función next:

```
echo "Next: ", next($helado), "<BR>;
```

La función prev devuelve el puntero al elemento anterior:

```
echo "Prev: ", prev($helado), "<BR>;
```

La función end mueve el puntero al último elemento de la matriz:

```
echo "End: ", end($helado), "<BR>;
```

Si desea retroceder al principio de la matriz, utilice la función reset:

```
reset($helado);
```

Éste es un ejemplo, phpnavigatearray.php, poniendo estas funciones a trabajar:

```
<html>
   <head>
            <title>
                Navegación por matrices
            </title>
    </head>

    <body>
      <h1>
      Navegación por matrices
    </h1>
    <?php
      $helado[0] = "chocolate";
      $helado[1] = "nuez";
      $helado[2] = "fresa";

      echo "Elemento actual: ", current($helado), "<br>";
      echo "Siguiente elemento: ", next($helado), "<br>";
      echo "Elemento anterior: ", prev($helado), "<br>";
      echo "Elemento final: ", end($helado), "<br>";
      echo "Reiniciando la matriz...<br>”;
```

FIGURA 3-14 Navegación por una matriz en PHP

```
            reset($helado);
            echo "Elemento actual: ", current($helado), "<br>";

        ?>
    </body>
</html>
```

Puede ver los resultados, donde ha recorrido la matriz, en la Figura 3-14.

Separación y combinación de matrices

Puede obtener secciones de matrices con la función array_slice, pasándola a la matriz de la que desea obtener una sección, la bifurcación en qué comenzar y la longitud de la matriz que desea crear. Éste es un ejemplo, phparraysplit.php:

```
<html>
  <head>
    <title>
      Separación de matrices
    </title>
  </head>

  <body>
    <h1>
      Separación de matrices
    </h1>
    <?php
      $helado["bueno”] = "naranja";
      $helado["mejor”] = "vainilla";
      $helado["el_mejor"] = "ron con pasas";
      $helado["superior"] = "limón";
      $submatriz = array_slice($helado, 1, 2);
```

```
      foreach ($submatriz as $valor) {
        echo "$valor <br>";
      }
      ?>
    </body>
</html>
```

Y puede ver qué produce esto en la Figura 3-15 —ha extraído una sección de una matriz.

También es posible combinar matrices con la función array_merge, como se ve en phpmergearrays.php:

```
<html>
  <head>
    <title>
      Combinación de matrices
    </title>
  </head>

  <body>
    <h1>
      Combinación de matrices
    </h1>
    <?php
      $budín = array("vainilla", "ron con pasas", "naranja");
      $helado = array("chocolate", "nuez", "fresa");

      $postres = array_merge($budín, $helado);

      foreach ($postres as $valor) {
        echo "$valor <br>";
      }
```

FIGURA 3-15 Cómo obtener una sección de una matriz en PHP

FIGURA 3-16 Combinación de matrices en PHP

```
    ?>
  </body>
</html>
```

Se aprecia el resultado en la Figura 3-16, donde se han combinado ambas matrices.

Otras funciones de matriz

Existen otras funciones de matriz que analizaremos también, como array_sum, que suma los elementos numéricos de una matriz. Por ejemplo, ésta es la forma en que se obtiene la calificación promedio de un grupo de estudiantes:

```
<?php
    $calificaciones = array(65, 61, 70, 64, 65);

    echo "La calificación promedio es ", array_sum($calificaciones) /
count($calificaciones);
?>
```

Esto se ve:

```
La calificación promedio es 65
```

Además, la función array_flip intercambiará las claves y valores de una matriz. Puede verla en acción en phparrayflip.php:

```
<html>
  <head>
    <title>
      Intercambio de claves y valores en matrices
    </title>
  </head>
```

```
  <body>
    <h1>
      Intercambio de claves y valores en matrices
    </h1>
    <?php
      $helado = array("sabor_1" => "vainilla", "sabor_2" => "ron con pasas",
        "sabor_3" => "naranja");

      foreach ($helado as $clave => $valor) {
        echo "Clave: $clave; Valor: $valor<br>";
      }
      echo "<br>";

      $helado = array_flip($helado);

      foreach ($helado as $clave => $valor) {
        echo "Clave: $clave; Valor: $valor<br>";
      }
    ?>
  </body>
</html>
```

Los resultados se aprecian en la Figura 3-17.

Por último, la función array_unique eliminará duplicados de una matriz. Observe este ejemplo, phparrayunique.php:

```
<?php
    $calificaciones = array(65, 61, 70, 64, 65);
    print_r($calificaciones);
    $calificaciones = array_unique($calificaciones);
    print_r($calificaciones);
?>
```

FIGURA 3-17 Intercambio de matrices en PHP

Esto se obtiene cuando ejecuta este script:

```
%php phparrayunique.php
Array
(
    [0] => 65
    [1] => 61
    [2] => 70
    [3] => 64
    [4] => 65
)
Array
(
    [0] => 65
    [1] => 61
    [2] => 70
    [3] => 64
)
```

Como puede apreciar, se ha eliminado el elemento duplicado de la matriz.

CAPÍTULO 4

Creación de funciones

Es el momento de dar el siguiente paso en el poder de la programación y ello significa crear sus propias funciones. Ya ha visto muchas funciones integradas de PHP —las funciones son aquellas secciones de código a las que se puede llamar y pasar datos, para devolver datos a usted.

Las funciones le permiten dividir su código, siguiendo ese antiguo mandato de la programación: divide y vencerás. ¿Tiene código que necesita ejecutar múltiples veces en su aplicación? Póngalo en una función. ¿Tiene código que no debe ejecutarse automáticamente cuando carga la página (como sucede con todo el código visto hasta ahora)? Póngalo en una función. ¿Tiene código demasiado extenso para depurarlo y mantenerlo con efectividad? Póngalo en una función.

Al poner código en una función lo saca del camino —la idea es ponerlo fuera de vista, de su mente—. Usted pasa datos a esa función y ésta le devolverá datos. Es posible que haya visto funciones en otros lenguajes, pero como notará, las funciones de PHP admiten muchas características únicas.

La restricción del código en funciones divide el código de forma especialmente útil: las variables usadas en una función no son visibles de forma predeterminada fuera de esa función. Eso significa que no deberá preocuparse por conflictos con nombres de variables: el uso de una variable $cuenta en dos secciones distintas de código pueden ocasionar conflicto, pero no si encierra código dentro de una función.

El uso de funciones es sólo el principio de la división del código —el uso de la programación orientada a objetos le permitirá integrar funciones y datos en clases y objetos—. Es decir, las funciones eran decisivas para permitir a los programadores segmentar su código; pero con el tiempo, las necesidades de los programadores fueron demasiado grandes para funciones simples, entonces la combinación de funciones y datos en objetos fue el siguiente paso. Sin embargo, todo comienza con funciones y ahora abordaremos ese tema.

Creación de funciones en PHP

¿Cómo se construyen funciones en PHP? Eso es bastante sencillo —así se hace, hablando formalmente:

```
function nombre_función([lista_argumentos...])
{
    [instrucciones]
    [return valor_retorno;]
}
```

Es el momento de ver un ejemplo. Podría crear una función llamada display que muestre texto en el navegador, en un ejemplo llamado phpdisplay.php. Para hacer que phpdisplay.php muestre texto indicando que está a punto de llamar a la función:

```
<?php
  echo "A punto de llamar a la función...<br>";
  echo "Llamando a la función...<br>";
  .
  .
  .
?>
```

Después, agregue el código para la función display; se ve como éste:

```
<?php
  echo "A punto de llamar a la función...<br>";
  echo "Llamando a la función...<br>";

  function display()
  {
     echo "Este texto fue mostrado por la función.";
  }
?>
```

Hay un par de cosas que deben observarse. El código contenido en una función de PHP, no se ejecuta hasta llamar a esa función. Es diferente de scripts vistos hasta ahora —si sólo coloca código en un script, fuera de una función, éste se ejecutará cuando cargue la página.

Observe también la sintaxis que define la función; esa definición comienza con la función keyword, seguida del nombre de la función, a su vez de un paréntesis (vacíos en este punto, pues no está pasando datos a la función). El cuerpo de la función —es decir, las instrucciones a ejecutarse cuando llame a la función— está encerrado entre llaves.

Bueno, eso crea la función; ¿cómo la llamará? Se invoca la función display, como haría en cualquier otro lenguaje de programación —utilizando su nombre como instrucción, seguido de paréntesis (paréntesis vacíos en este ejemplo, porque no pasará datos a la función display):

```
<html>
  <head>
    <title>
      Creación de funciones
    </title>
  </head>
  <body>
    <h1>
      Creación de funciones
    </h1>
    <?php
      echo "A punto de llamar a la función...<br>";
      echo "Llamando a la función...<br>";
      display();
```

FIGURA 4-1 Creación de la función display

```
      function display()
      {
         echo "Este texto fue mostrado por la función.";
      }
    ?>
  </body>
</html>
```

Puede ver este ejemplo en acción en la Figura 4-1 (observe que se llamó a la función display y ésta hizo su labor, mostrar texto). No está mal.

Paso de datos a funciones

¿Cómo se pasan datos a funciones para que puedan operar con ellos? Como en otros lenguajes, de eso trata la sección lista_argumentos, de la sintaxis de la función:

```
function nombre_función([lista_argumentos...])
{
    [instrucciones]
    [return valor_retorno;]
}
```

Éste es un ejemplo, phppassdata.php. Suponga que quisiera pasar los datos de texto a la función display que la función debe mostrar. Puede agregar un argumento de función a la lista de los argumentos ya existentes, de la siguiente forma, en la función display:

```
function display($saludo)
{
  .
  .
  .
}
```

Ahora puede hacer referencia a los datos procesados mediante un nombre, $saludo, de esta forma en el código de la función display:

```
function display($saludo)
{
   echo $saludo;
}
```

Así es como posibilita que una función admita datos que se le envían —usted asigna la nomenclatura que desea manejar para datos conformando la lista de argumentos de la función, para luego referirse a ellos en el código de la función.

Se pasan datos a una función, colocándolos entre paréntesis, inmediatamente después del nombre de la función cuando se llama a esa función. Por ejemplo, si deseara pasar el texto "¡Hola!" a la función display, para hacer que muestre ese texto, podría hacerlo de esta forma:

```
display("¡Hola!");
  .
  .
  .
function display($saludo)
{
   echo $saludo;
}
```

Ahora bien, cuando se llama a la función display de esta forma, al argumento $saludo (también llamado parámetro) se le asigna el valor "¡Hola!"; al ejecutar la línea echo $saludo, aparecerá "¡Hola!" en el navegador.

¿Qué sucedería en caso de pasar dos elementos de datos a una función? Puede listarlos en la lista de argumentos de la función, separados por comas. Por ejemplo, para lograr que la función display acepte dos argumentos, éstos se listan simplemente por nombre, por ejemplo, $saludo y $mensaje:

```
function display($saludo, $mensaje)
{
   echo $saludo;
}
```

Ahora, es posible referirse a $saludo y $mensaje por nombre en el código de su función:

```
function display($saludo, $mensaje)
{
   echo $saludo;
   echo $mensaje;
}
```

Cuando llama a la función display, usted lista los argumentos que desea pasar a esa función, separados por comas, de la siguiente forma en phppassdata.php:

```
<html>
  <head>
    <title>
      Paso de datos a funciones
    </title>
```

FIGURA 4-2 Paso de datos a funciones

```
  </head>
  <body>
    <h1>
      Paso de datos a funciones
    </h1>
    <?php
      echo "A punto de llamar a la función...<br>";
      echo "Pasando datos a la función...<br>";
      display("¡Hola", " a todos!");

      function display($saludo, $mensaje)
      {
         echo $saludo;
         echo $mensaje;
      }
    ?>
  </body>
</html>
```

Puede ver los resultados en la Figura 4-2, donde los datos se han asignado a display.

Paso de matrices a funciones

También puede pasar matrices a funciones con la facilidad que trabajaría en elementos de datos, cadenas o números. Por ejemplo, podría querer una función para promediar las calificaciones en un examen de un grupo de estudiantes y muestre ese promedio, llamada averager. Y podría desear pasar una matriz a la función averager.

Así es como se vería la matriz con las calificaciones del examen:

```
<?php
  $calificaciones = array(65, 32, 78, 98, 66);
        .
        .
        .
?>
```

Y es así como podría pasar la matriz $calificaciones a la función averager:

```
<?php
  $calificaciones = array(65, 32, 78, 98, 66);

  averager($calificaciones);
        .
        .
        .
?>
```

En la función averager, usted especifica el nombre de la matriz, a la que llamaremos simplemente $matriz aquí:

```
<?php
  $calificaciones = array(65, 32, 78, 98, 66);

  averager($calificaciones);
        .
        .
        .
  function averager($matriz)
  {
  }
?>
```

En la función averager, puede utilizar una instrucción foreach para recorrer la matriz:

```
<?php
  $scores = array(65, 32, 78, 98, 66);

  averager($calificaciones);
        .
        .
        .
  function averager($matriz)
  {
    foreach ($matriz as $valor) {
       .
       .
       .
    }
  }
?>
```

Luego podría sumar todos los elementos de la matriz en una variable llamada $total:

```
<?php
  $scores = array(65, 32, 78, 98, 66);

  averager($calificaciones);
        .
        .
        .
  function averager($matriz)
  {
    $total = 0;
    foreach ($matriz as $valor) {
       $total += $valor;
    }
  }
?>
```

Por último, puede determinar el valor promedio de los elementos de la matriz y mostrarlo en phpassarray.php:

```
<html>
  <head>
    <title>
      Paso de matrices a funciones
    </title>
  </head>
  <body>
    <h1>
      Paso de matrices a funciones
    </h1>
    <?php
      $calificaciones = array(65, 32, 78, 98, 66);
      averager($calificaciones);

      function averager($matriz)
      {
        $total = 0;
        foreach ($matriz as $valor) {
           $total += $valor;
        }
        if(count($matriz) > 0){
         echo "El promedio fue ", $total/count($matriz);
        } else {
         echo "¡No hay elementos para promediar!";
        }
  }
    ?>
  </body>
</html>
```

Y puede ver el resultado (el promedio fue 67.8, como aparece en la Figura 4-3).

FIGURA 4-3 Paso de matrices a funciones

Paso por referencia

Cuando se pasan datos a una función, lo que realmente se moviliza es una *copia* de esos datos. Así que, por ejemplo, si pasa una variable, se hace una copia de ella y ésta es la que realmente se toma en la función.

¿Qué sucedería si deseara pasar un valor real a la función? Suponga, por ejemplo, que deseara una función para alterar el valor de una variable. Por ejemplo, suponga que tuviera una variable $valor:

```
<?php
  $valor = 4;
    .
    .
    .
?>
```

y deseara tener una función para elevar al cuadrado la cantidad contenida en $valor. Podría intentarlo moviendo $valor a una función llamada, por ejemplo, squarer, elevando al cuadrado el número que se le asigna:

```
function squarer($número)
{
   $número *= $número;
}
```

Sin embargo, no funcionará, pues los argumentos que se pasan a la función squarer deben pasar por valor. Puede cambiar eso anteponiendo un signo & al argumento que desea pasar por referencia —que hará a PHP considerar el argumento como referencia—. Cuando se pasa un argumento por referencia, eso da al código de la función acceso directo al argumento de vuelta en el código que hace la llamada; de modo que para elevar al cuadrado el valor en $valor, todo lo que

debe hacer es anteponer al argumento el signo &, en la lista de argumentos como en phpreference.php:

```
<html>
  <head>
    <title>
      Paso de datos a funciones por referencia
    </title>
  </head>
  <body>
    <h1>
      Paso de datos a funciones por referencia
    </h1>
    <?php
      $valor = 4;

      echo "Antes de la llamada, \$valor contiene $valor <br>";
      squarer($valor);
      echo "Después de la llamada, \$valor contiene $valor <br>";

      function squarer(&$número)
      {
         $número *= $número;
      }
    ?>
  </body>
</html>
```

Y verá los resultados en la Figura 4-4 donde, como puede apreciar, se elevaron al cuadrado los datos en $valor, de vuelta al código que hizo la llamada. Formidable.

FIGURA 4-4 Paso de datos a funciones por referencia

Uso de argumentos predeterminados

¿Qué sucede si tiene una función llamada display, tomando dos argumentos de esta forma?:

```
function display($saludo, $mensaje)
{
   echo $saludo;
   echo $mensaje;
}
```

y la llama con un argumento:

```
display("No hay preocupaciones");
```

A PHP no le va a gustar eso; éste es el tipo de advertencia que recibe:

```
PHP Warning:  Missing argument 2 for display() in phpdisplay.php on line 5
```

Por otra parte, puede resolver este problema proporcionando un argumento *predeterminado* aquí.

Así es como funciona: usted agrega el argumento predeterminado a la lista de argumentos, utilizando un signo de igual, como se puede ver aquí:

```
function display($saludo, $mensaje = "¡Hola a todos!")
{
   echo $saludo;
   echo $mensaje;
}
```

Ahora bien, si nada del segundo argumento pasa a la función display, el argumento predeterminado se utiliza automáticamente. Así es como podría llamar a la función display en phpdefaultarguments.php:

```
<html>
  <head>
    <title>
      Uso de argumentos de función predeterminados
    </title>
  </head>
  <body>
    <h1>
      Uso de argumentos de función predeterminados
    </h1>
    <?php
      echo "A punto de llamar a la función...<br>";
      echo "Pasando datos a la función...<br>";
      display("El argumento predeterminado es: ");

      function display($saludo, $mensaje = "¡Hola a todos!")
      {
         echo $saludo;
         echo $mensaje;
      }
    ?>
  </body>
</html>
```

FIGURA 4-5 Uso de argumentos predeterminados

Y puede ver el resultado en la Figura 4-5, donde se usó en realidad el argumento predeterminado. ¿Desea proporcionar múltiples argumentos predeterminados? No hay problema:

```
function display($saludo, $mensaje = "¡Hola a todos!", $mensaje2 = "No hay preocupaciones.")
{
  echo $saludo;
  echo $mensaje;
  echo $mensaje2;
}
```

Puede dar valores predeterminados a más de un argumento, pero una vez que comience a asignarlos, tiene que extender la asignación a todos los argumentos siguientes, de modo que PHP no se confunda si falta más de un argumento.

Paso de números variables de argumentos

Mientras analizamos el paso de argumentos a funciones, cabe mencionar que puede configurar funciones para tomar un número variable de argumentos. En otras palabras, suponga que tiene una función llamada connector, enlazando palabras en una cadena de texto. La podría llamar de esta forma:

```
connector("Hola");
```

o así:

```
connector("Hola", "a todos");
```

o bien de esta forma:

```
connector("Hola", "a todos", "de nuevo");
```

¿Cómo se maneja un número variable de argumentos como éste en una función? Puede manejarlos con estas funciones:

- **func_num_args** Devuelve el número de argumentos pasados
- **func_get_arg** Devuelve un argumento
- **func_get_args** Devuelve todos los argumentos de una matriz

Por ejemplo, así es como se inicia la función connector —tomando una matriz de argumentos que se pasan a la función:

```
function connector()
{
    $argumentos = func_get_args();
        .
        .
        .
}
```

Luego puede recorrer los argumentos de esta forma —observe que puede obtener el número de argumentos de la función func_num_args:

```
function connector()
{
    $datos = "";
    $argumentos = func_get_args();
    for ($índice_ciclos = 0; $índice_ciclos < func_num_args(); $índice_ciclos++) {
        $datos .= $argumentos[$índice_ciclos] . " ";
    }
    echo $datos;
}
```

Note cómo la cadena final resultante se reproduce con echo. Así es como podría poner a trabajar la función connector en phpvariableargs.php:

```
<html>
  <head>
    <title>
      Paso de argumentos variables a funciones
    </title>
  </head>
  <body>
    <h1>
      Paso de argumentos variables a funciones
    </h1>
    <?php
      echo "Pasando '¿Cómo' 'van' 'las cosas?' a connector...<br>";
      echo "Se obtiene este resultado: ";
      connector("¿Cómo", "van", "las cosas?");
```

FIGURA 4-6 Paso de un número variable de argumentos a funciones

```
      function connector()
      {
        $datos = "";
        $argumentos = func_get_args();

        for ($índice_ciclos = 0; $índice_ciclos < func_num_args(); $índice_ciclos++)
{
          $datos .= $argumentos[$índice_ciclos] . " ";
        }
        echo $datos;
      }
    ?>
  </body>
</html>
```

Puede ver los resultados en la Figura 4-6, donde la función connector manejó los argumentos que se le pasaron.

Ahora, suponga que no deseaba que la función connector mostrara la cadena conectada, sino que la cadena fuera devuelta a su código para ponerla a trabajar usted mismo. Es muy fácil de hacer para las funciones —devolver valores que dan sus resultados)—.Y eso lo veremos a continuación.

Retorno de datos de funciones

Suponga que tiene una función simple llamada adder, para devolver la suma de dos números que se le asignan. Podría comenzar escribiendo la función adder de esta forma:

```
<?php
  function adder(operando_1, operando_2)
  {
    $suma = $operando_1 + $operando_2;
```

```
        .
        .
        .
}
?>
```

Ahora que ha obtenido la suma en la variable $suma, ¿cómo devuelve ese valor al código a cargo de hacer la llamada? Puede usar la instrucción return de PHP:

```
<?php
  function adder(operando_1, operando_2)
  {
    $suma = $operando_1 + $operando_2;

    return $suma;
}
?>
```

Esto devuelve el valor en $suma al código responsable de hacer la llamada. Así es como puede poner a trabajar la función adder —la expresión adder($valor_1, $valor_2), es reemplazada por el valor devuelto por la función adder, que es 5:

```
<?php
  $valor_1 = 2;
  $valor_2 = 3;

  echo "La suma de $valor_1 + $valor_2 es ", adder($valor_1, $valor_2);

  function adder(operando_1, operando_2)
  {
    $suma = $operando_1 + $operando_2;

    return $suma;
}
?>
```

Este script produce

```
La suma de 2 + 3 es 5
```

Éste es otro ejemplo usando la función connector del tema anterior. Sólo tiene que modificar la función connector para devolver sus datos de esta forma, como en phpretunrdata.php:

```
<html>
  <head>
    <title>
      Retorno de valores de funciones
    </title>
  </head>
  <body>
    <h1>
      Retorno de valores de funciones
    </h1>
```

FIGURA 4-7 Retorno de valores de funciones

```
    <?php
      echo "Pasando '¿Cómo' 'van' 'las cosas?' a connector...<br>";
      echo "Se obtiene este resultado: ", connector("¿Cómo", "van", "las cosas?");

      function connector()
      {
        $datos = "";
        $argumentos = func_get_args();

        for ($índice_ciclos = 0; $índice_ciclos < func_num_args(); $índice_ciclos++)
{
          $datos .= $argumentos[$índice_ciclos] . " ";
        }
        return $datos;
      }
    ?>
  </body>
</html>
```

Puede ver los resultados en la Figura 4-7, donde la función connector devolvió su cadena de resultado, que luego mostró el código haciendo la llamada.

Así se devuelven valores simples, como cadenas de funciones. ¿Qué tal las matrices? Ahora las analizaremos.

Retorno de matrices

Se pueden devolver matrices desde funciones tan fácil como se devuelven valores simples. Suponga que tiene una función, create_array, para crear matrices de la longitud especificada por usted, pasando un número a la función create_array. Esta función almacena 0 en el elemento cero, 1 en el primer elemento, 2 en el segundo elemento, y así sucesivamente.

De modo que si pasa un 3 a create_array, debe obtener la matriz [0, 1, 2]. Si pasa un 4, debe obtener [0, 1, 2, 3], etcétera. Así es como se inicia la función create_array:

```
function create_array($número)
{
  .
  .
  .
}
```

Así puede crear la matriz, $matriz, que devolverá esta función:

```
function create_array($número)
{
  for ($contador_ciclos = 0; $contador_ciclos < $número; $contador_ciclos++){
    $matriz[] = $contador_ciclos;
  }
  .
  .
  .
}
```

Esto creará la matriz llamada $matriz. Ahora puede devolverla desde la función create_array, como retornaría cualquier valor simple —con la instrucción return:

```
function create_array($número)
{
  for ($contador_ciclos = 0; $contador_ciclos < $número; $contador_ciclos++){
    $matriz[] = $contador_ciclos;
  }

  return $matriz;
}
```

Así puede poner a trabajar la función create_array, como en phpreturnarray.php:

```
<html>
  <head>
    <title>
      Retorno de matrices de funciones
    </title>
  </head>
  <body>
    <h1>
      Retorno de matrices de funciones
    </h1>
```

FIGURA 4-8 Retorno de matrices de funciones

```
    <?php
      $datos = create_array(3);
      echo "Ésta es la primera matriz:<br>";
      print_r($datos);
      echo "<br>";

      $datos_2 = create_array(4);
      echo "Ésta es la segunda matriz:<br>";
      print_r($datos_2);

      function create_array($número)
      {
        for ($contador_ciclos = 0; $contador_ciclos < $número; $contador_ciclos++){
          $matriz[] = $contador_ciclos;
        }
        return $matriz;
      }
    ?>
  </body>
</html>
```

Puede ver los resultados en la Figura 4-8, donde la función create_array generó la matriz que se le pidió crear y la devolvió. Muy conveniente.

Retorno de listas

Quizá recuerde la función list del capítulo 3, para convertir matrices en listas de variables. Ésa es la clave para hacer que sus funciones devuelvan también valores múltiples.

Normalmente, las funciones sólo pueden devolver valores individuales; pero si devuelve una matriz y la maneja como lista en el código para hacer la llamada, descubrirá que puede devolver tantos valores como desee desde funciones de PHP. Por ejemplo, suponga que quiere convertir la función create_array de la sección anterior en una función create_list:

```
function create_list($número)
{
  for ($contador_ciclos = 0; $contador_ciclos < $número; $contador_ciclos++){
   $matriz[] = $contador_ciclos;
  }

  return $matriz;
}
```

Ahora puede manejar la matriz devuelta, como lista en el código que hace la llamada en phpreturnlist.php de esta forma:

```
<html>
  <head>
    <title>
      Retorno de listas desde funciones
    </title>
  </head>
  <body>
    <h1>
      Retorno de listas desde funciones
    </h1>
    <?php
      list($primera, $segunda, $tercera) = create_list(3);
      echo "Ésta es la primera lista:<br>";
      echo "$primera, $segunda, $tercera<br>";

      list($primera, $segunda, $tercera, $cuarta) = create_list(4);
      echo "Ésta es la segunda lista:<br>";
      echo "$primera, $segunda, $tercera, $cuarta<br>";

      function create_list($número)
      {
        for ($contador_ciclos = 0; $contador_ciclos < $número; $contador_ciclos++){
         $matriz[] = $contador_ciclos;
        }

        return $matriz;
      }
    ?>
  </body>
</html>
```

Puede ver los resultados en la Figura 4-9, donde la función create_list devolvió una matriz que se manejaba correctamente como lista. Formidable.

FIGURA 4-9 Retorno de listas desde funciones

Retorno de referencias

Las funciones también pueden devolver referencias en PHP. No es algo que hará todo el tiempo; de tal modo, siéntase libre de pasar a la sección siguiente si lo desea. Pero si le interesa conocer la historia completa, siga leyendo.

Las referencias son elementos interesantes en PHP. Por ejemplo, suponga que tuviera una variable llamada $valor; podría obtener una referencia a esa variable con el operador de referencia, &:

```
$valor = 4;
$ref = & $valor;
```

Ahora $ref es una referencia a $valor y apuntará a los mismos datos en la memoria que $valor (si cambia una, también la otra):

```
$valor = 4;
$ref = & $valor;
$ref = 6;
```

Ahora $valor contendrá 6.

Observemos la acción de paso y devolución de una referencia a, así como desde una función, return_reference. Primero crearemos la función return_reference:

```
function &return_reference(& $ref)
{
  return $ref;
}
```

Observe que esta función toma una referencia como su argumento, indicado por el símbolo & en la lista de argumentos. También devuelve una referencia, según señala el símbolo & frente al nombre de la función.

De modo, se pasa la función return_reference como referencia a una variable y ésta se devuelve la misma referencia. Ahora pasemos una referencia a una variable a la función return_reference y obtendremos esa referencia de regreso:

```
<?php
  $valor = 4;
  echo "Valor actual: ", $valor, "\n";

  $ref = &return_reference($valor);
    .
    .
    .
```

Entonces podría incrementar la referencia $ref:

```
<?php
  $valor = 4;
  echo "Valor actual: ", $valor, "\n";

  $ref = &return_reference($valor);

  $ref++;
    .
    .
    .
```

También sería posible confirmar que la variable original se incrementó de esta forma en phpreturnreference.php:

```
<html>
  <head>
    <title>
      Retorno de referencias desde funciones
    </title>
  </head>
  <body>
    <h1>
      Retorno de referencias desde funciones
    </h1>
    <?php
      $valor = 4;
      echo "Valor actual: ", $valor, "<br>";

      $ref = &return_reference($valor);

      $ref++;

      echo "Nuevo valor: ", $valor, "<br>";

      function &return_reference(& $ref)
      {
        return $ref;
      }
    ?>
  </body>
</html>
```

FIGURA 4-10 Retorno de referencias desde funciones

Puede ver los resultados en la Figura 4-10, donde la referencia se creó y pasó a la función return_reference, que devolvió la misma referencia.Y luego se usó esa referencia para incrementar la variable a la que apunta.

Introducción de un ámbito variable en PHP

Parte de la idea tras el uso de funciones, consiste en que puede alojar sus variables en una función, ocultándolas del resto del código. Si hace esto, facilitará el desarrollo de su aplicación, pues no necesita preocuparse por conflictos entre variables.

Éste es un ejemplo mostrando cómo las variables del mismo nombre no entran en conflicto unas con otras, dentro y fuera de funciones. Este ejemplo maneja dos variables llamadas $valor, una afuera de una función cualquiera y la otra dentro de una función llamada scoper. Como verá más adelante, estas variables son independientes de las demás.

Este ejemplo comienza creando $valor y haciéndola igual a 4, para después reproducir ese valor con echo en el navegador:

```
<?php
  $valor = 4;

  echo "En el código que hace la llamada, = ", $valor, "<br>";
    .
    .
    .
?>
```

Luego llama a la función scoper, también con una variable llamada $valor, que se inicializa en 8000000 y reproduce esa variable mediante echo desde dentro de la función. Luego regresa una vez más al script scope y reproduce de nuevo con echo $valor, al nivel del script (que sigue siendo 4). Puede ver todo esto en phpscope.php:

```
<html>
  <head>
    <title>
      Manejo del ámbito en funciones
    </title>
  </head>
  <body>
    <h1>
      Manejo del ámbito en funciones
    </h1>
    <?php
      $valor = 4;

      echo "En el código que hace la llamada, \$valor = ", $valor, "<br>";

      scoper();

      echo "En el código que hace la llamada de nueva cuenta, \$valor es aún = ",
         $valor, "<br>";

      function scoper()
      {
        $valor = 8000000;
        echo "En la función, \$valor = ", $valor, "<br>";
      }
    ?>
  </body>
</html>
```

Puede ver los resultados en la Figura 4-11, donde se aprecia el ámbito de la variable $valor dentro y fuera de la función scoper.

FIGURA 4-11 Ámbito de la función

Acceso a datos globales

¿Qué sucedería si quisiera acceder datos fuera de una función, desde código contenido en una función? Por ejemplo, suponga que en la función scoper, en realidad deseaba tener acceso a la variable al nivel de script llamada $valor, no sólo la versión local de esa variable:

```
<?php
  $valor = 4;
    .
    .
    .
  function scoper()
  {
    $valor = 8000000;
    echo "En la función, \$valor = ", $valor, "<br>";
  }
?>
```

¿Cómo haría eso? En PHP, los datos al nivel de script se denominan datos *globales* y puede tener acceso a ellos desde dentro de una función, con la palabra clave global. Puede ver cómo funciona esto en phpglobalscope.php, que modifica el ejemplo anterior para llamar también a una función conocida como global_scoper:

```
<?php
  $valor = 4;

  echo "En el código que hace la llamada, \$valor = ", $valor, "<br>";

  scoper();
  global_scoper();

  echo "En el código que hace la llamada de nuevo, \$valor es aún = ", $valor, "<br>";

  function scoper()
  {
    $valor = 8000000;
    echo "En la función scoper, \$valor = ", $valor, "<br>";
  }

  function global_scoper()
  {
    .
    .
    .
  }
?>
```

Para tener acceso a la variable global $valor, desde dentro de global_scoper, puede declarar esa variable usando la palabra clave global:

```
<?php
  $valor = 4;
```

```
  echo "En el código que hace la llamada, \$valor = ", $valor, "<br>";

  scoper();
  global_scoper();

  echo "En el código que hace la llamada de nuevo, \$valor es aún = ", $valor, "<br>";

  function scoper()
  {
    $valor = 8000000;
    echo "En la función scoper, \$valor = ", $valor, "<br>";
  }

  function global_scoper()
  {
    global $valor;
    .
    .
    .
  }
?>
```

Ahora puede reproducir con echo la variable $valor dentro de la función global_scoper, para comprobar que en realidad se trata de la variable global $valor, de la siguiente forma en phpglobalscope.php:

```
<html>
  <head>
    <title>
      Manejo del ámbito local y global en funciones
    </title>
  </head>
  <body>
    <h1>
      Manejo del ámbito local y global en funciones
    </h1>
    <?php
      $valor = 4;

      echo "En el código que hace la llamada, \$valor = ", $valor, "<br>";

      scoper();
      global_scoper();

      echo "En el código que hace la llamada de nuevo, \$valor es aún = ", $valor, "<br>";

      function scoper()
      {
        $valor = 8000000;
        echo "En la función scoper, \$valor = ", $valor, "<br>";
      }
```

FIGURA 4-12 Ámbito local y global

```
      function global_scoper()
      {
        global $valor;
        echo "En la función scoper global, \$valor = ", $valor, "<br>";
      }
    ?>
  </body>
</html>
```

Puede ver los resultados en la Figura 4-12, donde se aprecia que puede acceder datos globales desde una función, empleando la palabra clave global.

Trabajo con variables estáticas

Un problema con funciones es que las variables contenidas en ellas se reinician cada vez que se las llama —es decir, los valores de esas variables no se preservan entre llamadas a funciones. Eso es un problema si realmente quiere que dichas variables mantengan sus valores.

Por ejemplo, suponga que deseara llevar el control del número de veces que ha llamado una función con la variable llamada $contador. Podría intentar algo como esto en phpcounter.php:

```
<html>
  <head>
    <title>
      Llevar la cuenta de llamadas a funciones
    </title>
  </head>
  <body>
    <h1>
      Llevar la cuenta de llamadas a funciones
    </h1>
```

```
    <?php
      echo "Ahora la cuenta es: ", count_function(), "<br>";
      echo "Ahora la cuenta es: ", count_function(), "<br>";
      echo "Ahora la cuenta es: ", count_function(), "<br>";
      echo "Ahora la cuenta es: ", count_function(), "<br>";
      echo "Ahora la cuenta es: ", count_function(), "<br>";

      function count_function()
      {
        $contador = 0;
        $contador++;
        return $contador;
      }
    ?>
  </body>
</html>
```

Pero se obtienen los resultados mostrados en la Figura 4-13, donde la cuenta es siempre 1. Ésa es la razón porque $contador se reinicia a 0 siempre que se llama la función count_function.

¿Cómo corregir tal cosa?

Existen dos formas sencillas para resolver este problema. Primero, puede hacer que $contador sea una variable global y acceder a ella como tal en la función count_function, de la siguiente forma:

```
<?php
  $contador = 0;
  echo "Ahora la cuenta es: ", count_function(), "<br>";
  echo "Ahora la cuenta es: ", count_function(), "<br>";
  echo "Ahora la cuenta es: ", count_function(), "<br>";
  echo "Ahora la cuenta es: ", count_function(), "<br>";
  echo "Ahora la cuenta es: ", count_function(), "<br>";
```

FIGURA 4-13 Problemas con variables de funciones

```
      function count_function()
      {
        global $contador;
        $contador++;
        return $contador;
      }
    ?>
```

Eso funciona y se obtiene:

```
Ahora la cuenta es: 1
Ahora la cuenta es: 2
Ahora la cuenta es: 3
Ahora la cuenta es: 4
Ahora la cuenta es: 5
```

Existe una forma más de hacerlo y no lo compromete insistiendo que use variables globales. También puede declarar la variable $contador en la función counter_function como *estática;* significa que preservará su valor entre llamadas a la función. Así es como se hace con la palabra clave static:

```
<html>
  <head>
    <title>
      Llevar la cuenta de llamadas a funciones
    </title>
  </head>
  <body>
    <h1>
      Llevar la cuenta de llamadas a funciones
    </h1>
    <?php
      echo "Ahora la cuenta es: ", count_function(), "<br>";
      echo "Ahora la cuenta es: ", count_function(), "<br>";
      echo "Ahora la cuenta es: ", count_function(), "<br>";
      echo "Ahora la cuenta es: ", count_function(), "<br>";
      echo "Ahora la cuenta es: ", count_function(), "<br>";

      function count_function()
      {
        static $contador = 0;
        $contador++;
        return $contador;
      }
    ?>
  </body>
</html>
```

Ahora puede ver los resultados en la Figura 4-14 —puede apreciar que la palabra clave static hizo su trabajo, reportó correctamente el número de veces que se llamó a la función.

FIGURA 4-14 Cómo llevar la cuenta de llamadas a funciones

Funciones condicionales de PHP

PHP es un lenguaje interpretado, lo que significa que el código no es accesible hasta que lo maneja realmente el intérprete. Ello no es un problema con funciones normales, pero usted puede definir funciones en instrucciones condicionales como instrucciones if —y su función no existe, hasta donde le compete a PHP, una vez se ejecute el código que la define.

Éste es un ejemplo. Podría tener una función normal como ésta:

```
<?php
  echo "normal_function() está lista para proceder tan pronto como se inicie el
script.<br>";

  normal_function;
    .
    .
    .
  function normal_function()
  {
    echo "Hola desde la función normal.<br>";
  }
?>
```

Este código llamará y ejecutará la función normal_function tan pronto se cargue la página, no hay problema con eso. Sin embargo, ahora suponga que tiene otra función, conditional_function, cuyo código está dentro de una instrucción if. Ésta no se encuentra disponible hasta ejecutar el código dentro de la instrucción if; de modo que tendrá problemas al llamar a la función conditional_function, antes de ejecutar la instrucción if, como aquí:

```
<?php
  echo "normal_function() está lista para proceder tan pronto se inicie el script.<br>";

  normal_function();
```

FIGURA 4-15 Intento de llamar a una función antes de que ésta exista

```
  $crear_función = TRUE;

  echo "conditional_function() no está lista sino hasta que se ejecute la ";
  echo "instrucción if.<br>";

  conditional_function();

  if ($crear_función) {
  function conditional_function()
  {
    echo "Hola desde la función condicional.<br>";
  }
}

  function normal_function()
  {
    echo "Hola desde la función normal.<br>";
  }
?>
```

Puede ver los resultados en la Figura 4-15 —según aparece, la función conditional_function antes de que ejecutar la instrucción if que la contiene, se obtiene un error fatal.

De tal modo, cambiemos esto para cerciorarnos de que se llama a la función conditional_function, sólo después de que ésta existe, que puede controlar con una variable llamada $función_creada:

```
<?php
  echo "normal_function() está lista para proceder tan pronto se inicie el script.<br>";

  normal_function();
```

```
  $crear_función = TRUE;

  echo "conditional_function() no está lista sino hasta que se ejecute la ";
  echo "instrucción if.<br>";

  $función_creada = FALSE;

  if ($crear_función) {
  function conditional_function()
  {
    echo "Hola desde la función condicional.<br>";
  }
  $función_creada = TRUE;
}

  function normal_function()
  {
    echo "Hola desde la función normal.<br>";
  }
?>
```

Y puede llamar a la función conditional_function después de creada, de esta forma en php-conditionalfunction.php:

```
<html>
  <head>
    <title>
      Creación de funciones condicionales
    </title>
  </head>
  <body>
    <h1>
      Creación de funciones condicionales
    </h1>
    <?php
      echo "normal_function() está lista para proceder tan pronto inicie el script.<br>";

      normal_function();

      $crear_función = TRUE;

      echo "conditional_function() no está lista hasta ejecutar la ";
      echo "instrucción if.<br>";

      $función_creada = FALSE;

      if ($crear_función) {
        function conditional_function()
        {
          echo "Hola desde la función condicional.<br>";
        }
        $función_creada = TRUE;
      }
```

FIGURA 4-16 Ejecución de una función condicional

```
      if ($función_creada){
        conditional_function();
      }

      function normal_function()
      {
        echo "Hola desde la función normal.<br>";
      }
    ?>
  </body>
</html>
```

Ahora puede ver los resultados en la Figura 4-16 —de acuerdo con el resultado, la función condicional se volvió accesible tras ejecutarse la instrucción if—. Formidable.

Funciones variables de PHP

Quizá recuerde que para las variables alternas de PHP sólo debía cargar el nombre de una variable en otra y así acceder a la primera variable. Bueno, puede hacer lo mismo con las funciones en PHP —asigne a una variable el nombre de una función y luego trate dicha variable como función mediante su nombre.

En phpvariablefunctions.php, comienza con tres funciones, rojo, blanco y azul:

```
<?php
  $función_variable = "rojo";
  {
    echo "En rojo() ahora.<br>";
  }
```

```
    function blanco($argumento)
    {
      echo "$argumento <br>";
    }

    function azul($argumento)
    {
      echo "$argumento <br>";
    }
  ?>
```

Ahora puede asignar el nombre de la función red a una variable que llamaremos $función_variable en este ejemplo —luego podrá tratar a $función_variable como el nombre de una función y llamarla de esta forma:

```
  <?php
    $función_variable = "rojo";
    $función_variable();
      .
      .
      .
    function rojo()
    {
      echo "En rojo() ahora.<br>";
    }

    function blanco($argumento)
    {
      echo "$argumento <br>";
    }


    function azul($argumento)
    {
      echo "$argumento <br>";
    }
  ?>
```

De manera similar, puede emplear la misma técnica para llamar a las funciones blanco y azul (observe que las funciones denominadas de esta forma pueden tener listas de argumentos diferentes):

```
<html>
  <head>
    <title>
      Creación de funciones variables
    </title>
  </head>
  <body>
    <h1>
      Creación de funciones variables
    </h1>
    <?php
      $función_variable = "rojo";
      $función_variable();
```

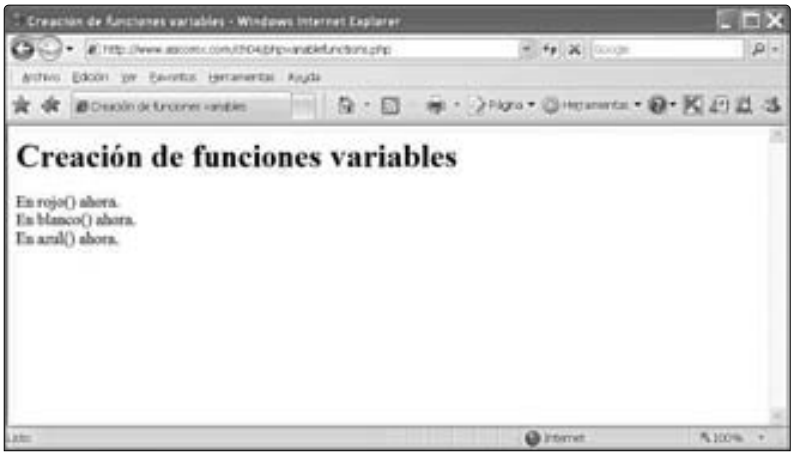

FIGURA 4-17 Ejecución de funciones variables

```
      $función_variable = "blanco";
      $función_variable("En blanco() ahora.");

      $función_variable = "azul";
      $función_variable("En azul() ahora.");

      function rojo()
      {
        echo "En rojo() ahora.<br>";
      }

      function blanco($argumento)
      {
        echo "$argumento <br>";
      }

      function azul($argumento)
      {
        echo "$argumento <br>";
      }
    ?>
  </body>
</html>
```

Los resultados aparecen en la Figura 4-17.

La capacidad para asignar funciones a variables de esta forma es una técnica poderosa. Puede escribir mucho código y adaptarlo de último minuto para llamar a las funciones deseadas —dependiendo de las condiciones en el momento de la ejecución— sin tener que editar ese código.

Anidación de funciones

También puede anidar definiciones de funciones en PHP. Observe que no se puede llamar a la función anidada hasta haber ejecutado la función que la anida —la función anidada no será accesible hasta entonces.

Éste es un ejemplo, phpnestedfunctions.php. Aquí se comienza con la definición de una función, outer_function, que encierra a otra función, inner_function:

```
<?php
  function outer_function()
  {
    echo "En la función externa.<br>";
  function inner_function()
    {
    echo "En la función interna.<br>";
    }
  }
?>
```

Ahora puede llamar a outer_function e inner_function —pero sólo en ese orden, pues la llamada a outer_function hace que PHP defina inner_function—, de la siguiente forma en phpnestedfunctions.php:

```
<html>
  <head>
    <title>
      Anidación de funciones
    </title>
  </head>
  <body>
    <h1>
      Anidación de funciones
    </h1>
    <?php
      outer_function();
      inner_function();

      function outer_function()
      {
        echo "En la función externa.<br>";
        function inner_function()
        {
          echo "En la función interna.<br>";
        }
      }
    ?>
  </body>
</html>
```

FIGURA 4-18 Anidación de funciones

Los resultados están en la Figura 4-18, donde se puede ver que las llamadas a outer_function e inner_function fueron exitosas.

Creación de archivos de inclusión

PHP le permite crear también archivos de *inclusión*, cuyo contenido se insertará en su archivo de código. Por ejemplo, suponga que desea llevar registro de algunas constantes, como la prima de su seguro de gastos médicos. Podría hacerlo definiendo una constante en un archivo llamado, por ejemplo, prima.inc:

```
<?php
    define("prima", 176.53);
?>
```

Ahora puede incluir prima.inc en su código con la instrucción *include* de PHP, de la siguiente forma:

```
<?php
  echo "Inclusión de constantes.inc....<br>";
  include("prima.inc");
    .
    .
    .
?>
```

FIGURA 4-19 Uso de archivos de inclusión

Después de incluir prima.inc, el contenido de ese archivo queda a disposición de su código. En este caso, eso significa que la prima con nombre asignado constante, se encuentre accesible para su código, de modo que puede hacer uso de esa constante así:

```
<html>
  <head>
    <title>
      Uso de archivos de inclusión
    </title>
  </head>
  <body>
    <h1>
      Uso de archivos de inclusión
    </h1>
    <?php
      echo "Inclusión de prima.inc....<br>";
      include("prima.inc");

      echo "Usted paga \$", como prima, " por mes.<br>";
    ?>
  </body>
</html>
```

Los resultados aparecen en la Figura 4-19, donde puede ver que la constante del archivo de inclusión en realidad procedió.

Retorno de errores de funciones

Es común que las funciones integradas de PHP devuelvan por valor FALSE, en caso de haber error y puede emplear la misma técnica en sus funciones. Cuando una función devuelve FALSE, puede usar la función die de PHP para imprimir un mensaje de error.

Éste es un ejemplo —suponga que tiene una función llamada reciprocal para calcular recíprocos (uno dividido entre un número)—. Pero no quiere calcular uno dividido entre cero, de tal modo que devuelva FALSE preguntándole si aprueba esto en la función reciprocal y TRUE si las cosas salieron según lo previsto:

```
<?php
  function reciprocal($valor)
  {
    if ($valor != 0) {
      echo 1 / $valor, "<br>";
      return TRUE;
    }
    else {
      return FALSE;
    }
  }
?>
```

Ahora puede llamar a la función reciprocal y agregar la cláusula "or die (*mensaje*)", donde *mensaje* es el texto que se mostrará si la función reciprocal devolviera un valor FALSE. Se ve así en este código (cuando se llama a la función die, la aplicación se cierra):

```
<html>
  <head>
    <title>
      Manejo de errores de funciones
    </title>
  </head>
  <body>
    <h1>
      Manejo de errores de funciones
    </h1>
    <?php
      echo "El recíproco de 2 es: ";
      reciprocal(2) or die ("No puede tomar el recíproco de cero.");
      echo "El recíproco de 0 es: ";
      reciprocal(0) or die ("No puede tomar el recíproco de cero.");

      function reciprocal($valor)
      {
        if ($valor != 0) {
          echo 1 / $valor, "<br>";
          return TRUE;
        }
        else {
          return FALSE;
        }
      }
    ?>
  </body>
</html>
```

Los resultados aparecen en la Figura 4-20, mostrando que el recíproco de 2 se ve bien; pero entonces cuando pide a la función reciprocal calcular el recíproco de cero, se atasca y devuelve FALSE, deteniendo la aplicación.

FIGURA 4-20 Manejo de resultados de error de funciones

CAPÍTULO 5

Lectura de datos en páginas Web

Éste es el capítulo que muchos programadores han esperado —trata sobre la conexión de controles HTML en páginas Web, como campos de texto, botones de radio, casillas de selección, etcétera, con PHP de vuelta en el servidor.

Ya conoce los fundamentos de PHP hasta ahora, ello significa que puede manipular el código necesario para trabajar con controles PHP y HTML. Así que nos conectaremos con HTML en este capítulo y veremos cómo leer datos ingresados por el usuario en controles HTML de páginas Web.

Configuración de páginas Web para comunicación con PHP

Para conectarse con PHP o cualquier código en el servidor, debe configurar sus páginas Web de una manera particular. Tiene que encerrar todos sus controles HTML en un formato HTML y debe indicar en esa forma a dónde se enviarán los datos alojados en esos controles. Por ejemplo, el usuario podría ingresar su nombre en un campo de texto y debe hacer saber al navegador a dónde enviar ese nombre, apenas el usuario haga clic en el botón Enviar.

Así se podría ver una forma HTML:

```
<form>
        .
        .
        .
</form>
```

Necesita especificar el método con que se enviarán sus datos —los dos métodos más comunes son "get" y "post" (profundizaremos acerca de la diferencia entre ambos más adelante, ya que los dos métodos llevarán sus datos al script PHP)— y podría elegir el método get aquí:

```
<form method="get">
        .
        .
        .
</form>
```

Bueno, esto indica al navegador cómo se supone que debe enviar los datos en la página Web (y también tiene que indicar al navegador a dónde enviar esos datos con una dirección URL). Usted asigna esa URL al atributo de acción del elemento <form>, de esta forma:

```
<form method="get" action="http://www.phpisgreat.com/phpreader.php">
        .
        .
        .
</form>
```

Este elemento <form> enviará sus datos a la URL http://www.phpisgreat.com/phpreader.php cuando se haga clic en el botón Enviar.

También puede asignar URL relativas al atributo de acción. Por ejemplo, si esta página HTML residiera en el servidor en un directorio específico, y el script phpreader.php residiera en el *mismo* directorio en el servidor, podría acortar la URL a ésta:

```
<form method="get" action="phpreader.php">
        .
        .
        .
</form>
```

Esta versión del elemento <form> supone que la página HTML que está actualmente en el navegador provino del mismo directorio que el script PHP en el que se encuentra. Por ejemplo, si esta página HTML fuera http://www.phpisgreat.com/input.html, entonces el script PHP especificado por la simple asignación de"phpreader.php"al atributo de acción enviaría los datos de la página HTML al script PHP en http://www.phpisgreat.com/phpreader.php.

Ésta es otra versión (puede omitir el elemento de acción por completo):

```
<form method="get">
        .
        .
        .
</form>
```

En este caso, los datos de la forma se envían de vuelta a la misma dirección URL en la que se encuentra el documento actual. Por ejemplo, si tiene un script de PHP que puede mostrar controles HTML, phpcomplete.php, entonces cuando se dirija a ese script en su navegador, verá esos controles HTML. Si no hay atributo de acción, los datos contenidos en la forma serán enviados de regreso a exactamente el mismo script cuando el usuario haga clic en el botón Enviar (o Submit).

Eso es algo que se suele hacer un script de PHP a cargo de la presentación de controles HTML y luego la lectura de datos contenidos en dichos controles HTML, cuando el usuario haga clic en el botón Enviar (o Submit).Verá cómo funciona esto en este libro.

Además de los controles HTML, como campos de texto y casillas de selección, también necesitará un botón para enviar en el diseño de su forma, ya que los datos contenidos en la forma se envían a su script PHP cuando se hace clic en ese botón para enviar. El botón de envío no necesita la leyenda"Enviar"—puede configurarla como desee asignándole el atributo de envío al valor del botón.

Ésta es la forma de crear un botón de envío —observe que debe estar dentro del elemento <form> de HTML— con la leyenda Enviar y que se usa un elemento <input> con el atributo de tipo"enviar", para crear un botón de envío:

```
<html>
  <head>
    <title>
      Conectando con PHP
    <title>
  </head>
  <body>
    <h1>
      Conectando con PHP
    <h1>
    <form method="get" action="phpreader.php">
           .
           .
           .
      <input type="submit" value="Enviar">
      <input type="reset" value="Reiniciar">
    </form>
  <body>
</html>
```

Observe que aquí también hay un botón de reinicio (opcional); cuando se hace clic en él, éste reinicia los datos contenidos en la forma en todos los controles HTML a sus valores predeterminados.

Suponga que deseara utilizar un campo de texto (es decir, un control HTML <uinput type = "texto">) para preguntar el nombre del usuario; podría hacerlo de esta manera en formato HTML:

```
<html>
  <head>
    <title>
      Conectando con PHP
    <title>
  </head>
  <body>
    </h1>
      Conectando con PHP
    <h1>
    <form method="get" action="phpreader.php">
      ¿Cuál es su nombre?

      <input name="data" type="text">
           .
           .
      <input type="submit" value="Enviar">
      <input type="reset" value="Reiniciar">
    </form>
  </body>
</html>
```

Observe que este segmento HTML asigna el nombre"data"al campo de texto. ¿Cómo puede tener acceso al nombre ingresado en este campo de texto en el servidor, en su script de PHP?

Si ha usado el método POST, puede encontrar tales datos en la matriz $_POST, como comenzaremos a ver en la siguiente sección, que trata sobre la recuperación de datos de campos de texto. Si ha utilizado el método GET, utiliza la matriz $_GET. Éstas son matrices "superglobales", que significa el grado de disponibilidad sin usar la palabra clave global. Asimismo, la matriz $_REQUEST aloja datos de $_GET y $_POST.

Eso significa que para recuperar el nombre ingresado por el usuario en el campo de texto "data", puede manejar la expresión $_RESQUEST["data"] en su script de PHP.

Pongamos todo esto en práctica con campos de texto reales, que veremos a continuación.

Manejo de campos de texto

Supongamos que ha reunido una página Web, phptext.html, con un campo de texto y un botón de envío en una forma, como puede ver aquí:

```
<html>
  <head>
    <title>
      Ingreso de datos en campos de texto
    </title>
  </head>
  <body>
    <h1>
      Ingreso de datos en campos de texto
    </h1>
    <form method="post" action="phptext.php">
      ¿Cuál es su nombre?

      <input name="data" type="text">           .

      <input type="submit" value="Enviar">
    </form>
  </body>
</html>
```

Puede ver esta página HTML en la Figura 5-1, esperando que usted ingrese su nombre.

De acuerdo, así ¿cómo lee los datos que el usuario ingresó en esta página Web —es decir, su nombre en un campo de texto al que ha llamado "data"— desde PHP en el servidor?

Esta página HTML está configurada para enviar sus datos al script de PHP phptext.php (observe que como ha asignado una dirección URL relativa al atributo de acción —simplemente el nombre del script PHP—, ese script debe almacenarse en el mismo directorio que phptext.html en el servidor):

```
<form method="post" action="phptext.php">
  ¿Cuál es su nombre?

  <input name="data" type="text">           .

  <input type="submit" value="Enviar">
</form>
```

FIGURA 5-1 Uso de un campo de texto en una página HTML

En phptext.php, puede acceder al texto en el campo llamado"data"como $_REQUEST["data"]. De modo que aquí está phptext.php:

```
<html>
  <head>
    <title>
      Lectura de datos de campos de texto
    </title>
  </head>
  <body>
    <h1>
      Lectura de datos de campos de texto
    </h1>
      Gracias por responder,
      <?php
        echo $_REQUEST["data"];
      ?>
  </body>
</html>
```

Todo lo que hace este script PHP, phptext.php, es reproducir con echo el nombre ingresado por el usuario en el campo de texto de datos. Asegúrese de que phptext.php se encuentre en el mismo directorio que phptext.html en el servidor y pruebe este ejemplo. Cuando haga clic en el botón enviar en phptext.html verá los resultados, algo como lo que aparece en la Figura 5-2, donde está el nombre del usuario.

Observe la dirección URL de la Figura 5-2: http://localhost/ch05/phptext.php?data=Steve. Esa URL incluye los datos que el usuario ingresó en el campo de texto, colocado ahí después de un signo de interrogación (?). Los datos enviados con el método"get"están siempre codificados en URL (los espacios se reemplazan con signos +, diferentes controles, pares de nombre/datos, se separan con un signo &) y se colocan en la URL.

FIGURA 5-2 Lectura de texto de un campo de texto en PHP

El método get funciona, pero como verá, puede crear resultados menos que profesionales. Si no desea que los datos del usuario aparezcan en la URL con que accede al servidor, mejor use el método"post", de esta forma en phptext.html:

```
<html>
  <head>
    <title>
      Ingreso de datos en campos de texto
    </title>
  </head>
  <body>
    <h1>
      Ingreso de datos en campos de texto
    </h1>
    <form method="post" action="phptext.php">
      ¿Cuál es su nombre?

      <input name="data" type="text">           .

      <input type="submit" value="Enviar">
    </form>
  </body>
</html>
```

Cuando se utiliza esta nueva versión de phptext.html, los datos del usuario se envían en los encabezados HTTP que el navegador manda al servidor, en lugar de la URL. Eso da como resultado una URL más limpia, como se aprecia en el área de texto URL del navegador, en la Figura 5-3. Puesto que los datos publicados se envían en encabezados HTTP y no en la URL, los datos enviados con el método post son ligeramente más seguros.

FIGURA 5-3 Lectura de texto de un campo de texto sin codificación URL visible

Excelente, ha podido leer datos de un campo de texto. A continuación veremos un control HTML similar, las áreas de texto.

Manejo de áreas de texto

Los campos de texto son adecuados sólo si desea una línea de texto; pero en caso de querer múltiples líneas de texto, debe recurrir a las áreas de texto.

Este ejemplo, phptextarea.html, presenta un área de texto al usuario y pregunta qué ingredientes desea en la pizza. Comienza de esta forma:

```
<html>
  <head>
    <title>
      Ingreso de datos en áreas de texto
    </title>
  </head>
  <body>
    <h1>
      Ingreso de datos en áreas de texto
    </h1>
    <form method="post" action="phptextarea.php">
      Indique los ingredientes que desea en su pizza: <br>
        .
        .
        .
      </textarea>
      <br>
      <input type="submit" value="Enviar">
    </form>
  </body>
</html>
```

Luego agrega un área de texto como ésta, donde los números 1 a 4 aparecerán en el área de texto:

```
<html>
  <head>
    <title>
      Ingreso de datos en áreas de texto
    </title>
  </head>
  <body>
    <h1>
      Ingreso de datos en áreas de texto
    </h1>
    <form method="post" action="phptextarea.php">
      Indique los ingredientes que desea en su pizza: <br>
      <textarea name="data" cols="50" rows="5">
1.
2.
3.
4.
      </textarea>
      <br>
      <input type="submit" value="Enviar">
    </form>
  </body>
</html>
```

Puede ver esta página en la Figura 5-4, en espera de que indique los ingredientes para su pizza:

FIGURA 5-4 Uso de un área de texto en una página HTML

¿Así que cómo lee los datos ingresados por el usuario en esta página Web —en el área de texto llamada"data"— en el servidor de PHP? Los datos alimentados por el usuario se enviarán a phptextarea.php en el servidor y puede acceder al texto del área de texto como $_REQUEST["data"] en ese script:

```
<html>
  <head>
    <title>
      Lectura de datos de áreas de texto
    </title>
  </head>
  <body>
    <h1>
      Lectura de datos de áreas de texto
    </h1>
    Usted ordenó una pizza con: <br>
    <?php
      $texto = $_REQUEST["data"];
        .
        .
        .
    ?>
  </body>
</html>
```

Observe que según el manejo de un área de texto, las líneas múltiples se rellenarán con caracteres de línea nueva, \n. Cuando muestre ese texto, el navegador pasará por alto las nuevas líneas; de modo que podría reemplazarlas con elementos <br> en su lugar, de la siguiente forma, donde mostramos los ingredientes de la pizza del usuario en phptextarea.php:

```
<html>
  <head>
    <title>
      Lectura de datos de áreas de texto
    </title>
  </head>
  <body>
    <h1>
      Lectura de datos de áreas de texto
    </h1>
    Usted ordenó una pizza con: <br>
    <?php
      $texto = $_REQUEST["data"];
      echo str_replace("\n", "<br>", $texto);
    ?>
  </body>
</html>
```

Y verá los resultados en la Figura 5-5, donde la aplicación no sólo ha reproducido con echo el texto ingresado por el usuario, sino también ha preservado la naturaleza del texto. Formidable.

FIGURA 5-5 Lectura de texto de un área de texto en PHP

Manejo de casillas de selección

El siguiente paso en los controles son las casillas de selección (esos controles cuadrados que puede seleccionar o deseleccionar con el mouse).

Las casillas de selección se crean con el elemento <input>, de la siguiente forma en una página Web, donde se pregunta al usuario si desea papas fritas en phpcheckbox.html:

```
<html>
  <head>
    <title>
      Ingreso de datos en casillas de selección
    </title>
  </head>
  <body>
    <h1>
      Ingreso de datos en casillas de selección
    </h1>
    <form method="post" action="phpcheckbox.php">
      ¿Desea papas fritas con su orden?
      <input name="check1" type="checkbox" value="sí">
      Sí
      <input name="check2" type="checkbox" value="no">
      No
      <br>
      <br>
      <input type="submit" value="Enviar">
    </form>
  </body>
</html>
```

Observe que el valor de la primera casilla de selección es“sí”y el valor de la segunda es“no” (esos valores que se enviarán a su script en el servidor).

Y puede ver los resultados en la Figura 5-6, donde la aplicación pregunta al usuario si desea papas fritas.

¿Cómo lee los datos de las casillas de selección, check1 y check2? Podría pensar que puede hacerlo simplemente de esta forma, donde sólo usa expresiones como $_REQUEST["check1"]:

```
<html>
  <head>
    <title>
      Lectura de datos de casillas de selección
    </title>
  </head>
  <body>
    <h1>
      Lectura de datos de casillas de selección
    </h1>
    Usted seleccionó:
    <?php
      echo $_REQUEST["check1"], "<br>";
      echo $_REQUEST["check2"], "<br>";
    ?>
  </body>
</html>
```

Por desgracia, es incorrecto —probablemente el usuario no haya hecho clic en una casilla de selección—, de tal modo que el intento de mostrar los datos de esa casilla de selección daría un error en PHP. Por ejemplo, si el usuario no ha seleccionado check1, entonces echo $_REQUEST["check1"] dará un error (porque la matriz $_REQUEST no tiene un elemento con el índice"check1").

En este caso, tiene que verificar primero si hay datos que lo estén esperando de una casilla de selección en particular antes de que intente mostrar esos datos. Puede revisar si una matriz tiene un elemento con cierto índice con la función isset; así que antes de reproducir

FIGURA 5-6 Configuración de casillas de selección

con echo $_REQUEST[“check1”], verifique si ese elemento de la matriz existe con isset($_REQUEST[“check1”]). Así es como funciona esto en phpcheckbox.php:

```
<html>
  <head>
    <title>
      Lectura de datos de casillas de selección
    </title>
  </head>
  <body>
    <h1>
      Lectura de datos de casillas de selección
    </h1>
    Usted seleccionó:
    <?php
      if (isset($_REQUEST["check1"])) {
        echo $_REQUEST["check1"], "<br>";
      }
      if (isset($_REQUEST["check2"])) {
        echo $_REQUEST["check2"], "<br>";
      }
    ?>
  </body>
</html>
```

Y puede ver los resultados en la Figura 5-7, donde lee qué casillas de selección fueron elegidas por el usuario.

Observe que el usuario podría hacer clic en ambas casillas de selección aquí —esto significaría que ambos deseaban papas fritas y no deseaban papas fritas—. Una mejor opción para los controles, en este caso serían los botones de radio, que sólo puede seleccionar uno a la vez y los veremos a continuación.

FIGURA 5-7 Lectura de datos de casillas de selección en PHP

Manejo de botones de radio

Las casillas de selección son buenas si desea que el usuario pueda seleccionar múltiples elementos de varias opciones. Pero si le interesa que el usuario seleccione sólo un elemento de varias opciones, debe usar botones de radio en su lugar, ya que éstos permiten al usuario hacer una selección a la vez.

Éste es un ejemplo, phpradiobutton.html que, como el ejemplo anterior, pregunta al usuario si desea papas fritas (pero a diferencia del ejemplo anterior, esta vez el usuario sólo puede seleccionar sí o no, no ambos; observe que daremos aquí el mismo nombre a ambos botones de radio,"radios"):

```
<html>
  <head>
    <title>
      Ingreso de datos con botones de radio
    </title>
  </head>
  <body>
    <h1>
      Ingreso de datos con botones de radio
    </h1>
    <form method="post" action="phpradiobutton.php">
      ¿Desea papas fritas con su orden?
      <input name="radios" type="radio" value="sí">
      Sí
      <input name="radios" type="radio" value="no">
      No
      <br>
      <br>
      <input type="submit" value="Enviar">
    </form>
  </body>
</html>
```

Puede ver esta página en la Figura 5-8, que espera su respuesta:

FIGURA 5-8 Uso de botones de radio en una página HTML

¿Cómo lee datos de botones de radio? Puede usar $_REQUEST de esta forma en phpradiobutton.php:

```
<html>
  <head>
    <title>
      Lectura de datos de botones de radio
    </title>
  </head>
  <body>
    <h1>
      Lectura de datos de botones de radio
    </h1>
    Usted seleccionó
    <?php
      echo $_REQUEST["radios"];
    ?>
  </body>
</html>
```

Pero tendría el mismo problema que con las casillas de selección —es posible que el usuario no haya seleccionado uno u otro botón de radio—. Por otra parte, sólo habrá una entrada en la matriz $_REQUEST bajo el índice"radios", de modo que sólo necesita una instrucción como ésta en phpradiobutton.php:

```
<html>
  <head>
    <title>
      Lectura de datos de botones de radio
    </title>
  </head>
  <body>
    <h1>
      Lectura de datos de botones de radio
    </h1>
    Usted seleccionó
    <?php
      if (isset($_REQUEST["radios"])) {
        echo $_REQUEST["radios"];
      }
    ?>
  </body>
</html>
```

De hecho, puede indicar que no se seleccionó un botón de radio si ése fuera el caso:

```
  <head>
    <title>
      Lectura de datos de botones de radio
    </title>
  </head>
```

FIGURA 5-9 Lectura de botones de radio en PHP

```
  <body>
    <h1>
      Lectura de datos de botones de radio
    </h1>
    Usted seleccionó
    <?php
      if (isset($_REQUEST["radios"])) {
        echo $_REQUEST["radios"];
      }
      else {
        echo "No se seleccionó un botón de radio. <br>";
      }
    ?>
  </body>
</html>
```

Los resultados aparecen en la Figura 5-9, donde la aplicación ha determinado con exactitud en qué botón de radio se hizo clic.

Manejo de cuadros de lista

Los cuadros de lista son también un control HTML común y requieren un manejo ligeramente especial. Suponga que deseara permitir al usuario seleccionar sus sabores de helado favoritos; podría comenzar de esta forma en phplistbox.html, donde creará la lista con un control <select> de HTML:

```
<html>
  <head>
    <title>
      Ingreso de datos con cuadros de lista
    </title>
  </head>
```

```
   <body>
    <h1>
      Ingreso de datos con cuadros de lista
    </h1>
    Seleccione sus sabores de helado favoritos:
    <form method="post" action="phplistbox.php">
      <select>
        .
        .
        .
      <select>
      <br>
      <br>
      <input type="submit" value="Enviar">
    </form>
  </body>
</html>
```

Para permitir al usuario seleccionar múltiples sabores de helado, vamos a hacer que éste sea un control de selección múltiple, lo cual se hace en HTML con el múltiplo de atributo independiente en el elemento <select>.Y éste es el truco que permitirá que este control de selección múltiple funcione con PHP (dará al control el nombre de una matriz, no sólo un nombre). Por ejemplo, si este control fuese de selección individual, podría llamarlo"ice_cream"(pero como es un control de selección múltiple, lo debe llamar"ice_cream[]", que indica a PHP que se trata de un control que permite selecciones múltiples:

```
<html>
  <head>
    <title>
      Ingreso de datos con cuadros de lista
    </title>
  </head>
  <body>
    <h1>
      Ingreso de datos con cuadros de lista
    </h1>
    Seleccione sus sabores de helado favoritos:
    <form method="post" action="phplistbox.php">
      <select name="ice_cream[]" multiple>
        .
        .
        .
      <select>
      <br>
      <br>
      <input type="submit" value="Enviar">
    </form>
  </body>
</html>
```

Ahora tiene libertad para agregar los sabores de helado como elementos <option> en el control <select> (los elementos contenidos en los controles <select> se dan como elementos <option> en HTML):

```
<html>
  <head>
    <title>
      Ingreso de datos con cuadros de lista
    </title>
  </head>
  <body>
    <h1>
      Ingreso de datos con cuadros de lista
    </h1>
    Seleccione sus sabores de helado favoritos:
    <form method="post" action="phplistbox.php">
      <select name="ice_cream[]" multiple>
        <option>vainilla</option>
        <option>fresa</option>
        <option>chocolate</option>
        <option>arenque</option>
      </select>
      <br>
      <br>
      <input type="submit" value="Enviar">
    </form>
  </body>
</html>
```

Puede ver esta página en la Figura 5-10, en espera de sus selecciones de helado.

FIGURA 5-10 Uso de cuadros de lista en una página HTML

Bueno, ¿qué tal si leemos las selecciones que el usuario ha hecho? Hasta ahora, usted ha manejado sus selecciones de esta forma, simplemente reproduciendo con echo $_REQUEST[“ice_cream”] en el navegador:

```
  <head>
    <title>
      Lectura de datos de cuadros de lista
    </title>
  </head>
  <body>
    <h1>
      Lectura de datos de cuadros de lista
    </h1>
    Sus sabores de helado:
    <BR>
    <?php
      foreach($_REQUEST["ice_cream"];
    ?>
  </body>
</html>
```

Pero eso no funcionará aquí, porque ice_cream es una matriz, no una variable individual. De modo que podría utilizar un ciclo foreach para mostrar las selecciones de helado del usuario de esta forma en phplistbox.php:

```
  <head>
    <title>
      Lectura de datos de cuadros de lista
    </title>
  </head>
  <body>
    <h1>
      Lectura de datos de cuadros de lista
    </h1>
    Sus sabores de helado:
    <BR>
    <?php
      foreach($_REQUEST["ice_cream"] as $sabor){
        echo $sabor, "<br>";
            }
    ?>
  </body>
</html>
```

Puede ver los resultados en la Figura 5-11, donde la página PHP ha reportado con exactitud las selecciones de helado del usuario.

FIGURA 5-11 Lectura de cuadros de lista en una página HTML

Manejo de controles de contraseña

Un uso común de PHP es la comprobación de contraseñas en el servidor, lo que da acceso al usuario a un recurso teniendo la contraseña indicada y puede usar controles de contraseña para eso.

Éste es un ejemplo que pide al usuario su contraseña, phppassword.html:

```
<html>
  <head>
    <title>
      Ingreso de datos con controles de contraseña
    </title>
  </head>
  <body>
    <h1>
      Ingreso de datos con controles de contraseña
    </h1>
    <form method="post" action="phppassword.php">
      Ingrese su contraseña:

      <input name="password" type="password">
      <br>
      <br>
      <input type="submit" value="Enviar">
    </form>
  </body>
</html>
```

Esta página se encuentra en la Figura 5-12, donde el usuario ha ingresado su contraseña.

FIGURA 5-12 Uso de controles de contraseña en una página HTML

Puede almacenar la contraseña en el servidor —que es el encanto de usar PHP para la verificación de contraseñas—. En este caso, se comprueba $_REQUEST[“password”] contra la contraseña, que en este caso es“permiso”, si coincide, muestra una página de bienvenida al usuario en phppassword.php:

```
  <head>
    <title>
      Lectura de datos de controles de contraseña
    </title>
  </head>
  <body>
    <h1>
      Lectura de datos de controles de contraseña
    </h1>
    <?php
      if ($_REQUEST["password"] == "permiso"){
    ?>
      <h2>
       Contraseña aceptada
      </h2>
      Bueno, ya está adentro.<br>
      Por favor actúe con responsabilidad.
        .
        .
        .
    ?>
  </body>
</html>
```

FIGURA 5-13 Cómo obtener acceso con una contraseña

Puede ver este resultado en la Figura 5-13.

Por otra parte, si la contraseña ingresada por el usuario no es correcta, puede mostrar una página de error —observe que esta página combina HTML y PHP:

```
<head>
  <title>
    Lectura de datos de controles de contraseña
  </title>
</head>
<body>
  <h1>
    Lectura de datos de controles de contraseña
  </h1>
  <?php
    if ($_REQUEST["password"] == "permiso"){
  ?>
    <h2>
     Contraseña aceptada
    </h2>
    Bueno, ya está adentro.<br>
    Por favor actúe con responsabilidad.
  <?php
    }
    else {
  ?>
    <h2>
     Contraseña rechazada
    </h2>
```

FIGURA 5-14 Cómo negar el acceso a una contraseña errónea

```
      No ingresó la contraseña correcta.<br>
      ¿Qué es usted, algún tipo de hacker?
    <?php
      }
    ?>
  </body>
</html>
```

Y puede ver el resultado con error en la Figura 5-14.

Manejo de controles ocultos

Los controles ocultos de HTML son un buen equivalente de scripts PHP, ya que le permiten almacenar datos en páginas Web, en general invisibles para el usuario y que usted puede utilizar en el servidor (el usuario puede ver datos ocultos revisando el código fuente de una página Web).

Este ejemplo usa un control oculto llamado customer_type, para almacenar lo que pensamos del cliente, tras bastidores. En este caso, phphidden.php, customer_type se define como"buen":

```
<html>
  <head>
    <title>
      Almacenaje de datos con controles ocultos
    </title>
  </head>
  <body>
    <h1>
      Almacenaje de datos con controles ocultos
    </h1>
```

```
    ¿Qué tipo de cliente pensamos es usted? <br>
    Haga clic en el botón para averiguarlo. <br>
    <form method="post" action="phphidden.php">
      <input name="tipo_cliente" type="hidden" value="buen">
      <br>
      <br>
      <input type="submit" value="Enviar">
    </form>
  </body>
</html>
```

Sin embargo, el tipo de cliente no es aparente con sólo observar la página Web, como se aprecia en la Figura 5-15.

No obstante, puede leer los datos del control oculto customer_type de PHP en el servidor. Se ve como esto:

```
  <head>
    <title>
      Lectura de datos de controles de contraseña
    </title>
  </head>
  <body>
    <h1>
      Lectura de datos de controles de contraseña
    </h1>
    Pensamos que usted es un
    <?php
     echo $_REQUEST["customer_type"];
     ?>
    cliente.
  <body>
</html>
```

FIGURA 5-15 Uso de controles ocultos en una página HTML

FIGURA 5-16 Lectura de controles ocultos en una página HTML

Y puede ver el resultado en la Figura 5-16 —el cliente es un buen cliente.

El uso de controles ocultos con PHP como éste, es popular para almacenar información acerca del usuario (otro recurso popular para registrar información del usuario consiste en usar cookies, también a revisarse más adelante en este libro).

Manejo de mapas de imagen

También puede emplear mapas de imagen —gráficos en los que se puede hacer clic que ve en los navegadores— con PHP, aunque hacerlo requiere cierto esfuerzo adicional.

Hay un ejemplo de mapa de imagen en phpimage.html, que maneja un elemento <input type="image"> para mostrar la correspondiente imagen:

```
<html>
  <head>
    <title>
      Ingreso de datos con mapas de imagen
    </title>
  </head>
  <body>
    <h1>
      Ingreso de datos con mapas de imagen
    </h1>
    <form method="post" action="phpimagemap.php">
      Haga clic en la imagen:
      <br>

      <input name="imap" type="image" src="map.jpg">
    </form>
  </body>
</html>
```

FIGURA 5-17 Uso de controles ocultos en una página HTML

Y se aprecia el mapa de imagen en la Figura 5-17. Observe que no necesita un botón Enviar aquí —al hacer clic en la imagen se enviará la ubicación de ese vínculo incrustado en la imagen al servidor.

Así que, ¿cómo maneja los clics en imágenes en PHP? Podría intentar algo como los datos en phpimagemap.php (se le asignó el nombre imap al mapa de imagen, de modo que podría hacer un primer intento con sólo utilizar $_REQUEST["imap"]):

```
  <head>
    <title>
      Lectura de datos de mapas de imagen
    </title>
  </head>
  <body>
    <h1>
      Lectura de datos de mapas de imagen
    </h1>
      Hizo clic en el mapa de imagen en
      <?php
        echo $_REQUEST["imap"];
      ?>
  </body>
</html>
```

Sin embargo, eso no funcionará, ya que la ubicación del mouse en el mapa de imagen consta de dos coordenadas, una *x* y otra *y*. En PHP, dichas coordenadas se almacenan con el nombre del mapa de imagen con _x y _y anexadas. Esto significa que puede mostrar la ubica-

ción en que el usuario hizo clic en la imagen (en coordenadas de imagen, (0, 0) es la esquina superior izquierda de la imagen, x positivo es a la derecha, y positivo es hacia abajo y todas las medidas son en píxeles), de tal forma en phpimagemap.php:

```
<head>
    <title>
      Lectura de datos de mapas de imagen
    </title>
  </head>
  <body>
    <h1>
      Lectura de datos de mapas de imagen
    </h1>
      Hizo clic en el mapa de imagen en la ubicación (
      <?php
        echo $_REQUEST["imap_x"], ", ", $_REQUEST["imap_y"];
      ?>
      ).
  </body>
</html>
```

Y el resultado está en la Figura 5-18 —la ubicación en que el usuario hizo clic en el mapa de imagen.

RECOMENDACIÓN *Los mapas de imagen son buenos para todo tipo de cosas en PHP —con ellos, puede permitir al usuario hacer selecciones o cambiar a otras direcciones URL.*

FIGURA 5-18 Lectura de controles ocultos en una página HTML

Manejo de cargas de archivos

También puede emplear páginas Web para cargar archivos, aspecto que no es raro ver en la Web. Sin embargo, muy pocos libros acerca de PHP abordan este tema (pero nosotros lo haremos).

Debe configurar la forma con múltiples partes si desea usar controles de carga de archivos. En este ejemplo, ello significa que el atributo enctype (tipo de codificación), del elemento <form>, se asigna a"multipart/form-data". También puede establecer el atributo de acción a la dirección URL, a donde desea que se envíen los datos del archivo —aquí se trataría de phpfile.php— y el método a"post":

```
<html>
  <head>
    <title>
      Ingreso de datos con cargas de archivos
    </title>
  </head>
  <body>
    <h1>
      Ingreso de datos con cargas de archivos
    </h1>
    <form method="post" action="phpimagemap.php"

      enctype="multipart/form-data"
      action="phpfile.php" method="post">
      .
      .
      .
      <input type="submit" value="Enviar archivo" />
    </form>
  </body>
</html>
```

Para cargar realmente el archivo, se usa un control de carga de archivos, <input type="file">. En este caso, usted podría nombrar el control de carga userfile, como sigue en phpfile.html:

```
<html>
  <head>
    <title>
      Ingreso de datos con cargas de archivos
    </title>
  </head>
  <body>
    <h1>
      Ingreso de datos con cargas de archivos
    </h1>
    <form method="post" action="phpimagemap.php"

      enctype="multipart/form-data"
      action="phpfile.php" method="post">
      Cargar el archivo: <input name="userfile" type="file" />
      <br>
      <br>
      <input type="submit" value="Enviar archivo" />
    </form>
  </body>
</html>
```

Éste es el archivo que vamos a probar su carga: file.txt:

```
Éste
es
el
contenido
del archivo.
```

Puede ver cómo luce esto en la Figura 5-19, donde el usuario ha buscado el archivo que va a cargar.

Bueno, eso controla los aspectos de HTML —¿y qué pasa con PHP?—. Se usa la matriz $_FILES en PHP para manejar archivos cargados; éstos son los elementos disponibles de la matriz:

- **$_FILES['userfile']['name']** El nombre del archivo en la máquina del usuario.
- **$_FILES['userfile']['type']** El tipo MIME del archivo. Por ejemplo, éste podría ser "image/jpeg" o "text/plain".
- **$_FILES['userfile']['size']** El tamaño del archivo cargado (en bytes).
- **$_FILES['userfile']['tmp_name']** El nombre temporal del archivo en el que se almacenó el archivo cargado en el servidor.
- **$_FILES['userfile']['error']** El código de error asociado con esta carga de archivo.

Como verá, hay mucho poder aquí. Para leer realmente el archivo cargado, vamos a usar algunas técnicas que se presentarán más adelante en este libro, tematizados bajo manejo de archivos.

FIGURA 5-19 Carga de un archivo

Se comienza adquiriendo un *manejador de archivos* en concordancia con el archivo. Este valor se corresponde con el archivo, hasta donde concierne a PHP y para conseguir un manejador de archivos, se abre un archivo. Las funciones de manejo de archivos de PHP comienzan a menudo con"f", y abrir un archivo no es la excepción —se abre un archivo con la función fopen—. Abrir un archivo da acceso a los datos contenidos en él; el nombre del archivo que abrirá, como se vio en la lista anterior, es $FILES['userfile']['tmp_name']; de modo que es así como se abre el archivo cargado —note el argumento"r"final, que abre el archivo para lectura—; en contraste con abrirlo para escritura, que sobreescribiría los datos contenidos en el archivo:

```
<html>
  <head>
    <title>Lectura de datos del archivo</title>
  </head>
  <body>
    <h1>Lectura de datos del archivo</h1>
    <br>
    El archivo contenía:
    <br>
    <?php
      $handle = fopen($_FILES['userfile']['tmp_name'], "r");
        .
        .
        .
    ?>
  </body>
</html>
```

Bueno, eso está muy bien —ha abierto el archivo cargado y tiene el manejador de archivos correspondiente—. Ahora puede leer los datos del archivo. Este ejemplo supone que los datos contenidos en el archivo son datos de texto y leeremos cadenas de texto del archivo en repetidas ocasiones hasta llegar al final del archivo. Puede comprobar, cuando haya llegado al final del archivo con la función feof, que devuelve el valor verdadero al final de un archivo y no hay más datos por leer.

La forma en que suele funcionar el proceso de lectura de archivos consiste en usar un ciclo while que se repite, mientras no se llegue al final del archivo. Tiene este aspecto:

```
<html>
  <head>
    <title>Lectura de datos del archivo</title>
  </head>
  <body>
    <h1>Lectura de datos del archivo</h1>
    <br>
    El archivo contenía:
    <br>
    <?php
      $handle = fopen($_FILES['userfile']['tmp_name'], "r");
      while (!feof($manejador)){
        .
        .
        .
      }
    ?>
  </body>
</html>
```

Cada repetición del ciclo podemos leer otra línea de texto del archivo, recurriendo a la función fgets, que lee una cadena del archivo:

```
<html>
  <head>
    <title>Lectura de datos del archivo</title>
  </head>
  <body>
    <h1>Lectura de datos del archivo</h1>
    <br>
    El archivo contenía:
    <br>
    <?php
      $handle = fopen($_FILES['userfile']['tmp_name'], "r");
      while (!feof($manejador)){
        $texto = fgets($manejador);
        .
        .
        .
      }
    ?>
  </body>
</html>
```

Este código lee una línea de texto del archivo y la almacena en la variable $texto. Ahora puede mostrar esa línea de texto en el navegador:

```
<html>
  <head>
    <title>Lectura de datos del archivo</title>
  </head>
  <body>
    <h1>Lectura de datos del archivo</h1>
    <br>
    El archivo contenía:
    <br>
    <?php
      $handle = fopen($_FILES['userfile']['tmp_name'], "r");
      while (!feof($manejador)){
        $texto = fgets($manejador);
        echo $texto, "<br>";
      }
    ?>
  </body>
</html>
```

Bueno, eso recorre todo el contenido del archivo, reproduciéndolo con echo, línea por línea, en el navegador. Cuando termine de trabajar con un archivo debe cerrarlo y eso funciona de la siguiente manera, donde se pasa el manejador del archivo a la función fclose:

```
<html>
  <head>
    <title>Lectura de datos del archivo</title>
  </head>
```

FIGURA 5-20 Cómo mostrar el contenido de un archivo cargado

```
  <body>
    <h1>Lectura de datos del archivo</h1>
    <br>
    El archivo contenía:
    <br>
    <?php
      $handle = fopen($_FILES['userfile']['tmp_name'], "r");
      while (!feof($manejador)){
        $texto = fgets($manejador);
        echo $texto, "<br>";
      }
      fclose($manejador);
    ?>
  </body>
</html>
```

Puede ver todo esto en acción en la Figura 5-20, donde se muestra el contenido del archivo cargado.

Aunque este ejemplo funcionó con un archivo de texto, también puede cargar archivos binarios, desde luego. Todo lo que debe hacer es emplear las técnicas de manejo de archivos binarios, que se verán en la exposición general del manejo de archivos en este libro.

Así que ahora ha cargado archivos enviados por el usuario. Genial.

Manejo de botones

Existe un control popular de HTML que aún no hemos abordado —los botones—. Aunque a menudo se ven botones en páginas Web, es más difícil trabajar con ellos a través de scripts en el servidor por una razón —rebotan—. Es decir, no hay datos almacenados que se envíen al servidor cuando se hace clic en un botón Enviar.

Así que, ¿cómo se manejan los botones en páginas Web con PHP? Aquí existe un par de soluciones y el resto de este capítulo se dedica a ellas —si no le interesan mucho los botones en sus páginas Web, siéntase en libertad de continuar con el capítulo siguiente.

Cómo hacer que persistan los datos de botones

Una solución obvia a la naturaleza fugaz de los datos de los botones, consiste en hacer que tales datos persistan. Podría lograrlo almacenando datos de texto en un control oculto, por ejemplo:

```
<html>
  <head>
    <title>Manejo de botones</title>
  <head>
  <body>
    <h1>Manejo de botones</H1>
    <form name="form1" action="phpbuttons.php" method="post">
      <input type="hidden" name="button">
        .
        .
        .
    </form>
  </body>
</html>
```

Luego puede agregar, por ejemplo, tres botones HTML estándar a la página:

```
<html>
  <head>
    <title>Manejo de botones</title>
  </head>
  <body>
    <h1>Manejo de botones</H1>
    <form name="form1" action="phpbuttons.php" method="post">
      <input type="hidden" name="button">
      <input type="button" value="Botón 1" onclick="setbutton1()">
      <input type="button" value="Botón 2" onclick="setbutton2()">
      <input type="button" value="Botón 3" onclick="setbutton3()">
    </form>
  </body>
</html>
```

Así pues, ¿de qué manera almacenar datos en el control oculto, cuando el usuario hace clic en un botón? Como éstos son botones HTML estándar, toda la acción debe realizarse en el navegador —ello significa usar un lenguaje de creación de scripts en el navegador, como JavaScript—; no hay otra opción. Así es como se almacena el nombre del botón en que se hizo clic en el control oculto:

```
<html>
  <head>
    <title>Manejo de botones</title>
    <script language="JavaScript">
      function setbutton1()
```

```
      {
        document.form1.button.value = "button 1"
        .
        .
        .
      }

      function setbutton2()
      {
        document.form1.button.value = "button 2"
        .
        .
        .
      }

      function setbutton3()
      {
        document.form1.button.value = "button 3"
        .
        .
        .
      }
    </script>
  </head>
  <body>
    <h1>Manejo de botones</H1>
    <form name="form1" action="phpbuttons.php" method="post">
      <input type="hidden" name="button">
      <input type="button" value="Botón 1" onclick="setbutton1()">
      <input type="button" value="Botón 2" onclick="setbutton2()">
      <input type="button" value="Botón 3" onclick="setbutton3()">
    </form>
  </body>
</html>
```

Después de que el nombre del botón se ha almacenado en el campo oculto, puede completar el proceso enviando la forma desde JavaScript:

```
<html>
  <head>
    <title>Manejo de botones</title>
    <script language="JavaScript">
      function setbutton1()
      {
        document.form1.button.value = "button 1"
        form1.submit()
      }

      function setbutton2()
      {
        document.form1.button.value = "button 2"
        form1.submit()
      }
```

```
      function setbutton3()
      {
        document.form1.button.value = "button 3"
        form1.submit()
      }
    </script>
  </head>
  <body>
    <h1>Manejo de botones</H1>
    <form name="form1" action="phpbuttons.php" method="post">
      <input type="hidden" name="button">
      <input type="button" value="Botón 1" onclick="setbutton1()">
      <input type="button" value="Botón 2" onclick="setbutton2()">
      <input type="button" value="Botón 3" onclick="setbutton3()">
    </form>
  </body>
</html>
```

Esta página, phpbuttons.php, se encuentra en la Figura 5-21.

Ahora resulta fácil leer los datos almacenados en un campo oculto utilizando PHP. Así se ve el script leído por el botón en que ha hecho clic, phpbuttons.php:

```
<html>
  <head>
    <title>
      Lectura de botones
    </title>
  </head>
  <body>
    <h1>Lectura de botones</h1>
```

FIGURA 5-21 Botones en una página Web

FIGURA 5-22 Cómo mostrar el botón en que se hizo clic

```
    Usted hizo clic en
    <?php
      if (isset($_REQUEST["button"])) {
        echo $_REQUEST["button"], "<br>";
      }
    ?>
  </body>
</html>
```

El resultado se encuentra en la Figura 5-22, donde el script PHP ha identificado correctamente el botón en que hizo clic.

Sin embargo, éste es un libro sobre PHP, no sobre JavaScript. ¿Existe mejor forma para manejar botones en páginas Web?

Uso de botones de envío como botones HTML

Puesto que la acción en PHP se lleva a cabo en el servidor, también puede usar botones de envío en lugar de botones HTML estándar. Los botones de envío tienen el mismo aspecto que los de HTML, así que el usuario no será más astuto.

Éste es un modo en que podría usar botones de envío para imitar botones HTML —crear tres formas para tres botones:

```
<html>
  <head>
    <title>Lectura de botones de envío</title>
  </head>

  <body>
    <h1>Lectura de botones de envío</h1>
    <form name="form1" action="phpsubmit.php" method="post">
```

```
      <input type="submit" value="Botón 1">
        .
        .
        .
    </form>
    <form name="form2" action="phpsubmit.php" method="post">
      <input type="submit" value="Botón 2">
        .
        .
        .
    </form>

    <form name="form3" action="phpsubmit.php" method="post">
      <input type="submit" value="Botón 3">
        .
        .
        .
    </form>
  </body>
</html>
```

Luego dé a cada forma su propio control oculto, con el nombre del botón de modo similar a phpsubmit.html:

```
<html>
  <head>
    <title>Lectura de botones de envío</title>
  </head>

  <body>
    <h1>Lectura de botones de envío</h1>
    <form name="form1" action="phpsubmit.php" method="post">
      <input type="hidden" name="button" value="button 1">
      <input type="submit" value=" Botón 1">
    </form>
    <form name="form2" action="phpsubmit.php" method="post">
      <input type="hidden" name="button" value="button 2">
      <input type="submit" value="Botón 2">
    </form>

    <form name="form3" action="phpsubmit.php" method="post">
      <input type="hidden" name="button" value="button 3">
      <input type="submit" value="Botón 3">
    </form>
  </body>
</html>
```

Puede ver el resultado en la Figura 5-23, donde los tres botones "aparentes" son en realidad botones de envío.

FIGURA 5-23 Uso de botones de envío como botones HTML

Ahora todo lo que necesita hacer es leer el valor almacenado en el control oculto en su script PHP, para determinar en qué botón hizo clic, como en phpsubmit.php:

```
<html>
  <head>
    <title>
      Lectura de botones de envío
    </title>
  </head>
  <body>
    <h1>
      Lectura de botones de envío
    </h1>
    Usted hizo clic en
    <?php
      if (isset($_REQUEST["button"])) {
        echo $_REQUEST["button"], "<br>";
      }
    ?>
  </body>
</html>
```

El resultado se ve en la Figura 5-24, donde el script PHP ha identificado correctamente el botón en que hizo clic. Nada mal.

De hecho, hay una forma más sencilla para usar botones de envío como botones HTML estándar —podría leer realmente el valor (es decir, las leyendas), de los botones de envío en PHP—. Así que no necesita controles ocultos para alojar el nombre de cada botón, según phpsubmit2.html:

```
<html>
  <head>
    <title>
      Uso de botones de envío con valores
```

FIGURA 5-24 Identificación de un botón en el que hizo clic

```
    </title>
  </head>
  <body>
    <h1>
      Uso de botones de envío con valores
    </h1>
    <form name="form1" action="phpsubmit2.php" method="post">
      <input type="submit" name="button" value="button 1">
    </form>

    <form name="form2" action="phpbuttons2.php" method="post">
      <input type="submit" name="button" value="button 2">
    </form>

    <form name="form3" action="phpbuttons2.php" method="post">
      <input type="submit" name="button" value="button 3">
    </form>
  </body>
</html>
```

Puede ver esta página, phpsubmit2.html, en la Figura 5-25.

Ahora leerá, simplemente, el valor de cada botón de envío en phpsubmit2.php:

```
<html>
  <head>
    <title>
      Lectura de botones de envío
    </title>
  </head>
  <body>
    <h1>
      Lectura de botones de envío
    </h1>
```

FIGURA 5-25 Uso de botones de envío con valores

```
    Usted hizo clic en
    <?php
      if (isset($_REQUEST["button"])) {
        echo $_REQUEST["button"], "<br>";
      }
    ?>
  </body>
</html>
```

Y notará los resultados de phpsumbit2.php en la Figura 5-26, donde se identificó el botón correcto.

FIGURA 5-26 Lectura de botones de envío con valores

De hecho, puede simplificar aún más esto —aquí no necesita tres formas por separado—; puede usar tres botones de envío diferentes en la misma forma. Se ve así en phpsubmit3.html:

```
<html>
  <head>
    <title>
      Uso de botones de envío en la misma forma
    </title>
  </head>

  <body>
    <h1>
      Uso de botones de envío en la misma forma
    </h1>
    <form name="form1" action="phpsubmit3.php" method="post">
      <input type="submit" name="button" value="button 1">

      <input type="submit" name="button" value="button 2">

      <input type="submit" name="button" value="button 3">
    </form>
  </body>
</html>
```

Puede ver esta página en la Figura 5-27.

FIGURA 5-27 Múltiples botones de envío en la misma forma

Entonces, puede leer con facilidad el valor del botón de envío en que hizo clic en phpsubmit3.php:

```
<html>
  <head>
    <title>
      Lectura de botones de envío
    </title>
  </head>
  <body>
    <h1>
      Lectura de botones de envío
    </h1>
    Usted hizo clic en
    <?php
      if (isset($_REQUEST["button"])) {
        echo $_REQUEST["button"], "<br>";
      }
    ?>
  </body>
</html>
```

Y el resultado se encuentra en la Figura 5-28, donde el script phpsumbit3.php hizo su función correctamente —determinó en qué botón se hizo clic.

FIGURA 5-28 Lectura de múltiples botones de envío en la misma forma

CAPÍTULO 6

Poder de manejo de navegadores de PHP

Este capítulo nos introduce al poder de PHP cuando se trata de trabajar con navegadores —uso de variables de servidor para tomar el control del navegador, determinar tipo de navegador, leer datos de páginas Web usando matrices personalizadas, etcétera—. El capítulo anterior nos inició en el manejo de la conexión entre navegador y servidor en PHP, este capítulo parte de ese punto, con la introducción de muchas herramientas nuevas y poderosas.

Asimismo, tendrá acceso a una exposición sobre cómo validar datos proporcionados por el usuario en este capítulo. Saber cómo comprobar y validar datos —¿el usuario ha ingresado datos en el campo requerido? ¿El usuario ha ingresado un número donde se requiere un número?— Es parte importante de la creación de aplicaciones Web. Así, en este capítulo veremos cómo implementar la validación de datos.

Uso de variables de servidor de PHP

Resulta que hay una matriz superglobal (es decir, accesible desde todas partes), $_SERVER, que contiene mucha información acerca de los eventos en su aplicación Web. Puede recurrir a $_SERVER['PHP_SELF'] para obtener el nombre del script actual, por ejemplo; $_SERVER ['REQUEST_METHOD'] aloja el método de solicitud usado ("GET","POST", etcétera), $_ SERVER['HTTP_USER_AGENT'], alberga el tipo del navegador del usuario, etcétera. Puede ver una muestra de las variables de servidor más útiles disponibles en $_SERVER, en la Tabla 6-1.

Este ejemplo (el script, phpserver.php, indica el nombre del script y el puerto usado para acceder a él en el servidor):

```
<html>
  <head>
    <title>
      Bienvenido a mi script
    </title>
  </head>

  <body>
    <h1>Bienvenido a mi script</h1>
```

```
    <?php
      echo "Ha tenido acceso a ", $_SERVER["PHP_SELF"], " en el puerto ",
       $_SERVER["SERVER_PORT"];
    ?>
  </body>
</html>
```

Variable de servidor	Descripción
'AUTH_TYPE'	Cuando se ejecuta con Apache, como módulo de autenticación HTTP, esta variable aloja el tipo de autenticación.
'DOCUMENT_ROOT'	El directorio raíz del documento bajo el que se ejecuta el script, como se define en el archivo de configuración del servidor.
'GATEWAY_INTERFACE'	Qué revisión de la especificación CGI utiliza el servidor; como 'CGI/1.1'.
'PATH_TRANSLATED'	Ruta basada en el sistema de archivos al script actual.
'PHP_AUTH_PW'	Cuando se ejecuta con Apache como módulo que realiza autenticación HTTP, esta variable aloja la contraseña proporcionada por el usuario.
'PHP_AUTH_USER'	Cuando se ejecuta con Apache como módulo, para autenticación HTTP, esta variable aloja el nombre de usuario proporcionado por el usuario.
'PHP_SELF'	Nombre de archivo del script actualmente en ejecución, relativo a la raíz del documento.
'QUERY_STRING'	La cadena de consulta, si hubiera una, con la que se tuvo acceso a la página.
'REMOTE_ADDR'	Dirección IP desde la que el usuario ve la página actual.
'REMOTE_HOST'	Nombre de host desde donde el usuario ve la página actual.
'REMOTE_PORT'	Puerto usado en la máquina del usuario para establecer comunicación con el servidor Web.
'REQUEST_METHOD'	Especifica qué método de solicitud se manejó para tener acceso a la página; como 'GET', 'HEAD', 'POST', 'PUT'.
'REQUEST_URL'	Dirección URL establecida para acceder a esta página, como '/index.html'.
'SCRIPT_FILENAME'	Nombre de ruta absoluto del script actualmente en ejecución.
'SCRIPT_NAME'	Contiene la ruta del script actual. Es útil para páginas que necesitan apuntar hacia sí mismas.
'SERVER_ADMIN'	Valor que se da a la directiva SERVER_ADMIN (para Apache), en el archivo de configuración del servidor Web.
'SERVER_NAME'	Nombre del host servidor en que se ejecuta el script.
'SERVER_PORT'	Puerto en el servidor usado por el servidor Web para comunicación. De forma predeterminada, es el puerto '80'.
'SERVER_PROTOCOL'	Nombre y revisión del protocolo de información a través del que se solicitó la página; como 'HTTP/1.0'.
'SERVER_SIGNATURE'	Cadena conteniendo la versión del servidor y nombre del host virtual, que se agregan a páginas generadas por el servidor.
'SERVER_SOFTWARE'	Cadena de identificación del servidor.

TABLA 6-1 Variables de servidor de PHP

FIGURA 6-1 Script de PHP identificado por sí solo

Puede ver los resultados de este script en la Figura 6-1, donde el script se ha identificado correctamente.

Uso de encabezados HTTP

Además de las variables de servidor, en la Tabla 6-1, también tiene acceso a encabezados HTTP en la matriz $_SERVER. Éstos los envía el navegador y contienen información acerca del mismo.

Un encabezado HTTP útil es HTTP_USER_AGENT, referente al tipo de navegador del usuario: es decir, $_SERVER['HTTP_USER_AGENT'] refiere el tipo de navegador del usuario. Como debe conectarse con el usuario a través de su navegador en aplicaciones Web, conocer su tipo de navegador puede ser invaluable en algunos casos.

Para ver los encabezados HTTP accesibles para usted, se encuentran en la Tabla 6-2.

Variable HTTP	Descripción
'HTTP_ACCEPT'	Texto en el encabezado Accept: de la solicitud actual, si es que hay una.
'HTTP_ACCEPT_CHARSET'	Texto en el encabezado Accept-Charset: de la solicitud actual, si hay una, como: '*, utf-8'.
'HTTP_ACCEPT_ENCODING'	Texto en el encabezado Accept-Encoding: de la solicitud actual, si hay una, como: 'zip'.
'HTTP_ACCEPT_LANGUAGE'	Texto en el encabezado Accept-Language: de la solicitud actual, si hay una, como: 'en' de Inglés.
'HTTP_CONNECTION'	Texto en el encabezado Connection: de la solicitud actual, si hay una, como: 'Keep-Alive'.
'HTTP_HOST'	Texto en el encabezado Host: de la solicitud actual, si hay una.
'HTTP_REFERER'	Dirección de la página (si hay una) referida por el agente del usuario a la página actual. Esto lo determina el navegador.
'HTTP_USER_AGENT'	Texto en el encabezado Host: de la solicitud actual, si hay una. Ésta es una cadena que denota al navegador el acceso a la página.

TABLA 6-2 Variables de servidor de HTTP

Uno de los encabezados HTTP más populares para utilizarse con PHP es HTTP_USER_AGENT, para ver a continuación.

Cómo determinar el tipo de navegador del usuario

A veces es importante conocer el tipo de navegador del usuario. Por ejemplo, podría usar el elemento de desplazamiento <marquee> —sólo disponible en Internet Explorer—. Comprobar si el usuario tiene Internet Explorer resulta decisivo en estos casos.

Puede manejar el encabezado HTTP_USER_AGENT para determinar qué tipo de navegador tiene el usuario. Este ejemplo, phpbrowser.html, se conecta a una página PHP, phpbrowser.php, para señalar qué tipo de navegador utiliza:

```
<html>
  <head>
     <title>
       Cómo determinar el tipo de navegador del usuario
     </title>
  </head>
  <body>
    <h1>Cómo determinar el tipo de navegador del usuario</h1>
    <form method=" post"  action=" phpbrowser.php" >
      Haga clic en el botón para determinar su tipo de navegador....
      <input type=" submit"  value=" Enviar" >
    </form>
  </body>
</html>
```

Puede ver esta página en la Figura 6-2.

FIGURA 6-2 Primera página en la aplicación para comprobar el navegador

Cuando el usuario hace clic en el botón de la Figura 6-2, el navegador contacta a phpbrowser.php en el servidor, enviando todos los encabezados HTTP habituales (incluyendo HTTP_USER_AGENT). La cadena devuelta por $_SERVER[“HTTP_USER_AGENT”] incluirá el texto “MSIE”, si el usuario cuenta con Microsoft Internet Explorer y puede reconocer tal hecho con el elemento <marquee>:

```
<html>
  <head>
        <title>Determinando el tipo de navegador</title>
  </head>
  <body>
    <h1>Determinando el tipo de navegador</h1>
    <br>
    <?php
      if(strpos($_SERVER["HTTP_USER_AGENT" ], "MSIE" )){
        echo("<marquee><h1>Usted utiliza Internet Explorer</h1></marquee>" );
      }
        .
        .
        .
    ?>
  </body>
</html>
```

Por otra parte, para comprobar si el usuario tiene Firefox, se busca “Firefox”en $_SERVER[“HTTP_USER_AGENT”] de esta forma:

```
<html>
  <head>
        <title>Determinando el tipo de navegador</title>
  </head>
  <body>
    <h1>Determinando el tipo de navegador</h1>
    <br>
    <?php
      if(strpos($_SERVER["HTTP_USER_AGENT" ], "MSIE" )){
        echo("<marquee><h1>Usted utiliza Internet Explorer</h1></marquee>" );
      }
      elseif (strpos($_SERVER["HTTP_USER_AGENT" ], "Firefox" )) {
        echo("<h1>Usted utiliza Firefox</h1>" );
      }
        .
        .
        .
    ?>
  </body>
</html>
```

Si el usuario no tiene estos navegadores, también puede indicar ese hecho:

```
<html>
  <head>
        <title>Determinando el tipo de navegador</title>
  </head>
  <body>
    <h1>Determinando el tipo de navegador</h1>
    <br>
    <?php
      if(strpos($_SERVER["HTTP_USER_AGENT" ], "MSIE" )){
        echo("<marquee><h1>Usted utiliza Internet Explorer</h1></marquee>" );
      }
      elseif (strpos($_SERVER["HTTP_USER_AGENT" ], "Firefox" )) {
        echo("<h1>Usted utiliza Firefox</h1>" );
      }
      else {
        echo("<h1>Usted no utiliza Internet Explorer o Firefox</h1>" );
      }
    ?>
  </body>
</html>
```

Puede ver los resultados en Internet Explorer en la Figura 6-3, en un elemento <marquee> de desplazamiento.

Y los resultados en Firefox están en la Figura 6-4.

FIGURA 6-3 Identificación de Internet Explorer

FIGURA 6-4 Identificación de Firefox

Cómo redirigir navegadores con encabezados HTTP

También es posible crear sus propios encabezados HTTP y devolverlos al navegador; en PHP se crean encabezados HTTP propios con la función header. Probablemente el encabezado HTTP más común para crear y devolver al navegador sea el encabezado de redirección, que reorienta el navegador a una nueva dirección URL.

Aquí hay un ejemplo; suponga que tiene dos páginas HTML a las que desea redirigir a los usuarios (welcome.html):

```
<html>
  <head>
    <title>Bienvenido</title>
  </head>
  <body>
    <h1>Bienvenido</h1>
    Bienvenido a esta aplicación.
  </body>
</html>
```

y hello.html:

```
<html>
  <head>
    <title>Hola</title>
  </head>
```

```
  <body>
    <h1>Hola</h1>
    Hola desde esta aplicación.
  </body>
</html>
```

Éste es phpredirect.html, pasa el nombre de la página a la que se redirigirá usando el valor (es decir, la leyenda) en el botón Enviar de dos formas: una para welcome.html y otra para hello.html:

```
<html>
  <head>
    <title>Redirigiendo el navegador</title>
  </head>
  <body>
    <h1>Redirigiendo el navegador</h1>
    ¿Qué página le gustaría ver?
    <form name=" form1"  action=" phpredirect.php"  method=" post" >
      <input type=" submit"  name=" button"  value=" bienvenido" >
    </form>
    <form name=" form2"  action=" phpredirect.php"  method=" post" >
      <input type=" submit"  name=" button"  value=" hola" >
    </form>
  </body>
</html>
```

Y puede ver esta página, phpredirect.html, en la Figura 6-5.

FIGURA 6-5 phpredirect.html

El script de redirección, phpredirect.php, asume el nombre del script al que se reorientará, desde el valor del botón Enviar en que se hizo clic ($_REQUEST['button']) de esta forma:

```
<?php
    $redirigir = "Location: " . $_REQUEST['button'] . ".html" ;
         .
         .
         .
?>
```

Ahora la variable $redirigir aloja el texto de un nuevo encabezado de redirección: "Location: welcome.html" o "Location: hello.html". Puede crear el encabezado HTTP real y devolverlo al navegador, mediante la función header de PHP de esta forma:

```
<?php
    $redirigir = "Location: " . $_REQUEST['button'] . ".html";
    echo header($redirigir);
?>
```

Este script, phpredirect.php, se aprecia en la Figura 6-6, donde la persona ha hecho clic en el botón bienvenido y ha sido redirigida a welcome.html.

Como notará, puede lograr fácilmente que el navegador haga lo que usted desea desde su código PHP, dirigiéndolo a donde usted desea.

Otro uso para los encabezados de redirección es con mapas de imagen, las imágenes en que puede hacer clic como se vio en el capítulo anterior. Por ejemplo, en PHP, usted podría comprobar si el usuario hizo clic en un punto en particular de un mapa de imagen:

```
<?php
    if ($REQUEST["map_x" ] > 20 && $REQUEST["map_x" ] < 80) {
          if ($REQUEST["map_y" ] > 10 && $REQUEST["map_y" ] < 40) {
         .
         .
         .
         }
    }
?>
```

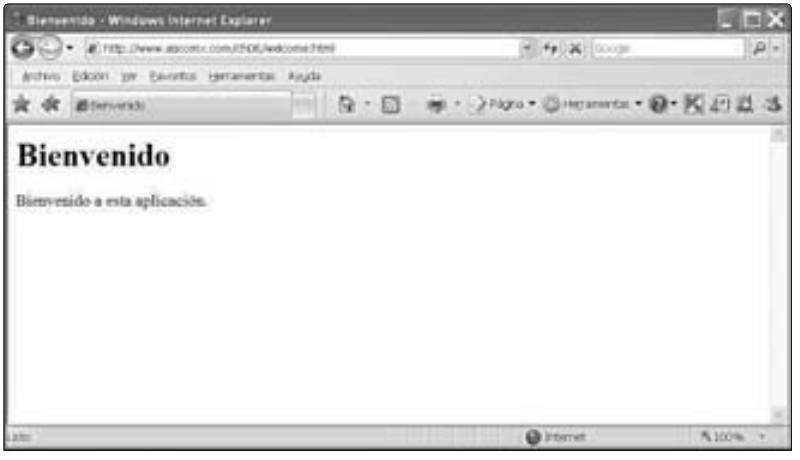

FIGURA 6-6 welcome.html

Y si el usuario hiciera clic en un punto de la imagen, puede lograr que su navegador se dirija a una nueva dirección URL con sólo redirigirlo de esta forma:

```
<?php
    if ($REQUEST["map_x" ] > 20 && $REQUEST["map_x" ] < 80) {
          if ($REQUEST["map_y" ] > 10 && $REQUEST["map_y" ] < 40) {
              $redirigir = "Location: www.php.net" ;
              echo header($redirigir);
          }
    }
?>
```

Formidable.

Vaciado de todos los datos en una forma de una sola vez

Mientras analizamos el tema del manejo de datos enviados a sus scripts de PHP desde el navegador, resulta útil tener a la mano un script que vacíe todos los datos de una forma HTML. Este script es útil para depurar sus aplicaciones Web.

Por ejemplo, dé un vistazo a phpform.html, mostrando una forma con varios controles:

```
<html>
  <head>
    <title>
      Vaciado de datos de una forma
    </title>
  </head>
  <body>
   <h1>Vaciado de datos de una forma</h1>
     <form method=" post"  action=" phpform.php" >
     ¿Cuántos años tiene?<input name=" edad"  type=" text" >
     <br>
     <br>
     ¿Cuántas hijas tiene?<input name=" número_hijas"  type=" text" >
     <br>
     <br>
     Seleccione su sabor o sabores de helado favoritos:
     <select name=" helado[]"  multiple>
            <option>Vainilla</option>
            <option>Chocolate</option>
            <option>Fresa</option>
            <option>Sardina</option>
     </select>
     <br>
     <br>
     <input type=" submit"  value=" Enviar" >
     </form>
  </body>
</html>
```

Puede ver esta página, phpform.html, en acción en la Figura 6-7, donde el usuario ha ingresado datos en la forma.

Es bastante fácil crear un script que vacíe todos los datos presentes en la forma phpform.html. Puede comenzar recorriendo en ciclo todos los pares clave/valor de la matriz $_REQUEST:

```
<html>
  <head>
    <title>
      Vaciado de datos de una forma
    </title>
  </head>
  <body>
    <h1>Vaciado de datos de una forma</h1>
    Éstos son los datos de la forma:
    <br>
    <?php
      foreach($_REQUEST as $key => $value){
        .
        .
        .
      }
    ?>
  </body>
</html>
```

Si el elemento de datos de forma con que trabaja es una matriz —como la que contiene sabores de helado en phpform.html, puede recorrer esa matriz en ciclos, mostrando todos los

FIGURA 6-7 Ingreso de datos en una forma

elementos que contiene de la siguiente manera—, observe que comprobamos si el elemento de datos es una matriz con la función is_array de PHP:

```
<html>
  <head>
    <title>
      Vaciado de datos de una forma
    </title>
  </head>
  <body>
    <h1>Vaciado de datos de una forma</h1>
    Éstos son los datos de la forma:
    <br>
    <?php
      foreach($_REQUEST as $key => $value){
        if(is_array($value)){
          foreach($value as $item){
            echo $key, " => ", $item, "<br>";
          }
        }
        .
        .
        .
      }
    ?>
  </body>
</html>
```

Si el elemento datos de la forma no es una matriz, es un simple elemento de datos y puede reproducirlo con echo al navegador, de esta manera:

```
<html>
  <head>
    <title>
      Vaciado de datos de una forma
    </title>
  </head>
  <body>
    <h1>Vaciado de datos de una forma</h1>
    Éstos son los datos de la forma:
    <br>
    <?php
      foreach($_REQUEST as $key => $value){
        if(is_array($value)){
          foreach($value as $item){
            echo $key, " => ", $item, "<br>";
          }
        }
```

FIGURA 6-8 Vaciado de los datos en una forma

```
        else {
          echo $key, " => ", $value, "<br>" ;
        }
      }
    ?>
  </body>
</html>
```

Bueno, ése es el script phpform.php, un script genérico para vaciar datos de una forma. Puede ver este script en acción, vaciando los datos que se le pasan desde phpform.html, en la Figura 6-8.

Manejo de datos de una forma con matrices personalizadas

Quizá recuerde del capítulo anterior que PHP maneja datos de controles de selección múltiple, como cuadros de lista utilizando matrices. Así se ve esto:

```
Seleccione sus sabores de helado favoritos:
<form method=" post"  action=" phplistbox.php" >
  <select name=" ice_cream[]"  multiple>
    <option>vainilla</option>
    <option>fresa</option>
    <option>chocolate</option>
    <option>arenque</option>
  </select>
```

También puede indicar a PHP que agrupe datos en matrices personalizadas con sintaxis similar. Por ejemplo, suponga que deseara agrupar datos de tal manera, que se le envían a usted en una matriz llamada data.

Podría tener un campo de texto tomando el nombre del usuario y a cuyos datos deseara referirse como data['name']. Asimismo, sería posible configurar dicho control en una página Web así:

```
<html>
  <head>
    <title>
      Uso de matrices de forma personalizadas
    </title>
  </head>
  <body>
    <h1>Uso de matrices de forma personalizadas</h1>
    <form method=" post"  action=" phparray.php" >
      ¿Cuál es su nombre?
      <input name=" data[name]"  type=" text" >
      <br>
      <br>
        .
        .
        .
      <input type=" submit"  value=" Enviar" >
    </form>
  </body>
</html>
```

De manera similar, podría preguntar al usuario cuál es su sabor de helado favorito, almacenando ese texto en el elemento de matriz data['flavor'] de esta forma:

```
<html>
  <head>
    <title>
      Uso de matrices de forma personalizadas
    </title>
  </head>
  <body>
    <h1>Uso de matrices de forma personalizadas</h1>
    <form method=" post"  action=" phparray.php" >
      ¿Cuál es su nombre?
      <input name=" data[name]"  type=" text" >
      <br>
      <br>
      ¿Cuál es su sabor de helado favorito?
      <input name=" data[flavor]"  type=" text" >
      <br>
      <br>
      <input type=" submit"  value=" Enviar" >
    </form>
  </body>
</html>
```

Ahora, phparray.html, puede verse en la Figura 6-9.

Puede tener una matriz alojando datos de la otra matriz $_REQUEST y referirse a ellos mediante las claves ya especificadas. Esto se ve así en phparray.php:

```
<html>
  <head>
    <title>
      Uso de matrices de forma
    </title>
  </head>
  <body>
    <h1>Uso de matrices de forma</h1>
    Su nombre es
    <?php
      $data = $_REQUEST['data'];
      echo $data['name'], "<br>" ;
    ?>

    Su sabor favorito es
    <?php
      $data = $_REQUEST['data'];
      echo $data['flavor'], "<br>" ;
    ?>
  </body>
</html>
```

Los resultados están en la Figura 6-10, donde la aplicación tuvo éxito al recuperar datos de una matriz personalizada.

FIGURA 6-9 Almacenaje de los datos de una forma en una matriz personalizada

FIGURA 6-10 Obtención de datos de una forma de matrices personalizadas

Cómo poner todo en una página

Hasta ahora, las páginas elaboradas se sustentan en una página HTML de inicio, en conexión con una página PHP. Sin embargo, es más común manejar todo con una sola página PHP.

En este ejemplo, se pregunta al usuario cuál es su sabor de helado favorito. Tenga en mente que está diseñado para manejar todo con una página; en tal caso, los datos ingresados por el usuario —su sabor de helado favorito— se almacenarán en la matriz $_REQUEST con el nombre "flavor". Eso significa que si hay datos esperando en la matriz $_REQUEST bajo el nombre "flavor", deberá mostrarlos. De no existir los datos, debe mostrar la página de bienvenida que pregunta el sabor favorito del usuario.

Puede comprobar si tiene datos en espera con la función isset de PHP, que devuelve el valor TRUE mientras exista un elemento de matriz:

```
<html>
  <head>
    <title>Uso de una página para aceptar y procesar datos</title>
  </head>
  <body>
    <h1>Uso de una página para aceptar y procesar datos</h1>
    <?php
      if(isset($_REQUEST["flavor" ])){
    ?>
        .
        .
        .
    <?php
      }
    ?>
  </body>
</html>
```

Si hay datos esperando, puede mostrarlos de esta forma:

```
<html>
  <head>
    <title>Uso de una página para aceptar y procesar datos</title>
  </head>
  <body>
    <h1>Uso de una página para aceptar y procesar datos</h1>
    <?php
      if(isset($_REQUEST["flavor" ])){
    ?>
      Su sabor de helado favorito es
    <?php
      echo $_REQUEST["flavor" ];
      }
    ?>
        .
        .
        .
  </body>
</html>
```

Si, por otra parte, no hay datos en espera, el usuario no ha ingresado datos aún; así, es tiempo para preguntar cuál es su sabor de helado favorito. Así se ve esto:

```
<html>
  <head>
    <title>Uso de una página para aceptar y procesar datos</title>
  </head>
  <body>
    <h1>Uso de una página para aceptar y procesar datos</h1>
    <?php
      if(isset($_REQUEST["flavor" ])){
    ?>
      Su sabor de helado favorito es
    <?php
      echo $_REQUEST["flavor" ];
      }
      else {
    ?>
    <form method=" post"  action=" phpone.php" >
      ¿Cuál es su sabor de helado favorito?
      <input name=" flavor"  type=" text" >
      <br>
      <br>
      <input type=submit value=Enviar>
    </form>
    <?php
      }
    ?>
  </body>
</html>
```

FIGURA 6-11 ¿Cuál es su sabor favorito?

Un ejemplo de esto, phpone.php, se encuentra la Figura 6-11 —aún no hay datos esperando que la aplicación los muestre, de modo que presenta la página de bienvenida, preguntando cuál es su sabor favorito.

Después de indicar un sabor y hacer clic en el botón Enviar, la aplicación muestra su selección, como en la Figura 6-12.

No está mal —ha manejado toda esta aplicación con una página PHP, sin necesidad de una página HTML—. La aplicación completa giró en torno a un punto: comprobar si algunos datos estaban listos o no para ser leídos. Si los datos estaban listos, fueron procesados; si no, se mostró la página de bienvenida y el usuario ingresó los datos. De esta forma puede crear aplicaciones Web completas que recurren sólo a una página PHP.

FIGURA 6-12 Su sabor favorito

Ejecutar la validación de datos

En los siguientes temas desarrollaremos ejemplos para mostrarle cómo trabajar con la validación de datos en PHP. ¿Ingresó el usuario un número entero para indicar su edad? Si no, puede indicárselo con un mensaje de error, dándole otra oportunidad para hacerlo. ¿Ingresó una cadena de texto para indicar su nombre? Si no, puede informarle.

En el tema anterior, usted comprobó si había datos en espera verificando los datos almacenados para un control de entrada de datos en particular, pero no es una técnica muy general —por ejemplo, podría no requerir del usuario que ingrese datos en sus controles—. Por esa razón, estos ejemplos utilizarán un campo oculto, welcome_already_seen, para determinar si la página de bienvenida ya se ha visto. Si este elemento está presente en $_REQUEST, el usuario ya visitó la página de bienvenida y es tiempo de procesar los datos que ha ingresado; si no está presente ese elemento, debemos mostrar la página de bienvenida. En código, podría verse así:

```
if(isset($_REQUEST["welcome_already_seen" ])){
.
.
.
}
else {
    show_welcome();
}
```

Si el usuario ya vio la página de bienvenida, presumiblemente ha ingresado sus datos (eso es lo que estamos por comprobar) y debe procesar esos datos. Puede verificar si los datos ingresados están en el formato deseado con una función llamada, por ejemplo, check_data, que almacenará cualquier error encontrado en una matriz global llamada $errors_array:

```
$errors_array = array();
if(isset($_REQUEST["welcome_already_seen"])){
    check_data();
    .
    .
    .
}
else {
    show_welcome();
}
```

Ahora puede comprobar si hubo errores —la matriz $errors_array estará vacía si no hubo errores, de tal modo, podría comprobar la existencia de errores con el siguiente método—, mostrar los errores con una función llamada show_errors y luego presentar nuevamente la página de bienvenida:

```
$errors_array = array();
if(isset($_REQUEST["welcome_already_seen" ])){
    check_data();
    if(count($errors_array) != 0){
        show_errors();
```

```
            show_welcome();
        }
        .
        .
        .
    }
    else {
        show_welcome();
    }
```

Si no hubo errores, puede manejar los datos que el usuario ingresó en otra función llamada, por ejemplo, handle_data:

```
$errors_array = array();
    if(isset($_REQUEST["welcome_already_seen" ])){
        check_data();
        if(count($errors_array) != 0){
            show_errors();
            show_welcome();
        }
        else {
        handle_data();
        }
    }
    else {
        show_welcome();
    }
```

De tal modo, la idea es comprobar los datos que el usuario ingresó en una función llamada check_data:

```
    function check_data()
    {
        .
        .
        .
    }
```

La matriz $errors_array será llenada con cualquier error encontrado por la función check_data y la función show_errors mostrará dichos errores:

```
    function show_errors()
    {
        global $errors_array;
        foreach ($errors_array as $err){
            echo $err, "<br>" ;
        }
    }
```

La función show_welcome dependerá de datos que desee solicitar al usuario, desde luego. Esta función muestra los controles que el usuario debe utilizar. Una cosa que será igual en todos los casos es que deseamos cerciorarnos de crear el campo "welcome_already_seen" aquí:

```
function show_welcome()
{
    echo "<form method='post' action=phpvalidate.php'>;
    .
    .
    .
    echo "<input type='submit' value='Enviar'>" ;
    echo "<input type='hidden' name='welcome_already_seen'
        value='already_seen'>" ;
    echo "</form>" ;            }
```

Así es como funciona nuestro ejemplo de estructura de validación de datos. Ahora pongámosla a trabajar.

Cómo comprobar si el usuario ingresó los datos requeridos

Bueno, pongamos a trabajar nuestra estructura de validación en un ejemplo requiriendo del usuario ingrese algunos datos. Este ejemplo que sigue preguntará el sabor de helado favorito del usuario y, si el usuario no ingresa ningún texto, la aplicación objetará.

Comenzamos este ejemplo, phprequireddata.php, con el código que ya hemos desarrollado para realizar la validación:

```
<html>
  <head>
    <title>
      Comprobación de datos requeridos
    </title>
  </head>
  <body>
    <h1>Comprobación de datos requeridos</h1>
    <?php
      $errors_array = array();
      if(isset($_REQUEST["welcome_already_seen" ])){
         check_data();
         if(count($errors_array) != 0){
           show_errors();
           show_welcome();
         }
         else {
           handle_data();
         }
      }
      else {
        show_welcome();
      }
```

Ahora debe escribir las funciones llamadas por este código. Podría comenzar con la página show_welcome, que muestra la página de bienvenida preguntando al usuario su sabor de helado favorito y lo devuelve a la misma página bajo la clave "flavor" (observe que la forma no

necesita un atributo de acción, pues quiere que el navegador devuelva sus datos a la misma página PHP):

```
function show_welcome()
{
  echo "<form method='post'>" ;
  echo "¿Cuál es su sabor de helado favorito?" ;
  echo "<br>" ;
  echo "<input name='flavor' type='text'>" ;
  echo "<br>" ;
  echo "<br>" ;
  echo "<input type='submit' value='Enviar'>" ;
  .
  .
  .
  echo "</form>" ;
}
```

Asimismo, no olvide crear aquí el campo oculto welcome_already_seen —si existe este campo oculto, la aplicación sabe que el usuario ya ha visto la página de bienvenida:

```
function show_welcome()
{
  echo "<form method='post'>" ;
  echo "¿Cuál es su sabor de helado favorito?" ;
  echo "<br>" ;
  echo "<input name='flavor' type='text'>" ;
  echo "<br>" ;
  echo "<br>" ;
  echo "<input type='submit' value='Enviar'>" ;
  echo "<input type='hidden' name='welcome_already_seen'
   value='already_seen'>" ;
  echo "</form>" ;
}
```

A continuación, la función check_data comprobará si el usuario ha ingresado algo indicando su sabor de helado favorito. El sabor se almacenará en la matriz $_REQUEST con la clave 'flavor'; de modo que podría comprobar si el usuario ha dejado en blanco el campo de texto correspondiente, de esta forma en check_data:

```
function check_data()
{
  if($_REQUEST["flavor" ] == "" ) {
  .
  .
  .
  }
}
```

Si el usuario no ha ingresado texto, puede agregar una entrada a la matriz $errors_array de esta forma —observe que este error mostrará su mensaje en color rojo,"Indique su sabor favorito":

```
function check_data()
{
  global $errors_array;
  if($_REQUEST["flavor" ] == "" ) {
    $errors_array[] = "<font color='red'>Indique su sabor favorito</font>" ;
  }
}
```

La función show_errors simplemente muestra los errores en el navegador usando la instrucción echo. Todo lo que debe hacer esta función es recorrer en ciclos la matriz $errors_array y mostrar cada error:

```
function show_errors()
{
  global $errors_array;
  foreach ($errors_array as $err){
    echo $err, "<br>";
  }
}
```

Si todo está bien —no hubo errores en la validación de los datos—, la aplicación llamará a la función handle_data y, en este ejemplo, esa función muestra el sabor de helado favorito del usuario:

```
function handle_data()
{
  echo "Su sabor de helado favorito es ";
  echo $_REQUEST["flavor"];
}
```

Puede ver esta aplicación en acción en la Figura 6-13, donde espera que el usuario ingrese su sabor de helado favorito.

FIGURA 6-13 No se ingresaron los datos requeridos

FIGURA 6-14 Error de datos requeridos

Si el usuario no ingresa más que clics en el botón Enviar, éste verá un error como el que aparece en la Figura 6-14, pidiendo ingresar su sabor de helado favorito.

Si, entonces, el usuario ingresa un sabor y hace clic en Enviar, la aplicación muestra ese sabor, como se ve en la Figura 6-5. Muy bien.

Bueno, eso se encarga del caso donde usted desea comprobar que el usuario ha ingresado ciertos datos. Pero hay otras cosas que puede comprobar —por ejemplo, si el usuario ingresó una cadena de texto.

FIGURA 6-15 Cómo corregir un error de datos requeridos

Requerir números

¿Qué tal si requiere que el usuario ingrese datos enteros? Este ejemplo, phprequirednumber.php, hace eso. Comienza con el código fijo para comprobar la validación:

```
<html>
  <head>
    <title>
      Se requiere entrada de números enteros
    </title>
  </head>
  <body>
    <h1>Se requiere entrada de números enteros</h1>
    <?php
      $errors_array = array();
        if(isset($_REQUEST["welcome_already_seen" ])){
          check_data();
          if(count($errors_array) != 0){
            show_errors();
            show_welcome();
          }
          else {
            handle_data();
          }
        }
        else {
          show_welcome();
      }
```

Bueno, ahora es el momento de escribir funciones a las que llama este código, comenzando con la función show_welcome, pregunta al usuario su edad, pasando esa información bajo la clave "number" de vuelta al servidor:

```
function show_welcome()
{
  echo " <form method='post'>" ;
  echo " Ingrese su edad como un número entero." ;
  echo " <br>" ;
  echo " <input name='number' type='text'>" ;
  echo " <br>" ;
  echo " <br>" ;
  echo " <input type='submit' value='Enviar'>" ;
  echo " <input type=hidden name='welcome_already_seen'
    value='already_seen'>" ;
  echo " </form>" ;
}
```

Luego sigue la función check_data, donde se supone que usted comprueba si la edad indicada por el usuario está en la forma correcta. Así que ¿cómo comprueba si la cadena $_REQUEST[“number”] contiene un entero en formato de cadena? Una forma de comprobarlo consiste en convertir la cadena a entero, mediante la función intval de PHP y convertirla de nuevo en cadena usando la función strval —comparándola después con la cadena original—. Si ambas cadenas son iguales, la cadena representa un entero. Así se ve en código:

```
function check_data()
{
  if(strcmp($_REQUEST["number"], strval(intval($_REQUEST["number"])))) {
  .
  .
  .
  }
}
```

Si las cadenas son iguales, strcmp devuelve 0, a interpretarse como FALSE. Por otra parte, si las cadenas no son iguales, strcmp devolverá un resultado distinto de cero, interpretado como TRUE. Si el resultado es TRUE, debe agregar un nuevo error a la matriz global $errors_array, indicando que el usuario debe ingresar un entero:

```
function check_data()
{
  global $errors_array;

  if(strcmp($_REQUEST["number"], strval(intval($_REQUEST["number"])))) {
    $errors_array[] = "<font color='red'>Ingrese un número entero</font>";
  }
}
```

Y en la función show_errors puede mostrar los errores, de haber alguno:

```
function show_errors()
{
  global $errors_array;
  foreach ($errors_array as $err){
    echo $err, "<br>";
  }
}
```

La función handle_data indica la edad del usuario, en caso de que haya ingresado en realidad un entero:

```
function handle_data()
{
  echo "Su edad es ";
  echo $_REQUEST["number"];
}
```

FIGURA 6-16 Edad del usuario con letras

Bueno, eso es todo. Puede ver la aplicación, phprequirednumber.php, en la Figura 6-16. El usuario ha decidido inexplicablemente indicar su edad con letras en vez de ingresar un número.

Y los resultados están en la Figura 6-17 (de hecho se reportó un error).

Corregir el error ingresando un número da el resultado en la Figura 6-18 —todo está bien.

FIGURA 6-17 Corrección de un error de datos requeridos

FIGURA 6-18 Corrección de los datos requeridos

Requerir texto

También puede escribir aplicaciones requiriendo del usuario ingresar texto o incluso texto específico. Por ejemplo, quizá desee preguntar al usuario cuál es su sabor de helado favorito —y luego asegurarse de que ingrese el sabor preferido por usted, pistache—. Puede iniciar esta aplicación, phprequiredtext.php, así:

```
<html>
  <head>
    <title>
      Requerir entrada de texto
    </title>
  </head>

  <body>
    <h1>Requerir entrada de texto</h1>
    <?php
      $errors_array = array();
        if(isset($_REQUEST["welcome_already_seen"])){
          check_data();
          if(count($errors_array) != 0){
            show_errors();
            show_welcome();
          }
          else {
            handle_data();
          }
        }
        else {
          show_welcome();
      }
```

También puede preguntar al usuario su sabor favorito de esta forma en la función show_welcome:

```
function show_welcome()
{
  echo "<form method='post'>";
  echo "¿Cuál es su sabor de helado favorito?";
  echo "<br>";
  .
  .
  .
  echo "<input type='submit' value='Enviar'>";
  echo "<input type=hidden name='welcome_already_seen'
    value='already_seen'>";
  echo "</form>";
}
```

Y asegurarse de que el sabor sea enviado al servidor bajo la clave 'flavor' de esta forma en show_welcome:

```
function show_welcome()
{
  echo "<form method='post'>";
  echo "¿Cuál es su sabor de helado favorito?";
  echo "<br>";
  echo "<input name='flavor' type='text'>";
  echo "<br>";
  echo "<br>";
  echo "<input type='submit' value='Enviar'>";
  echo "<input type=hidden name='welcome_already_seen'
    value='already_seen'>";
  echo "</form>";
}
```

En el servidor, es posible comprobar el sabor ingresado por el usuario en la función check_data. Dicha función manipula texto usando *expresiones regulares*. Las expresiones regulares (la especificación completa se encuentra en www.perldoc.com/perl5.6/pod/perle.html) permiten empatar texto en su código PHP. Por ejemplo, si desea insistir en que la entrada del usuario incluya la palabra "pistache", puede usar la función preg_match de PHP de la siguiente forma (la "I" aquí obliga una búsqueda distinguiendo mayúsculas de minúsculas):

```
function check_data()
{
  global $errors_array;
  if(!preg_match('/pistache/i', $_REQUEST["flavor"])){
  .
  .
  .
  }
}
```

Aunque esta comprobación de expresiones regulares fue para asegurarse de que una palabra en particular apareciera en la entrada de texto del usuario, también es viable usar expresiones regulares para asegurarse de que su entrada de datos tiene un formato particular —por ejemplo, podría desear que la persona ingrese un número de seguro social, con el formato xxx-xx-xxxx, donde cada "*x*" es un dígito—. Tal cosa es posible empatando su texto con la expresión regular "\d\d\d-\d\d-\d\d\d\d", donde "\d" representa la forma de expresar un dígito a la usanza de las expresiones regulares. O bien, tal vez querría que la persona ingrese un número del seguro social, con el formato xxx-xx-xxxx, donde cada *x* es un dígito.

Si la respuesta del usuario no incluye la palabra "pistache", puede objetar con un mensaje de error como éste:

```
function check_data()
{
  global $errors_array;
  if(!preg_match('/pistache/i', $_REQUEST["flavor"])){
    $errors_array[] = "<font color='red'>Su sabor favorito debe ser \"pistache\".</
font>";
   }
  }
```

En la función show_errors, es posible mostrar cualquier error colocado en la matriz global $errors_array:

```
function show_errors()
{
  global $errors_array;

  foreach ($errors_array as $err){
    echo $err, "<br>";
  }
}
```

Y la función handle_data, asimismo, puede mostrar el sabor favorito de la persona (más vale que sea pistache):

```
function handle_data()
{
  echo "Su sabor favorito es ";
  echo $_REQUEST["flavor"];
}
```

Esta aplicación, phprequiredtext.php, se encuentra en la Figura 6-19. Ahí, el usuario intenta indicar que su sabor de helado favorito es fresa.

Los resultados están en la Figura 6-20 —se reportó un error, ya que su sabor favorito no es pistache.

FIGURA 6-19 Ingreso de datos incorrectos

Ingresar pistache como el sabor favorito del usuario da los resultados en la Figura 6-21 (ahora todo está bien).

FIGURA 6-20 Corrección de un error de datos requeridos

FIGURA 6-21 Los datos corregidos

Datos de usuario persistentes

A veces podría tener múltiples controles para que el usuario ingrese datos y, si éste ingresa datos incorrectos en un control, es posible que aún desee mostrar los datos correctos en los otros controles, para no comenzar desde el principio. De eso trata el ejemplo siguiente, phppersist.php.

Comienza con nuestra estructura de validación de datos:

```
<html>
  <head>
    <title>
      Datos de usuario persistentes
    </title>
  </head>
  <body>
    <h1>Datos de usuario persistentes</h1>
    <?php
      $errors_array = array();
      if(isset($_REQUEST["welcome_already_seen"])){
        check_data();
        if(count($errors_array) != 0){
          show_errors();
          show_welcome();
        } else {
          handle_data();
        }
      }
      else {
        show_welcome();
      }
```

La función show_welcome debe comprobar si el usuario ya ingresó datos en algunos controles y, si lo ha hecho, puede presentarlos. Así es como funciona —usted inserta información en las variables $nombre y $apellido, que el usuario ya ingresó, o una cadena vacía,"":

```
function show_welcome()
{
  $nombre = isset($_REQUEST["first"]) ? $_REQUEST["first"] : "";
  $apellido = isset($_REQUEST["last"]) ? $_REQUEST["last"] : "";
  .
  .
  .
}
```

Luego, al mostrar los campos de texto en la página de bienvenida, puede usar $nombre y $apellido como valores de esos controles:

```
function show_welcome()
{
  $nombre = isset($_REQUEST["first"]) ? $_REQUEST["first"] : "";
  $apellido = isset($_REQUEST["last"]) ? $_REQUEST["last"] : "";
  echo "<form method='post'>";
  echo "Ingrese su nombre: ";
  echo "<input name='nombre' type='text' value='", $nombre, "'>";
  echo "<br>";
  echo "<br>";
  echo "Ingrese su apellido: ";
  echo "<input name='apellido' type='text' value='", $apellido, "'>";
  echo "<br>";
  echo "<br>";
  echo "<input type='submit' value='Enviar'>";
  echo "<input type=hidden name='welcome_already_seen'
    value='already_seen'>";
  echo "</form>";
}
```

Eso hará que reaparezca lo ya ha ingresado por el usuario, si a éste se le debe notificar un error. La función check_data comprueba si el usuario ha dejado algún campo de texto en blanco y, si lo ha dejado, genera un error:

```
function check_data()
{
  global $errors_array;

  if($_REQUEST["first"] == "") {
  $errors_array[] = "<font color='red'>Ingrese su nombre</font>";
  }
  if($_REQUEST["last"] == "") {
  $errors_array[] = "<font color='red'>Ingrese su apellido</font>";
  }
}
```

La función show_errors muestra cualquier error presente:

```
function show_errors()
{
  global $errors_array;
  foreach ($errors_array as $err){
    echo $err, "<br>";
  }
}
```

Y la función handle_data presenta los datos ingresados por el usuario:

```
function handle_data()
{
  echo "Éste es su nombre: ";
  echo $_REQUEST["first"];
  echo "<br>Éste es su apellido: ";
  echo $_REQUEST["last"];
}
```

Puede ver esta aplicación en la Figura 6-22, donde el usuario sólo ha ingresado su nombre.

Cuando el usuario hace clic en el botón Enviar, la aplicación indica su error —también vuelve a mostrar los datos que ingresó correctamente,"Edward", ahorrando al usuario al esfuerzo de escribirlo de nueva cuenta, como en la Figura 6-23.

FIGURA 6-22 El usuario ingresa sólo el nombre

FIGURA 6-23 Detección del error

Validación de datos en el cliente

Realizar la validación de sus datos en PHP tiene una desventaja: necesita hacer un viaje de ida y vuelta al servidor para comprobar qué datos ingresó el usuario. Existen algunas ventajas de realizar la validación de datos en el navegador, sin enviar nada en absoluto al servidor.

Éste es un ejemplo breve que realiza la validación en el cliente —o sea, en el navegador— usando JavaScript. Este ejemplo, phpclientside.html, pregunta al usuario su fecha de cumpleaños a través de un campo de texto en una forma HTML:

```
<body>
  <h1>
    Validación de datos en el cliente
  </h1>
  <form name="form1" action="script.php" method="post"
    Indique su fecha de cumpleaños (mm/dd/aaaa):
      <input type="text" name="fecha">
      <br>
      <input type="submit" value="Enviar">
  </form>
</body>
```

Observe que este ejemplo publica sus datos en un script (inexistente) llamado script.php. Para comprobar los datos del usuario antes de enviar la forma, puede conectar el atributo onsubmit de la forma con una función de JavaScript, check_data. Si esa función devuelve un valor falso, la forma no se enviará:

```
<body>
  <h1>
    Validación de datos en el cliente
  </h1>
```

```
<form name="form1" action="script.php" method="post"
      onsubmit="javascript:return check_data()">
      Indique su fecha de cumpleaños (mm/dd/aaaa):
        <input type="text" name="fecha">
        <br>
        <input type="submit" value="Enviar">
    </form>
  </body>
```

Puede crear la función check_data de JavaScript, en la sección <head> de la página, en un elemento <script>:

```
<head>
    <title>
      Validación de datos en el cliente
    </title>

    <script language="javascript">

      function check_data()
      {
        .
        .
        .
      }
    </script>
  </head>
```

JavaScript admite expresiones regulares, de modo que puede crear una expresión regular para comprobar fechas utilizando el formato dd/mm/aaaa de esta forma:

```
<head>
    <title>
      Validación de datos en el cliente
    </title>
    <script language="javascript">
      function check_data()
      {
        var regexp = /^(\d{1,2})\/(\d{1,2})\/(\d{4})$/
        .
        .
        .
      }
    </script>
  </head>
```

Luego, comprobar la fecha que ingresó el usuario contra esa expresión regular:

```
<head>
    <title>
      Validación de datos en el cliente
    </title>
```

```
    <script language="javascript">
      function check_data()
      {
        var regexp = /^(\d{1,2})\/(\d{1,2})\/(\d{4})$/
        var result = document.form1.date.value.match(regexp)
        if (result == null) {
          alert("Ingrese una fecha en el formato dd/mm/aaaa.");
          document.form1.text1.value = "";
          return false;
        .
        .
        .
      }
    </script>
  </head>
```

Si el resultado del intento por empatar la expresión regular con lo ingresado por el usuario fue nulo, el usuario no ingresó la fecha en el formato señalado. Puede mostrar un error usando un cuadro de alerta de JavaScript y devolver falso de la función, lo cual significa que la forma no se enviará, que da al usuario la oportunidad de corregir el problema:

```
<head>
    <title>
      Validación de datos en el cliente
    <title>

    <script language="javascript">

      function check_data()
      {
        var regexp = /^(\d{1,2})\/(\d{1,2})\/(\d{4})$/
        var result = document.form1.date.value.match(regexp)
        if (result == null) {
          alert("Ingrese una fecha en el formato dd/mm/aaaa.");
          document.form1.date.value = "";
          return false;
               .
        .
        .
      }
    </script>
  </head>
```

Si los datos son correctos, puede enviar la forma desde JavaScript con la función submit de JavaScript:

```
<head>
    <title>
      Validación de datos en el cliente
    </title>
    <script language=”javascript”>
      function check_data()
      {
        var regexp = /^(\d{1,2})\/(\d{1,2})\/(\d{4})$/
        var result = document.form1.date.value.match(regexp)
```

FIGURA 6-24 Ingreso de una fecha en un formato incorrecto

```
        if (result == null) {
          alert("Ingrese una fecha en el formato dd/mm/aaaa.");
          document.form1.text1.value = "";
          return false;
        } else {
          document.form1.submit();
          return true;
        }
      }
    </script>
  </head>
```

Y esta aplicación, phpclientside.html, se ve en la Figura 6-24. Ahí, el usuario ingresó su fecha de cumpleaños en el formato erróneo.

Los resultados se ven en la Figura 6-25 —el error se presenta en un cuadro de alerta de JavaScript—. Genial.

FIGURA 6-25 Detección de un error en una fecha en JavaScript

Manejo de etiquetas HTML en la entrada del usuario

A veces, es asequible que los usuarios integren elementos HTML en el texto enviado por ellos y conviene estar pendiente de eso —si muestra ese texto en una página Web, éste podría incluir scripts malintencionados que obligarían a su página Web cerrarse o dirigirse a otra página.

PHP incluye funciones para desactivan código HTML no deseado. Este ejemplo, phphtml.php, permite al usuario ingresar código HTML —pero desactiva ese código antes de mostrarse en un navegador—. La forma en que esto funciona es, simplemente, aplicando la función htmlentities de PHP a datos provistos por el usuario, de esta forma:

```
<html>
  <head>
    <title>Manejo de código HTML en la entrada del usuario</title>
  </head>
  <body>
    <H1>Manejo de código HTML en la entrada del usuario</H1>
    <?php
      $errors_array = array();

      if(isset($_REQUEST["welcome_already_seen"])){
                check_data();
                if(count($errors_array) != 0){
                    show_errors();
                    show_welcome();
                }
                else {
                    handle_data();
                }
            }
      else {
                show_welcome();
            }
      function check_data()
      {
        global $errors_array;
        if($_REQUEST["flavor"] == "") {
          $errors_array[] = "<font color='red'>Ingrese su sabor favorito</font>";
        }
      }

      function show_errors()
      {
        global $errors_array;

        foreach ($errors_array as $err){
          echo $err, "<br>";
        }
      }
```

```
      function handle_data()
      {
        echo "Su sabor favorito es ";
        $ok_text = strip_tags($_REQUEST["flavor"]);
        echo $ok_text;
      }
      function show_welcome()
      {
        echo "<form method='post'>";
        echo "¿Cuál es su sabor de helado favorito?<br>";
        echo "<input name='flavor' type='text'>";
        echo "<br><br>";
        echo "<input type='submit' value='Enviar'>";
        echo "<input type='hidden' name='welcome_already_seen'
          VALUE='already_seen'>";
        echo "</form>";
      }
    ?>
  </body>
</html>
```

Ahora el usuario puede ingresar código HTML en sus datos, como luce en la Figura 6-26.

El uso de la función htmlentities escapa a cualquier código HTML incluido; de modo que el texto"fresa"se convierte a"fresa"y eso así como se muestra, tal cual se ve en la Figura 6-27.

FIGURA 6-26 Inclusión de código HTML en los datos del usuario

FIGURA 6-27 Escape del código HTML en los datos del usuario

También puede utilizar la función strip_tags así:

```
function handle_data()
{
  echo "Su nombre es ";
  $ok_text = strip_tags($_REQUEST["flavor"]);
  echo $ok_text;
}
```

El uso de la función strip_tags elimina cualquier etiqueta HTML del texto —y el resultado aparece en la Figura 6-28.

FIGURA 6-28 Eliminación de código HTML en los datos del usuario

CAPÍTULO 7

Programación orientada a objetos

PHP no está diseñado para ser un lenguaje orientado a objetos de principio a fin, pero tiene un número increíble de características integradas orientadas a objetos, incluidas muchas agregadas en PHP 5. Verá esas características en este capítulo y el siguiente.

Por otra parte, no necesita programación orientada a objetos (POO) para crear aplicaciones Web en PHP; de ese modo, no se verá obligado a leer este material si no le compete. La programación orientada a objetos está destinada a aplicaciones de mayor tamaño. Por eso se creó (para permitirle dividir aplicaciones de mayores dimensiones cuando el simple empaquetado de código en funciones no es de ayuda).

La programación comenzó como un asunto muy lineal —sólo se escribía el código, línea por línea—. Luego se introdujeron funciones y estas permitían dividir el código —el código en las funciones no se ejecutaría hasta ser invocado—. Y eso está bien, hasta cierto punto. Pero cuando se tienen aplicaciones todavía más grandes se necesita otra solución.

Esa solución consiste en usar objetos. La programación orientada a objetos podría parecer terrorífica si nunca la ha usado, aunque no debería —la idea es facilitarle la vida—. Por ejemplo, suponga que en la vida real tiene todas las partes interconectadas de un refrigerador en su cocina. Para refrigerar alimentos debe encender bombas, ventiladores, hacer circular el fluido del refrigerador, operar el compresor, etcétera. Eso suena a mucho trabajo —es mejor integrarlo todo en un *objeto* fácilmente concebido (un refrigerador)—. Ahora todos los instrumentos internos están ocultos a la vista y puede pensar: ahí está el refrigerador. Refrigera alimentos. Fácil.

Ésa es la idea tras la programación orientada a objetos. Divide su código en objetos —por ejemplo, un administrador de bases de datos o un administrador de pantallas—. Ésa es la forma en que los programadores pueden concebir sus aplicaciones: en términos de objetos con funciones y responsabilidades específicas. Todas las funciones necesarias para el funcionamiento interno del objeto, se encuentran integradas en él y ocultas de la vista de los demás. Los datos que utiliza el objeto también están escondidos dentro del objeto y ésa es una de las formas principales en que los objetos difieren del simple uso de funciones (no sólo puede integrar muchas funciones en un objeto, sino también los datos usados por esas funciones).

Así que en lugar de 200 funciones y 500 variables necesarias para poder imprimir en su aplicación, puede integrarlo todo en un objeto llamado, por ejemplo, printer. Todas las funciones y variables están ocultas dentro de ese objeto y no son accesibles fuera de éste. Las funciones en programación orientada a objetos reciben un nuevo nombre —métodos—. Mientras los elementos de datos almacenados en un objeto se denominan propiedades.

El objeto printer podría ocultar casi todos sus métodos internamente, pero también *dejará expuestos*, intencionalmente, algunos métodos para uso público y éstos son los que usted utiliza para trabajar con el objeto printer. Por ejemplo, éste puede tener un método llamado print y pasar texto a ese método para imprimir: printer("Ésta es una prueba").

Bueno, comencemos a analizar los detalles. Para crear un objeto, primero necesita crear una *clase*. Las clases son para los objetos lo que las cortadoras de galletas son para las galletas (se crean objetos utilizando clases). De hecho, un objeto se conoce como *instancia* de una clase. Veremos todo eso a continuación.

Creación de clases

Toda programación orientada a objetos comienza con clases. Las clases son el *tipo* de objetos, de la misma forma que "entero" puede ser el tipo de una variable. Tiene que crear el tipo antes de crear objetos específicos de ese tipo. Es importante comprender esto —una clase es un tipo de objeto—. Y dado que puede estructurar objetos como desee, debe especificar su estructura al crear clases. Después de crear una clase es posible crear objetos de esa clase, como verá ahora.

La clase es el tipo del objeto y éste una instancia de la clase. Normalmente usted crea objetos mediante clases y luego utiliza esos objetos en su código.

Éste es un ejemplo, una clase llamada Person. En un par de páginas crearemos objetos persona, a partir de la clase Person; la clase es la especificación de lo que contendrán esos objetos. Así es como se crea una clase en PHP, con la palabra clave class:

```
class Person
{
    .
    .
    .
}
```

Bien, eso crea una clase llamada Person. En PHP, las clases suelen recibir nombres que comienzan con mayúscula y a los objetos se les dan nombres comenzando con minúscula. Como bien sabe, las clases pueden contener elementos de datos, llamados propiedades; de tal modo, demos un nombre a la persona, $name. En la clase, usted declara propiedades y métodos que se integrarán a los objetos de esta clase —ello significa que usa la instrucción var para declarar propiedades en PHP—. Así es como se agrega una propiedad llamada $name a la clase Person:

```
class Person
{
    var $name;
        .
        .
        .
}
```

También puede agregar métodos (funciones en lenguaje, no de programación orientada a objetos) a la clase Person. Por ejemplo, necesitará alguna forma para establecer el nombre de la persona, así que podría usar un método llamado set_name para conseguirlo. Así es como se agrega ese método a la clase Person —observe que toma el nombre de la persona como argumento, llamado $data:

```
class Person
{
    var $name;
    function set_name($data)
    {
        .
        .
        .
    }
}
```

Ahora debe almacenar el nombre pasado a la función set_name en la variable $name. Podría hacerlo accediendo a $name como variable global:

```
class Person
{
    var $name;
    function set_name($data)
    {
        global $name;
        .
        .
        .
    }
}
```

Luego puede asignar el argumento pasado a set_name, $data, al nombre almacenado internamente, $name, de esta forma:

```
class Person
{
    var $name;
    function set_name($data)
    {
        global $name;
        $name = $data;
    }
}
```

Eso almacena el nombre de la persona. También va a necesitar alguna forma de lectura del nombre de la persona; así, podría agregar otro método llamado get_name:

```
class Person
{
    var $name;
    function set_name($data)
    {
        global $name;
        $name = $data;
    }
```

```
    function get_name()
    {
        .
        .
        .
    }
}
```

En el método get_name, puede acceder al nombre interno, $name y devolverlo de esta forma:

```
class Person
{
    var $name;

    function set_name($data)
    {
        global $name;
        $name = $data;
    }

    function get_name()
    {
        global $name;
        return $name;
    }
}
```

Luce bien y funcionará, pero hay que hacer un cambio. Normalmente no verá programación orientada a objetos usando la palabra clave global. En su lugar, generalmente nos referimos a propiedades de clase usando la palabra clave $this. Esta apunta al objeto actual. En términos de PHP, esto:

```
global $name;
$name = $data;
```

se puede reemplazar con esto:

```
$this->name = $data;
```

Observe la sintaxis aquí —se utiliza $this, seguido del operador ->—. Además, se omite el signo de pesos $ frente a la propiedad a que hace referencia, de esta forma: $this->name.

Utilizando esta sintaxis, la normal a encontrar en la programación orientada a objetos en PHP, puede escribir los métodos en la clase Person de esta manera:

```
class Person
{
    var $name;

    function set_name($data)
    {
        $this->name = $data;
    }
```

```
    function get_name()
    {
        return $this->name;
    }}
```

Esto es algo que debe saber cuando se habla de crear propiedades: no puede asignar a una propiedad un valor calculado en una clase. Por ejemplo, esto es correcto; asigna el nombre "Ralph" a $name:

```
class Person
{
    var $name = "Ralph";

    function set_name($data)
    {
        $this->name = $data;
    }

    function get_name()
    {
        return $this->name;
    }
}
```

Significa que cuando se creen objetos de la clase Person a $name, se asignará el nombre "Ralph". Sin embargo, no puede asignar más de un valor constante a una propiedad en una clase. Por ejemplo, ni siquiera puede asignar la expresión"Ralph" . "Kramden", que eslabona "Ralph"y"Kramden", a la propiedad $name. Esto no funcionará:

```
class Person
{
    var $name = "Ralph" . "Kramden";   //NO ES CORRECTO.

    function set_name($data)
    {
        $this->name = $data;
    }

    function get_name()
    {
        return $this->name;
    }
}
```

Es algo con lo que debemos tener cuidado al crear propiedades. ¿Existen restricciones similares en los métodos? No realmente, salvo que no debe iniciar el nombre de un método con una línea de subrayado doble (__); existen varios métodos integrados que comienzan de esa forma y podría tener conflictos con ellos.

Bueno, ahora ha creado una clase de PHP, Person. Ahora es momento de ponerla a trabajar, creando algunos objetos.

Creación de objetos

Para crear un objeto nuevo se usa el operador new de PHP. Tan sólo necesita definir su clase y luego puede usar el operador new para crear un objeto de esa clase. Los objetos se almacenan como variables, de modo que el proceso de creación de objetos mediante el operador new tiene este aspecto:

```
class Person
{
    var $name
    function set_name($data)
    {
        $this->name = $data;
    }

    function get_name()
    {
        return $this->name;
    }
}
$ralph = new Person;
```

Genial, ha creado un nuevo objeto Person llamado $ralph. Fácil.

Todos los métodos y propiedades de la clase Person están integrados en el objeto $ralph. Así que, ¿cómo llamar al método set_name, por ejemplo, para establecer el nombre almacenado en el objeto?

Puede llamar al método set_name, del objeto $ralph, de la siguiente forma, utilizando de nuevo el operador ->:

```
$ralph = new Person;
$ralph->set_name("Ralph");
```

Así es como funciona en PHP; usted crea un objeto y luego puede llamar a los métodos integrados en ese objeto usando el operador ->. Ejecutar la instrucción $ralph->set_name("Ralph"); establece el nombre almacenado internamente en $ralph como "Ralph".

Y se vuelve posible leer ese nombre del objeto, a través del método get_name del objeto. Esto se ve así en phpobject.php, donde puede mostrar el nombre interno del objeto:

```
<html>
  <head>
    <title>
      Creación de un objeto
    </title>
  </head>

  <body>
    <h1>Creación de un objeto</h1>
    <?php
      class Person
      {
        var $name;
```

```
        function set_name($data)
        {
            $this->name = $data;
        }
        function get_name()
        {
            return $this->name;
        }
    }
    $ralph = new Person;
    $ralph->set_name("Ralph");
    echo "El nombre de su amigo es ", $ralph->get_name(), ".";
  ?>
 </body>
</html>
```

Puede ver los resultados en la Figura 7-1 —este ejemplo crea una clase y luego un objeto de esa clase, configura ese objeto llamando a su método set_name—, luego llama a get_name para obtener el nombre almacenado. No está mal.

Pudo ver que, después de crear un objeto, puede tener acceso a los métodos de ese objeto de la siguiente forma:

```
$ralph = new Person;
$ralph->set_name("Ralph);
```

También, de manera predeterminada, puede tener acceso a las propiedades contenidas en un objeto de la misma forma. Por ejemplo, podría leer el nombre almacenado internamente

FIGURA 7-1 Creación de un objeto

(como $name) en el objeto $ralph, sin llamar al método get_name. De hecho, puede acceder a esa propiedad de esta forma:

```
<?php
  class Person
  {
    var $name;
    function set_name($data)
    {
        $this->name = $data;
    }
    function get_name()
    {
        return $this->name;
    }
  }
  $ralph = new Person;
  $ralph->set_name("Ralph");
  echo "El nombre de su amigo es ", $ralph->name(), ".";
?>
```

De modo que puede recuperar el nombre almacenado en el objeto usando el método get_name:

```
$ralph->get_name()
```

o de esta forma, recurriendo sólo a la propiedad:

```
$ralph->name
```

En la programación orientada a objetos, en general se llama a un método para obtener datos de un objeto, no se tiene acceso directo a las propiedades del objeto. Los métodos que devuelven valores de propiedades se denominan métodos *accessor* (o de acceso) en la programación orientada a objetos y son populares porque dan control sobre la forma en que se establecen datos internos contenidos en un objeto. Por ejemplo, alguien puede pasar una cadena vacía al método set_name y, en tal caso, quizá desee establecer el nombre interno del objeto a un valor predeterminado en su lugar:

```
function set_name($data)
{
    if($data != ""){
      $this->name = $data;
    }
    else {
      $this->name = "Ralph";
    }
{
```

Si permite que código fuera del objeto establezca el valor de la propiedad $name directamente, no tendría este tipo de control sobre los valores aceptables de la propiedad $name.

Bueno, podría restringir el acceso a los datos internos de un objeto y querer código para llamar a get_name en vez de acceder a la propiedad $name directamente. Pero, ¿cómo evitaría que el código acceda esa propiedad directamente? Puede hacerlo.

Cómo establecer el acceso a propiedades y métodos

En este punto, cualquier código accede a la propiedad $name en su objeto. Eso se debe a que, de forma predeterminada, todos los miembros (propiedades y métodos), de una clase u objeto, se declaran públicos. Eso significa que dichos miembros son accesibles desde cualquier parte de su código. Eso quiere decir que este código funciona, el cual tiene acceso a la propiedad $name desde fuera del objeto:

```
<?php
  class Person
  {
    var $name;
    function set_name($data)
    {
        $this->name = $data;
    }
    function get_name()
    {
        return $this->name;
    }
  }
  $ralph = new Person;
  $ralph->set_name("Ralph");
  echo "El nombre de su amigo es ", $ralph->name(), ".";
?>
```

Puede restringir el acceso a miembros de una clase u objeto con los *modificadores de acceso* de PHP, que son éstos:

- **public** Significa "Accesible a todos"
- **private** Significa "Accesible en la misma clase"
- **protected** Significa "Accesible en la misma clase y clases derivadas de esa clase"

Acceso público

El acceso público tiene menos restricciones de entre todos y es el predeterminado. Puede declarar propiedades y métodos públicos de forma explícita con la palabra clave public:

```
<?php
  class Person
  {
    public $name;
```

```
        public function set_name($data)
        {
            $this->name = $data;
        }
        public function get_name()
        {
            return $this->name;
        }
      }
      $ralph = new Person;
      $ralph->set_name("Ralph");
      echo "El nombre de su amigo es ", $ralph->name(), ".";
    ?>
```

Este código funciona, reproduciendo con echo el nombre almacenado internamente en el objeto, ya que ese nombre se ha almacenado con acceso público. Ahora restrinjamos el acceso a ese nombre.

Acceso privado

Puede hacer privado el miembro de una clase u objeto con la palabra clave private. Cuando hace un miembro privado no puede acceder a éste fuera de la clase u objeto. En este ejemplo, phpprivate.php, la propiedad $name se ha hecho privada para la clase Person e intentamos acceder a ella desde fuera del objeto $ralph:

```
<html>
  <head>
    <title>
      Creación de un objeto
    </title>
  </head>
  <body>
    <h1>Creación de un objeto</h1>
    <?php
      class Person
      {
        private $name;
        function set_name($data)
        {
            $this->name = $data;
        }
        function get_name()
        {
            return $this->name;
        }
      }
```

```
    $ralph = new Person;
    $ralph->set_name("Ralph");
    echo "El nombre de su amigo es ", $ralph->name, ".";
  ?>
 </body>
</html>
```

Puede ver este ejemplo en acción en la Figura 7-2 —como notará, obtiene un error porque la propiedad a que intenta acceder es privada:

```
El nombre de su amigo es PHP Fatal error: Cannot access private property
Person:$name in C:\Inetpub\wwwroot\ch07\phpprivate.php on line 29
```

Eso significa que ha forzado código fuera del objeto para tener acceso al nombre almacenado en el objeto $ralph, utilizando el método de acceso get_name, en vez de acceder directamente a la propiedad $name:

```
<?php
  class Person
  {
    private $name;

    function set_name($data)
    {
        $this->name = $data;
    }

    function get_name()
    {
        return $this->name;
    }
  }
```

FIGURA 7-2 Intento de acceso a una propiedad privada

```
    $ralph = new Person;
    $ralph->set_name("Ralph");
    echo "El nombre de su amigo es ", $ralph->get_name(), ".";
  ?>
```

Ahora todo funciona correctamente, como puede ver en la Figura 7-3. También puede hacer métodos privados, como se encuentra aquí:

```
<html>
  <head>
    <title>
      Creación de un objeto
    </title>
  </head>

  <body>
    <h1>Creación de un objeto</h1>
    <?php
      class Person
      {
        var $name;

        function set_name($data)
        {
            $this->name = $data;
        }

        private function get_name()
        {
            return $this->name;
        }
      }
```

FIGURA 7-3 Acceso a una propiedad privada

```
    $ralph = new Person;
    $ralph->set_name("Ralph");

    echo "El nombre de su amigo es ", $ralph->get_name(), ".";
  ?>
 </body>
</html>
```

Pero ahora este código no va a funcionar, ya que intentamos llamar a get_name fuera del objeto —y el método get_name es privado—. Eso significa que puede llamarlo sólo desde el código, dentro del mismo objeto. Por ejemplo, código como éste es legal (aunque no muy útil), pues llama al ahora método privado get_name, desde el interior del código mismo del objeto:

```
<?php
  class Person
  {
    var $name;

    function set_name($data)
    {
        $this->name = get_name();
    }

    private function get_name()
    {
        return $this->name;
    }
  }

  $ralph = new Person;
  $ralph->set_name("Ralph");

  echo "El nombre de su amigo es ", $ralph->get_name(), ".";
?>
```

Daremos un vistazo al otro modificador de acceso, el modificador protegido, tras hablar acerca de la herencia, que veremos pronto en este capítulo.

Uso de constructores para inicializar objetos

Como ha visto, puede crear objetos con el operador new y luego emplear métodos como set_name, para establecer datos internos en ese objeto:

```
function set_name($data)
{
  $this->name = get_name();
}
```

De esta forma puede inicializar los datos en un objeto antes de comenzar a trabajar con ese objeto.

¿No sería conveniente si pudiera crear e inicializar un objeto al mismo tiempo? PHP le permite hacerlo con *constructores* que, como en otros lenguajes con soporte para programación orientada a objetos, son métodos especiales de ejecución automática cuando se crea un objeto.

Los constructores tienen un nombre especial en PHP (__construct; es decir,"construct"precedido por dos líneas de subrayado). Éste es un ejemplo:

```
function __construct($data)
{
        .
        .
        .
}
```

Como advertirá, este constructor toma un argumento, $data. Puede asignar esos datos al nombre interno almacenado en el objeto de esta forma:

```
function __construct($data)
{
    $this-<name = $data;
}
```

¿Cómo se utiliza este constructor? Pasa datos al constructor cuando utiliza el operador new, para crear nuevos objetos y pasa datos al constructor, encerrando dichos datos entre paréntesis luego del nombre de la clase. En este ejemplo, phpconstructor.php, se inicializa un objeto con el nombre"Dan":

```
<html>
  <head>
    <title>
      Uso de un constructor
    </title>
  </head>

  <body>
    <h1>Uso de un constructor</h1>
    <?php
      class Person
      {
        var $name;

        function __construct($data)
        {
          $this->name = $data;
        }

        function set_name($data)
        {
            $this->name = $data;
        }
```

```
        function get_name()
        {
            return $this->name;
        }
    }

    $dan = new Person("Dan");

    echo "El nombre de su amigo es ", $dan->get_name(), ".";
  ?>
 </body>
</html>
```

Puede ver los resultados en la Figura 7-4, donde el constructor realmente inicializó el objeto.

Puede pasar tantos argumentos a constructores como necesite —en tanto el constructor esté configurado para tomar tales argumentos:

```
$dan = new Person("Dan", "cabello castaño", "ojos azules");
```

Toda clase de PHP viene con un constructor predeterminado que no toma argumentos —es el constructor predeterminado al que se llama cuando se ejecuta código como éste:

```
$ralph = new Person;
$ralph->set_name("Ralph);
```

Sin embargo, tan pronto cree su propio constructor, sin importar cuántos argumentos tome, el constructor predeterminado ya no será accesible.

FIGURA 7-4 Uso de un constructor

También puede dar al constructor el nombre de la clase (como en otros lenguajes, con soporte para programación orientada a objetos) en lugar de __construct, de esta forma para la clase Person:

```
class Person
{
  var $name;
  function Person($data)
  {
    $this->name = $data;
  }
  .
  .
  .
}
```

Uso de destructores para eliminar objetos

Además de constructores, PHP cuenta también con soporte para *destructores*, a los cuales se llama cuando se elimina un objeto. Se utilizan destructores para hacer limpieza después de usar un objeto —cierre conexiones con bases de datos o Internet—. En PHP, se llama a los destructores cuando se destruye explícitamente un objeto o todas las referencias al objeto quedan fuera de alcance.

Los destructores se llaman __destruct en PHP, de esta forma (no se pasan argumentos a destructores):

```
function __destruct()
{
}
```

Éste es un ejemplo, phpdestructor.php:

```
<html>
  <head>
    <title>
      Uso de un destructor
    </title>
  </head>

  <body>
    <h1>Uso de un destructor</h1>
    <?php
      class Person
      {
        var $name;

        function __construct($data)
        {
          echo "Construyendo ", $data, "...<br>";
```

```
        $this->name = $data;
      }

      function set_name($data)
      {
          $this->name = $data;
      }

      function get_name()
      {
          return $this->name;
      }

      function __destruct()
      {
        echo "Destruyendo ", $this->name, "...<br>";
      }
    }

    $dan = new Person("Dan");

    echo "El nombre de su amigo es ", $dan->get_name(), ".<br>";
  ?>
 </body>
</html>
```

Puede ver los resultados en la Figura 7-5, donde el destructor se ejecutó al destruir el objeto $dan, tras terminar el script.

FIGURA 7-5 Uso de un destructor

Basar una clase en otra con herencia

La clase Person está bien hasta donde llega, pero, ¿qué sucedería si deseara personalizarla para sus amigos, de modo que cada uno de ellos pudiera tener su propio mensaje? Por ejemplo, Dan podría decir,"Hola, soy Dan."; Ralph podría decir,"Ralph aquí.", y así sucesivamente.

Puede agregar funcionalidad a sus clases, como la clase Friends, a través de la *herencia*. La herencia en PHP actúa de forma muy similar a otros lenguajes, como Java. Así es como funciona; la clase Person da soporte a los métodos set_name y get_name:

```
class Person
{
  var $name;

  function set_name($data)
  {
      $this->name = $data;
  }

  function get_name()
  {
      return $this->name;
  }
}
```

Ahora suponga que desea crear una nueva clase, Friend, con todas las funciones de la clase Person (es decir, los métodos set_name y get_name), pero que dé soporte también a nuevos métodos. Puede basar la clase Friend en la clase Person, con la palabra clave *extends*, de la siguiente forma:

```
class Friend extends Person
{
  .
  .
  .
}
```

Ahora la clase Friend da soporte a los métodos Person, set_name y get_name. Puede agregar también sus propios métodos a la clase Friend. Si deseara que cada objeto Friend tuviera sus propios mensajes personalizados, podría agregar un método set_message, para permitirle establecer el mensaje de cada objeto y un método speak que devuelva ese mensaje:

```
class Friend extends Person
{
  var $message;

  function set_message($msg)
  {
      $this->message = $msg;
  }
```

```
    function speak()
    {
      echo $this->message;
    }
  }
```

Ahora los objetos Friend darán soporte a los métodos set_name, get_name, set_message y speak. Éste es un ejemplo, phpinheritance.php, que muestra esto en acción:

```
<html>
  <head>
    <title>
      Herencia con constructores
    </title>
  </head>

  <body>
    <h1>Herencia con constructores</h1>
    <?php
      class Person
      {

        var $name;

        function set_name($data)
        {
          $this->name = $data;
        }

        function get_name()
        {
            return $this->name;
        }
      }

      class Friend extends Person
      {
        var $message;

        function set_message($msg)
        {
          $this->message = $msg;
        }

        function speak()
        {
          echo $this->message;
        }
      }

      $tony = new Friend;
      $tony->set_name("Tony");
      $tony->set_message("Hola dice Tony.");
```

FIGURA 7-6 Herencia de clases

```
      echo "El nombre de su amigo es ", $tony->get_name(), ".<br>";
      echo $tony->get_name(), " dice \"", $tony->speak(), "\"<br>";
    ?>
  </body>
</html>
```

Puede ver los resultados en la Figura 7-6, donde se han usado todos los métodos en el código: set_name, get_name, set_message y speak.

Así funciona la herencia —se basa una clase en otra y la clase derivada hereda funcionalidad de la clase de base—. La herencia de propiedades y métodos de la clase de base dependerá de modificadores de acceso usados en la clase de base. Los miembros públicos serán visibles en la clase derivada (y todas partes), pero los miembros declarados privados en la clase de base no estarán accesibles en la clase derivada.

¿Qué sucede si desea que un miembro esté accesible en la clase de base —y la clase derivada— pero no para el código fuera de la clase de base y las clases derivadas? Para ello se usa el acceso protegido.

Acceso protegido

El uso de la palabra clave protected hace que los miembros de la clase estén accesibles sólo en la clase en que están declarados y cualquier clase derivada de esa clase. Por ejemplo, puede hacer el método set_name protegido en la clase Person:

```
<?php
  class Person
  {
    var $name;

    protected function set_name($data)
    {
      $this->name = $data;
    }
```

```
        function get_name()
        {
            return $this->name;
        }
    }
```

Eso significa que sólo el código en Person y clases derivadas de ella puede llamar al método set_name. Por ejemplo, este código no funcionará:

```
$tony = new Friend;
$tony->set_name("Tony");
```

Para hacer el método set_name accesible a código fuera del objeto, podría crear un nuevo método, set_name_public, que llame a set_name internamente, de esta forma en la clase Friend en phpprotected.php:

```
<html>
  <head>
    <title>
      Herencia con constructores
    </title>
  </head>

  <body>
    <h1>Herencia con constructores</h1>
    <?php
      class Person
      {
        var $name;

        protected function set_name($data)
        {
          $this->name = $data;
        }

        function get_name()
        {
            return $this->name;
        }
      }

      class Friend extends Person
      {
        var $message;

        function set_message($msg)
        {
          $this->message = $msg;
        }

        function speak()
        {
          echo $this->message;
        }
```

```
        function set_name_public($name)
        {
          $this->set_name($name);
        }
      }

      $tony = new Friend;
      $tony->set_name_public("Tony");
      $tony->set_message("Hola dice Tony.");

      echo "El nombre de su amigo es ", $tony->get_name(), ".<br>";
      echo $tony->get_name(), " dice \"", $tony->speak(), "\"<br>";
    ?>
  </body>
</html>
```

Ahora esto funciona, porque set_name_public tiene acceso público (cualquier código puede llamarlo). Ésa es la idea detrás del modificador de acceso protegido —el acceso público da a todo código acceso a un miembro, el acceso privado restringe el acceso a la clase actual y el acceso protegido restringe el acceso a la clase actual y cualquier otra clase derivada de esa clase.

Constructores y herencia

¿Qué tal si usamos constructores y herencia? De forma específica, ¿cómo se devuelven datos al constructor de la clase de base? Por ejemplo, vio un constructor para la clase Person que tomó el nombre de la persona:

```
class Person
{
  var $name;

  function __construct($data)
  {
    $this->name = $data;
  }
```

¿Cómo podría llamar a este constructor desde la clase Friend, basada en la clase Person? Puede hacerlo de esta forma en el constructor de la clase Friend —parent::__construct llama al constructor de la clase parent (de la clase Friend):

```
function __construct($data);
{
  parent::__construct($data);
}
```

Y puede pasar otros datos a este constructor también —datos dirigidos a la clase Friend—. Se ve cómo funciona esto en phpinheritconstructor.php:

```
<html>
  <head>
    <title>
      Herencia con constructores
    </title>
  </head>
```

```
  <body>
    <h1>Herencia con constructores</h1>
    <?php
      class Person
      {
        var $name;

        function __construct($data)
        {
          $this->name = $data;
        }

        function set_name($data)
        {
            $this->name = $data;
        }

        function get_name()
        {
            return $this->name;
        }
      }

      class Friend extends Person
      {
        var $message;
        function __construct($data, $msg)
        {
          parent::__construct($data);
          $this->message = $msg;
        }

        function speak()
        {
          echo $this->message;
        }
      }

      $nancy = new Friend("Nancy", "Hola, aquí Nancy.");

      echo "El nombre de su amigo es ", $nancy->get_name(), ".<br>";
      echo $nancy->get_name(), " dice \"", $nancy->speak(), "\"<br>";
    ?>
  </body>
</html>
```

Los resultados están en la Figura 7-7, donde el constructor Friend devolvió datos al constructor Person.

Llamada a métodos de clases de base

Cuando ha heredado contenido de otra clase, ¿cómo llama a los métodos de esa clase de base? Normalmente, no es un problema, ya que esos métodos están disponibles para usted, simple-

FIGURA 7-7 Uso de la herencia y constructores

mente porque los ha heredado de la clase de base. ¿Pero qué sucede si la clase de base tiene métodos protegidos —y desea hacerlos públicos permitiendo que cualquier código los llame?

Como pudo ver con los constructores, puede llamar a métodos de clases de base agregando "parent::"frente al nombre del método de la clase de base. Aquí hay un ejemplo, phpbasemethods.php. Los métodos get_name y set_name se han protegido en la clase Person:

```
<?php
  class Person
  {
    var $name;

    protected function set_name($data)
    {
        $this->name = $data;
    }

    protected function get_name()
    {
        return $this->name;
    }
  }
```

Y la clase Friend tiene dos métodos, get_name_public y set_name_public, pues llaman a los métodos protegidos usando la sintaxis parent::, así:

```
<html>
  <head>
    <title>
      Llamada a métodos de clases de base
    </title>
  </head>
```

```
  <body>
    <h1>Llamada a métodos de clases de base</h1>
    <?php
      class Person
      {
        var $name;

        protected function set_name($data)
        {
            $this->name = $data;
        }

        protected function get_name()
        {
            return $this->name;
        }
      }

      class Friend extends Person
      {
        var $message;

        function set_message($msg)
        {
          $this->message = $msg;
        }

        function speak()
        {
          echo $this->message;
        }

        function set_name_public($data)
        {
            parent::set_name($data);
        }
        function get_name_public()
        {
            return parent::get_name();
        }
      }

      $britta = new Friend;
      $britta->set_name_public("Britta");
      $britta->set_message("Hola de Britta.");
      echo "El nombre de su amigo es ", $britta->get_name_public(), ".<br>";
      echo $britta->get_name_public(), " dice \"", $britta->speak(), "\"<br>";
    ?>
  </body>
</html>
```

FIGURA 7-8 Llamada a métodos de clases de base

Eso es todo; el código crea entonces un amigo llamado Britta y utiliza los métodos get_name_public y set_name_public para llamar a las versiones protegidas de estos métodos, que hace el código con la sintaxis parent::. Puede ver los resultados en la Figura 7-8.

NOTA *También puede usar la sintaxis parent:: para acceder las propiedades de la clase de base.*

¿Qué hace usted si ha heredado contenido de una clase que también ha heredado contenido de otra clase? ¿Puede usar la sintaxis parent::? No, pero puede anteponer, simplemente, el miembro al que desee acceder con el nombre de la clase en que se encuentra. Eso se vería así en este ejemplo, donde el miembro que usted busca está en la clase Person:

```
<?php
      class Person
      {
        var $name;

        protected function set_name($data)
        {
            $this->name = $data;
        }

        protected function get_name()
        {
            return $this->name;
        }
      }
```

```
class Friend extends Person
{
  var $message;

  function set_message($msg)
  {
    $this->message = $msg;
  }

  function speak()
  {
    echo $this->message;
  }

  function set_name_public($data)
  {
      Person::set_name($data);
  }

  function get_name_public()
  {
      return Person::get_name();
  }
}
  .
  .
  .
```

Sustitución de métodos

En PHP, como la programación orientada a objetos en muchos lenguajes, puede *sustituir* métodos. Eso quiere decir que puede redefinir un método de clase de base, en una clase derivada.

Éste es un ejemplo, phpoverride.php. Aquí se sustituye el método set_name en la clase de base Person:

```
<?php
  class Person
  {
    var $name;

    function set_name($data)
    {
      $this->name = $data;
    }

    function get_name()
    {
      return $this->name;
    }
}
```

Este método se sustituye en la clase Friend con sólo redefinirlo. La versión de sustitución de este método pone el nombre con mayúscula antes de almacenarlo:

```
<html>
  <head>
    <title>
      Sustitución de métodos
    </title>
  </head>

  <body>
    <h1>
       Sustitución de métodos
    </h1>
    <?php
      class Person
      {
        var $name;

        function set_name($data)
        {
          $this->name = $data;
        }

        function get_name()
        {
          return $this->name;
        }
      }

      class Friend extends Person
      {
        var $name;

        function speak()
        {
          echo $this->name, " está hablando<br>";
        }

        function set_name($data)
        {
          $this->name = strtoupper($data);
        }
      }

      echo "Creando su nuevo amigo...<BR>";
      $friend = new Friend;
      $friend->set_name("Susan");
      $friend->speak();
      ?>
  </body>
</html>
```

FIGURA 7-9 Sustitución de métodos

Eso es todo —ahora, cuando cree un objeto $susan y dé nombre al objeto, ese nombre lo pondrá en mayúscula la versión sustituida del método set_name—. Puede ver los resultados en la Figura 7-9.

Sobrecarga de métodos

Además de sustituir métodos, también puede *sobrecargar* métodos en PHP. Sustituir un método significa redefinirlo, pero sobrecargarlo significa crear una versión alternativa con una lista de argumentos diferente. En lenguajes de programación orientada a objetos estándar, un método se sobrecarga definiendo una nueva versión del método con una lista de argumentos distinta. Éste es un ejemplo, donde se definen dos versiones del método set_name: uno que toma el nombre de la persona y otro tomando el nombre de la persona y su mensaje:

```
function set_name($data)
{
  $this->name = $data;
}

function set_name($data, $msg)
{
    $this->name = $data;
  $this->message = $msg;
}
```

Ahora podría llamar al método set_name con uno o dos argumentos y el lenguaje de programación orientada a objetos llamaría a la versión correcta de set_name, dependiendo de cuántos argumentos se pasarán:

```
$friend->set_name("Susan");
$friend->set_name("Ted", "Aquí Ted");
```

Así es como funcionan las cosas en los lenguajes de programación orientada a objetos estándar —pero PHP es diferente.

En PHP implementó la sobrecarga de métodos con el método __call. En una clase, éste es el método al que se llama cuando se invoca un método inexistente.

¿Me lo puede repetir? Suponga que tiene una clase con un método __call, pero no un método set_name. Cuando llama al método set_name en objetos de esa clase, al que se llama en realidad es a __call. Al método __call se le pasa el nombre del método faltante, además de una matriz conteniendo la lista de argumentos que se pasó a ese método faltante:

```
function __call($method, $arguments)
{
    .
    .
    .
}
```

Como los argumentos pasados al método faltante se envían en matriz, puede implementar la sobrecarga de métodos que usan el método __call. Por ejemplo, si deseará sobrecargar el método set_name para llamarlo set_name($name) o como set_name($name, $message), podría dejar ese método sin escribir nada en él y manipularlo con un método __call:

```
function __call($method, $arguments)
{
    .
    .
    .
}
```

En el método __call, primero podría verificar si el método al que se llama —y no se encuentra— era en realidad set_name:

```
function __call($method, $arguments)
{
  if($method == "set_name"){
    .
    .
    .
  }
}
```

Se puede llamar al método set_name con uno o dos argumentos ($name o $name y $message), de modo que antes pueda comprobar si se pasó un solo argumento:

```
function __call($method, $arguments)
{
  if($method == "set_name"){
```

```
    if(count($arguments) == 1){
      .
      .
      .
    }
  }
}
```

Si se pasó un solo argumento, ése es el nombre de la persona y se puede almacenar de esta manera:

```
function __call($method, $arguments)
{
  if($method == "set_name"){

    if(count($arguments) == 1){
      $this->name = $arguments[0];
    }
  }
}
```

Por otra parte, si se pasaron dos argumentos a la función, esos argumentos son $name y $message:

```
function __call($method, $arguments)
{
  if($method == "set_name"){

    if(count($arguments) == 1){
      $this->name = $arguments[0];
}

    if(count($arguments) == 2){
      $this->name = $arguments[0];
      $this->message = $arguments[1];
    }

  }
}
```

Así es como se maneja la sobrecarga; llamando al mismo nombre de método con diferentes números de argumentos en la programación orientada a objetos con PHP. Ahora puede llamar a set_name con uno o dos argumentos, como se ve en phpoverload.php:

```
<html>
  <head>
    <title>
      Sobrecarga de métodos
    </title>
  </head>
```

```
<body>
  <h1>
     Sobrecarga de métodos
  </h1>
  <?php
    class Friend
    {
      var $name;
      var $message;

      function speak()
      {
        echo $this->name, " dice \"", $this->message, "\"<br>";
      }

      function set_message($msg)
      {
        $this->message = $msg;
      }

      function __call($method, $arguments)
      {
        if($method == "set_name"){

          if(count($arguments) == 1){
            $this->name = $arguments[0];
          }

          if(count($arguments) == 2){
            $this->name = $arguments[0];
            $this->message = $arguments[1];
          }

        }
      }
    }
    echo "Creando su nuevo amigo...<br>";
    $friend = new Friend;
    $friend->set_name("Susan");
    $friend->set_message("Hola de Susan");
    $friend->speak();

    $friend->set_name("Ted", "Aquí Ted");
    $friend->speak();
  ?>
  </body>
</html>
```

Puede ver los resultados en la Figura 7-10, donde el código pudo llamar dos versiones diferentes del método set_name sobrecargado.

FIGURA 7-10 Sobrecarga de métodos

Carga automática de clases

Si va a crear muchas clases o sus definiciones de clases son extensas, podría almacenarlas en archivos por separado. Podría incluir implícitamente cada uno de estos archivos, pero hay una forma más fácil de hacerlo: puede usar la función __autoload.

A esta función se le pasan nombres de cualesquier clase que PHP busque y no puede hallar en el archivo en uso. Eso significa que puede cargar la clase faltante usando require o include de esta forma, donde la clase class_name está en un archivo class_name.php:

```
function _autoload($class_name)
{
    require $class_name . '.php';
}
```

Veamos esto en acción. Ha visto ejemplos con la clase Person y la clase Friend, así que pongamos esas clases en archivos externos. Éste es Person.php (observe que se requiere la marca <?php...?>):

```
<?php
      class Person
      {
        var $name;

        function set_name($data)
        {
            $this->name = $data;
        }
```

```
        function get_name()
        {
            return $this->name;
        }
    }
?>
```

Y éste es Friend.php:

```
<?php
    class Friend extends Person
    {
        var $message;

        function set_message($msg)
        {
            $this->message = $msg;
        }

        function speak()
        {
            echo $this->message;
        }
    }
?>
```

Ahora, en phpautoload.php, puede usar la función __autoload para cargar automáticamente Friend.php y Person.php según lo necesite, de esta forma:

```
<html>
  <head>
    <title>
      Carga automática de clases
    </title>
  </head>

  <body>
    <h1>
      Carga automática de clases
    </h1>

    <?php
      function __autoload($class_name)
      {
        require $class_name . '.php';
      }

      $tony = new Friend;
      $tony->set_name("Tony");
      $tony->set_message("Hola dice Tony.");
```

```
    echo "El nombre de su amigo es ", $tony->get_name(), ".<br>";
    echo $tony->get_name(), " dice \"", $tony->speak(), "\"<br>";
  ?>
 </body>
</html>
```

Los resultados se encuentran en la Figura 7-11, donde se cargaron correctamente las clases Person y Friend, como las necesitaba PHP. Ésa es una buena técnica, pues a medida que sus clases se hagan más extensas, es probable que desee colocarlas en archivos por separado.

FIGURA 7-11 Carga automática de clases

CAPÍTULO 8

Programación avanzada orientada a objetos

El capítulo anterior puso a rodar la bola con la programación orientada a objetos. En este capítulo continúa su recorrido guiado, proporcionándole cuanto necesita para convertirse en un maestro de la programación orientada a objetos en PHP. Aquí aprenderá a manejar miembros de clase estáticos, abstracción, interfaces y reflexión —todos los aspectos de cualquier lenguaje moderno con soporte para la programación orientada a objetos.

Creación de métodos estáticos

Suponga que deseara crear una clase utilitaria Math, en la que pudiese almacenar rutinas matemáticas útiles:

```
<?php
  class Math
  {
    .
    .
    .
  }
?>
```

Por ejemplo, colocaría un método en la clase Math llamado squarer, para mostrar el cuadrado de números pasados por usted:

```
<?php
  class Math
  {
    function squarer($op)
    {
      echo $op, "<sup>2</sup> = ", $op * $op, "<br>";
    }
  }
?>
```

Ahora puede crear un objeto de la clase Math y usar su método squarer para presentar los cuadrados de números de esta forma en phpmath.php:

```
<html>
  <head>
    <title>
      Uso de la clase Math
    </title>
  </head>

  <body>
    <h1>
       Uso de la clase Math
    </h1>
    <?php
      class Math
      {
        function squarer($op)
        {
          echo $op, "<sup>2</sup> = ", $op * $op, "<br>";
        }
      }

      echo "Uso de un objeto Math...<BR>";
      $math = new Math();
      echo $math->squarer(2);
    ?>
  </body>
</html>
```

Y los resultados se encuentran en la Figura 8-1.

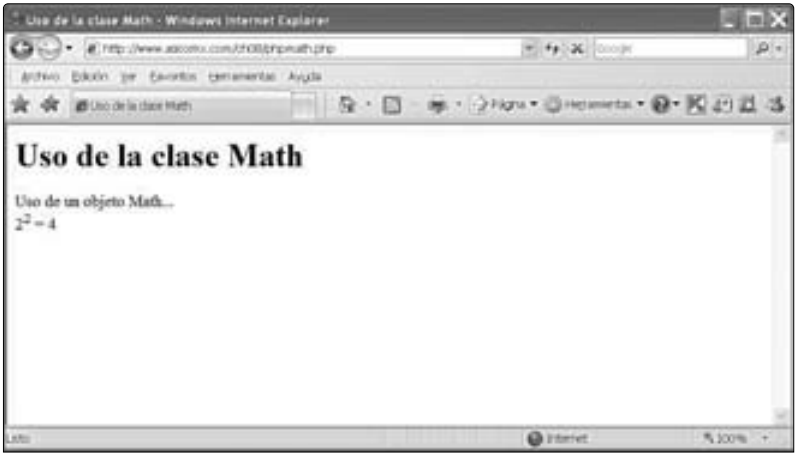

FIGURA 8-1 Uso de un objeto Math

Eso está bien, ¿pero por qué necesitó crear un objeto Math como éste?

```
echo "Uso de un objeto Math...<BR>";
$math = new Math();
echo $math->squarer(2);
```

La clase Math es simplemente una clase utilitaria; ¿no debería ser posible utilizar tan sólo métodos contenidos en ella, como el método squarer, directamente? ¿No debería usted simplemente llamar a estos métodos en la *clase*, en lugar de un *objeto* de esa clase?

Creación de un método estático

Cuando se crean métodos estáticos se puede hacer eso (llamar al método sin antes tener que crear un objeto de esa clase). Los métodos estáticos son métodos de clase; están diseñados para llamarse en la clase, no en un objeto. Veamos cómo funciona esto, alterando la clase Math para dar soporte a métodos estáticos, creando una vez más la clase Math:

```
<?php
  class Math
  {
    .
    .
    .
  }
?>
```

Por ejemplo, podría iniciar con un método estático llamado say_hi que muestra un mensaje:

```
<?php
  class Math
  {
    public function say_hi()
    {
      echo "La clase Math dice 'Hola a todos'. "<br>";
    }
  }
?>
```

Para hacer este método estático, debe usar la palabra clave static frente a la palabra clave function:

```
<?php
  class Math
  {
    public static function say_hi()
    {
      echo "La clase Math dice 'Hola a todos'. "<br>";
    }
  }
?>
```

Ahora puede tener acceso al método say_hi de la clase Math (sin crear antes un objeto de esa clase), como Math::say_hi( ):

```
<?php
  class Math
  {
    public static function say_hi()
    {
      echo "La clase Math dice 'Hola a todos'. "<br>";
    }
  }

    echo "Uso de la clase Math...<br>";
    Math::say_hi();
?>
```

Por los resultados en la Figura 8-2, se ve cómo funcionó la llamada al método estático; también llamado método de clase.

Observe la sintaxis aquí: Math::say_hi( ). Se nota que no puede usar el operador -> aquí, como haría trabajando con un objeto, pues no interviene objeto alguno. En su lugar, usa el operador dos puntos dobles (::) conocido también como operador de resolución de ámbito (conocido como, por extraño que parezca, Paamayim Nekudotayim en PHP), que significa "dos puntos dobles" en hebreo).

Ésta es otra sugerencia: realmente no es necesario usar la palabra clave static para hacer un método estático en PHP. Si utiliza un método de forma estática; es decir, llamarlo mediante el nombre de clase en vez de invocarlo como parte de un objeto, PHP tratará el método como estático. Es así como funciona; sin la palabra clave static:

```
<?php
  class Math
  {
    public function say_hi()
```

FIGURA 8-2 Llamada a un método de clase

```
    {
      echo "La clase Math dice 'Hola a todos'. "<br>";
    }
  }

    echo "Uso de la clase Math...<br>";
    Math::say_hi();
?>
```

Sin embargo, conviene usar la palabra clave static para mantener las cosas en orden en su código.

RECOMENDACIÓN *¿Puede pasar también datos a un método estático? Ciertamente.*

Paso de datos a un método estático

Suponga que deseara escribir el método squarer y agregarlo a la clase Math. Debe pasar ese método al número que desea elevar al cuadrado:

```
<?php
  class Math
  {
    static data;

    public static function say_hi()
    {
      echo "La clase Math dice 'Hola a todos'. "<br>";
    }

    function squarer($op);
    {
    .
    .
    .
    }
?>
```

Y puede utilizar los datos pasados a este método en código, dentro del método:

```
<?php
  class Math
  {
    statis $data;

    public static function say_hi()
    {
      echo "La clase Math dice 'Hola a todos'. "<br>";
    }

    function squarer($op)
    {
      echo $op, "<sup>2</sup> = ", $op + $op, "<br>;
    }
?>
```

Así es como se llama y pasan datos al método squarer en phpstaticmethod.php:

```
<html>
  <head>
    <title>
      Creación de métodos estáticos
    </title>
  </head>
  <body>
    <h1>
       Creación de métodos estáticos
    </h1>
    <?php
      class Math
      {
        static $data;

        public static function say_hi()
        {
          echo "La clase Math dice 'Hola a todos'. <br>";
        }

        function squarer($op)
        {
          echo $op, "<sup>2</sup> = ", $op * $op, "<br>";
        }

      echo "Uso de la clase Math...<br>";
      Math::say_hi();
      Math::squarer(2);
    ?>
  </body>
</html>
```

Puede ver los resultados en la Figura 8-3 —el método squarer tomó sus datos y presenta el cuadrado resultante.

Puede ver que es posible usar métodos estáticos con clases —¿qué hay de las propiedades estáticas?

Uso de propiedades en métodos estáticos

Algunas veces, usted desea usar propiedades cuando hace su trabajo y no es un problema usando objetos. Pero, ¿puede emplear propiedades trabajando con la clase en código, no en un objeto? Sí, en tanto sean propiedades estáticas.

Por ejemplo, suponga que tuviera un método estático llamado set_data pasado a un argumento:

```
<?php
  class Math
  {
    public static function set_data($op)
    {
```

FIGURA 8-3 Creación de un objeto

```
        .
        .
        .
        }
    }
?>
```

Ahora suponga que desea guardar los datos pasados a este método estático en una propiedad class. Resulta que puede hacer eso en PHP, aunque no interviene objeto alguno. Debe declarar la propiedad estática de esta forma:

```
<?php
  class Math
  {
    static $data;

    public static function set_data($op)
    {
    .
    .
    .
    }
  }
?>
```

Ahora tiene la libertad para almacenar datos en esa propiedad. Pero, ¿cómo referir esa propiedad en código? No puede usar la sintaxis $this->data, pues $this refiere el objeto actual y no hay objeto aquí. En cambio, debe utilizar la palabra clave self cuando trabaje con propiedades estáticas en métodos estáticos. Así que puede almacenar los datos pasados a set_data en la propiedad $data, de la siguiente manera:

```
<?php
  class Math
  {
    static $data;

    public static function set_data($op)
    {
      self::$data = $op;
      .
      .
      .
    }
  }
?>
```

Y reproducir con echo el valor alojado en esa propiedad, para probar que se ha manipulado correctamente:

```
<?php
  class Math
  {
    static $data;

    public static function set_data($op)
    {
      self::$data = $op;
      echo "self::\$data = ", self::$data, "<br>;
      .
      .
      .

    }
  }
?>
```

Estaría en condiciones para incrementar el valor en $data, de esta forma:

```
<?php
  class Math
  {
    static $data;

    public static function set_data($op)
    {
      self::$daa = $op;
      echo "self::\$data = ", self::$data, "<br>;
      echo "Sumando 1 a self::\$data <br>";
      self::$data++;
      echo "Ahora self::\$data = ", self::$data, "<br>";
    }
  }
?>
```

Así es como pondría a trabajar el método set_data en phpstaticproperty.php, pasándole un valor de 5:

```
<html>
  <head>
    <title>
      Creación de propiedades estáticas
    </title>
  </head>
  <body>
    <h1>
       Creación de propiedades estáticas
    </h1>
    <?php
      class Math
      {
        static $data;

        public static function set_data($op)
        {
          self::$data = $op;
          echo "self::\$data = ", self::$data, "<br>";
          echo "Sumando uno a self::\$data <br>";
          self::$data++;
          echo "Ahora self::\$data = ", self::$data, "<br>";
        }
      }
      echo "Uso de la clase Math...<br>";
      Math::set_data(5);
    ?>
  </body>
</html>
```

Y los resultados lucen en la Figura 8-4, donde la propiedad estática hizo lo suyo.

¿Puede usar también variables en métodos estáticos simples? Sí puede. Por ejemplo, es posible crear una variable nueva llamada $item y asignarle valor en la propiedad $data:

```
    <?php
      class Math
      {
        static $data;
        public static function set_data($op)
        {
          self::$data = $op;
          echo "self::\$data = ", self::$data, "<br>";
          echo "Sumando uno a self::\$data <br>";
          self::$data++;
          echo "Ahora self::\$data = ", self::$data, "<br>";
          echo "Asignando self::\$data a \$item. <br>";
          $item = self::$data;
```

FIGURA 8-4 Creación de una propiedad estática

```
          .
          .
          .
        }
      }
    ?>
```

E incrementar $item y mostrar los resultados:

```
<html>
  <head>
    <title>
      Creación de propiedades estáticas
    </title>
  </head>

  <body>
    <h1>
       Creación de propiedades estáticas
    </h1>
    <?php
      class Math
      {
        static $data;

        public static function set_data($op)
        {
          self::$data = $op;
          echo "self::\$data = ", self::$data, "<br>";
          echo "Sumando uno a self::\$data <br>";
          self::$data++;
          echo "Ahora self::\$data = ", self::$data, "<br>";
          echo "Asignando self::\$data a \$item. <br>";
          $item = self::$data;
```

FIGURA 8-5 Uso de variables en métodos estáticos

```
            echo "Sumando uno más a \$item. <br>";
            $item++;
            echo "Ahora \$item = ", $item, "<br>";
          }
        }

        echo "Uso de la clase Math...<br>";
        Math::set_data(5);
      ?>
    </body>
</html>
```

Los resultados se encuentran en la Figura 8-5, donde tuvo éxito en el uso de variables en métodos estáticos.

Miembros estáticos y herencia

¿Qué sucede con los miembros estáticos cuando interviene la herencia? ¿Puede referirse a miembros estáticos de una clase de base, desde una clase heredada? Sí puede.

Éste es un ejemplo, phpstaticinheritance.php. Agreguemos una propiedad estática, $message, conteniendo un mensaje de confirmación, a nuestra clase Math:

```
<?php
  class Math
  {
    static $data;
    static $message = "Sin preocupaciones.";

    public static function say_hi()
    {
      echo "La clase Math dice 'Hola a todos'. <br>";
    }
```

```
      public static function squarer($op)
      {
        echo $op, "<sup>2</sup> = ", $op * $op, "<br>";
      }
    }
  ?>
```

Ahora puede extender la clase Math a una nueva clase, New_Math:

```
<?php
  class Math
  {
    static $data;
    static $message = "Sin preocupaciones.";
    .
    .
    .
  }

  class New_Math extends Math
  {
    .
    .
    .
  }
?>
```

Y acceder a la propiedad estática $message, de la clase Math, desde un método en New_Math (el método show_message):

```
<?php
  class Math
  {
    static $data;
    static $message = "Sin preocupaciones.";
    .
    .
    .
  }

  class New_Math extends Math
  {
    public static function show_message()
    {
      echo "En New_Math, el mensaje de la clase Math es: '", Math::$message,
"'<br>";
    }
  }
?>
```

Ahora puede llamar a New_Math::show_message de esta forma, en phpstaticinheritance.php:

```
<html>
  <head>
    <title>
```

```
      Creación de métodos estáticos
    </title>
  </head>
  <body>
    <h1>
       Creación de métodos estáticos
    </h1>
    <?php
      class Math
      {
        static $message = "Sin preocupaciones.";
       .
       .
       .
      {

  class New_Math extends Math
  {
    public static function show_message()
    {
      echo "En New_Math, el mensaje de la clase Math es: '",
         Math::$message, "'<br>";
    }
  }

  echo "Uso de la clase Math...<br>";
  New_Math::show_message();
  ?>
  </body>
</html>
```

Los resultados están en la Figura 8-6, donde la propiedad estática $message, de la clase de base Math, en realidad estaba accesible desde el método estático show_message, en la clase New_Math derivada. Genial.

FIGURA 8-6 Uso de miembros estáticos con herencia

Creación de clases abstractas

PHP también permite crear clases abstractas. No puede utilizar clases abstractas con objetos creados directamente. En su lugar, deberá heredar una clase de otra clase abstracta primero. ¿Para qué sirve eso? Puede marcar métodos en particular en la clase abstracta como abstractos, significa que la clase a heredar debe definirlos antes de poder crear objetos de la clase que hereda.

La idea es obligar a cualquier persona que desee usar una clase abstracta a implementar sus propios métodos. Por ejemplo, suponga que usted es editor y que tiene una clase Novel que envía a los autores. Es posible que esa clase Novel tenga un método estándar, get_publisher_info, para devolver información acerca de usted como editor. Podría tener un método abstracto, get_text, que devuelva el texto de la novela a escribir por el autor. Como get_text es abstracto, el autor debe implementar el método get_text por sí mismo, exactamente lo que usted desea; al autor escribiendo su propia novela.

Demos un vistazo a esto en phoabstract.php. Puede comenzar creando la clase abstracta Novel (para contener métodos abstractos, una clase debe ser abstracta) que usted, como editor, envía a los autores:

```
<?php
  abstract class Novel
  {
    .
    .
    .
  }
?>
```

Esta clase puede tener un método estándar, por ejemplo get_publisher_info, para devolver información acerca de usted, el editor:

```
<?php
  abstract class Novel
  {
    function get_publisher_info()
    {
      return "SteveCo Publishing <br>";
    }
    .
    .
    .
  }
?>
```

Además, puede dar soporte a un método get_text, que devuelve texto de la novela. Ése es el método que usted desea implemente el autor, de modo que lo hace abstracto. Para ello, no se le da un cuerpo, tan sólo se termina su declaración con punto y coma:

```
<?php
  abstract class Novel
  {
    function get_publisher_info()
    {
      return "SteveCo Publishing <br>";
    }

    abstract function get_text();
  }
?>
```

El autor no puede crear un objeto de la clase Novel directamente —debe agregar código al método get_text por sí mismo—. El autor podría comenzar creando una nueva clase, My_Novel, extendiendo su clase abstracta Novel:

```
<?php
  abstract class Novel
  {
    function get_publisher_info()
    {
      return "SteveCo Publishing <br>";
    }
    abstract function get_text();
  }

  class My_Novel extends Novel
  {
    .
    .
    .
  }
?>
```

Para evitar objeciones de PHP, este código tiene que implementar la función get_text, que se podría ver como ésta:

```
<?php
  abstract class Novel
  {
    function get_publisher_info()
    {
      return "SteveCo Publishing <br>";
    }

    abstract function get_text();
  }
```

```
    class My_Novel extends Novel
    {
      public function get_text()
      {
        return "Era una noche oscura y lluviosa...";
      }
    }
  ?>
```

Éste es phpabstract.php, para crear y usar un objeto de la clase My_Novel:

```
<html>
  <head>
    <title>
      Creación de clases abstractas
    </title>
  </head>

  <body>
    <h1>
       Creación de clases abstractas
    </h1>
    <?php
      abstract class Novel
      {
        function get_publisher_info()
        {
          return "SteveCo Publishing <br>";
        }

        abstract function get_text();
      }

      class My_Novel extends Novel
      {
        public function get_text()
        {
          return "Era una noche oscura y lluviosa...";
        }
      }

      $mynovel = new My_Novel();
      echo "Esta novela proviene de ", $mynovel->get_publisher_info();
      echo "La novela dice '", $mynovel->get_text(), "'<br>";
    ?>
  </body>
</html>
```

Como verá en la Figura 8-7, la clase abstracta hizo lo suyo —el autor necesitó implementar el método abstracto get_text, antes de crear cualquier objeto.

FIGURA 8-7 Uso de clases abstractas

Creación de interfaces

En la programación orientada a objetos, las interfaces son un poco como clases abstractas —especifican qué métodos debe implementar una clase—. Sin embargo, las interfaces son sólo especificaciones para métodos; no pueden incluir código.

Por ejemplo, podría tener una interfaz llamada iDatabase, especificando qué métodos debe implementar una clase de base de datos (get_record y set_record, por ejemplo). Cualquier clase de base de datos que implementara la interfaz iDatabase, tendría que implementar también los métodos get_record y set_record.

Al igual que la mayoría de lenguajes con soporte para programación orientada a objetos, PHP no soporta herencia múltiple —es decir, no puede heredar de múltiples clases en el mismo nivel, ya que no puede listar múltiples clases con la palabra clave extends—. Es decir, puede hacer esto:

```
class My_Novel extends Novel
{
   .
   .
   .
}
```

Pero no puede hacer esto:

```
class My_Novel extends Novel, Novella, Story
{
   .
   .
   .
}
```

Se considera a veces que el uso de interfaces le permite usar herencia múltiple, ya que puede implementar múltiples interfaces al mismo tiempo, de esta forma:

```
class My_Novel implements iNovel
{
  .
  .
  .
}
```

Sin embargo, no se deje engañar (esto no es herencia múltiple). Las interfaces son simplemente especificaciones de métodos y las interfaces no incluyen ningún código para esos métodos. Las interfaces son útiles cuando, por ejemplo, desea crear docenas de clases y estandarizar los métodos en cada clase. Los nombres y las listas de argumentos de los métodos serán los mismos en todas las clases que implementen la interfaz (aunque las implementaciones de código reales de los métodos pueden variar por clase).

Éste es un ejemplo, que modifica la clase Novel abstracta que acaba de ver en una interfaz, iNovel. Para crear una interfaz, se usa la palabra clave interface:

```
<?php
      interface iNovel
      {
        .
        .
        .
      }
?>
```

Luego lista los métodos que desea que contenga cualquier clase que implemente esta interfaz. Observe que como sucede con los métodos abstractos, no se incluye ningún cuerpo del método. En este caso, podría agregar dos métodos a iNovel: get_dedication, que devuelve la dedicatoria de la novela, y get_text, que devuelve el texto de la novela:

```
<?php
      interface iNovel
      {
        function get_dedication();
        function get_text();
      }
?>
```

Para implementar esta interfaz en la clase My_Novel, se utiliza la palabra clave implements:

```
<?php
      interface iNovel
      {
        function get_dedication();
        function get_text();
      }
```

```
  class My_Novel implements iNovel
  {
    .
    .
    .
  }
?>
```

Como ha implementado la interfaz iNovel, debe implementar todos los métodos de esa interfaz. Aquí, se trata de get_dedication y get_text, y tiene este aspecto en My_Novel:

```
<?php
      interface iNovel
      {
        function get_dedication();
        function get_text();
      }

  class My_Novel implements iNovel
  {
    public function get_dedication()
    {
      return "Para mi adorada.";
    }

    public function get_text()
    {
      return "Era una noche oscura y lluviosa...";
    }
  }
?>
```

Así es como se puede crear un objeto usando My_Novel y ponerlo a trabajar:

```
<html>
  <head>
    <title>
      Uso de interfaces
    </title>
  </head>

  <body>
    <h1>
       Uso de interfaces
    </h1>
    <?php
      interface iNovel
      {
        function get_dedication();
        function get_text();
      }
```

```
    class My_Novel implements iNovel
    {
      public function get_dedication()
      {
        return "Para mi adorada.";
      }

      public function get_text()
      {
        return "Era una noche oscura y lluviosa...";
      }
    }

    $mynovel = new My_Novel();
    echo "La novela está dedicada '", $mynovel->get_dedication(), "'<br>";
    echo "La novela dice '", $mynovel->get_text(), "'<br>";
  ?>
 </body>
</html>
```

Puede ver los resultados en la Figura 8-8, donde la interfaz iNovel se implementó en la clase My_Novel. No está mal.

Tenga en mente que las interfaces no hacen realmente todo eso por usted —tan sólo permiten especificar que métodos deben implementar las clases de implementación—.Y eso está bien si desea mantener un conjunto de métodos consistente entre múltiples clases.

¿Desea otro ejemplo de interfaz? Ahora se lo damos.

FIGURA 8-8 Uso de interfaces

Soporte a la iteración de objetos

¿Desea permitir que sus objetos contengan una serie de elementos para recorrerlos utilizando foreach? Puede hacerlo si implementa la interfaz Iterator de PHP, conteniendo estos métodos:

- **public function current()** Devuelve el elemento actual de su colección
- **public function key()** Devuelve la clave actual
- **public function next()** Devuelve el siguiente elemento
- **public function valid()** Devuelve el valor verdadero, si el elemento actual es válido
- **public function rewind()** Inicia de nuevo las operaciones desde el principio

Pongamos esto a trabajar en un ejemplo simple, phpiterator, para crear objetos que puede utilizar en ciclos foreach. La clase aquí se llamará DataHandler y para hacer éste un ejemplo simple, usará una matriz internamente, puesto que las matrices ya cuentan con soporte para métodos de la interfaz Iterator, que facilita nuestro trabajo.

Primero se crea la clase DataHandler, implementando la interfaz Iterator de PHP:

```
<?php
  class DataHandler implements Iterator
  {
    .
    .
    .
  }
?>
```

Esta clase almacenará una matriz internamente que usted pasa a su constructor y permite realizar iteraciones por la matriz. Así es como el constructor almacena la matriz, como propiedad privada $array:

```
<?php
  class DataHandler implements Iterator
  {
      private $array = array();
      public function __construct($arr)
      {
           if (is_array($arr)) {
               $this->array = $arr;
           }
       }
    .
    .
    .
  }
?>
```

Ahora puede usar funciones de matriz de PHP para escribir métodos Iterator —es decir, para permitirle usar matrices en instrucciones foreach, las matrices implementan ya esos métodos—; de modo que partiremos de ese hecho aquí. Por ejemplo, el método DataHandler current

usará el método de matriz current en la matriz privada $array, el método key emplea el método de matriz key, en la matriz privada $array y así sucesivamente. Así es como se implementan los métodos Iterator en dataHandler:

```
<?php
      class DataHandler implements Iterator
      {
          private $array = array();
          public function __construct($arr)
          {
              if (is_array($arr)) {
                  $this->array = $arr;
              }
          }

          public function current()
          {
              return current($this->array);
          }

          public function key()
          {
              return key($this->array);
          }

          public function next()
          {
              return next($this->array);
          }

          public function valid()
          {
              return $this->current() !== false;
          }

          public function rewind()
          {
              reset($this->array);
          }
      }
    ?>
```

Todo lo que resta por hacer es crear un objeto de la clase DataHandler, inicializándola con una matriz y luego usar un ciclo foreach para hacer iteraciones por ese objeto, que luce así en phpiterator.php:

```
<html>
  <head>
    <title>
      Iteración de objetos
    </title>
  </head>
```

```
  <body>
    <h1>
       Iteración de objetos
    </h1>
    <?php
      class DataHandler implements Iterator
      {
          private $array = array();
          public function __construct($arr)
          {
              if (is_array($arr)) {
                  $this->array = $arr;
              }
          }
        .
        .
        .
      }
      $new_array = array("a", "b", "c", "d", "e");
      $object = new DataHandler($new_array);
      echo "Iteración por el objeto: <br>";
      foreach ($object as $key => $value) {
          echo $key, ' => ', $value, "<br>";
      }
    ?>
  </body>
</html>
```

Los resultados aparecen la Figura 8-9, donde el ciclo foreach no tuvo problemas con el objeto DataHandler. Genial.

FIGURA 8-9 Iteración de objetos

Comparación de objetos

La comparación de dos objetos en PHP requiere algo de meditación. Cuando se usa el operador de comparación (==), las variables de objeto se comparan simplemente —dos objetos son iguales si tienen los mismos atributos y valores, así como si son objetos de la misma clase—. Si utiliza el operador de identidad (===), los objetos son idénticos si y sólo si se refieren al mismo objeto.

Este ejemplo, phpcompare.php, muestra cómo funciona esto. En este ejemplo, $object_1 es una instancia de Class_1, $object_1ª es otra instancia de Class_1, $object_1copy es una copia de $object_1 y $object_2 es una instancia de Class_2:

```
<html>
  <head>
    <title>
      Comparación de objetos
    </title>
  </head>
  <body>
    <h1>
       Comparación de objetos
    </h1>
    <?php
      class Class_1
      {
        public $data;

        function __construct($item)
        {
          $this->data = $item;
        }
      }

      class Class_2
      {
        public $data;
        function __construct($item)
        {
          $this->data = $item;
        }
      }

      $object_1 = new Class_1("a");
      $object_1copy = $object_1;
      $object_1a = new Class_1("a");
      $object_2 = new Class_2("a");

      if($object_1 == $object_1a){
        echo '$object_1 == $object_1a is TRUE <br>';
      }
```

```
    else {
      echo '$object_1 == $object_1a is FALSE <br>';
    }

    if($object_1 == $object_2){
      echo '$object_1 == $object_2 is TRUE <br>';
    }
    else {
      echo '$object_1 == $object_2 is FALSE <br>';
    }

    if($object_1 === $object_1a){
      echo '$object_1 === $object_1a is TRUE <br>';
    }
    else {
      echo '$object_1 === $object_1a is FALSE <br>';
    }

    if($object_1 === $object_1copy){
      echo '$object_1 === $object_1copy is TRUE <br>';
    }
    else {
      echo '$object_1 === $object_1copy is FALSE <br>';
    }

  ?>
 </body>
</html>
```

Puede ver los resultados en la Figura 8-10, donde las diferentes comparaciones funcionan como deben hacerlo.

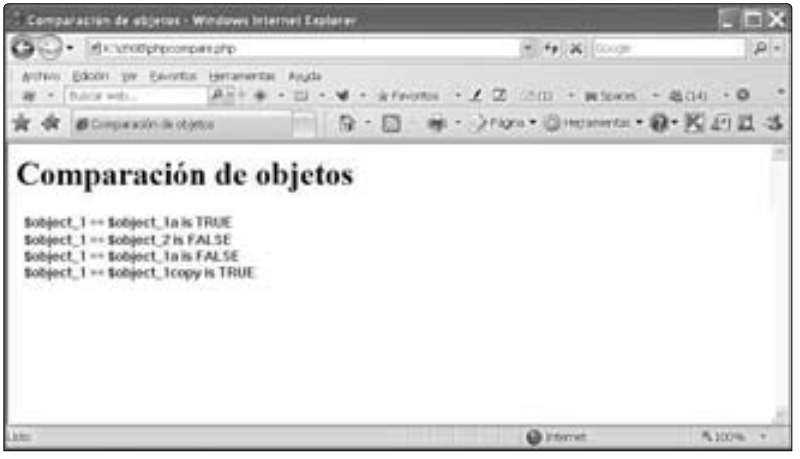

FIGURA 8-10 Comparación de objetos

Creación de constantes de clase

También puede crear constantes de clase en clases de PHP. Éstas se diseñan para ser utilizadas por clases, no por objetos. Éste es un ejemplo, phpconstant.php, que declara una clase Math de la siguiente forma:

```
<?php
  class Math
  {
    .
    .
    .
  }
?>
```

Y puede definir una constante en la clase Math, pi:

```
<?php
  class Math
  {
    const pi = 3.14159;
    .
    .
    .
  }
?>
```

Ahora puede referirse a la constante pi en el código dentro de la clase Math como self::pi. Usará la constante con self:: en vez de $this->, porque para las constantes de clase no interviene objeto. Así es como se podría mostrar self::pi utilizando un método público, display_pi:

```
<?php
  class Math
  {
    const pi = 3.14159;
    function display_pi()
    {
      echo  'Pi del interior de la clase (self::pi): ', self::pi , "<br>";
    }
  }
?>
```

¿Puede acceder a pi fuera de la clase? Sí puede (como Math::pi). Observe que utiliza el nombre de clase, Math, no un nombre de objeto:

```
<?php
  class Math
  {
    const pi = 3.14159;
```

```
    function display_pi()
    {
        echo  'Pi del interior de la clase (self::pi): ', self::pi , "<br>";
    }
  }
  echo 'Pi del exterior de la clase (Math::pi): ', Math::pi , "<br>";
    .
    .
    .
?>
```

También puede mostrar pi utilizando objetos, si llama al método público display_pi:

```
<?php
  class Math
  {
    const pi = 3.14159;
    function display_pi()
    {
        echo  'Pi del interior de la clase (self::pi): ', self::pi , "<br>";
    }
  }
  echo 'Pi del exterior de la clase (Math::pi): ', Math::pi , "<br>";
  $object = new Math();
  $object->display_pi();
    .
    .
    .
?>
```

Sin embargo, no puede tener acceso a la constante de clase pi, mediante expresiones de objeto como $object::pi u $object->pi de esta forma en phpconstant.php:

```
<html>
  <head>
    <title>
      Uso de constantes de clase
    </title>
  </head>
  <body>
    <h1>
       Uso de constantes de clase
    </h1>
    <?php
      class Math
      {
        const pi = 3.14159;
```

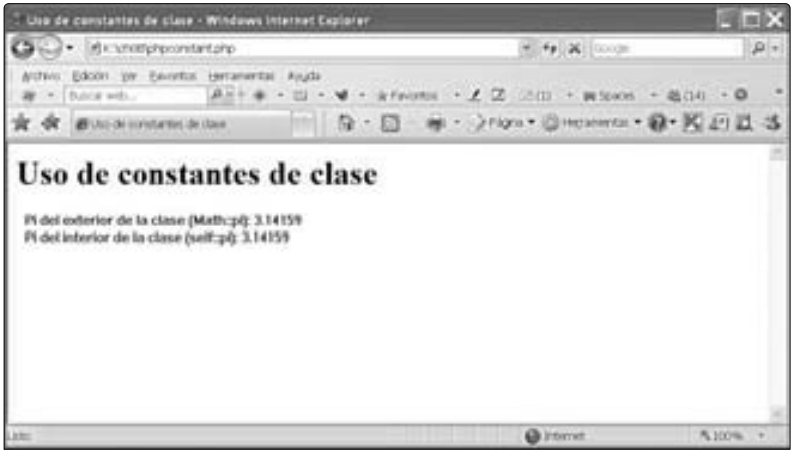

FIGURA 8-11 Uso de constantes de clase

```
        function display_pi()
        {
          echo  'Pi del interior de la clase (self::pi): ', self::pi , "<br>";
        }
      }

      echo 'Pi del exterior de la clase (Math::pi): ', Math::pi , "<br>";

      $object = new Math();
      $object->display_pi();

      //Ninguno de éstos funcionará:
      //echo $object::pi;
      //echo $object->pi;
    ?>
  </body>
</html>
```

En la Figura 8-11 puede ver los resultados, donde el valor de la constante se muestra en realidad.

De modo que las constantes de clase están disponibles para usarse fuera del código de una clase y, cuando se les antepone nombre de clase y operador de resolución de ámbito —pero eso es todo.

Uso de la palabra clave final

Quizá recuerde que puede sustituir clases como ésta. En este ejemplo tomado del capítulo anterior, el método set_name se sustituye con una nueva versión que escribe el nombre con mayúscula:

```
<html>
  <head>
    <title>
      Métodos finales
    </title>
  </head>

  <body>
    <h1>
       Métodos finales
    </h1>
    <?php
      class Person
      {
        var $name;

        function set_name($data)
        {
          $this->name = $data;
        }

        function get_name()
        {
          return $this->name;
        }
      }

      class Friend extends Person
      {
        var $name;

        function speak()
        {
          echo $this->name, " está hablando<br>";
        }

        function set_name($data)
        {
          $this->name = strtoupper($data);
        }
      }

      echo "Creando su nuevo amigo...<BR>";
      $friend = new Friend;
      $friend->set_name("Susan");
      $friend->speak();
      ?>
  </body>
</html>
```

¿Qué sucede si no desea permitir se sustituya un método? Puede hacer eso en PHP, como con otros lenguajes con soporte para programación orientada a objetos, con la palabra clave

final. Todo lo que debe hacer es usar esta palabra clave declarando un método que desea restringir, de esta forma en la clase de base Person, en phpfinal.php:

```
class Person
{
  var $name;

  final function set_name($data)
  {
    $this->name = $data;
  }

  function get_name()
  {
    return $this->name;
  }
}
```

Marcar set_name como final debe darnos un error; puede ver los resultados en la Figura 8-12, donde PHP nos informa que no se puede sustituir set_name, ya que es final. Genial.

RECOMENDACIÓN *El uso de la palabra clave final es una buena idea en clases que dará a otras desde las cuales pueden heredar, si no desea que se sustituyan métodos.*

De hecho, puede declarar clases completas como finales, significa que no puede extenderlas. Así luce esto:

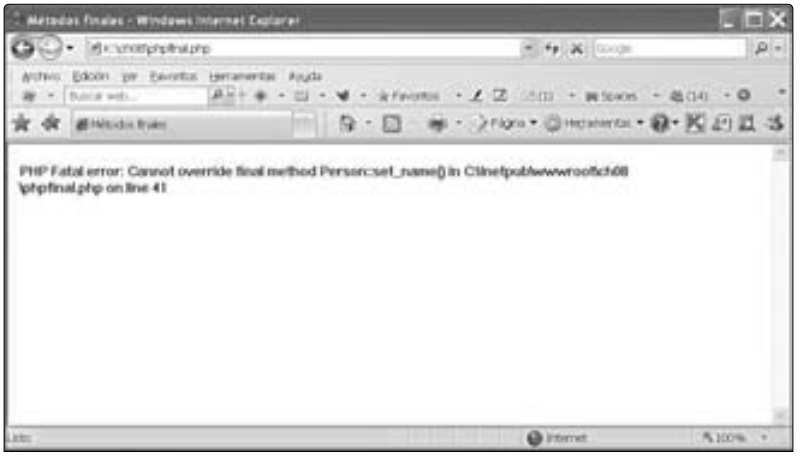

FIGURA 8-12 Cómo deshabilitar la sustitución de un método

```
<html>
  <head>
    <title>
      Clases finales
    </title>
  </head>
  <body>
    <h1>
       Clases finales
    </h1>
    <?php
      final class Person
      {
        var $name;

        function set_name($data)
        {
          $this->name = $data;
        }

        function get_name()
        {
          return $this->name;
        }
      }

      class Friend extends Person
      {
        var $name;

        function speak()
        {
          echo $this->name, " está hablando<br>";
        }

        function set_name($data)
        {
          $this->name = strtoupper($data);
        }
      }

      echo "Creando su nuevo amigo...<BR>";
      $friend = new Friend;
      $friend->set_name("Susan");
      $friend->speak();
      ?>
  </body>
</html>
```

Ahora que la clase Person se ha declarado final, este código dará un error, ya que intenta sustituir esa clase.

Clonación de objetos

Cuando se copian objetos, debe meditar un poco el proceso. Puede usar la palabra clave clone para hacer una copia de un objeto en PHP:

```
$object_copy = clone $object_1;
```

Esta instrucción hace una copia de $object_1, $object_copy (si usara el operador de asignación, =, terminaría con dos nombres de variable que apuntarían al mismo objeto).

Sin embargo, hay un problema aquí: si $object_1 tiene propiedades internas conteniendo subobjetos, las propiedades del objeto clonado apuntarán también a los mismos subobjetos. Y quizá eso no sea lo que usted desea —podría querer que la copia del objeto contenga nuevos subobjetos—. Puede hacer que suceda mediante el método __clone.

Éste es un ejemplo para aclarar esto, phpclone.php. Podría tener una clase, BigClass, conteniendo dos objetos LittleClass:

```
<?php
  class BigClass
  {
    public $little_1;
    public $little_2;

    public function __construct()
    {
        $this->little_1 = new LittleClass();
        $this->little_2 = new LittleClass();
    }
    .
    .
    .
    }
?>
```

Puede agregar un método __clone a la clase BigClass, que clonará expresamente $little_1:

```
<?php
  class BigClass
  {
    public $little_1;
    public $little_2;

    public function __construct()
    {
        $this->little_1 = new LittleClass();
        $this->little_2 = new LittleClass();
    }
```

```
    function __clone()
    {
      $this->little_1 = clone $this->little_1;
    }
  }
?>
```

En LittleClass, puede dar un número único a cada instancia, almacenado en la propiedad $number, a incrementarse en cada ocasión que usted crea un nuevo objeto LittleClass:

```
<?php
  class LittleClass
  {
    static $counter = 0;
    public $number;

    public function __construct()
    {
         $this->number = ++self::$counter;
    }

    public function __clone() {
      $this->number = ++self::$counter;
    }
  }
?>
```

Y puede agregar un método __clone en LittleClass, que incrementará $number cuando se clone un objeto LittleClass:

```
<?php
  class LittleClass
  {
    static $counter = 0;
    public $number;

    public function __construct()
    {
         $this->number = ++self::$counter;
    }

    public function __clone() {
      $this->number = ++self::$counter;
    }
  }
?>
```

Ahora puede crear un objeto BigClass y luego clonarlo —entonces observe las propiedades del primer objeto y la propiedad del clon de esta forma:

```
html>
  <head>
    <title>
```

```
    Clonación de objetos
  </title>
</head>

<body>
  <h1>
     Clonación de objetos
  </h1>
  <?php
    class BigClass
    {
      public $little_1;
      public $little_2;

      public function __construct()
      {
         $this->little_1 = new LittleClass();
         $this->little_2 = new LittleClass();
       }

       function __clone()
       {
         $this->little_1 = clone $this->little_1;
       }
     }

    class LittleClass
    {
      static $counter = 0;
      public $number;

      public function __construct()
      {
          $this->number = ++self::$counter;
      }

      public function __clone() {
        $this->number = ++self::$counter;
      }
    }

    $object_1 = new BigClass();
    $object_2 = clone $object_1;

    echo "\$object_1: <br>";
    print_r($object_1);

    echo "<br><br>";
    echo "\$object_2: <br>";
    print_r($object_2);

  ?>
</body>
</html>
```

FIGURA 8-13 Clonación de objetos

Puede ver los resultados en la Figura 8-13. Observe que para $object_1, $numbger = 1 de $little_1 y $number = 2 de $little_2. Para $object_2, $number = 3 de $little_1, mientras que $number = 2 de $little_2. Es decir, $little_2 de $object_1 y $little_2 de $object_2 apuntan al mismo objeto. Por otra parte, como había un método __clone que manejaba $little_1 de forma diferente, $little_1 de $object_1 y $little_1 de $object_2 apuntan a diferentes objetos. Muy bien.

Reflexión

El uso de la reflexión en la programación orientada a objetos significa que puede examinar su propio código en tiempo de ejecución. Éste es un ejemplo, phpreflection.php, mostrando cómo obtener información acerca de un método. En este caso el método se llama tracker:

```
<?php
  function tracker()
  {
    static $counter = 10;
    $counter++;
    return $counter;
  }
?>
```

Para obtener información acerca de este método puede usar la clase ReflectionFunction:

```
<?php
  function tracker()
  {
    static $counter = 10;
    $counter++;
    return $counter;
  }
```

```
    $method = new ReflectionFunction('tracker');
      .
      .
      .
  ?>
```

Ahora puede usar este nuevo objeto, $method, para hallar el nombre del método que representa, las líneas en que se extiende el método en el código, que sucede cuando pasa datos al método, qué propiedades estáticas tiene el método y más. Así se ve en phpreflection.php:

```
<html>
  <head>
    <title>
      Reflexión
    </title>
  </head>

  <body>
    <h1>
       Reflexión
    </h1>
    <?php
      function tracker()
      {
        static $counter = 10;
        $counter++;
        return $counter;
      }

      $method = new ReflectionFunction('tracker');

      echo  "El método llamado  ", $method->getName();
      echo " es ", $method->isInternal() ? "definido por PHP <br>." :
        "definido por el usuario. <br>";
      echo "Está en ", $method->getFileName(), "<br>";
      echo "Comienza en la línea ", $method->getStartLine();
      echo " y termina en la línea ",  $method->getEndline(), "<br>";

      if ($method->getStaticVariables()){
        $statics = $method->getStaticVariables();
        echo "Tiene esta variable estática: ", var_export($statics, 1), "<br>";
      }

     echo "La invocación del método da como resultado ", var_dump($method->invoke()), "<br>";
    ?>
  </body>
</html>
```

FIGURA 8-14 Uso de la reflexión

Puede ver los resultados en la Figura 8-14, donde, como notará, hay mucha información acerca del método tracker.

NOTA *Puede realizar también reflexión en clases, objetos, propiedades y más.*

CAPÍTULO 9

Manejo de archivos

En este capítulo se analiza el manejo de archivos con PHP. Almacenar datos en el servidor es especialmente poderoso para aplicaciones Web, pues permite que los datos "persistan"; es decir, se mantengan cerca entre accesos a la página. Blogs, libros de visitas, páginas de comentarios; todos son posibles cuando se trabaja con archivos en el servidor.

En este capítulo veremos mucha tecnología PHP, comenzando con la forma de abrir archivos.

Cómo abrir archivos usando fopen

Para trabajar con un archivo en PHP, primero debe abrir el archivo, como en la mayoría de lenguajes.

```
$filehandle = fopen (filename, mode [, use_include_path [, zcontext]])
```

En esta llamada a función, *filename* es el nombre de archivo que va a abrir, *mode* indica cómo desea abrirlo archivo (por ejemplo, leer su contenido o escribir en él), *use_include_path* puede establecerse en 1 o TRUE, para especificar que desea buscar el archivo en la ruta de inclusión de PHP y *zcontext* aloja un contexto de archivo opcional (los contextos modifican o mejoran el comportamiento de los flujos de datos desde y hacia archivos). Éstos son los modos posibles:

- **'r'** Abrir sólo para lectura.
- **'r+'** Abrir para lectura y escritura.
- **'w'** Abrir sólo para escritura y truncar el archivo en longitud cero. Si el archivo no existe, intente crearlo.
- **'w+'** Abrir para lectura y escritura y truncar el archivo a longitud cero. Si el archivo no existe, intente crearlo.
- **'a'** Abrir sólo para anexar. Si el archivo no existe, intente crearlo.
- **'a+'** Abrir para lectura y escritura, comenzando al final del archivo. Si el archivo no existe, intente crearlo.
- **'x'** Crear y abrir sólo para escritura. Si el archivo ya existe, la llamada fopen fallará devolviendo FALSE.
- **'x+'** Crear y abrir para lectura y escritura. Si el archivo ya existe, la llamada fopen fallará devolviendo FALSE.

Observe que diferentes sistemas operativos tienen diferentes convenciones para finalizar líneas. Cuando escribe un archivo de texto y desea insertar un cambio de línea, necesita utilizar el o los caracteres de final de línea correctos para su sistema operativo. Los sistemas basados en Unix usan \n como carácter de final de línea; los sistemas con Windows utilizan \r\n, mientras los sistemas Macintosh utilizan \r para el mismo efecto.

En Windows, puede emplear una bandera de traducción en modo de texto ('t'), que traducirá \n a \r\n cuando trabaje con el archivo. En contraste, también puede usar 'b' para forzar el modo binario, que no traducirá sus datos. Para usar estas banderas, especifique 'b' o 't' como el último carácter del parámetro modo, como 'wt'.

En este momento, el modo predeterminado está en binario para todas las plataformas distinguiendo entre modo binario y de texto. Si tiene problemas con sus scripts, pruebe usar la bandera 't'.

Este ejemplo abre el archivo /home/file.txt para lectura (puede usar diagonales como éstas en nombres de ruta, incluso en Windows (también puede usar diagonales inversas, mientras las cierre de esta forma: \\):

```
$handle = fopen("/home/file.txt", "r");
```

Cuando abre un archivo consigue un manejador de archivos, que representa un archivo abierto. De ahí en adelante, usa este manejador para trabajar con el archivo. Ahora que ha abierto el archivo, puede leerlo usando las diferentes funciones de lectura de datos de las cuales hablaremos en un par de páginas, como fread.

Este ejemplo abre un archivo para escritura:

```
$handle = fopen("/home/file.txt", "w");
```

Este ejemplo abre un archivo para escritura binaria:

```
$handle = fopen("/home/file.txt", "wb");
```

En Windows, debe tener cuidado de cerrar cualquier diagonal inversa usada en la ruta al archivo (o use diagonales normales):

```
$handle = fopen("c:\\data\\file.txt", "r");
```

Tampoco está limitado a archivos en el sistema local de archivos, Así es como podría abrir un archivo en un sitio Web diferente, como especifica la dirección URL:

```
$handle = fopen("http://www.superduperbigco.com/file.txt", "r");
```

También puede abrir archivos usando el protocolo FTP:

```
$handle = fopen("ftp://usuario:contraseña@superduperbigco.com/file.txt", "w");
```

Cuando abre un archivo consigue un manejador de archivos con que trabajar y pasarlo a otras funciones de archivo para trabajar con el archivo.

Éste es un ejemplo. Suponga que tiene un archivo, file.txt, con este contenido:

```
Éstos
son
sus
datos.
```

Podría abrir este archivo para lectura en phpfopen.php:

```
<?php
  $handle = fopen("file.txt", "r");
    .
    .
    .
?>
```

Si falla la operación de apertura, fopen devuelve FALSE, de modo que puede comprobar si el archivo se abrió de esta forma en phpfopen.php:

```
<html>
  <head>
    <title>
      Abriendo un archivo
    </title>
  </head>
  <body>
    <h1>
       Abriendo un archivo
    </h1>
    <?php
      $handle = fopen("file.txt", "r");
      if($handle){
        echo "Archivo abierto correctamente.";
      }
    ?>
  </body>
</html>
```

Puede ver los resultados en la Figura 9-1, donde el archivo file.txt se abrió correctamente.

FIGURA 9-1 Cómo abrir un archivo

También puede abrir direcciones URL en Internet con fopen, que lo prepara para leer el contenido de páginas Web, de esta forma:

```
$handle = fopen(http://www.php.net, "r");
```

Bueno, ahora el archivo se abrió correctamente —¿cómo lee el texto contenido en él?

Recorrido cíclico del contenido de un archivo con feof

El archivo que está leyendo, file.txt, tiene muchas líneas y verá de qué forma leer esas líneas, de una en una. ¿Cómo se recorren todas las líneas del archivo, ahora que ha abierto el archivo?

Puede usar un ciclo while y la función feof. Se pasa un manejador de archivos a esta función y devolverá TRUE, si está al final del archivo. Así que aquí vemos cómo recorrer el contenido del archivo, línea por línea, en phpread.php:

```
<html>
  <head>
    <title>
      Lectura del contenido de un archivo
    </title>
  </head>
  <body>
    <h1>
       Lectura del contenido de un archivo
    </h1>
    <?php
      $handle = fopen("file.txt", "r");
      while (!feof($handle)){
        .
        .
        .
      }
    ?>
  </body>
</html>
```

Para hacer este ciclo activo, debe leer realmente el contenido del archivo (de lo contrario, nunca llegará al final del archivo y este ciclo nunca terminará). Puede hacer eso con la función fgets.

Lectura de texto de un archivo utilizando fgets

Puede recurrir a la función fgets para obtener una cadena de texto de un archivo; así es como se usa en general:

```
fgets (manejador [, longitud])
```

Esta función se pasa del manejador de archivos correspondiente a un archivo abierto y una longitud opcional. La función devuelve una cadena con una longitud de hasta –1 byte leído, del archivo correspondiente al manejador de archivos. La lectura termina cuando se ha leído

longitud –1 byte, en una línea nueva (incluida en el valor de retorno) o al encontrar el final del archivo, lo que ocurra primero. Si no se especifica una longitud, ésta se fija de forma predeterminada en 1024 bytes.

Así es como podría leer una línea de texto del archivo, file.txt, si phpread.php:

```
<?php
  $handle = fopen("file.txt", "r");
  while (!feof($handle)){
    $text = fgets($handle);
    .
    .
    .
  }
?>
```

Y podría reproducir con echo el texto leído, línea por línea:

```
<html>
  <head>
    <title>
      Lectura del contenido de un archivo
    </title>
  </head>
  <body>
    <h1>
       Lectura del contenido de un archivo
    </h1>
    <?php
      $handle = fopen("file.txt", "r");
      while (!feof($handle)){
        $text = fgets($handle);
        echo $text, "<br>";
      }
    ?>
  </body>
</html>
```

Eso abre el archivo y lee su contenido. Hay un paso más que seguir para completar la operación de lectura del archivo; cerrar el archivo.

Cómo cerrar un archivo

Cuando termine de trabajar con un archivo, debe cerrarlo en PHP. Cerrar el archivo libera recursos conectados con ese archivo y evita conflictos más tarde en su código, en caso de que recicle variables de manejador de archivos.

Para cerrar un archivo se usa fclose de esta forma:

```
fclose($filehandle);
```

Esta función devuelve TRUE, si el archivo se cerró correctamente y FALSE en caso contrario.

Así es como se pone a trabajar fclose en phpread.php:

```
<html>
  <head>
    <title>
      Lectura del contenido de un archivo
    </title>
  </head>
  <body>
    <h1>
       Lectura del contenido de un archivo
    </h1>
    <?php
      $handle = fopen("file.txt", "r");
      while (!feof($handle)){
        $text = fgets($handle);
        echo $text, "<br>";
      }
      fclose($handle);
    ?>
  </body>
</html>
```

Puede ver los resultados en la Figura 9-2, donde el archivo file.txt se abrió correctamente, se leyó su contenido y luego se cerró.

Ésa es una forma de leer texto (línea por línea). ¿Qué tal carácter por carácter? Lo veremos a continuación.

FIGURA 9-2 Lectura del contenido de un archivo

Lectura del contenido de un archivo carácter por carácter con fgetc

Puede leer caracteres individuales de un archivo de texto a través de la función fgetc:

```
fgetc($filehandle):
```

Esta función devuelve el carácter leído. Es útil para tener control preciso sobre operaciones de lectura.

Éste es un ejemplo, phpfgetc.php. Se comienza abriendo el archivo para leer su contenido, file.txt:

```
<?php
  $handle = fopen("file.txt", "r");
  .
  .
  .
?>
```

Puede leer un carácter individual de file.txt de esta forma:

```
<?php
  $handle = fopen("file.txt", "r");

  ($char = fgetc($handle)) {
  .
  .
  .
  }
?>
```

Para recorrer todos los caracteres del archivo, puede colocar la instrucción anterior en la condición de un ciclo while; cuando fgetc devuelve FALSE, no hay más caracteres por leer:

```
<?php
  $handle = fopen("file.txt", "r");

  while ($char = fgetc($handle)) {
  .
  .
  .
  }
?>
```

Y reproducir con echo cada carácter conforme lo lee:

```
<?php
  $handle = fopen("file.txt", "r");

  while ($char = fgetc($handle)) {

    echo "$char";

  }
?>
```

FIGURA 9-3 Lectura de caracteres de un archivo

Los resultados están en la Figura 9-3, donde se abrió el archivo file.txt y luego se leyó su contenido, carácter por carácter.

Hasta donde llega está bien, pero ¿qué hay de los caracteres de línea nueva? El archivo file.txt, tiene este contenido:

```
Éstos
son
sus
datos.
```

Pero como verá la Figura 9-4, los caracteres de nueva línea del archivo se enviaron simplemente al navegador, que no muestra caracteres de nueva línea —debe convertirlos en elementos <br> en su lugar—. Esto se ve así en código:

```
<?php
  $handle = fopen("file.txt", "r");

  while ($char = fgetc($handle)) {
    if($char == "\n"){
      $char = "<br>";
    }

    echo "$char";
  }
?>
```

Todo lo que resta por hacer es cerrar el archivo con fclose en phpfgetc.php:

```
<html>
  <head>
    <title>
      Lectura de caracteres de un archivo
    </title>
  </head>

  <body>
    <h1>
       Lectura de caracteres de un archivo
    </h1>
      <?php
        $handle = fopen("file.txt", "r");

        while ($char = fgetc($handle)) {
          if($char == "\n"){
            $char = "<br>";
          }
          echo "$char";
        }

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

Puede ver los resultados en la Figura 9-4, donde se abrió el archivo file.txt, su contenido se leyó carácter por carácter; incluido manejo apropiado de líneas nuevas y luego se cerró.

FIGURA 9-4 Lectura de caracteres de un archivo y manejo de nuevas líneas

Lectura de un archivo completo con file_get_contents

Puede leer el contenido completo de un archivo con la función file_get_contents:

```
file_get_contents (filename [, use_include_path [, context [, offset [, maxlen]]]])
```

Aquí, *filename* es el nombre del archivo, *use_include_path* se establece a TRUE si desea buscar la ruta de inclusión de PHP, *context* es un contexto para la operación, *offset* es la bifurcación en el archivo desde el que se leerá y *maxlen* es la longitud máxima de los datos a leer.

Éste es el ejemplo, phpfilegetcontents.php. Aquí se comienza leyendo todo el contenido del archivo file.txt en la variable $text:

```
<?php
  $text = file_get_contents("file.txt");
  .
  .
  .
?>
```

Luego el código convierte todas las líneas nuevas en elementos <br> mediante la función str_replace de PHP:

```
<?php
  $text = file_get_contents("file.txt");
  $fixed_text = str_replace("\n", "<br>", $text);
  .
  .
  .
?>
```

Por último, el texto convertido se reproduce con echo en el navegador, de esta forma en phpfilegetcontents.php:

```
<html>
  <head>
    <title>
      Lectura de un archivo completo de una vez
    </title>
  </head>

  <body>
    <h1>
       Lectura de un archivo completo de una vez
    </h1>
      <?php
        $text = file_get_contents("file.txt");

        $fixed_text = str_replace("\n", "<br>", $text);
```

```
        echo $fixed_text;
      ?>
  </body>
</html>
```

Puede ver los resultados en la Figura 9-5, donde se abrió el archivo file.txt, se leyó todo su contenido y luego se mostró.

¿Desea leer una página Web completa de una sola vez? Tan sólo ábrala y léala, como aquí, donde leemos la página de inicio de PHP:

```
<html>
  <head>
    <title>
      Lectura de un archivo completo de una vez
    </title>
  </head>

  <body>
    <h1>
       Lectura de un archivo completo de una vez
    </h1>
      <?php
        $text = file_get_contents("http://www.php.net");

        $fixed_text = str_replace("\n", "<br>", $text);

        echo $fixed_text;
      ?>
  </body>
</html>
```

FIGURA 9-5 Lectura de un archivo completo de una vez

Lectura de un archivo en una matriz con file

Puede usar la función file para leer un archivo completo en una matriz de una sola vez; cada línea se convierte en elemento de la matriz. Así se usa la función file:

```
file (filename [, use_include_path [, context]] )
```

En este caso, *filename* es el nombre del archivo que desea leer, *use_include_path* debe establecerse en TRUE si desea buscar la ruta de inclusión del archivo en PHP y *context* es un contexto para la operación. Esta función devuelve una matriz o FALSE si fallara la operación.

Esta función es muy útil en caso de que quiera escribir sus archivos de bases de datos. Por ejemplo, suponga que deseara conservar las calificaciones de estudiantes en un archivo de base de datos —y pudiera leer todas esas calificaciones en una matriz con una única instrucción.

Éste es un ejemplo, phpfarray.php, que lee el archivo file.txt en una matriz, $data. Primero, cargaría el contenido de ese archivo en la matriz:

```
<?php
  $data = file('file.txt');
  .
  .
  .
?>
```

Eso es todo —ahora ha leído un archivo completo en la matriz $data. Ahora cada línea de file.txt es un elemento de la matriz $data.

Puede mostrar ahora los datos en la matriz con un ciclo foreach:

```
<?php
  $data = file('file.txt');

  foreach ($data as $number => $line) {
  .
  .
  .
  }
?>
```

Y todo lo que debe hacer es mostrar cada línea:

```
<?php
  $data = file('file.txt');

  foreach ($data as $line) {
     echo $line , "<br>";
  }
?>
```

De hecho, puede hacer más que eso; también mostrar números de línea. Así funciona en phpfarray.php:

```
<html>
  <head>
    <title>
      Lectura de un archivo en una matriz
    </title>
  </head>

  <body>
    <h1>
       Lectura de un archivo en una matriz
    </h1>
      <?php
        $data = file('file.txt');

        foreach ($data as $number => $line) {
           echo "Line $number: " , $line , "<br>";
        }
      ?>
  </body>
</html>
```

Y ver el resultado en la Figura 9-6, donde se leyó el archivo completo, file.txt, en una matriz y luego se mostró.

También puede abrir páginas Web y leerlas en matrices usando la función file. Este ejemplo, phpurlaray.php, lee http://www.php.net y la muestra:

```
<html>
  <head>
    <title>
      Lectura de una página Web en una matriz
    </title>
  </head>
```

FIGURA 9-6 Lectura de un archivo en una matriz

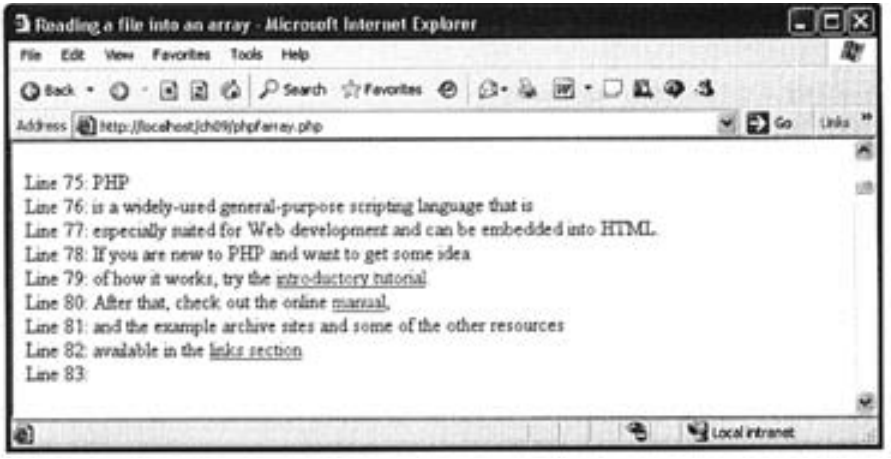

FIGURA 9-7 Lectura de una dirección URL en una matriz

```
  <body>
    <h1>
       Lectura de una página Web en una matriz
    </h1>
      <?php
        $data = file('http://www.php.net');

        foreach ($data as $number => $line) {
           echo "Line $number: " , $line , "<br>";
        }
      ?>
  </body>
</html>
```

Puede ver parte del resultado en la Figura 9-7, donde se leyó la dirección URL y mostró su código HTML, línea por línea.

Esto es algo a tomar en cuenta: cada elemento de la matriz tiene aún un carácter de nueva línea al final de éste. Si desea deshacerse de ese carácter de nueva línea, puede usar la función rtrim de PHP.

Comprobar si existe un archivo con file_exists

Si intenta trabajar con un archivo que no existe obtendrá un error. Para evitar eso puede comprobar si un archivo existe con la función file_exists:

```
file_exists (filename )
```

Se pasa un nombre de archivo a esta función (que puede incluir una ruta) y ésta devuelve TRUE, mientras el archivo exista y FALSE en caso contrario.

Éste es un ejemplo, phpfileexists.php. Comprueba un archivo que no existe, does_not_exist.txt:

```
<?php
  $filename = "does_not_exist.txt";
  .
  .
  .
?>
```

Entonces el código comprueba si el archivo existe:

```
<?php
  $filename = "does_not_exist.txt";

  if (file_exists($filename)) {
  .
  .
  .
  }
  .
  .
  .
?>
```

Si el archivo existe, puede leerlo y mostrar su contenido:

```
<?php
  $filename = "does_not_exist.txt";

  if (file_exists($filename)) {
    $data = file($filename);

    foreach ($data as $number => $line) {
      echo "Line $number: " , $line , "<br>";
    }
  }
  .
  .
  .
?>
```

Cuando no es así, puede mostrar un mensaje para ese efecto (que produce un mensaje de error de PHP):

```
<html>
  <head>
    <title>
      Comprobando si existe un archivo
    </title>
  </head>
```

FIGURA 9-8 Comprobando si existe un archivo

```
  <body>
    <h1>
       Comprobando si existe un archivo
    </h1>
      <?php
        $filename = "does_not_exist.txt";

        if (file_exists($filename)) {
          $data = file($filename);

          foreach ($data as $number => $line) {
            echo "Line $number: " , $line , "<br>";
          }
        }
        else {
          echo "El archivo $filename no existe";
        }
      ?>
  </body>
</html>
```

Puede ver el resultado en la Figura 9-8, donde este ejemplo determinó que el archivo no existía, antes de intentar abrirlo y, por tanto, no generó error.

Cálculo del tamaño del archivo con filesize

Puede lograr que la función filesize indique el tamaño de un archivo como un número entero:

```
filesize (filename )
```

Tan solo se pasa el nombre del archivo (incluyendo la ruta si desea agregarla) a filesize y obtendrá un entero o FALSE, en caso de que el archivo no exista.

FIGURA 9-9 Lectura de una dirección URL en una matriz

Este ejemplo, phpfilesize.php, muestra el tamaño de file.txt:

```
<html>
  <head>
    <title>
      Obtener el tamaño del archivo
    </title>
  </head>
  <body>
    <h1>
       Obtener el tamaño del archivo
    </h1>
      <?php
        echo "El archivo file.txt tiene ", filesize("file.txt"), " bytes de longitud.";
      ?>
  </body>
</html>
```

Puede ver el resultado en la Figura 9-9, donde este ejemplo determinó que el archivo tiene 24 bytes de longitud.

Ya ha obtenido mucha información sobre cómo manipular archivos de texto; ¿qué tal ahora archivos binarios?

Interpretación de lecturas binarias con fread

Hasta ahora, los archivos vistos aquí se han manipulado como archivos de texto, pero resulta simple manipular archivos en forma binaria y también usar funciones como fread:

```
fread (handle, length)
```

Esta función lee hasta *length* bytes del archivo referido por *handle*. La lectura se detiene cuando se han leído *length* bytes o cuando se llega al final del archivo (EOF).

En sistemas como Windows, debe abrir archivos para lectura binaria, modo 'rb', para trabajar con fread. Dado que agregar 'b' al modo no daña otros sistemas, lo incluiremos aquí para aumentar la portabilidad. Éste ejemplo, phpfread.php, trata a file.txt como archivo binario.

Este ejemplo comienza abriendo file.txt para lectura binaria:

```
<?php
  $handle = fopen("file.txt", "rb");
  .
  .
  .
?>
```

Puede usar filesize para determinar el tamaño del archivo y leerlo todo en una variable, $text, utilizando fread:

```
<?php
  $handle = fopen("file.txt", "rb");

  $text = fread($handle, filesize, filesize("file.txt"));
  .
  .
  .
?>
```

Ahora puede reemplazar líneas nuevas con elementos <br>:

```
<?php
  $handle = fopen("file.txt", "rb");

  $text = fread($handle, filesize, filesize("file.txt"));

  $fixed_text = str_replace("\n", "<br>", $text);
  .
  .
  .
?>
```

Y reproducir $text con echo, forzando a PHP para tratarlo como texto:

```
<?php
  $handle = fopen("file.txt", "rb");

  $text = fread($handle, filesize, filesize("file.txt"));

  $fixed_text = str_replace("\n", "<br>", $text);

  echo $fixed_text;
  .
  .
  .
?>
```

Todo lo que resta por hacer es cerrar el archivo, como se ve en phpfread.php:

```
<html>
  <head>
    <title>
      Lectura de datos binarios
    </title>
  </head>

  <body>
    <h1>
       Lectura de datos binarios
    </h1>
      <?php
        $handle = fopen("file.txt", "rb");

        $text = fread($handle, filesize("file.txt"));

        $fixed_text = str_replace("\n", "<br>", $text);

        echo $fixed_text;

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

Puede ver los resultados en la Figura 9-10, el ejemplo lee file.txt al usar una operación de lectura binaria.

FIGURA 9-10 Lectura de datos binarios

Análisis gramatical de archivos con fscanf

También puede analizar la gramática de archivos con la función fscanf:

```
fscanf (handle, format)
```

Esta función toma un manejador de archivos y una cadena de formato del mismo tipo que usaría con sprintf.

Este ejemplo, phpscanf.php, lee el archivo, actors.txt, donde nombres y apellidos de los actores están separados con tabuladores:

```
Cary  Grant
Myrna Loy
Jimmy Stewart
June  Allyson
```

Este ejemplo se inicia abriendo actors.txt:

```
<?php
  $handle = fopen("actors.txt", "r");
  .
  .
  .
?>
```

En tal caso el formato es "%s\t%s\n" (cadena, tabulador, cadena, carácter de nueva línea); así que de esta forma lee y analiza una línea de datos de un archivo, en una matriz llamada $name en un ciclo while:

```
<?php
  $handle = fopen("actors.txt", "r");

  while ($name = fscanf($handle, "%s\t%s\n")) {
  .
  .
  .
  }
?>
```

Después asigna los valores de la matriz $name a las variables $firstname y $lastname y luego muestra esos nombres:

```
<html>
  <head>
    <title>
      Análisis gramatical de archivos
    </title>
  </head>
  <body>
    <h1>
       Análisis gramatical de archivos
    </h1>
      <?php
        $handle = fopen("actors.txt", "r");
```

FIGURA 9-11 Análisis gramatical de datos del archivo

```
        while ($name = fscanf($handle, "%s\t%s\n")) {
          list ($firstname, $lastname) = $name;
          echo $firstname, " ", $lastname, "<br>";
        }

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

Puede ver el resultado en la Figura 9-11, donde se analizó el archivo actors.txt y se mostraron los datos contenidos.

Análisis gramatical de archivos ini con parse_ini_file

De forma muy similar a fscanf, parse_ini_file permite analizar archivos; en este caso, archivos de inicialización .ini. Así se utiliza parse_ini_file:

```
parse_ini_file (filename [, process_sections] )
```

Esta función se carga en el archivo ini *filename* y devuelve la configuración contenida en una matriz asociativa. Estableciendo el último parámetro process_sections a TRUE, se obtiene una matriz multidimensional, con nombres de sección y configuración incluidos. El valor predeterminado para process_sections es FALSE.

Observe que esta función no debe usarse con el archivo de inicialización empleado por PHP, php.ini. Este archivo se analiza y lee antes de que PHP siquiera se inicie.

ADVERTENCIA *Existen palabras reservadas que no deben usarse, como claves en archivos ini. Éstas incluyen; null, yes, no, true y false. Los valores null, no y false producen ""; yes y true producen "1". Los caracteres {}|&~![()" no se deben utilizar en cualquier parte de la clave.*

Por ejemplo, éste es un archivo .ini de muestra, sample.ini (observe que puede iniciar comentarios con puntos y comas en archivos .ini):

```
; Éste es un archivo .ini de muestra
[first_section]
first_color = rojo
second_color = blanco
third_color = azul
[second_section]
file = "/usr/local/code.data"
URL = "http://www.php.net"
```

Este ejemplo muestra cómo leer sample.ini, phpparseinifile.php. Puede leer el contenido de sample.ini en una matriz, $array:

```
<?php
  $array = parse_ini_file("sample.ini");
  .
  .
  .
?>
```

Ahora puede recuperar los valores del archivo .ini usando la matriz $array. Por ejemplo, recuperar el valor bajo la clave first_color (que es "rojo") de esta forma: $array["first_color"]. En este ejemplo, recorrería la matriz, mostrando claves y valores:

```
<html>
  <head>
    <title>
      Análisis gramatical de archivos .ini
    </title>
  </head>

  <body>
    <h1>
       Análisis gramatical de archivos .ini
    </h1>
      <?php
        $array = parse_ini_file("sample.ini");

        foreach ($array as $key => $value) {
          echo "$key => $value <br>";
        }
      ?>
  </body>
</html>
```

Puede ver el resultado en la Figura 9-12, donde se leyeron y mostraron los datos contenidos en sample.ini.

FIGURA 9-12 Análisis gramatical de un archivo .ini

Obtención de información de archivos con stat

La función stat proporciona información acerca de un archivo:

```
stat (filename)
```

La función devuelve una matriz conteniendo información con más sentido en sistemas Unix. Éstos son datos almacenados en la matriz con estos índices numéricos (también puede utilizar las claves de texto que se dan aquí entre paréntesis):

- **0 (dev)** Número de dispositivo
- **1 (ino)** Número inode
- **2 (mode)** Modo de protección inode
- **3 (nlink)** Número de vínculos
- **4 (uid)** Identificación de usuario del propietario
- **5 (gid)** Identificación de grupo del propietario
- **6 (rdev)** Tipo de dispositivo, si es un dispositivo inode
- **7 (size)** Tamaño en bytes
- **8 (atime)** Hora del último acceso (sello de hora de Unix)
- **9 (mtime)** Hora de la última modificación (sello de hora de Unix)
- **10 (ctime)** Hora del último cambio inode (sello de hora de Unix)
- **11 (blksize)** Tamaño de bloque de E/S del sistema de archivos
- **12 (blocks)** Número de bloques asignados

La mayoría de estos elementos sólo tienen significado en Unix; si no hay contrapartes en Windows, obtendrá un valor –1.

En este ejemplo ponemos tales conceptos a trabajar, phpstat.php, informa el tamaño del archivo file.txt. Comienza con el uso de stat en file.txt:

```
<?php
  $array = stat("file.txt");
  .
  .
  .
?>
```

Y encuentra el tamaño del archivo file.txt y mostrándolo de esta manera:

```
<html>
  <head>
    <title>
      Uso de stat
    </title>
  </head>

  <body>
    <h1>
       Uso de stat
    </h1>
      <?php
        $array = stat("file.txt");

        echo "El archivo tiene ", $array["size"]. " bytes de longitud.";
      ?>
  </body>
</html>
```

Puede ver el resultado en la Figura 9-13, donde aparece el tamaño del archivo.

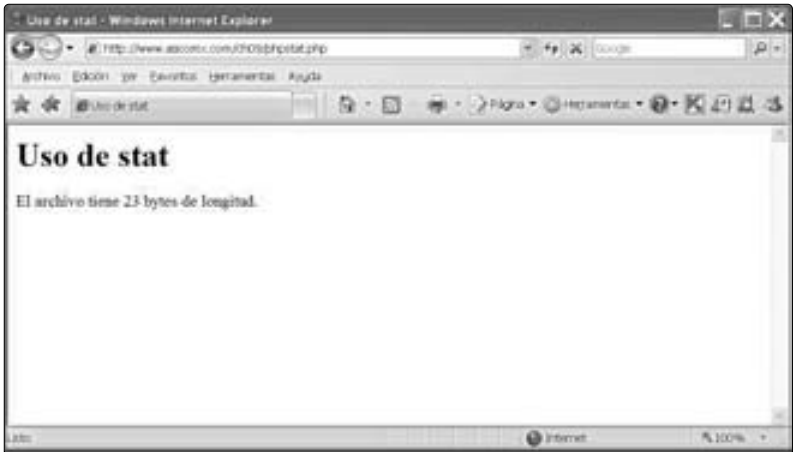

FIGURA 9-13 Cómo hallar el tamaño de un archivo

Cómo establecer la ubicación del puntero del archivo con fseek

PHP usa punteros de archivo para registrar dónde se encuentra en un archivo y desde dónde se lleva a cabo la siguiente operación de lectura o escritura. Por ejemplo, cuando abre un archivo, el puntero del archivo se sitúa en el principio del archivo; cuando abre un archivo para hacerle adiciones, el puntero del archivo se ubica en el final del archivo. Puede usar fseek para ubicar usted mismo el puntero del archivo:

```
fseek(handle, offset, [start_point]);
```

Aquí, *handle* es el manejador del archivo en que se ubicará el puntero del archivo, *offset* es el número de bytes en que desea establecer el puntero y *start_point* indica un punto de partida para el puntero, que es una de estas constantes:

- **SEEK_SET** El principio del archivo
- **SEEK_CUR** La ubicación actual del puntero
- **SEEK_END** El final del archivo

Puede establecer el valor offset en valores negativos.

Copia de archivos con copy

Puede copiar archivos con la función copy:

```
copy (source, destination)
```

Aquí, *source* es el nombre del archivo de origen y *destination* el nombre de la copia (incluyendo rutas de acceso, si se aplica). Esta función devuelve TRUE si tiene éxito y FALSE en caso contrario.

Este ejemplo, phpcopy.php, hará una copia del archivo file.txt, copy.txt:

```
<?php
  $file = 'file.txt';
  $copy = 'copy.txt';
  .
  .
  .
?>
```

Aquí intentamos hacer la copia e informamos del éxito si se logra:

```
<?php
  $file = 'file.txt';
  $copy = 'copy.txt';

  if (copy($file, $copy)) {
    echo "Se copió $file.";
  }
  .
  .
  .
?>
```

Si falla la operación de copia, podemos informar de esa falla así en phpcopy.php:

```
<html>
  <head>
    <title>
      Copia de archivos
    </title>
  </head>

  <body>
    <h1>
       Copia de archivos
    </h1>
      <?php
        $file = 'file.txt';
        $copy = 'copy.txt';

        if (copy($file, $copy)) {
          echo "Se copió $file.";
        }
        else {
          echo "No se pudo copiar $file.";
        }
      ?>
  </body>
</html>
```

Puede ver el resultado en la Figura 9-14, donde se copió el archivo según lo planeado.

NOTA *Si sólo desea mover un archivo, use entonces la función rename.*

FIGURA 9-14 Copia de archivos

Eliminación de archivos con unlink

¿Desea eliminar un archivo? Puede hacerlo con la función unlink:

```
unlink (filename [, context] )
```

Aquí, *filename* es el nombre del archivo y *context* es un contexto opcional. Esta función devuelve TRUE si se eliminó el archivo y FALSE en caso contrario.

Puede encontrar un ejemplo en phpunlink.php. Si el código logra eliminar copy.txt, se lo informa:

```
<?php
  if(unlink("copy.txt")){
    echo "Se eliminó el archivo.";
  }
  .
  .
  .
?>
```

Si no se pudo eliminar el archivo, este ejemplo, phpunlink.php, también lo indica:

```
<html>
  <head>
    <title>
      Eliminación de archivos
    </title>
  </head>

  <body>
    <h1>
       Eliminación de archivos
    </h1>
      <?php
        if(unlink("copy.txt")){
          echo "Se eliminó el archivo.";
        }
        else {
          echo "No se pudo eliminar el archivo.";
        }
      ?>
  </body>
</html>
```

Puede ver el resultado en la Figura 9-15, donde se eliminó el archivo.

FIGURA 9-15 Eliminación de un archivo

Cómo escribir en un archivo con fwrite

¿Qué sucedería si deseara escribir una cadena en un archivo? Podría utilizar fwrite:

```
fwrite (handle, string [. length])
```

Se pasa a fwrite un manejador de archivos, la cadena que va a escribir y, opcionalmente, la longitud máxima de datos que va a escribir. Esta función devuelve el número de bytes escritos o FALSE si hubo error.

Observe que configurar su sistema para escribir archivos puede requerir algo de trabajo por las protecciones del sistema de archivos implicadas. Por ejemplo, en Windows, debe hacer clic derecho en la carpeta donde desea escribir archivos, seleccionar Propiedades, hacer clic en la ficha Compartir, seleccionar el botón de radio Compartir esta carpeta y permitir la escritura. En Unix, asegúrese de tener privilegios de escritura en la carpeta en que desea escribir.

Este ejemplo, phpfwrite.php, escribirá texto en un archivo, data.txt. Se comienza abriendo el archivo para escritura:

```
<?php
  $handle = fopen("data.txt", "w");
  .
  .
  .
?>
```

Luego puede crear la cadena de texto a escribir en el archivo, $text. Observe que si desea incluir texto en múltiples líneas, debe agregar usted mismo los caracteres de nueva línea:

```
<?php
  $handle = fopen("data.txt", "w");
```

```
    $text = "Aquí\nestá\nel\ntexto.";
    .
    .
    .
  ?>
```

Es el momento de escribir el archivo con fwrite:

```
  <?php
    $handle = fopen("data.txt", "w");
    $text = "Aquí\nestá\nel\ntexto.";
    fwrite($handle, $text);
  ?>
```

Para mejorar esto, puede comprobar si ha fallado la operación de escritura, así, en phpfwrite.php:

```
  <?php
    $handle = fopen("data.txt", "w");

    $text = "Aquí\nestá\nel\ntexto.";

    if (fwrite($handle, $text) == FALSE) {
      echo "No se puede escribir data.txt.";
    }
  ?>
```

Si la operación fue exitosa, puede indicarlo al usuario de esta forma:

```
<html>
    <head>
        <title>
            Escritura de archivos
        </title>
    </head>
    <body>
      <h1>
        Escritura de archivos
      </h1>

      <?php
        $handle = fopen("data.txt", "w");

        $text = "Aquí\nestá\nel\ntexto.";

        if (fwrite($handle, $text) == FALSE) {
          echo "No se puede escribir data.txt.";
        }
        else {
          echo "Se creó data.txt.";
        }
```

FIGURA 9-16 Escritura de un archivo

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

Puede ver el resultado en la Figura 9-16, donde se escribió el archivo. Genial. Éste es el contenido de data.txt:

```
Éste
es
el
texto.
```

RECOMENDACIÓN *¿Desea comprobar si puede escribir en un archivo antes de si quiera intentarlo? Llame a la función is_writable, pasándole el nombre del archivo; esta función devolverá TRUE si puede escribir en el archivo y FALSE en caso contrario.*

Lectura y escritura de archivos binarios

Puede escribir datos binarios con fwrite y leerlos con fread, pero se requiere trabajo. Puede empacar datos binarios en cadenas usando la función pack y también desempacar datos binarios con la función unpack.

Este ejemplo, phpwritebinary.php, escribe el número 512 en formato binario (no de cadena). Comienza por abrir el archivo para escritura binaria de esta forma:

```
<?php
  $number = 512;
  $handle = fopen ("data.dat", "wb");
```

```
        .
        .
        .
    ?>
```

Después, empaca los datos en formato entero largo de esta forma:

```
pack ("L", $number);
```

Éstos son los formatos para la función pack, como "L" de entero largo:

- **a** Cadena rellena con NUL
- **A** Cadena rellena con SPACE
- **h** Cadena hexadecimal, primero el segmento bajo
- **H** Cadena hexadecimal, primero el segmento alto
- **c** Carácter con signo
- **C** Carácter sin signo
- **s** Corto con signo (siempre 16 bits, orden de bytes de máquina)
- **S** Corto sin signo (siempre 16 bits, orden de bytes de máquina)
- **n** Corto sin signo (siempre 16 bits, orden de bytes "big endian")
- **v** Corto sin signo (siempre 16 bits, orden de bytes "little endian")
- **i** Entero con signo (tamaño y orden de bytes dependientes de la máquina)
- **I** Entero sin signo (tamaño y orden de bytes dependientes de la máquina)
- **l** Largo con signo (siempre 32 bits, orden de bytes de máquina)
- **L** Largo sin signo (siempre 32 bits, orden de bytes de máquina)
- **N** Largo sin signo (siempre 32 bits, orden de bytes "big endian")
- **V** Largo sin signo (siempre 32 bits, orden de bytes "little endian")
- **f** Flotante (tamaño y representación dependientes de la máquina)
- **d** Doble ((tamaño y representación dependientes de la máquina)
- **x** Byte NUL
- **X** Copia de seguridad de un byte
- **@** Rellenar con NUL hasta la posición absoluta

Así es como este ejemplo escribe los datos en el archivo:

```
<?php
  $number = 512;
  $handle = fopen ("data.dat", "wb");
  if (fwrite ($handle, pack ("L", $number)) == FALSE) {
    .
    .
    .
  }
?>
```

Y, por último, este código indica éxito o fracaso al usuario, de esta forma en phpwritebinary.php:

```
<html>
  <head>
    <title>
      Escritura de archivos binarios
    </title>
  </head>
  <body>
    <h1>
      Escritura de archivos binarios
    </h1>

    <?php
      $number = 512;
      $handle = fopen ("data.dat", "wb");
      if (fwrite ($handle, pack ("L", $number)) == FALSE) {
        echo "No se puede escribir data.dat.";
      }
      else {
        echo "Se creó data.dat. y se almacenó $number.";
      }

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

Puede ver el resultado en la Figura 9-17, donde se escribió el archivo binario.

FIGURA 9-17 Escritura de un archivo binario

Para leer los datos binarios del archivo, puede usar la función unpack en phpreadbinary.php. Primero debe abrir el archivo para lectura binaria:

```
<?php
  $handle = fopen ("data.dat", "rb");
    .
    .
    .
?>
```

Luego use fread para leer los datos binarios, indicando que desea cuatro bytes (la longitud de un entero largo):

```
<?php
  $handle = fopen ("data.dat", "rb");
  $data = fread ($handle, 4);
    .
    .
    .
?>
```

Luego puede usar unpack, para desempacar los datos en una matriz con un elemento bajo el índice "data", conteniendo un valor largo:

```
<?php
  $handle = fopen ("data.dat", "rb");
  $data = fread ($handle, 4);
  $array = unpack ("Ldata", $data);
    .
    .
    .
?>
```

Ahora puede recuperar los datos binarios de la matriz mediante la clave "data":

```
<?php
  $handle = fopen ("data.dat", "rb");
  $data = fread ($handle, 4);
  $array = unpack ("Ldata", $data);
  $data = $array["data"];
    .
    .
    .
?>
```

Todo lo que resta por hacer es mostrar esos datos, en phpreadbinary.php:

```
<html>
  <head>
    <title>
      Lectura de archivos binarios
    </title>
  </head>
```

FIGURA 9-18 Lectura de datos de un archivo binario

```
  <body>
    <h1>
      Lectura de archivos binarios
    </h1>

    <?php
      $handle = fopen ("data.dat", "rb");
      $data = fread ($handle, 4);
      $array = unpack ("Ldata", $data);
      $data = $array["data"];
      echo "Lea este valor de data.dat: ", $data;
    ?>
  </body>
</html>
```

Puede ver el resultado en la Figura 9-18, donde se leyeron los datos binarios del archivo data.dat apropiadamente.

Anexión de información a archivos con fwrite

También puede anexar datos a archivos mediante fwrite, si abre el archivo para realizar una anexión explícita. Este ejemplo, phpappend.php, anexará texto al archivo data.txt. Comienza por abrir el archivo data.txt para hacer la anexión:

```
<?php
  $handle = fopen("data.txt", "a");
    .
    .
    .
?>
```

Y reúne el texto que desea anexar al archivo, que es

```
Aquí
hay
más
texto.
```

Puede colocar este texto en una variable, $text:

```
<?php
  $handle = fopen("data.txt", "a");

  $text = "\nAquí\nhay\nmás\ntexto.";
    .
    .
    .
?>
```

Ahora debe usar fwrite para anexar texto a data.txt (y hacerle saber al usuario si falló la operación):

```
<?php
  $handle = fopen("data.txt", "a");

  $text = "\nAquí\nhay\nmás\ntexto.";

  if (fwrite($handle, $text) == FALSE) {
    echo "No se puede anexar texto a data.txt.";
  }
    .
    .
    .
?>
```

Por otra parte, si la operación de anexión fue exitosa, puede hacer saber eso al usuario de esta forma, en phpappend.php:

```
<html>
  <head>
    <title>
      Anexión de texto a archivos
    </title>
  </head>
  <body>
    <h1>
      Anexión de texto a archivos
    </h1>

    <?php
      $handle = fopen("data.txt", "a");

      $text = "\nAquí\nhay\nmás\ntexto.";
```

```
      if (fwrite($handle, $text) == FALSE) {
        echo "No se puede anexar texto a data.txt.";
      }
      else {
        echo "Se anexó texto a data.txt.";
      }

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

Puede ver el resultado en la Figura 9-19, donde se hizo una anexión al archivo. Éste es el nuevo contenido de data.txt:

```
Éste
es
el
texto.
Aquí
hay
más
texto.
```

Genial.

RECOMENDACIÓN *¿Desea hacer lo opuesto y truncar el archivo? Use ftruncate, especificando el nuevo tamaño del archivo.*

```
ftruncate (handle, size)
```

FIGURA 9-19 Anexión de texto a un archivo

Escritura en todo el contenido del archivo con file_put_contents

Existe un atajo, si desea escribir texto en un archivo (utilice la función file_put_contents). Esta función escribe una cadena en un archivo y así se utiliza en general:

```
file_put_contents (filename, data [, flags [, context]])
```

En este caso, *filename* es el nombre del archivo en que desea escribir, *data* es la cadena de texto que desea escribir, *flags* puede ser FILE_USE_INCLUDE_PATH o FILE_APPEND y *context* es el contexto de un archivo. La función devuelve el número de bytes escritos en el archivo o FALSE, si no pudo escribir en el archivo.

Usar esta función es muy semejante a llamar funciones fopen, fwrite y fclose automáticamente —no tiene por qué abrir o cerrar el archivo usted mismo ni necesita un manejador de archivos—. Éste es un ejemplo, phpfileputcontents.php. Se inicia con el texto que desea escribir en el archivo:

```
<?php
  $text = "Éste\nes\nel\ntexto.";
    .
    .
    .
?>
```

A continuación, puede usar file_put_contents para escribir en el archivo data.txt:

```
<?php
  $text = "Éste\nes\nel\ntexto.";

  file_put_contents("data.txt", $text);
    .
    .
    .
?>
```

Puede mejorar esto, como ya ha visto antes, comprobando el valor de retorno de la función. Si hubo un problema, puede hacérselo saber al usuario:

```
<?php
  $text = "Éste\nes\nel\ntexto.";

  if (file_put_contents("data.txt", $text) == FALSE) {
    echo "No se puede escribir en data.txt.";
  }
    .
    .
    .
?>
```

En su defecto, puede hacerle saber al usuario que la operación fue un éxito, de esta forma en phpfileputcontents.php:

```
<html>
  <head>
    <title>
      Escritura en archivos con file_put_contents
    </title>
  </head>
  <body>
    <h1>
      Escritura en archivos con file_put_contents
    </h1>

    <?php
      $text = "Éste\nes\nel\ntexto.";

      if (file_put_contents("data.txt", $text) == FALSE) {
        echo "No se puede escribir en data.txt.";
      }
      else {
        echo "Se escribió en data.txt.";
      }
    ?>
  </body>
</html>
```

Observe que no necesita cerrar el archivo (o abrirlo, para ese efecto). El resultado está en la Figura 9-20, donde se creó el archivo.

FIGURA 9-20 Escritura en un archivo con file_put_contents

Bloqueo de archivos

En un ambiente con múltiples usuarios, como un servidor Web, es posible que múltiples usuarios accedan a sus scripts simultáneamente —que significa múltiples copias del mismo script operando al mismo tiempo—. Si su script o scripts tienen acceso a archivos, puede haber un conflicto cuando dos scripts o copias del mismo script, intenten escribir en el mismo archivo simultáneamente. Para corregir eso, use la función de bloqueo de archivos, flock:

```
flock (handle, operation [, &wouldblock] )
```

Aquí, *handle* es el manejador del archivo que desea bloquear y *operation* es uno de éstos:

- Para adquirir un bloqueo compartido (lector), establezca operation a LOCK_SH.
- Para adquirir un bloqueo exclusivo (escritor), establezca operation a LOCK_EX.
- Para liberar un bloqueo (compartido o exclusivo), establezca operation a LOCK_UN.

Y el tercer argumento opcional se establece a TRUE, si el bloqueo se llega a establecer.

Esta función devuelve TRUE, cuando se efectúa el bloqueo y FALSE en caso contrario.

Se logra un bloqueo en archivos de forma notificativa —todo su código para acceso a archivos debe comprobar si puede lograr un bloqueo en un archivo antes de tener acceso a éste—. Si su código no puede lograr un bloqueo en un archivo, algún otro código está usando el archivo y su código presente debe esperar. Los bloqueos notifican (salvo en Windows, donde son obligatorios), lo que significa que otro código puede trabajar con archivos bloqueados; y posiblemente provoque un desastre; así que debe estar seguro de lograr un bloqueo antes de escribir en un archivo. Cuando termine, desbloquee el archivo para dar a otro código acceso a éste.

La función flock normalmente bloquea (es decir, no retorna) hasta que se pueda asegurar un bloqueo. Si no desea esperar; y posiblemente dejar su código bloqueado; establezca con OR la constante LOCK_NB a la operación, como: LOCK_EX | LOCK_NB. Eso hará que la función flock retorne de inmediato y, si no consiguió un bloqueo, puede esperar un segundo (con la función sleep de PHP de esta forma: sleep(1), que obliga una pausa de un segundo en el código) y reintentarlo, con un límite de 15 intentos.

Este ejemplo, phpflock.php, bloquea el archivo data.txt antes de escribir en él. Puede abrir el archivo e intentar un bloqueo de esta forma:

```
<?php
  $handle = fopen("data.txt", "w");

  $text = "Éste\nes\nel\ntexto.";

  if (flock($handle, LOCK_EX | LOCK_NB)) {
     .
     .
     .
?>
```

Si consigue el bloqueo, puede escribir en el archivo y luego desbloquearlo:

```
<?php
  $handle = fopen("data.txt", "w");
```

```
    $text = "Éste\nes\nel\ntexto.";

    if (flock($handle, LOCK_EX | LOCK_NB)) {
      echo "Se bloqueó el archivo. <br>";
      if (fwrite($handle, $text) == FALSE) {
        echo "No se puede escribir en data.txt. <br>";
      }
      else {
        echo "Se escribió en data.txt.<br>";
      }
      flock($handle, LOCK_UN);
      echo "Se desbloqueó el archivo. <br>";
    }
  ?>
```

Este código sólo intenta bloquear el archivo una vez y muestra de inmediato un error si no puede hacerlo, pero usted podría repetir la operación 15 o 20 segundos, intentando continuamente de bloquear el archivo hasta que se diera por vencido.

Si no pudiera obtener el bloqueo, otro código está usando el archivo y debe hacer saber esto al usuario de esta forma en phpflock.php:

```
<html>
  <head>
    <title>
      Bloqueo y desbloqueo de archivos
    </title>
  </head>
  <body>
    <h1>
      Bloqueo y desbloqueo de archivos
    </h1>

    <?php
      $handle = fopen("data.txt", "w");

      $text = "Éste\nes\nel\ntexto.";

      if (flock($handle, LOCK_EX | LOCK_NB)) {
        echo "Se bloqueó el archivo. <br>";
        if (fwrite($handle, $text) == FALSE) {
          echo "No se puede escribir en data.txt. <br>";
        }
        else {
          echo "Se escribió en data.txt. <br>";
        }
        flock($handle, LOCK_UN);
        echo "Se desbloqueó el archivo. <br>";
      }
      else {
        echo "No se pudo bloquear el archivo. <br>";
      }
```

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

Eso es todo; puede ver el resultado en la Figura 9-21, donde se bloqueó el archivo, se escribió en él y se desbloqueó. Fenomenal.

Tan sólo tenga en mente que si no puede bloquear un archivo, lo más probable es que otro código lo esté utilizando y que debe intentarlo de nuevo más tarde.

FIGURA 9-21 Bloqueo y escritura en un archivo

10
CAPÍTULO

Trabajo con bases de datos

La conexión de PHP a bases de datos en el servidor es natural y en este capítulo analizamos dicha conexión. Verá cómo conectar PHP con tablas de bases de datos y similares en este capítulo, actualizando los datos según desee usted a esa base de datos.

PHP integra amplio soporte para bases de datos integradas y eso es bueno, pues una de las formas más populares para trabajar con PHP consiste en la manipulación de bases de datos en el servidor. Puede ver las bases de datos admitidas por PHP en la Tabla 10-1.

Si desea usar el soporte integrado de PHP para los diferentes servidores de bases de datos descritos en la Tabla 10-1, puede hallar manuales para ellos en www.php.net/*dbname*, donde *dbname* es el nombre de la base de datos, como mysql, sybase, mssql, etc. Para ODBC, utilice el nombre uodbc; para Oracle, use oci8.

Por mucho, el sistema de bases de datos de uso más frecuente con PHP es MySQL; de este modo el presente capítulo se centra en MySQL, que se puede obtener sin costo en www.mysql.com, aunque hablaremos también de otros servidores de bases de datos. Todas estas bases de datos son compatibles con PHP y la única diferencia entre ellas, en su mayor parte, es la forma en que se conecta con ellas —apenas sepa cómo trabajar con MySQL, el proceso se extiende fácilmente a otras bases de datos.

Adabas	Ingres	Oracle
dBase	InterBase	Ovrimos
Empress	FrontBase	PstgreSQL
FilePro	mSQL	Solid
Hyperwave Direct	MS-SQL	Sybase
IBM DB2	MySQL	Velocis
Informix	ODBC	Unix dbm

TABLA 10-1 Bases de datos soportadas por PHP

¿Qué es una base de datos?

Así pues, ¿qué es una base de datos? Daremos un vistazo a lo que comprende una base de datos de manera breve (y si ya está familiarizado con las bases de datos, tablas, etc., naturalmente puede saltarse esta introducción).

Las bases de datos organizan información para facilitar acceso y uso por parte de programas. La construcción más popular de una base de datos es la tabla y aquí daremos un vistazo a ellas. Suponga, por ejemplo, que enseña PHP a un grupo de estudiantes y desea llevar registro de sus calificaciones. Podría crear una tabla con dos columnas, Nombre y Calificación:

```
Nombre  Calificación
```

Puede almacenar el nombre del primer estudiante en la columna Nombre, de esta forma:

```
Nombre  Calificación
Ana
```

Esto crea una entrada en la tabla para Ana (es decir, una nueva fila). Cada fila de una tabla de base de datos es un *registro*, y este corresponde a la estudiante llamada Ana. Cada columna de un registro se conoce como *campo* y ha dado al campo Nombre el valor "Ana". De forma similar, es posible asignar a Ana una calificación en el campo Calificación:

```
Nombre     Calificación
Ana        C
```

Y puede agregar también registros para otros estudiantes:

```
Nombre     Calificación
Ana        C
Marco      B
Eduardo    A
Francisco  A
Teodoro    A
Marcela    B
Rafael     B
Tomás      B
```

Entonces, ¿qué es una base de datos? En su forma más convencional, una base de datos es sólo una colección de una o más tablas. Y para tener acceso a los datos contenidos en esas tablas se utiliza SQL en PHP, que veremos en el tema siguiente.

Algunos elementos esenciales de SQL

Para interactuar con bases de datos en PHP se usa el Structured Query Language, SQL (lenguaje de consulta estructurado). Aquí analizaremos algunos elementos de SQL —observe que el lenguaje SQL completo escapa al contenido de este libro, centrado sólo en la conexión con bases de datos y otras cosas de PHP—; los elementos de SQL empleados por usted para trabajar con sus bases de datos son decisión personal.

Por ejemplo, suponga que tiene una tabla de base de datos llamada frutas y desea recuperar los registros de esa tabla. Esto es lo que podría contener esa tabla ahora mismo:

```
Nombre      Número
manzanas    1020
naranjas    3329
plátanos    8582
peras       235
```

Para trabajar con esta tabla en su código, puede ejecutar una instrucción de SQL, llamada *consulta de SQL*, en la tabla. Esta consulta devolverá todos los registros de la tabla:

```
SELECT * FROM frutas
```

La ejecución de esta consulta devuelve un conjunto de registros conteniendo todos los registros coincidentes de la tabla de frutas. Como especificó que desea hallar coincidencias con el comodín * aquí, la consulta devuelve todos los registros de la tabla de frutas. En PHP, ese conjunto de registros aparece como una matriz y puede recorrerla en ciclos de maneras que verá en este capítulo.

¿Cómo se aprecia esto en PHP? ¿Cómo se ejecuta realmente una consulta SQL en PHP? Para interactuar con MySQL, utilizaría la función mysql_query para ejecutar dicha consulta de SQL, algo como esto, devolviendo la tabla de frutas completa, almacenándose en $result:

```
$query = "SELECT * FROM frutas";
$result = mysql_query($query) or die("Falló la consulta: " . mysql_error());
```

Ahora $result aloja una matriz con registros de la tabla de frutas y accediendo a los campos de cada registro por nombre.

¿Qué tal si vemos un poco más de SQL? Puede seleccionar también campos específicos de una tabla de esta forma, donde seleccionamos los campos nombre y número de la tabla de frutas:

```
SELECT nombre, número FROM frutas
```

Usando la cláusula WHERE, puede configurar criterios de selección que los registros del conjunto generado por la consulta deben cumplir. Por ejemplo, para seleccionar todos los registros de la tabla de frutas, donde el campo nombre es igual a manzanas, puede ejecutar esta instrucción: SELECT * FROM frutas WHERE name=“manzanas”.

No es necesario usar un signo igual aquí; puede probar campos empleando estos operadores:

- < (menor que)
- <= (menor que o igual a)
- > (mayor que)
- >= (mayor que o igual a)

Puede usar una cláusula IN para especificar un conjunto de valores que los campos sean capaces de igualar. Por ejemplo, así es como se pueden recuperar registros con valores en el campo nombre, coincidentes con manzanas o naranjas:

```
SELECT * FROM frutas WHERE nombre IN ("manzanas", "naranjas")
```

También puede utilizar operaciones lógicas en las cláusulas de sus instrucciones SQL. Éste es un ejemplo donde especificamos dos criterios: el campo nombre debe contener"manzanas"o "naranjas"y debe haber algún valor en el campo número. Se utiliza la palabra clave NULL, para probar si hay algo en un campo:

```
SELECT * FROM frutas WHERE nombre NOT IN ("manzanas", "naranjas") AND número IS NOT
NULL
```

Puede usar estos operadores lógicos para conectar cláusulas: AND, OR y NOT. El uso de AND significa que ambas cláusulas deben ser verdaderas; OR involucra que una u otra puede ser verdadera, mientras NOT alterna el valor de una cláusula TRUE a FALSE o de FALSE a TRUE.

Como podría esperar, también podría ordenar los el conjunto registros que produce una instrucción de SQL. En este es ejemplo ordenamos los registros en la tabla de frutas, usando el campo nombre:

```
SELECT * FROM frutas ORDER BY nombre
```

También puede clasificar registros en orden descendente con la palabra clave DESC:

```
SELECT * FROM frutas ORDER BY nombre DESC
```

Puede usar la instrucción DELETE para eliminar registros de esta forma, donde suprimimos todos los registros de la tabla frutas con los valores de nombre que no son manzanas o naranjas:

```
DELETE * FROM frutas WHERE nombre NOT IN ("manzanas", "naranjas")
```

Se usa la instrucción UPDATE para actualizar una base de datos cuando desea hacer cambios. Por ejemplo, así se cambia el valor del campo número, en el registro conteniendo el número de manzanas:

```
UPDATE frutas SET número = "2006" WHERE nombre = "manzanas"
```

Puede insertar también datos nuevos en una tabla. Éste es un ejemplo introduce una nueva fila en la tabla frutas:

```
INSERT INTO frutas (nombre, número) VALUES('chabacanos', '203')
```

Bueno, hemos aprendido tantos elementos de SQL como necesitaremos aquí. El paso siguiente es crear una base de datos para trabajar con ella en PHP.

Creación de una base de datos MySQL

Utilizamos bases de datos MySQL en este capítulo y, para tener algo con qué trabajar en código, ahora vamos a crear una base de datos en MySQL. Puede adquirir MySQL sin costo en www.mysql.com (y, de hecho, su computadora podría tenerlo ya instalado). Para comprobar si ya está instalado, pruebe esto en una ventana de comandos (recuerde que % representa la línea de comandos genérica en este libro):

```
%mysql
```

Si ve una respuesta de MySQL, ¡felicidades!, ya lo tiene instalado. En caso contrario, necesitará descargarlo e instalarlo. De hecho, MySQL solía venir con PHP ya no, por asuntos de licencias.

Dependiendo de su sistema y la versión de MySQL, quizás necesite arrancar el servidor MySQL antes de trabajar con el producto. Puede iniciar el servidor de MySQL con esta línea de comandos:

```
%mysqld --console
```

En algunos sistemas, como Windows con versiones recientes de MySQL, no necesita iniciar el servidor de MySQL en absoluto (ya se está ejecutando). En ese caso, obtendrá un error si intenta iniciar el servidor MySQL por segunda vez; el error tendrá que ver con recursos compartidos.

Después, debe activar una sesión de MySQL que se conectará al servidor y usarlo para crear su propia base de datos. Necesita nombre de usuario y contraseña para trabajar con MySQL; por ejemplo, su nombre de usuario es"usuario"y su contraseña es"contraseña". Puede iniciar MySQL de esta forma en la línea de comandos:

```
%mysql –u usuario –p
```

Esto le pedirá ingresar su contraseña:

```
%mysql –u usuario –p
Enter password: *********
```

Una vez que haya iniciado una sesión, verá una respuesta parecida a ésta:

```
%mysql –u usuario –p
Enter password: *********
Welcome to the MySQL monitor.  Commands end with ; or \g.
mysql>
```

La última línea, mysql>, solicita ingresar comandos.

Si no tiene un nombre de usuario ni contraseña, escriba **mysql –u root** o simplemente **mysql**, para comenzar:

```
%mysql –u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
mysql>
```

Para comenzar, escriba **SELECT VERSION( ), CURRENT_DATE;** para confirmar que MySQL está funcionando:

```
mysql> SELECT VERSION(), CURRENT_DATE;
+-------------+----------------+
|  VERSION()  |  CURRENT_DATE  |
+-------------+----------------+
|  5.0.19-nt  |  2007-05-10    |
+-------------+----------------+
1 row in set (0.01 sec)
```

Este comando indica la versión de MySQL y fecha actual. Observe que los comandos de MySQL, como éste, terminan con punto y coma.

MySQL viene con algunas bases de datos integradas (que puede comprobar con el comando SHOW DATABASES;:

```
mysql> SHOW DATABASES;
+------------------------+
| Database               |
+------------------------+
| mysql                  |
| test                   |
+------------------------+
2 rows in set (0.08 sec)
```

Estas bases de datos ya existen en MySQL (mysql es una base de datos usado MySQL, para administración interna y test es una base de datos para pruebas).

Las tablas de bases de datos se almacenan en bases de datos; así que el primer paso consiste en crear una base de datos. Podría utilizar ésta para alojar información acerca de diversas frutas y vegetales, por ejemplo, y de tal modo crear una base de datos llamada, por ejemplo, produce. Puede crear la base de datos produce con el comando CREATE DATABASE en el monitor de MySQL:

```
mysql> CREATE DATABASE produce;
Query OK, 1 row affected (0.01 sec)
```

Eso crea una base de datos nueva, vacía, que puede examinar con el comando SHOW DATABASES;:

```
mysql> CREATE DATABASE produce;
Query OK, 1 row affected (0.01 sec)

mysql> SHOW DATABASES;
+------------------------+
| Database               |
+------------------------+
| mysql                  |
| test                   |
| produce                |
+------------------------+
3 rows in set (0.08 sec)
```

Ahora haga que la base de datos produce sea la predeterminada, mediante el comando USE, de esta forma en MySQL:

```
mysql> CREATE DATABASE produce;
Query OK, 1 row affected (0.01 sec)

mysql> SHOW DATABASES;
+------------------------+
| Database               |
+------------------------+
| mysql                  |
| test                   |
| produce                |
+------------------------+
3 rows in set (0.08 sec)
```

```
mysql> USE produce
Database changed
```

¿Existen tablas en esta nueva base de datos? Puede usar el comando SHOW TABLES para comprobar eso:

```
mysql> USE produce
Database changed
mysql> SHOW TABLES;
Empty set (0.01 sec)
```

Esta respuesta ("Empty set") indica que la base de datos no contiene tablas aún.

Creación de una nueva tabla

Puede cambiar eso generando una tabla nueva, por ejemplo: frutas, conteniendo diferentes frutas. Para crear una tabla de base de datos, debe crear los diferentes campos de esa tabla, que significa establecer su formato de datos. Éstos son algunos de los formatos de datos más populares:

- **VARCHAR(*length*)** Crea una cadena de longitud variable
- **INT** Crea un entero
- **DECIMAL(totaldigits, decimalplaces)** Crea un valor decimal
- **DATETIME** Crea un objeto de fecha y hora, como 2008-11-15 20:00:00

Los registros de la tabla frutas contendrán cadenas de 20 caracteres (el nombre de una fruta y número de unidades que se tiene de esa fruta). Así se crea la tabla de frutas en el monitor de MySQL:

```
mysql> CREATE TABLE frutas (name VARCHAR(20), number VARCHAR(20));
Query OK, 0 rows affected (0.13 sec)
```

Ello creará la tabla de frutas, que puede comprobar con el comando SHOW TABLES; de esta forma:

```
mysql> CREATE TABLE frutas (name VARCHAR(20), number VARCHAR(20));
Query OK, 0 rows affected (0.13 sec)

mysql> SHOW TABLES;
+--------------------+
| Tables_in_produce  |
+--------------------+
| frutas             |
+--------------------+
1 row in set (0.00 sec)
```

De hecho, puede comprobar esta tabla, obteniendo el formato de sus campos con el comando DESCRIBE de SQL de esta forma:

```
mysql> CREATE TABLE frutas (name VARCHAR(20), number VARCHAR(20));
Query OK, 0 rows affected (0.13 sec)

mysql> SHOW TABLES;
+--------------------+
| Tables_in_produce  |
+--------------------+
| frutas             |
+--------------------+
1 row in set (0.00 sec)

mysql> DESCRIBE frutas;
+--------+-------------+------+------+----------+---------+
| Field  | Type        | Null | Key  | Default  | Extra   |
+--------+-------------+------+------+----------+---------+
| name   | varchar(20) | YES  |      | NULL     |         |
| number | varchar(20) | YES  |      | NULL     |         |
+--------+-------------+------+------+----------+---------+
2 rows in set (0.01 sec)
```

Genial (ahora tiene una nueva tabla de base de datos, frutas). Es tiempo de añadirle datos.

Colocación de datos en la nueva base de datos

Suponga que lleva el registro del inventario de frutas de su tienda de abarrotes local. Usted cuenta estas frutas:

- manzanas 1020
- naranjas 3329
- plátanos 8582
- peras 235

Bueno, ¿qué tal si almacenamos esas cifras en la nueva tabla de la base de datos de frutas? Tiene dos campos en cada registro de la tabla de frutas, como se aprecia en la descripción de MySQL de la tabla:

```
mysql> DESCRIBE frutas;
+--------+-------------+------+------+----------+---------+
| Field  | Type        | Null | Key  | Default  | Extra   |
+--------+-------------+------+------+----------+---------+
| name   | varchar(20) | YES  |      | NULL     |         |
| number | varchar(20) | YES  |      | NULL     |         |
+--------+-------------+------+------+----------+---------+
2 rows in set (0.01 sec)
```

Ahora puede insertar dos cadenas en el registro de cada tipo de fruta (nombre de la fruta y número de unidades de la fruta). Puede crear registros en la tabla de frutas con el comando INSERT de SQL. Por ejemplo, para ingresar un registro del número de manzanas en existencia, usted ejecutaría este comando:

```
mysql> INSERT INTO frutas VALUES ('manzanas', '1020');
Query OK, 1 row affected (0.00 sec)
```

Formidable, así genera un nuevo registro para las manzanas. Puede comprobar ese registro con el comando SELECT * FROM frutas;:

```
mysql> INSERT INTO frutas VALUES ('manzanas', '1020');
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM frutas;
+----------+--------+
| name     | number |
+----------+--------+
| manzanas | 1020   |
+----------+--------+
1 row in set (0.00 sec)
```

Como verá, hay un nuevo registro para manzanas aquí.

Así es como se agregan las otras frutas con instrucciones INSERT:

```
mysql> INSERT INTO frutas VALUES ('manzanas', '1020');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO frutas VALUES ('naranjas', '3329');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO frutas VALUES ('plátanos', '8582');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO frutas VALUES ('peras', '235');
Query OK, 1 row affected (0.00 sec)
```

Y comprobar la nueva tabla de frutas, así como su contenido con SELECT * FROM frutas; de la siguiente forma:

```
mysql> SELECT * FROM frutas;
+----------+---------+
| name     | number  |
+----------+---------+
| manzanas | 1020    |
| naranjas | 3329    |
| plátanos | 8582    |
| peras    | 235     |
+----------+---------+
4 rows in set (0.00 sec)
```

Ello crea una base de datos, produce, y una tabla en la base de datos, frutas. Puede abandonar el monitor de MySQL de esta forma:

```
mysql> SELECT * FROM frutas;
+----------+---------+
| name     | number  |
+----------+---------+
| manzanas | 1020    |
| naranjas | 3329    |
| plátanos | 8582    |
| peras    | 235     |
+----------+---------+
4 rows in set (0.00 sec)
mysql>quit
```

Eso crea y cierra su base de datos (es momento de tener acceso a ella en PHP).

Acceso a la base de datos en PHP

Cuando instala PHP, puede seleccionar entre varias extensiones. Para instalar soporte para MySQL, haga clic en el nodo Extensions, del instalador para abrir ese nodo y elegir el nodo MySQL, a fin de instalar dicha extensión. Es posible que su instalación de PHP se haya realizado con soporte para MySQL (la mayoría de instalaciones en servidores Web son así).

El soporte para MySQL en PHP consta de varias funciones a las que puede llamar para interactuar con MySQL y son:

- **mysql_affected_rows** Indica número de filas afectadas por la operación anterior de MySQL.
- **mysql_change_user** Cambiar al usuario que inició sesión.
- **mysql_client_encoding** Devuelve el nombre del conjunto de caracteres actual.
- **mysql_close** Cierra una conexión MySQL.
- **mysql_connect** Abre una conexión con un servidor MySQL.
- **mysql_create_db** Crea una base de datos MySQL.
- **mysql_data_seek** Busca información en la base de datos.
- **mysql_db_name** Indica nombre de la base de datos.
- **mysql_db_query** Envía una consulta MySQL.
- **mysql_drop_db** Desecha (es decir, elimina) una base de datos MySQL.
- **mysql_error** Devuelve el texto del mensaje de error de la operación anterior de MySQL.
- **mysql_fetch_array** Captura una fila de resultado como matriz asociativa, matriz numérica o ambas.
- **mysql_fetch_assoc** Captura una fila de resultado como matriz asociativa.

- **mysql_fetch_row** Obtiene una fila de resultado como matriz enumerada.
- **mysql_field_len** Devuelve la longitud de un campo dado.
- **mysql_field_name** Obtiene el nombre del campo dado en un resultado.
- **mysql_field_seek** Busca hasta una bifurcación de campo determinada.
- **mysql_field_table** Obtiene el nombre de la tabla en que está el campo determinado.
- **mysql_field_type** Obtiene el tipo del campo dado en un resultado.
- **mysql_get_server_info** Obtiene información del servidor MySQL.
- **mysql_info** Obtiene información acerca de la consulta más reciente.
- **mysql_list_dbs** Lista las bases de datos disponibles en un servidor MySQL.
- **mysql_list_fields** Lista campos de una tabla MySQL.
- **mysql_list_tables** Lista las tablas en una base de datos MySQL.
- **mysql_num_fields** Obtiene número de campos en el resultado.
- **mysql_num_rows** Obtiene número de filas en el resultado.
- **mysql_pconnect** Abre una conexión persistente con un servidor MySQL.
- **mysql_query** Envía una consulta MySQL.
- **mysql_result** Obtiene datos de resultado.
- **mysql_select_db** Selecciona una base de datos MySQL.
- **mysql_tablename** Obtiene el nombre de tabla de un campo.

Vamos a poner a trabajar estas funciones en este capítulo. Por ejemplo, podría reunir un ejemplo, phpdatatable.php, que lee y muestra la tabla de frutas de la base de datos produce.

Conexión al servidor de bases de datos

PHP se conecta a bases de datos usando objetos de *conexión*. Para crear un objeto de conexión para MySQL, use mysql_context:

```
mysql_context ( [servidor [, nombre de usuario [, contraseña [, new_link [, client_
flags]]]]] )
```

Aquí, *servidor* es el servidor de MySQL, que pueden ser direcciones URL, números de puerto, etc. Los argumentos *nombre de usuario* y *contraseña* son nombre de usuario y contraseña de MySQL.

El argumento new_link, si se establece en TRUE, obliga a PHP para establecer un nuevo vínculo con la base de datos, incluso si tiene ya uno de estos vínculos. En su defecto, si intenta abrir un segundo vínculo con la base de datos llamando a mysql_connect por segunda vez, es posible que PHP emplee en su lugar su vínculo ya establecido.

El parámetro client_flags puede ser una combinación (creada comparando valores con OR junto con el operador OR, |), de las siguientes constantes: MYSQL_CLIENT_SSL, MYSQL_CLIENT_COMPRESS, MYSQL_CLIENT_IGNORE_SPACE o MYSQL_CLIENT_INTERACTIVE.

La función mysql_connect devuelve un objeto de conexión, si es exitosa y FALSE en caso contrario.

Bueno, conectémonos con MySQL desde PHP, empleando mysql_connect. En este caso, MySQL y PHP están en la misma máquina; de modo que el servidor es tan sólo"localhost". Ésta es la forma de crear el objeto de conexión (ingrese su nombre de usuario y contraseña, desde luego):

```
<?php
  $connection = mysql_connect("localhost","root","*********")
    .
    .
    .
```

Aumentemos esto un poco, haciendo que muestre un mensaje si hubiera un error y salgamos, mediante la función die:

```
<?php
  $connection = mysql_connect("localhost","root","*********")
    or die ("No se pudo conectar al servidor");
    .
    .
    .
```

Conexión a la base de datos

Bueno, nos hemos conectado a MySQL; el paso siguiente consiste en seleccionar la base de datos a trabajar. La base de datos se selecciona con mysql_select_db, que funciona de esta manera:

```
mysql_select_db (database_name [, link_identifier] )
```

Aquí, *database_name* es el nombre de la base de datos, que aquí es"produce"y *link_identifier* como el objeto de conexión. Así trabaja mysql_select_db en phpdatatable.php:

```
<?php
  $connection = mysql_connect("localhost","root","password")
    or die ("No se pudo conectar al servidor");
  $db = mysql_select_db("produce",$connection)
    or die ("No se pudo seleccionar la base de datos");
    .
    .
    .
?>
```

Genial (se ha conectado al servidor de bases de datos y elegido la base de datos con que desea trabajar). Escoger la base de datos es fundamental (si no selecciona una base de datos antes de proceder, se producirán errores).

Lectura de la tabla

El objetivo del ejemplo phpdatatable.php es leer y mostrar la tabla de base de datos llamada frutas. Para devolver esa tabla al servidor MySQL, puede enviar a ese servidor una consulta MySQL mediante la función mysql_query:

```
mysql_query (consulta [ link_identifier] )
```

Aquí, *consulta* es la petición a SQL que desea enviar al servidor de bases de datos y *link_identifier* el objeto de conexión, representando la conexión con ese servidor; si tiene sólo una conexión abierta, PHP usará esa conexión.

Para SELECT, SHOW, DESCRIBE, EXPLAIN y otras instrucciones devolviendo datos de tabla, mysql_query devuelve un recurso si hay un acierto o FALSE en caso de haber un error. Para otro tipo de instrucciones SQL, UPDATE, DELETE, DROP, etc., mysql_query devuelve TRUE si hay un acierto o FALSE por un error. El recurso devuelto se debe pasar a mysql_fetch_array y otras funciones para manejar tablas de resultados para acceder a los datos devueltos.Y usar mysql_num_rows para averiguar cuántas filas se devolvieron de una instrucción SELECT.

La instrucción SQL para obtener todos los registros de la tabla de frutas es"SELECT * FROM frutas"; así que ésa será nuestra consulta:

```
<?php
  $connection = mysql_connect("localhost","root","password")
    or die ("No se pudo conectar al servidor");
  $db = mysql_select_db("produce",$connection)
    or die ("No se pudo seleccionar la base de datos");
  $query = "SELECT * FROM frutas";
    .
    .
    .
?>
```

Ahora puede obtener una tabla de datos devueltos repleta de hileras de la base de datos, de la siguiente forma con mysql_query:

```
<?php
  $connection = mysql_connect("localhost","root","password")
    or die ("No se pudo conectar al servidor");
  $db = mysql_select_db("produce",$connection)
    or die ("No se pudo seleccionar la base de datos");
  $query = "SELECT * FROM frutas";
  $result = mysql_query($query)
    .
    .
    .
?>
```

Observe que si la consulta SQL falla, puede finalizar la aplicación y mostrar un error mediante la función mysql_error:

```
<?php
  $connection = mysql_connect("localhost","root","password")
    or die ("No se pudo conectar al servidor");
```

```
  $db = mysql_select_db("produce",$connection)
    or die ("No se pudo seleccionar la base de datos");

  $query = "ELECT * FROM frutas";
  $result = mysql_query($query)
    or die("La consulta falló: " . mysql_error());
    .
    .
    .
?>
```

Muy bien, ya ha progresado. Ahora es momento de descifrar los datos recuperados de la base de datos.

Presentación de los datos de la tabla

Puede mostrar los datos de la tabla de frutas en una tabla HTML en la página Web devolviendo este ejemplo. Los dos campos en la tabla de frutas son nombre y número, de modo que puede comenzar creando una tabla HTML con los encabezados Nombre y Número:

```
<?php
  $connection = mysql_connect("localhost","root","password")
    or die ("No se pudo conectar al servidor");

  $db = mysql_select_db("produce",$connection)
    or die ("No se pudo seleccionar la base de datos");

  $query = "SELECT * FROM frutas";
  $result = mysql_query($query)
    or die("La consulta falló: " . mysql_error());

  echo "<table border='1'>";
  echo "<tr>";
  echo "<th>Nombre</th><th>Número</th>";
  echo "</tr>";
    .
    .
    .
?>
```

Ahora es momento de manipular el resultado recibido de su consulta SQL. Se trata de un recurso PHP; es la tabla de frutas en forma de código. Para extraer los registros de esa tabla, puede usar la función mysql_fetch_array, ésta devuelve una matriz correspondiente al registro actual, es decir, la fila actual, en la tabla de datos y usted puede recorrer los registros utilizando ciclos. Así se usa la función mysql_fetch_array:

```
mysql_fetch_array (resultado [, result_type] )
```

Aquí, *resultado* es el recurso devolviendo la función mysql_fetch_array (es la tabla de datos recuperada de la base de datos). Y *result_type* es el tipo de matriz requerida por usted.

Este valor es una constante y puede tomar los siguientes valores: MYSQL_ASSOC, MYSQL_NUM y el valor predeterminado de MYSQL_BOTH.

Puede utilizar mysql_fetch_array para capturar filas de la tabla de frutas y recorrer esas filas con un ciclo while. Así se ve eso, donde se almacena cada fila de la tabla en una variable llamada $row:

```
<?php
  $connection = mysql_connect("localhost","root","password")
    or die ("No se pudo conectar al servidor");

  $db = mysql_select_db("produce",$connection)
    or die ("No se pudo seleccionar la base de datos");

  $query = "SELECT * FROM frutas";
  $result = mysql_query($query)
    or die("La consulta falló: " . mysql_error());

  echo "<table border='1'>";
  echo "<tr>";
  echo "<th>Nombre</th><th>Número</th>";
  echo "</tr>";

  while ($row = mysql_fetch_array($result))
  {
    .
    .
    .
  }
?>
```

Esto asigna la fila actual de la tabla de frutas a $row. Puede acceder al campo del nombre de esa fila de esta forma: $row['name'] y al campo número en la fila como $row['number'].

Eso significa que mostraría los datos en ambos campos de la fila actual, nombre y número, de esta forma en phpdatatable.php:

```
<?php
  $connection = mysql_connect("localhost","root","password")
    or die ("No se pudo conectar al servidor");

  $db = mysql_select_db("produce",$connection)
    or die ("No se pudo seleccionar la base de datos");

  $query = "SELECT * FROM frutas";
  $result = mysql_query($query)
    or die("La consulta falló: " . mysql_error());

  echo "<table border='1'>";
  echo "<tr>";
  echo "<th>Nombre</th><th>Número</th>";
  echo "</tr>";
```

```
  while ($row = mysql_fetch_array($result))
  {
    echo "<tr>";
    echo "<td>", $row['name'], "</td><td>", $row['number'], "</td>";
    echo "</tr>";
  }
    .
    .
    .
?>
```

Todo lo que resta por hacer es cerrar la conexión con la base de datos.

Cierre de la conexión

Puede cerrar la conexión hacia la base de datos con mysql_close:

```
mysql_close ( [link_identifier] )
```

Aquí, *link_identifier* es el objeto de conexión representando la conexión con la base de datos. Así es como se cierra la conexión en phpdatatable.php:

```
<html>
  <head>
    <title>
       Mostrar tablas con MySQL
    </title>
  </head>

  <body>
    <h1>Mostrar tablas con MySQL</h1>

    <?php
      $connection = mysql_connect("localhost","root","password")
        or die ("No se pudo conectar al servidor");

      $db = mysql_select_db("produce",$connection)
        or die ("No se pudo seleccionar la base de datos");

      $query = "SELECT * FROM frutas";
      $result = mysql_query($query)
        or die("La consulta falló: ".mysql_error());

      echo "<table border='1'>";
      echo "<tr>";
      echo "<th>Nombre</th><th>Número</th>";
      echo "</tr>";

      while ($row = mysql_fetch_array($result))
      {
        echo "<tr>";
        echo "<td>", $row[`name'], "</td><td>", $row['number'], "</td>";
        echo "</tr>";
      }
```

FIGURA 10-1 La tabla frutas

```
      echo "</table>";

      mysql_close($connection);
    ?>
  </body>
</html>
```

Puede ver los resultados en la Figura 10-1, donde se aprecia la tabla de frutas completa. Formidable.

Bueno, ha logrado recuperar información de la tabla de base de datos. Excelente. Ahora, ¿qué tal si modifica esos datos?

Actualización de bases de datos

Si alguien compra sus productos en línea, necesitará actualizar su base de datos. Suponga, por ejemplo, que alguien compra una pera en línea, ello significa que su existencia pasa de 235 a 234 peras:

- manzanas 1020
- naranjas 3329
- plátanos 8582
- peras 234

¿Cómo haría este cambio desde PHP a la tabla de frutas? Podría conectarse a la base de datos produce y usar una consulta UPDATE de SQL para efectuar el cambio.

Puede ver cómo funciona esto en phpdataupdate.php. Primero se realiza una conexión con el servidor de bases de datos (ingrese su nombre de usuario y contraseña, desde luego):

```
<?php
  $connection = mysql_connect("localhost","root","password")
    or die ("No se pudo conectar al servidor");
   .
   .
   .
```

Luego elige la base de datos produce:

```
<?php
  $connection = mysql_connect("localhost","root","password")
    or die ("No se pudo conectar al servidor");

  $db = mysql_select_db("produce",$connection)
    or die ("No se pudo seleccionar la base de datos");
   .
   .
   .
```

Así, ¿cómo actualiza el número de peras? Puede usar la instrucción UPDATE de SQL "UPDATE frutas SET number = 234 WHERE name ='peras'" de esta forma:

```
<?php
  $connection = mysql_connect("localhost","root","password")
    or die ("No se pudo conectar al servidor");

  $db = mysql_select_db("produce",$connection)
    or die ("No se pudo seleccionar la base de datos");

    $query = "UPDATE frutas SET number = 234 WHERE name = 'peras'";
   .
   .
   .
```

Ahora puede ejecutar esta instrucción SQL con mysql_query:

```
<?php
  $connection = mysql_connect("localhost","root","password")
    or die ("o se pudo conectar al servidor");

  $db = mysql_select_db("produce",$connection)
    or die ("No se pudo seleccionar la base de datos");

    $query = "UPDATE frutas SET number = 234 WHERE name = 'peras'";

    $result = mysql_query($query)
      or die("La consulta falló: ".mysql_error());
   .
   .
   .
```

Bueno, ahora confirmemos que se ha hecho el cambio a la tabla de frutas, actualizando el número de peras. Puede mostrar el nuevo contenido de la tabla de frutas de esta forma en phpdataupdate.php:

```
<html>
  <head>
    <title>
      Actualización de bases de datos
    </title>
  </head>

  <body>
    <h1>Actualización de bases de datos</h1>

    <?php
       $connection = mysql_connect("localhost","root","password")
         or die ("No se pudo conectar al servidor");

       $db = mysql_select_db("produce",$connection)
         or die ("No se pudo seleccionar la base de datos");

       $query = "UPDATE frutas SET number = 234 WHERE name = 'peras'";

       $result = mysql_query($query)
         or die("La consulta falló: ".mysql_error());

       $query = "SELECT * FROM frutas";

       $result = mysql_query($query)
         or die("La consulta falló: " . mysql_error());

       echo "<table border='1'>";
       echo "<tr>";
       echo "<th>Nombre</TH><TH>Número</th>";
       echo "</tr>";

       while ($row = mysql_fetch_array($result))
       {
         echo "<tr>";
         echo "<td>", $row['name'], "</td><td>", $row['number'], "</td>";
         echo "</tr>";
       }

       echo "</table>";

       mysql_close($connection);
     ?>
  </body>
</html>
```

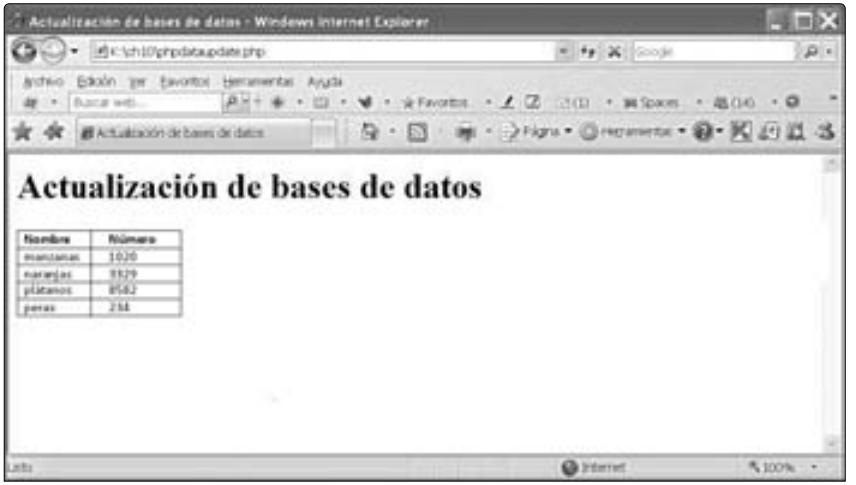

FIGURA 10-2 Actualización de la tabla de frutas

Puede ver los resultados en la Figura 10-2, donde el número de peras se ha actualizado a 234. Genial.

Bueno, hasta ahora todo va bien. Ahora ha leído y actualizado los datos de una base. ¿Qué tal si inserta datos nuevos?

Inserción de nuevos elementos en una base de datos

El negocio va bien y decide agregar más frutas a su sitio Web, comenzando con chabacanos. Su primer embarque es de 203 chabacanos, así que podría agregarlos a su existencia de frutas:

- manzanas 1020
- naranjas 3329
- plátanos 8582
- peras 234
- **chabacanos 203**

Pero no hay un registro para los chabacanos, así que es momento de agregar uno, que puede hacer con el comando INSERT de SQL en un nuevo ejemplo, phpdatainsert.php.

Primero se conecta al servidor de bases de datos:

```
<?php
  $connection = mysql_connect("localhost","root","password")
    or die ("No se pudo conectar al servidor");
    .
    .
    .
```

Luego seleccione la base de datos de frutas:

```
<?php
  $connection = mysql_connect("localhost","root","password")
    or die ("No se pudo conectar al servidor");

$db = mysql_select_db("produce",$connection)
    or die ("No se pudo seleccionar la base de datos");
    .
    .
    .
```

Puede insertar un nuevo registro en una tabla con la instrucción INSERT de SQL. Para introducir un nuevo registro para chabacanos en la tabla de frutas, puede ejecutar esta instrucción SQL: "INSERT INTO frutas (nombre, número) VALUES('chabacanos','203'):

```
<?php
  $connection = mysql_connect("localhost","root","password")
    or die ("No se pudo conectar al servidor");

  $db = mysql_select_db("produce",$connection)
    or die ("No se pudo seleccionar la base de datos");

  $query = "INSERT INTO frutas (name, number) VALUES('chabacanos', '203')";

  $result = mysql_query($query)
    or die("La consulta falló: " . mysql_error());
    .
    .
    .
```

Si da resultado, inserta un nuevo registro en la tabla frutas. Puede confirmarlo presentando la tabla frutas en phpdatainsert.php:

```
<html>
  <head>
    <title>
      Inserción de nuevos datos
    </title>
  </head>

  <body>
    <h1>Inserción de nuevos datos</h1>

    <?php
      $connection = mysql_connect("localhost","root","password")
        or die ("No se pudo conectar al servidor");

      $db = mysql_select_db("produce",$connection)
        or die ("No se pudo seleccionar la base de datos");

      $query = "INSERT INTO frutas (name, number) VALUES('chabacanos', '203')";
```

```
    $result = mysql_query($query)
      or die("La consulta falló: " . mysql_error());

    $query = "SELECT * FROM frutas";

    $result = mysql_query($query)
      or die("La consulta falló: " . mysql_error());

    echo "<table border='1'>";
    echo "<tr>";
    echo "<th>Nombre</TH><TH>Número</th>";
    echo "</tr>";

    while ($row = mysql_fetch_array($result))
    {
      echo "<TR>";
      echo "<td>", $row['name'], "</td><td>", $row['number'], "</td>";
      echo "</tr>";
    }

    echo "</TABLE>";

    mysql_close($connection);
  ?>
 </body>
</html>
```

Y ver los resultados en la Figura 10-3, donde hay un nuevo registro para los chabacanos. Genial.

FIGURA 10-3 Insertar un nuevo registro en la tabla de frutas

Eliminación de registros

Ahora suponga que su suministro de chabacanos se ha agotado, así que necesita retirarlos de la oferta en su sitio Web. ¿Cómo se elimina un registro? Puede usar la instrucción DELETE de SQL.

En este ejemplo, phpdatadelete.php, se elimina el registro de chabacanos de la tabla frutas. Se comienza de la forma habitual, conectándose al servidor de bases de datos y eligiendo la base de datos, que produce esta forma:

```
<?php
  $connection = mysql_connect("localhost","root","password")
    or die ("No se pudo conectar al servidor");

  $db = mysql_select_db("produce",$connection)
    or die ("No se pudo seleccionar la base de datos");
    .
    .
    .
```

Ahora puede eliminar el registro de chabacanos de la tabla de frutas mediante la instrucción de SQL "DELETE FROM frutas WHERE name ='chabacanos'":

```
<?php
  $connection = mysql_connect("localhost","root","password")
    or die ("No se pudo conectar al servidor");

  $db = mysql_select_db("produce",$connection)
    or die ("No se pudo seleccionar la base de datos");

  $query = "DELETE FROM frutas WHERE name = 'chabacanos'";
    .
    .
    .
?>
```

Y ejecute la consulta en la base de datos:

```
<?php
  $connection = mysql_connect("localhost","root","password")
    or die ("No se pudo conectar al servidor");

  $db = mysql_select_db("produce",$connection)
    or die ("No se pudo seleccionar la base de datos");

  $query = "DELETE FROM frutas WHERE name = 'chabacanos'";

  $result = mysql_query($query)
    or die("La consulta falló: " . mysql_error());
    .
    .
    .
?>
```

```
<html>
    <head>
        <title>
        Eliminación de registros
    </title>
</head>

<body>
  <h1>Eliminación de registros</h1>

  <?php
    $connection = mysql_connect("localhost","root","*******")
      or die ("No se pudo conectar al servidor");

    $db = mysql_select_db("produce",$connection)
      or die ("No se pudo seleccionar la base de datos");

    $query = "DELETE FROM frutas WHERE name = 'chabacanos'";

    $result = mysql_query($query)
      or die("La consulta falló: " . mysql_error());

    $query = "SELECT * FROM frutas";

    $result = mysql_query($query)
      or die("La consulta falló: " . mysql_error());

    echo "<table border='1'>";
    echo "<tr>";
    echo "<th>Nombre</th><th>Número</th>";
    echo "</tr>";

    while ($row = mysql_fetch_array($result))
    {
      echo "<tr>";
      echo "<td>", $row['name'], "</td><td>", $row['number'], "</td>";
      echo "</tr>";
    }

    echo "</table>";

    mysql_close($connection);
  ?>
</body>
</html>
```

Vea los resultados en la Figura 10-4, donde se han eliminado los chabacanos. Genial.

FIGURA 10-4 Eliminación de registros

Creación de nuevas tablas

También puede crear tablas propias de bases de datos usando PHP y SQL. En este ejemplo, phpcreatetable.php, se crea una nueva tabla llamada vegetales, con los siguientes registros:

- Nombre Número
- maíz 2083
- espinacas 1993
- remolachas 437

Comienza conectándose con el servidor y, como es habitual, tras elegir la base de datos se produce:

```
<?php
  $connection = mysql_connect("localhost","root","********")
    or die ("No se pudo conectar al servidor");

  $db = mysql_select_db("produce",$connection)
    or die ("No se pudo seleccionar la base de datos");
    .
    .
    .
?>
```

Cree tablas usando la instrucción CREATE de SQL. Por ejemplo, para crear una nueva tabla de vegetales con campos de nombre y número, podría ejecutar la instrucción de SQL "CREATE TABLE vegetales (nombre VARCHAR(20), número VARCHAR(20))":

```
<?php
  $connection = mysql_connect("localhost","root","********")
    or die ("No se pudo conectar al servidor");
```

```
  $db = mysql_select_db("produce",$connection)
    or die ("No se pudo seleccionar la base de datos");

  $query = "CREATE TABLE vegetales (nombre VARCHAR(20),
    número VARCHAR(20))";

  $result = mysql_query($query)
    or die("La consulta falló: " . mysql_error());
    .
    .
    .
?>
```

Es momento de agregar algunos vegetales a la nueva tabla, que puede hacer con la instrucción INSERT de SQL. Así es como insertaría un nuevo registro para el maíz, por ejemplo:

```
<?php
  $connection = mysql_connect("localhost","root","********")
    or die ("No se pudo conectar al servidor");

  $db = mysql_select_db("produce",$connection)
    or die ("No se pudo seleccionar la base de datos");

  $query = "CREATE TABLE vegetales (nombre VARCHAR(20),
    número VARCHAR(20))";

  $result = mysql_query($query)
    or die("La consulta falló: " . mysql_error());

  $query = "INSERT INTO vegetales (name, number) VALUES(
    'maíz', '2083')";

  $result = mysql_query($query)
    or die("La consulta falló: " . mysql_error());
    .
    .
    .
?>
```

Y así se crearían los otros dos registros en la tabla de vegetales:

```
<?php
  $connection = mysql_connect("localhost","root","********")
    or die ("No se pudo conectar al servidor");

  $db = mysql_select_db("produce",$connection)
    or die ("No se pudo seleccionar la base de datos");

  $query = "CREATE TABLE vegetales (nombre VARCHAR(20),
    número VARCHAR(20))";

  $result = mysql_query($query)
    or die("La consulta falló: " . mysql_error());

  $query = "INSERT INTO vegetales (name, number) VALUES(
    'maíz', '2083')";
```

```
    $result = mysql_query($query)
      or die("La consulta falló: " . mysql_error());

    $query = "INSERT INTO vegetales (name, number)
      VALUES('espinacas', '1993')";

    $result = mysql_query($query)
      or die("La consulta falló: " . mysql_error());

    $query = "INSERT INTO vegetales (name, number)
      VALUES('remolachas', '437')";

    $result = mysql_query($query)
      or die("La consulta falló: " . mysql_error());
      .
      .
      .
  ?>
```

Por último, puede mostrar la nueva tabla de la forma siguiente en phpcreatetable.php:

```
<html>
  <head>
    <title>
      Creación de una nueva tabla
    </title>
  </head>
  <body>
    <h1>
      Creación de una nueva tabla
    </h1>
    <?php
      $connection = mysql_connect("localhost","root","********")
        or die ("No se pudo conectar al servidor");

      $db = mysql_select_db("produce",$connection)
        or die ("No se pudo seleccionar la base de datos");

      $query = "CREATE TABLE vegetales (nombre VARCHAR(20),
        número VARCHAR(20))";

      $result = mysql_query($query)
        or die("La consulta falló: " . mysql_error());

      $query = "INSERT INTO vegetales (name, number) VALUES(
        'maíz', '2083')";

      $result = mysql_query($query)
        or die("La consulta falló: " . mysql_error());

      $query = "INSERT INTO vegetales (name, number)
        VALUES('espinacas', '1993')";

      $result = mysql_query($query)
        or die("La consulta falló: " . mysql_error());
```

```
    $query = "INSERT INTO vegetales (name, number)
      VALUES('remolachas', '437')";

    $result = mysql_query($query)
      or die("La consulta falló: " . mysql_error());

    $query = "SELECT * FROM vegetales";

    $result = mysql_query($query)
      or die("La consulta falló: " . mysql_error());

    echo "<table border='1'>";
    echo "<tr>";
    echo "<th>Nombre</th><th>Número</th>";
    echo "</tr>";

    while ($row = mysql_fetch_array($result))
    {
      echo "<tr>";
      echo "<td>", $row['name'], "</td><td>",
        $row['number'], "</td>";
      echo "</tr>";
    }

    echo "</table>";

    mysql_close($connection);
  ?>
 </body>
</html>
```

Los resultados se aprecian en la Figura 10-5, donde aparece la nueva tabla. Formidable.

FIGURA 10-5 Creación de una nueva tabla

Creación de una nueva base de datos

Incluso puede crear bases de datos totalmente nuevas, con sus propias tablas, mediante PHP y SQL. Por ejemplo, una nueva base de datos nueva, alimentos, con una tabla de antojitos poseyendo estos registros:

- Nombre Número
- tacos 218
- pizza 193
- hamburguesas 112

Así se hace en phpcreatedatabase.php. Primero, conéctese al servidor de bases de datos:

```
<?php
  $connection = mysql_connect("localhost","root","password")
    or die ("No se pudo conectar al servidor");


    .
    .
    .
?>
```

Ahora puede crear una base de datos completamente nueva, alimentos, con la instrucción SQL"CREATE DATABASE IF NOT EXISTS alimentos":

```
<?php
  $connection = mysql_connect("localhost","root","password")
    or die ("No se pudo conectar al servidor");

  $query = "CREATE DATABASE IF NOT EXISTS alimentos";

  $result = mysql_query($query)
    or die("La consulta falló: " . mysql_error());
    .
    .
    .
?>
```

Después, elija la base de datos alimentos:

```
<?php
  $connection = mysql_connect("localhost","root","password")
    or die ("No se pudo conectar al servidor");

  $query = "CREATE DATABASE IF NOT EXISTS alimentos";

  $result = mysql_query($query)
    or die("La consulta falló: " . mysql_error());

  $db = mysql_select_db("alimentos", $connection)
    or die ("No se pudo seleccionar la base de datos");
    .
    .
    .
?>
```

Luego, es tiempo de crear la tabla de antojitos:

```
<?php
  $connection = mysql_connect("localhost","root","password")
    or die ("No se pudo conectar al servidor");

  $query = "CREATE DATABASE IF NOT EXISTS alimentos";

  $result = mysql_query($query)
    or die("La consulta falló: " . mysql_error());

  $db = mysql_select_db("alimentos", $connection)
    or die ("No se pudo seleccionar la base de datos");

  $query = "CREATE TABLE antojitos (name VARCHAR(20), number
    VARCHAR(20))";

  $result = mysql_query($query)
    or die("La consulta falló: " . mysql_error());
    .
    .
    .
?>
```

Entonces debe colocar en la tabla algunos antojitos:

```
<?php
  $connection = mysql_connect("localhost","root","password")
    or die ("No se pudo conectar al servidor");


  $query = "CREATE DATABASE IF NOT EXISTS alimentos";

  $result = mysql_query($query)
    or die("La consulta falló: " . mysql_error());

  $db = mysql_select_db("alimentos", $connection)
    or die ("No se pudo seleccionar la base de datos");

  $query = "CREATE TABLE antojitos (name VARCHAR(20), number
    VARCHAR(20))";

  $result = mysql_query($query)
    or die("La consulta falló: " . mysql_error());

  $query = "INSERT INTO antojitos (name, number)
    VALUES('tacos', '218')";

  $result = mysql_query($query)
    or die("La consulta falló: " . mysql_error());

  $query = "INSERT INTO antojitos (name, number)
    VALUES('pizza', '193')";
```

```
    $result = mysql_query($query)
      or die("La consulta falló: " . mysql_error());

    $query = "INSERT INTO antojitos (name, number)
      VALUES('hamburguesas', '112')";

    $result = mysql_query($query)
      or die("La consulta falló: " . mysql_error());
      .
      .
      .
  ?>
```

Ahora puede mostrar la nueva tabla de antojitos:

```
<html>
  <head>
    <title>
      Creación de una nueva base de datos
    </title>
  </head>

  <body>
    <h1>
      Creación de una nueva base de datos
    </h1>

    <?php
      $connection = mysql_connect("localhost","root","password")
        or die ("No se pudo conectar al servidor");

      $query = "CREATE DATABASE IF NOT EXISTS alimentos";

      $result = mysql_query($query)
        or die("La consulta falló: " . mysql_error());

      $db = mysql_select_db("alimentos", $connection)
        or die ("No se pudo seleccionar la base de datos");

      $query = "CREATE TABLE antojitos (name VARCHAR(20), number
        VARCHAR(20))";

      $result = mysql_query($query)
        or die("La consulta falló: " . mysql_error());

      $query = "INSERT INTO antojitos (name, number)
        VALUES('tacos', '218')";

      $result = mysql_query($query)
        or die("La consulta falló: " . mysql_error());

      $query = "INSERT INTO antojitos (name, number)
        VALUES('pizza', '193')";
```

```
    $result = mysql_query($query)
      or die("La consulta falló: " . mysql_error());

    $query = "INSERT INTO antojitos (name, number)
      VALUES('hamburguesas', '112')";

    $result = mysql_query($query)
      or die("La consulta falló: " . mysql_error());
    $query = "SELECT * FROM antojitos";

    $result = mysql_query($query)
      or die("La consulta falló: " . mysql_error());

    echo "<table border='1'>";
    echo "<tr>";
    echo "<th>Nombre</th><th>Número</th>";
    echo "</tr>";

    while ($row = mysql_fetch_array($result))
    {
      echo "<tr>";
      echo "<td>", $row['name'], "</td><td>",
        $row['number'], "</td>";
      echo "</tr>";
    }
    echo "</table>";

    mysql_close($connection);
  ?>
 </body>
</html>
```

Los resultados están en la Figura 10-6, donde figura la nueva tabla, en la nueva base de datos.

FIGURA 10-6 Creación de una nueva base de datos

Clasificación de sus datos

Puede continuar su trabajo con bases de datos trabajando SQL en PHP (cualquier posibilidad con bases de datos, también es posible con SQL). Éste es otro ejemplo, phpdatasort.php, se recurre a SQL para clasificar los datos de la tabla de frutas antes de mostrarlos:

```
<html>
  <head>
    <title>
      Clasificación de sus datos
    </title>
  </head>

  <body>
    <h1>
      Clasificación de sus datos
    </h1>

    <?php
      $connection = mysql_connect("localhost","root","********")
        or die ("No se pudo conectar al servidor");

      $db = mysql_select_db("produce",$connection)
        or die ("No se pudo seleccionar la base de datos");

      $query = "SELECT * FROM frutas ORDER BY name";

      $result = mysql_query($query)
        or die("La consulta falló: " . mysql_error());

      echo "<table border='1'>";
      echo "<tr>";
      echo "<th>Nombre</th><th>Número</th>";
      echo "</TR>";

      while ($row = mysql_fetch_array($result))
      {
        echo "<tr>";
        echo "<td>", $row['name'], "</td><td>",
          $row['number'], "</td>";
        echo "</tr>";
      }

      echo "</table>";

      mysql_close($connection);
    ?>
  </body>
</html>
```

FIGURA 10-7 Clasificación de sus datos

Los resultados están en la Figura 10-7, donde se clasificaron sus datos.

Ahora que puede utilizar SQL con PHP, sus capacidades de trabajo con bases de datos son ilimitadas.

CAPÍTULO 11

Sesiones, cookies y FTP

En este capítulo veremos gran parte del poder de PHP —sesiones, cookies, FTP, correo electrónico y más—. Estos temas son de interés central para programadores de PHP, de modo que este capítulo será interesante.

Por su naturaleza, Internet es un lugar sin estado —es decir, las páginas Web no almacenan datos—. La siguiente vez que cargue la mayoría de páginas Web en un navegador, será lo mismo que la primera visita —la página Web se reinicializa y muestra de nueva cuenta—. Sin embargo, se ha vuelto un lugar más serio en términos de programación y ello significa escribir aplicaciones Web, funcionando a manera de páginas múltiples: esto significa almacenar datos persistentes del usuario de una página a la siguiente. Verá dos formas para hacerlo en este capítulo: sesiones y cookies. Y comenzaremos con las cookies.

Cómo establecer una cookie

Las cookies son segmentos de texto que puede almacenar en la computadora del usuario y PHP cuenta con buen soporte para establecer y leer cookies. Esos segmentos de texto persistirán, incluso cuando la computadora del usuario está apagada, de modo que se almacene información acerca del usuario con facilidad. Eso es formidable si quiere personalizar una aplicación Web —el usuario puede elegir una combinación de colores, por ejemplo—. Las cookies se utilizan para todos los fines, desde los bien intencionados hasta los más siniestros —como registrar qué anuncios ya ha visto un usuario y a los que ha respondido—. Establecer y leer cookies en PHP no es difícil.

Las cookies se alojan en la máquina del usuario con la función setcookie de PHP:

```
setcookie(name [, value [, expire [, path [, domain [, secure]]]]])
```

Éste es el significado de los parámetros:

- ***name*** El nombre de la cookie.
- ***value*** El valor de la cookie. (Este valor se almacena en la computadora del cliente, de modo que no conserve información confidencial.)
- ***expire*** El tiempo en que expira o caduca la cookie. Éste se da en segundos desde el 1 de enero de 1970. Lo más probable es que establezca este valor con la función time de PHP, más el tiempo en segundos que ha de transcurrir antes de que caduque.
- ***path*** Ruta de acceso en el servidor al que estará disponible la cookie.

- ***domain*** El dominio para el que está disponible la cookie.
- ***secure*** Indica que la cookie sólo debe transmitirse a través de una conexión HTTPS segura. Cuando se establezca a 1, la cookie sólo se activará si existe conexión segura. El valor predeterminado es 0.

¿Desea un ejemplo real de una cookie simple? Aquí está, phpsetcookie.php. Este ejemplo establece una cookie llamada message hacia el texto"Sin preocupaciones.":

```
<?php
  setcookie("message", "Sin preocupaciones.");
?>
```

Así es como se ve este código PHP en phpsetcookie.php:

```
<html>
  <head>
    <title>
      Cómo establecer una cookie
    </title>
    <?php
      setcookie("message", "Sin preocupaciones.");
    ?>
  </head>

  <body>
    <h1>
      Cómo establecer una cookie
    </h1>
    Se ha establecido la cookie.
    Vaya a <a href="phpgetcookie.php">phpgetcookie.php</a> para leerla.
  <body>
</html>
```

Y puede ver los resultados en la Figura 11-1, donde se estableció la cookie.

Esta página de PHP incluye un hipervínculo a otra página, phpgetcookie.php, donde se leerá la cookie:

```
<html>
        .
        .
        .
  <body>
    <h1>
      Cómo establecer una cookie
    </h1>
    Se ha establecido la cookie.
    Vaya a <a href="phpgetcookie.php">phpgetcookie.php</a> para leerla.
  <body>
</html>
```

Y esa página —phpgetcookie.php— la veremos a continuación.

FIGURA 11-1 Cómo establecer una cookie

Lectura de una cookie

Cuando se define una cookie, ésta no estará visible para sus scripts hasta la siguiente vez que cargue una página. Eso se debe a que la cookie se envía al servidor desde la máquina del usuario; así, inmediatamente después de establecer una cookie, no podrá leerla; necesita recibir primero una página del navegador. Observe también que las cookies se envían en encabezados HTTP de páginas recibidas por el navegador y, si su página de manejo de cookies está en el dominio A (como www.ultragiantbigco.com), sólo se le enviarán cookies procedentes del dominio A.

Una vez establecidas las cookies, se puede acceder a ellas en la siguiente carga de página con la matriz $_COOKIE. Establecimos una cookie llamada message en el tema anterior (y la leeremos aquí). Los valores de las cookies se cargan automáticamente en la matriz global llamada $_COOKIES, de forma muy similar al almacenamiento de valores de datos de páginas Web en $_REQUEST, facilitando este proceso.

En phpgetcookie.php, puede comenzar comprobando si se estableció la cookie (evitando un error cuando intente mostrar $_COOKIE[‘message’] y la cookie no se ha definido):

```
<?php
  if (isset($_COOKIE['message'])) {
  .
  .
  .
  }
?>
```

Si la cookie se estableció en realidad, puede mostrar su texto de esta forma:

```
<?php
  if (isset($_COOKIE['message'])) {
    echo $_COOKIE['message'];
  }
?>
```

Así es como se ve esto en phpgetcookie.php:

```
<html>
  <head>
    <title>
      Lectura de una cookie
    </title>
  </head>

  <body>
    <h1>
      Lectura de una cookie
    </h1>
      La cookie dice:
      <?php
        if (isset($_COOKIE['message'])) {
          echo $_COOKIE['message'];
        }
      ?>
  <body>
</html>
```

Los resultados se aprecian en la Figura 11-2, donde se leyó la cookie.

Bueno, eso permite establecer cookies y leerlas. Sin embargo, esta cookie se eliminará apenas el usuario cierre el navegador. ¿Cómo establecer el tiempo de caducidad de una cookie? Eso lo veremos a continuación.

FIGURA 11-2 Lectura de una cookie

Cómo establecer la caducidad de una cookie

Esto se define al crearla con la función setcookie:

```
setcookie(name [, value [, expire [, path [, domain [, secure]]]]])
```

Aquí, el tiempo almacenado en este parámetro es un sello de hora de Unix, alojando el tiempo en segundos desde el 1 de enero de 1970, para especificar la hora de eliminación de la cookie. Para obtener el tiempo actual en segundos desde el 1 de enero de 1970, use la función time de PHP. Por ejemplo, para especificar que una cookie debe eliminarse en una hora, puede hacer algo como lo siguiente, donde se especifica un tiempo de caducidad time( ) + 3600 segundos:

```
<?php
  setcookie("mycookie", $value, time() + 3600;
?>
```

Este ejemplo, phpsetexpiration.php, establece una cookie con fecha de caducidad de 30 días a partir de ahora, definiendo su longevidad a time()+60*6**24*30:

```
<?php
  setcookie("message", "Sin preocupaciones por 30 días.", time()+60*60*24*30);
?>
```

Así es como se ve en phpsetexpiration.php:

```
<html>
  <head>
    <title>
      Cómo establecer el tiempo de caducidad de una cookie
    </title>
    <?php
      setcookie("message", "Sin preocupaciones por 30 días.", time()+60*60*24*30);
    ?>
  </head>

  <body>
    <h1>
      Cómo establecer el tiempo de caducidad de una cookie
    </h1>
      La caducidad de la cookie se ha establecido en 30 días. Vaya a
      <a href="phpgetcookie.php">phpgetcookie.php</a> a continuación.
  <body>
</html>
```

Vea los resultados en la Figura 11-3, donde la cookie se estableció para 30 días.

Cuando hace clic en el hipervínculo de este ejemplo, éste lo llevará a la página de lectura de cookies, phpreadcookie.php, y apreciará el texto de la cookie en la Figura 11-4.

Incluso podría querer más control sobre las cookies (por ejemplo, eliminar cookies, que veremos a continuación).

FIGURA 11-3 Cómo establecer una cookie por 30 días

Eliminación de cookies

¿Desea eliminar una cookie? Si puede o no hacerlo dependerá del navegador que usa. Es posible establecer el tiempo de caducidad de una cookie fijándola en un hora ya transcurrida y, en teoría, eso deberá forzar que el navegador elimine la cookie, pero es posible que el navegador no la suprima hasta que el usuario cierre el navegador (y posiblemente ni siquiera lo haga). Así que una precaución apropiada consiste en establecer primero el texto de la cookie en una cadena vacía,"":

```
<?php
  setcookie("message", "");
?>
```

FIGURA 11-4 Lectura de una cookie

Luego, establezca el tiempo de caducidad a algún momento en el pasado también:

```
<?php
  setcookie("message", "", time() - 3600);
?>
```

Eso es todo (así se elimina una cookie; no hay función deletecookie especial). Así se ve en phpdeletecookie.php:

```
<html>
  <head>
    <title>
      Eliminación de cookies
    </title>
    <?php
      setcookie("message", "", time() - 3600);
    ?>
  </head>
  <body>
    <h1>
      Eliminación de cookies
    </h1>
    Se eliminó la cookie. Compruébelo en
    <a href="phpgetcookie.php">phpgetcookie.php</a>.
  <body>
</html>
```

Verá los resultados en la Figura 11-5, donde se eliminó la cookie.

Puede ver lo que dice la página phpgetcookie.php acerca de la cookie eliminada en la Figura 11-6 (no halló la cookie).

FIGURA 11-5 Eliminación de una cookie

FIGURA 11-6 Intento de leer una cookie eliminada

Trabajando con FTP

El uso de FTP en Internet es formidable para mover archivos a diferentes servidores. Éste puede ser un medio muy bueno para implementar código, obtener datos de configuración de otras fuentes, leer páginas Web y más. A continuación, el soporte para FTP integrado en PHP (las funciones incluyendo nb en sus nombres no producen bloqueo —no esperan a completar su tarea antes de regresar):

- **ftp_alloc** Asigna espacio para cargar un archivo
- **ftp_cdup** Cambia al directorio de origen
- **ftp_chdir** Cambia directorios en un servidor FTP
- **ftp_chmod** Establece permisos en un archivo a través de FTP
- **ftp_close** Cierra una conexión FTP
- **ftp_connect** Abre una conexión FTP
- **ftp_delete** Elimina un archivo del servidor FTP
- **ftp_exec** Solicita la ejecución de un programa en el servidor FTP
- **ftp_fget** Descarga un archivo de un servidor FTP y lo guarda en un archivo abierto
- **ftp_fput** Carga un archivo abierto al servidor FTP
- **ftp_get_option** Recupera comportamientos en tiempo de ejecución del flujo FTP en curso
- **ftp_get** Descarga un archivo del servidor FTP
- **ftp_login** Inicia sesión en una conexión FTP
- **ftp_mdtm** Devuelve la hora de la última modificación del archivo dado
- **ftp_mkdir** Crea un directorio
- **ftp_nb_continue** Continúa la recuperación/envío de un archivo

- **ftp_nb_fget** Recupera un archivo del servidor FTP y lo escribe en un archivo abierto
- **ftp_nb_fput** Almacena un archivo de un archivo abierto en el servidor FTP
- **ftp_nlist** Devuelve una lista de archivos en el directorio dado
- **ftp_put** Carga un archivo en el servidor FTP
- **ftp_pwd** Devuelve el nombre del directorio actual
- **ftp_quit** Alias de ftp_close
- **ftp_raw** Envía un comando arbitrario a un servidor FTP
- **ftp_rawlist** Devuelve una lista detallada de archivos en el directorio dado
- **ftp_rename** Cambia el nombre a un archivo en el servidor FTP
- **ftp_rmdir** Elimina un directorio
- **ftp_set_option** Establece opciones FTP mixtas en tiempo de ejecución
- **ftp_site** Envía un comando SITE al servidor
- **ftp_size** Devuelve el tamaño del archivo dado
- **ftp_ssl_connect** abre una conexión SSL-FTP segura
- **ftp_systype** Devuelve el identificador del tipo del sistema del servidor FTP remoto

Este ejemplo, phpftp.php, iniciará sesión en un directorio remoto y obtendrá un listado de directorios. Comienza conectándose al servidor remoto mediante ftp_connect:

```
ftp_connect(host [, port [, timeout]])
```

Dicha función abre una conexión FTP con el *host* dado (observe que el host no debe tener diagonal antes ni prefijo ftp://). El parámetro *port* especifica un puerto alternativo al cual conectarse; si se omite o establece en cero, entonces se utilizará el puerto FTP predeterminado, 21. *timeout* especifica tiempo límite para todas las operaciones de red subsecuentes. Si se omite, el valor predeterminado es 90 segundos. Esta función devuelve un flujo FTP, si hay un acierto o FALSE en caso de error.

Así se conecta el código en phpftp.php (indique su servidor aquí):

```
<?php
  $connect = ftp_connect("ftp.ispname.com");
    .
    .
    .
?>
```

Después, debe iniciar sesión en el servidor FTP y puede usar ftp_login para eso:

```
ftp_login(ftp_stream, username, password)
```

Esta función pasa su *nombre de usuario* y *contraseña* al servidor FTP, respondiendo TRUE si hay un acierto o FALSE si encuentra una falla. De obtener un resultado de TRUE, habrá iniciado sesión en el servidor FTP.

Así es como en phpftp.php se usa ftp_login para iniciar sesión (escriba su nombre y contraseña):

```
<?php
  $connect = ftp_connect("ftp.ispname.com");

  $result = ftp_login($connect, "username", "password");
    .
    .
    .
?>
```

Bueno, hasta ahora todo va bien. El objetivo de phpftp.php es listar directorios de un directorio remoto en el servidor y, en este ejemplo, se llamará code22. Puede usar la función ftp_nlist para obtener la lista del directorio remoto:

```
array ftp_nlist(ftp_stream, directory)
```

Aquí, *ftp_stream* es el objeto de conexión obtenido de ftp_connect y *directory* el directorio remoto del que quiere una lista. La función devuelve una matriz de nombres de archivo del directorio especificado si hay un acierto o FALSE de haber un error.

Así es como se obtiene el listado de directorios de code22:

```
<?php
  $connect = ftp_connect("ftp.ispname.com");

  $result = ftp_login($connect, "username", "password");

  array = ftp_nlist($connect, "code22");
    .
    .
    .
?>
```

Ahora puede recorrer la matriz de nombres de archivo, $array, listando nombres de archivos:

```
<?php
  $connect = ftp_connect("ftp.ispname.com");

  $result = ftp_login($connect, "username", "password");

  $array = ftp_nlist($connect, "code22");

  foreach($array as $value){
    echo $value, "<br>";
  }
    .
    .
    .
?>
```

Genial. Así se ve este script PHP en phpftp.php:

```
<html>
  <head>
    <title>
```

```
      Lectura de un directorio remoto con FTP
    </title>
  </head>

  <body>
    <h1>
      Lectura de un directorio remoto con FTP
    </h1>

    Esto es lo que hay en el directorio remoto:
    <br>
    <br>
    <?php
      $connect = ftp_connect("ftp.ispname.com");

      $result = ftp_login($connect, "username", "password");

      $array = ftp_nlist($connect, "code22");

      foreach($array as $value){
        echo $value, "<br>";
      }
    ?>
  <body>
</html>
```

Puede ver el resultado de phpftp.php en la Figura 11-7 (se conectó al directorio remoto y mostró los archivos del directorio). No está mal.

FIGURA 11-7 Cómo obtener un listado de directorios remotos con FTP

Descarga de archivos con FTP

¿Qué tal si descargamos un archivo utilizando FTP? El ejemplo que sigue, phpftpget.php, hace precisamente eso. Comienza conectándose con un servidor (use su servidor aquí):

```
<?php
  $connect = ftp_connect("ftp.ispname.com");
    .
    .
    .
?>
```

Luego, intenta iniciar sesión (use su nombre de usuario y contraseña aquí):

```
<?php
  $connect = ftp_connect("ftp.ispname.com");

  $result = ftp_login($connect, "username", "password");
    .
    .
    .
?>
```

Si no inicia sesión, puede informar al usuario y cerrar la aplicación:

```
<?php
  $connect = ftp_connect("ftp.ispname.com");

  $result = ftp_login($connect, "username", "password");

  if(!$result){
    echo "No se pudo conectar.";
    exit;
  }
    .
    .
    .
?>
```

Por otra parte, si el usuario logró iniciar sesión, es momento para descargar el archivo. Para ello, use ftp_get:

```
ftp_get(ftp_stream, local_file, remote_file, mode [, pos])
```

Esta función recupera *remote_file* del servidor FTP y lo guarda localmente en *local_file* (que puede incluir nombre de ruta de acceso). El *modo* de transferencia especificado debe ser FTP_ASCII o FTP_BINARY. El argumento *pos* es la posición del archivo remoto desde la que se iniciará la descarga. Esta función devuelve TRUE si hay un acierto o FALSE de haber falla.

Así es como se descarga el archivo a.php y guarda bajo script.php localmente:

```
<?php
  $connect = ftp_connect("ftp.ispname.com");
```

```
    $result = ftp_login($connect, "username", "password");

    if(!$result){
      echo "No se pudo conectar.";
      exit;
    }

    $result = ftp_get($connect, "script.php", "a.php", FTP_ASCII);
      .
      .
      .
  ?>
```

¿Realmente consiguió el archivo? Puede comprobarlo viendo si $result es verdadero o falso. Si $result es TRUE, consiguió el archivo; si es FALSE, no lo obtuvo:

```
  <?php
    $connect = ftp_connect("ftp.ispname.com");

    $result = ftp_login($connect, "username", "password");

    if(!$result){
      echo "No se pudo conectar.";
      exit;
    }

    $result = ftp_get($connect, "script.php", "a.php", FTP_ASCII);

    if($result){
      echo "Se consiguió el archivo.";
    }
    else {
      echo "No se consiguió el archivo.";
    }
    .
    .
    .
  ?>
```

Por último, conviene cerrar la conexión FTP, que se consigue con ftp_close, de la siguiente forma en phpftpget.php:

```
<html>
  <head>
    <title>
      Descarga de un archivo con FTP
    </title>
  </head>

    <body>
      <h1>Descarga de un archivo con FTP</h1>
      Descargando el archivo....
      <br>
      <?php
```

FIGURA 11-8 Descarga de un archivo con FTP

```
        $connect = ftp_connect("ftp.ispname.com");

        $result = ftp_login($connect, "username", "password");

        if(!$result){
          echo "No se pudo conectar.";
          exit;
        }

        $result = ftp_get($connect, "script.php", "a.php", FTP_ASCII);

         if($result){
           echo "Se consiguió el archivo.";
         }
         else {
           echo "No se consiguió el archivo.";
         }

         ftp_close($connect);
      ?>
  <body>
</html>
```

Puede apreciar los resultados en la Figura 11-8 —la aplicación se conectó al servidor remoto y descargó el archivo—. Excelente.

Carga de archivos con FTP

Acaba de ver cómo se descargan archivos con FTP y PHP; también puede cargar archivos. Todo lo que necesita es ftp_put:

```
ftp_put(ftp_stream, remote_file, local_file, mode [, pos])
```

Esta función aloja *local_file* (que puede especificarse con una ruta de acceso) en el servidor FTP como *remote_file*. El modo de transferencia especificado debe ser FTP_ASCII o FTP_BINARY. El argumento *pos* especifica el sitio desde donde leer datos en el archivo local. La función devuelve TRUE si hay un acierto o FALSE de haber falla.

En ejemplo, phpftpput.php, el script opera la carga por sí solo. Comienza conectándose con el servidor FTP:

```
<?php
  $connect = ftp_connect("ftp.ispname.com");
    .
    .
    .
?>
```

Luego puede usted iniciar sesión:

```
<?php
  $connect = ftp_connect("ftp.ispname.com");

  $result = ftp_login($connect, "username", "password");
    .
    .
    .
?>
```

Y mostraría los resultados del intento de conexión si éste falla, saliendo del programa de esta forma:

```
<?php
  $connect = ftp_connect("ftp.ispname.com");

  $result = ftp_login($connect, "username", "password");

  if(!$result){
    echo "No se pudo conectar.";
    exit;
  }
    .
    .
    .
?>
```

Ahora puede cargar el archivo utilizando ftp_put:

```
<?php
  $connect = ftp_connect("ftp.ispname.com");

  $result = ftp_login($connect, "username", "password");
    .
    .
    .
?>
```

Asimismo, comprobar los resultados de esta operación, reportando el acierto al usuario:

```
<?php
  $connect = ftp_connect("ftp.ispname.com");

  $result = ftp_login($connect, "username", "password");

  if(!$result){
    echo "No se pudo conectar.";
    exit;
  }

  $result = ftp_get($connect, "script.php", "a.php", FTP_ASCII);

  if($result){
    echo "Se cargó el archivo.";
  }
    .
    .
    .
?>
```

O bien, si la operación falló, también reportarlo. Por último, podría cerrar la conexión:

```
<html>
  <head>
    <title>
      Carga de un archivo con FTP
    </title>
  </head>

  <body>
    <h1>
      Carga de un archivo con FTP
    </h1>
    Cargando el archivo....
    <br>
    <?php
      $connect = ftp_connect("ftp.ispname.com");

      $result = ftp_login($connect, "username", "password");

      if(!$result){
        echo "No se pudo conectar.";
        exit;
      }

      $result = ftp_put($connect, "phpftpput.php", "phpftpput.php", FTP_ASCII);
```

FIGURA 11-9 Carga de un archivo con FTP

```
      if($result){
        echo "Se cargó el archivo.";
      }
      else {
        echo "No se cargó el archivo.";
      }

      ftp_close($connect);
    ?>
  <body>
</html>
```

Bueno, los resultados se encuentran en la Figura 11-9 —la aplicación se conectó al servidor remoto y cargó el archivo.

Eliminación de un archivo con FTP

¿Desea deshacerse de un archivo en el servidor FTP? Use ftp_delete:

```
ftp_delete (ftp_stream, path)
```

Aquí, *ftp_stream* es el objeto de conexión y *path* el nombre del archivo —incluida su ruta de acceso— que desea eliminar en el servidor. Esta función devuelve TRUE cuando acierta o FALSE si falla.

En este ejemplo, phpftpdelete.php, primero debe conectarse e iniciar sesión:

```
<?php
  $connect = ftp_connect("ftp.ispname.com");

  $result = ftp_login($connect, "username", "password");

  if(!$result){
```

```
    echo "No se pudo conectar.";
    exit;
  }
    .
    .
    .
```

Luego eliminar el archivo cargado antes, phpftpput.php:

```
<?php
  $connect = ftp_connect("ftp.ispname.com");

  $result = ftp_login($connect, "username", "password");

  if(!$result){
    echo "No se pudo conectar.";
    exit;
  }

  $result = ftp_delete($connect, "phpftpput.php");
    .
    .
    .
?>
```

Si el archivo se eliminó, podría reportarlo:

```
<?php
  $connect = ftp_connect("ftp.ispname.com");

  $result = ftp_login($connect, "username", "password");

  if(!$result){
    echo "No se pudo conectar.";
    exit;
  }

  $result = ftp_delete($connect, "phpftpput.php");

  if($result){
    echo "Se eliminó el archivo.";
  }
    .
    .
    .
?>
```

En su defecto, reporte la falla. De una u otra forma, debe cerrar la conexión FTP de esta forma en phpftpdelete.php:

```
<html>
  <head>
    <title>
      Eliminación de un archivo con FTP

    </title>
```

```
  </head>

  <body>
    <h1>
      Eliminación de un archivo con FTP
    </h1>
    Deleting the file....
    <br>
    <?php
      $connect = ftp_connect("ftp.ispname.com");

      $result = ftp_login($connect, "username", "password");

      if(!$result){
        echo "No se pudo conectar.";
        exit;
      }

      $result = ftp_delete($connect, "phpftpput.php");

      if($result){
        echo "Se eliminó el archivo.";
      }
      else {
        echo "No se eliminó el archivo.";
      }

      ftp_close($connect);
    ?>
  <body>
</html>
```

Vea los resultados en la Figura 11-10 (el archivo se eliminó en realidad).

FIGURA 11-10 Eliminación de un archivo con FTP

Creación y eliminación de directorios con FTP

Ambas operaciones son posibles en servidores FTP remotos mediante PHP. Se crean directorios utilizando ftp_mkdir (de"make directory"o"crear directorio"):

```
ftp_mkdir (ftp_stream, directory )
```

Y eliminan directorios usando ftp_rmdir (de"remove directory"o"eliminar directorio"):

```
ftp_rmdir (ftp_stream, directory )
```

Para ambas funciones, *ftp_stream* es el objeto de conexión FTP y *directory* es el alojamiento afectado.

Éste es un ejemplo, phpftpdir.php. Tras conectarse e iniciar sesión, el código intenta crear un nuevo directorio llamado backup con ftp_mkdir, reportando al usuario si ha tenido éxito:

```
<?php
  $connect = ftp_connect("ftp.ispname.com");

  $result = ftp_login($connect, "username", "password");

  if(!$result){
    echo "No se pudo conectar.";
    exit;
  }

  $result = ftp_mkdir($connect, "backup");

  if($result){
    echo "Se creó el directorio. <br>";
  }
  else {
    echo "No se pudo crear el directorio. <br>";
  }
    .
    .
    .
?>
```

Y tras crear el nuevo directorio, puede eliminarlo con ftp_rmdir de la siguiente forma en phpftpdir.php:

```
<html>
  <head>
    <title>
      Creación y eliminación de directorios con FTP
    </title>
  </head>

  <body>
    <h1>
      Creación y eliminación de directorios con FTP
    </h1>
```

```
    <?php
      $connect = ftp_connect("ftp.ispname.com");

      $result = ftp_login($connect, "username", "password");

      if(!$result){
        echo "No se pudo conectar.";
        exit;
      }

      $result = ftp_mkdir($connect, "backup");

      if($result){
        echo "Se creó el directorio. <br>";
      }
      else {
        echo "No se pudo crear el directorio. <br>";
      }

      $result = ftp_rmdir($connect, "backup");

      if($result){
        echo "Se eliminó el directorio. <br>";
      }
      else {
        echo "No se pudo eliminar el directorio. <br>";
      }

      ftp_close($connect);
    ?>
  <body>
</html>
```

Vea los resultados en la Figura 11-11 (el directorio se creó y luego eliminó).

FIGURA 11-11 Creación y eliminación de un directorio con FTP

Envío de correo electrónico

Sus scripts de PHP también pueden enviar correo electrónico. Para habilitar el correo electrónico, edite esta sección del archivo de inicialización de PHP, php.ini:

```
[mail function]
; For Win32 only.
SMTP = localhost
smtp_port = 25
; For Win32 only.
;sendmail_from = me@example.com
; For Unix only.  You may supply arguments as well (default: "sendmail -t -i").
;sendmail_path =
```

Los usuarios de Windows deben indicar el host SMTP que desean utilizar (como mail.ispname.com o smtp.ispname.com) y establecer su dirección de respuesta. Es posible que usuarios de Linux y Unix no deban hacer cambios si la utilidad sendmail ya está en su ruta de acceso, pero si las cosas no funcionan como están, elimine los comentarios de sendmail_path y establezca el valor apropiado (como /usr/bin/sendmail).

Para enviar correo se utiliza la función mail:

```
mail(to, subject, message [, additional_headers [, additional_parameters]])
```

Esto envía un mensaje a la dirección de correo electrónico en el campo *to* (para), el tema *subject* y el mensaje *message*. También puede establecer encabezados y parámetros de correo adicionales para sendmail.

Puede ver un ejemplo aquí (phpemail.html y phpemail.php). La página phpemail.html permite indicar el correo electrónico al que desea enviar y lo manda en phpemail.php:

```
<html>
  <head>
    <title>
      Envío de correo electrónico
    </title>
  </head>

  <body>
    <h1>Envío de correo electrónico</h1>
    <br>
    <form method="post" action="phpemail.php">
        .
        .
        .
    </form>
  <body>
</html>
```

En esta página, el usuario ingresa sus comentarios en un área de texto llamada mensaje y hace clic en el botón Enviar, para mandar esos comentarios a phpemail.php:

```
<html>
  <head>
    <title>
```

```
      Envío de correo electrónico
    </title>
  </head>

  <body>
    <h1>Envío de correo electrónico</h1>
    <br>
    <form method="post" action="phpemail.php">
      Escriba sus comentarios y haga clic en Enviar:
      <br>
      <textarea name="message" cols="50" rows="5"></textarea>
      <br>
      <br>
      <input type="submit" value="Enviar">
    </form>
  <body>
</html>
```

Puede ver esta página en la Figura 11-12, donde el usuario ha ingresado un mensaje de correo electrónico.

El script phpemail.php lee el mensaje enviado por el usuario (es decir, $_REQUEST ["message"]) y lo manda por correo de esta forma:

```
<html>
  <head>
    <title>
      Se envió su correo electrónico
    </title>
  </head>
```

FIGURA 11-12 Cómo escribir un correo electrónico

FIGURA 11-13 Envío de un correo electrónico

```
  <body>
    <h1>Se envió su correo electrónico</h1>
    Gracias por su mensaje.
    <br>
    <?php
      mail("steve@ispname.com", "Web mail", $_REQUEST["message"]);
    ?>
  <body>
</html>
```

Los resultados se aprecian en la Figura 11-13, donde se envío el correo electrónico.

Envío de correo electrónico avanzado

También es posible enviar correo electrónico con encabezados adicionales, como cc y bcc. Así funciona en phpadvemail.html y phpadvemail.php. En phpadvemail.html, el usuario ingresa su correo electrónico:

```
<html>
  <head>
    <title>
      Envío de correo electrónico con encabezados
    </title>
  </head>

  <body>
    <h1> Envío de correo electrónico con encabezados</H1>

    <form method="post" action="phpemail.php">
      Escriba sus comentarios y haga clic en Enviar:
      <br>
      <br>
```

```
      <textarea name="message" cols="50" rows="5"></textarea>
      <br>
      <input type="submit" value="Enviar">
    </form>
  <body>
</html>
```

Donde también puede incluir campos de texto para los encabezados cc y bcc:

```
<html>
  <head>
    <title>
      Envío de correo electrónico con encabezados
    </title>
  </head>

  <body>
    <h1> Envío de correo electrónico con encabezados</H1>

    <form method="post" action="phpemail.php">
      Escriba sus comentarios y haga clic en Enviar:
      <br>
      cc: <input type="text" name="cc">
      bcc: <input type="text" name="bcc">
      <br>
      <textarea name="message" cols="50" rows="5"></textarea>
      <br>
      <input type="submit" value="Enviar">
    </form>
  <body>
</html>
```

Esta página se ve en la Figura 11-14; el usuario ha ingresado correo electrónico y otras direcciones de correo en los campos cc y bcc.

FIGURA 11-14 Ingreso de correo electrónico con encabezados

El mensaje de correo electrónico de usuario y encabezados cc y bcc se envían a phpadvemail.php. Para emplear los encabezados de correo electrónico, deben ensamblarse en una cadena de texto separada por"\r\n". Por ejemplo, si el usuario ingresó un encabezado cc, usted puede agregarlo a una cadena llamada $headers de la siguiente forma:

```
<?php
  $headers = "";

  if(isset($_REQUEST["cc"])){
    $headers .= "cc:" . $_REQUEST["cc"] . "\r\n";
  }
    .
    .
    .
```

Y obtener el encabezado bcc también, si acaso hay uno:

```
<?php
  $headers = "";

  if(isset($_REQUEST["cc"])){
    $headers .= "cc:" . $_REQUEST["cc"] . "\r\n";
  }

  if(isset($_REQUEST["bcc"])){
    $headers .= "bcc:" . $_REQUEST["bcc"] . "\r\n;
  }
    .
    .
    .
?>
```

Por último, podría enviar el correo electrónico, incluidos los encabezados, en phpadvemail.php:

```
<html>
  <head>
    <title>
      Se envió su correo electrónico
    </title>
  </head>

  <body>
    <h1>Se envió su correo electrónico</h1>
    Gracias por su mensaje con encabezados.
    <br>
    <?php
      $headers = "";

      if(isset($_REQUEST["cc"])){
        $headers .= "cc:" . $_REQUEST["cc"] . "\r\n";
      }
```

FIGURA 11-15 Envío de correo electrónico con encabezados

```
    if(isset($_REQUEST["bcc"])){
      $headers .= "bcc:" . $_REQUEST["bcc"] . "\r\n;
    }

    $result = mail("steve@ispname.com", "Web mail", $_REQUEST["message"]),
      $headers);
  ?>
 <body>
</html>
```

Podría ver los resultados en la Figura 11-15, donde envió el correo electrónico —completo con encabezados cc y bcc.

Cómo agregar archivos adjuntos al correo electrónico

También puede enviar archivos adjuntos con su correo electrónico desde PHP, aunque requiere trabajo adicional. Éste es un ejemplo, phpemailattachments.php, para enviar un archivo de imagen, image.jpg.

Comenzaremos clasificando los datos necesarios para este correo electrónico en diferentes variables —observe que $attachment contiene el nombre del archivo de imagen a ser adjunto y $attachment_MIME_type define el tipo MIME (Multipurpose Internet Mail Extension) del archivo adjunto:

```
<?php
  $to = "steve@ispname.com";
  $subject = "Correo Web";
  $message = "Este correo electrónico tiene un archivo adjunto.";
  $attachment = "image.jpg";
  $attachment_MIME_type = "image/jpeg";
    .
    .
    .
```

Ahora puede trabajar con el archivo adjunto y codificarlo para su inclusión en el mensaje. Es aquí donde se requiere más trabajo, pues debe crear una forma de múltiples partes. Puede comenzar leyendo los datos del archivo adjunto en una variable llamada $data:

```
<?php
  $to = "steve@ispname.com";
  $subject = "Correo Web";
  $message = "Este correo electrónico tiene un archivo adjunto.";
  $attachment = "image.jpg";
  $attachment_MIME_type = "image/jpeg";

  $handle = fopen ($attachment, "rb");
  $data = fread ($handle, filesize($attachment));
  fclose ($handle);
    .
    .
    .
```

Ahora vamos a trabajar en la creación de nuestra forma de múltiples partes, comenzando con la línea divisoria separando mensaje de correo electrónico, de datos adjuntos:

```
<?php
  $to = "steve@ispname.com";
  $subject = "Correo Web";
  $message = "Este correo electrónico tiene un archivo adjunto.";
  $attachment = "image.jpg";
  $attachment_MIME_type = "image/jpeg";

  $handle = fopen ($attachment, "rb");
  $data = fread ($handle, filesize($attachment));
  fclose ($handle);

  $boundary = "---Multipart_Boundary---";

  $headers = "\nMIME-Version: 1.0\n" .
  "Content-Type: multipart/mixed;\n" .
  " boundary=\"" . $boundary . "\"";
    .
    .
    .
```

Ahora debe codificar los datos adjuntos binarios en texto usados por el correo electrónico y para hacerlo use las funciones chunk_split y base64_encode de PHP:

```
<?php
  $to = "steve@ispname.com";
  $subject = "Correo Web";
  $message = "Este correo electrónico tiene un archivo adjunto.";
  $attachment = "image.jpg";
  $attachment_MIME_type = "image/jpeg";
```

```
$handle = fopen ($attachment, "rb");
$data = fread ($handle, filesize($attachment));
fclose ($handle);

$boundary = "---Multipart_Boundary---";

$headers = "\nMIME-Version: 1.0\n" .
"Content-Type: multipart/mixed;\n" .
" boundary=\"" . $boundary . "\"";

$data = chunk_split(base64_encode($data));
  .
  .
  .
```

Luego ensamble el texto del mensaje, incluidos los datos adjuntos codificados en texto, y envíe por correo todo el paquete de esta manera:

```
<?php
  $to = "steve@ispname.com";
    .
    .
    .
  $boundary = "---Multipart_Boundary---";

  $headers = "\nMIME-Version: 1.0\n" .
  "Content-Type: multipart/mixed;\n" .
  " boundary=\"" . $boundary . "\"";

  $data = chunk_split(base64_encode($data));

  $text = "--" . $boundary . "\n" .
  "Content-Type:text/plain\nContent-Transfer-Encoding: 7bit\n\n" .
  $message . "\n\n--" . $boundary . "\n" .
  "Content-Type: " . $attachment_MIME_type . ";\n name=\"" .
  $attachment . "\"\nContent-Transfer-Encoding: base64\n\n" .
  $data . "\n\n--" . $boundary . "--\n";

  $result = mail($to, $subject, $text, $headers);
    .
    .
    .
?>
```

Por último, reporte al usuario lo que sucedió, de la siguiente forma en phpemailattachments.php:

```
<html>
  <head>
    <title>
```

```
      Envío de correo electrónico con archivos adjuntos
    </title>
  </head>

  <body>
    <h1>
      Envío de correo electrónico con archivos adjuntos
    </h1>
      <?php
      $to = "steve@ispname.com";
      $subject = "Correo Web";
      $message = "Este correo electrónico tiene un archivo adjunto.";
      $attachment = "image.jpg";
      $attachment_MIME_type = "image/jpeg";

      $handle = fopen ($attachment, "rb");
      $data = fread ($handle, filesize($attachment));
      fclose ($handle);

      $boundary = "---Multipart_Boundary---";

      $headers = "\nMIME-Version: 1.0\n" .
      "Content-Type: multipart/mixed;\n" .
      " boundary=\"" . $boundary . "\"";

      $data = chunk_split(base64_encode($data));

      $text = "--" . $boundary . "\n" .
      "Content-Type:text/plain\nContent-Transfer-Encoding: 7bit\n\n" .
      $message . "\n\n--" . $boundary . "\n" .
      "Content-Type: " . $attachment_MIME_type . ";\n name=\"" .
      $attachment . "\"\nContent-Transfer-Encoding: base64\n\n" .
      $data . "\n\n--" . $boundary . "--\n";

      $result = mail($to, $subject, $text, $headers);
      if($result) {
        echo "Se envió su correo electrónico.";
      } else {
        echo "No se envió su correo electrónico.";
      }
    ?>
  <body>
</html>
```

Genial. Vea los resultados en la Figura 11-16, donde se envió el correo electrónico, incluido el archivo adjunto.

FIGURA 11-16 Envío de correo electrónico con archivos adjuntos

Almacenaje de datos en sesiones

Por su naturaleza, las páginas Web sólo alojan datos temporales, a menos que usted almacene específicamente sus datos en el servidor. Una forma de almacenar datos en el servidor consiste en usar *sesiones*. Mediante sesiones puede almacenar y recuperar datos por nombre. Para trabajar con una sesión, se comienza llamando a la función session_start:

```
session_start();
        .
        .
        .
```

Para almacenar datos en la sesión, use matriz $_SESSION. Por ejemplo, aquí almacenamos "Rear Window"bajo la clave"movie":

```
session_start();
$SESSION['movie'] = "Rear Window";
        .
        .
        .
```

Ahora en otro acceso a la página —la misma o una página diferente—, puede acceder los datos bajo la clave"movie"usando $_SESSION nuevamente:

```
session_start();
$movie = $SESSION['movie'];
        .
        .
        .
```

Como apreciará, puede preservar datos entre accesos a páginas. Dichos datos no se conservarán por siempre —desaparecerán en algún momento, en general de 30 a 180 minutos, dependiendo de su instalación PHP.

El comportamiento de las sesiones se ve afectado por tales parámetros en el archivo de inicialización de PHP, php.ini —observe, por ejemplo, que puede especificar el tiempo de almacenamiento de datos de la sesión, definiendo un valor para session.cache_expire en minutos:

- session.save_path“/tmp”
- session.name“PHPSESSID”
- session.save_handler“files”
- session.auto_start“0”
- session.gc_probability“1”
- session.gc_divisor“100”
- session.gc_maxlifetime“1440”
- session.serialize_handler“php”
- session.cookie_lifetime“0”
- session.cookie_path“/”
- session.cookie_domain“”
- session.cookie_secure“”
- session.use_cookies“1”
- session.use_only_cookies“0”
- session.referer_check“”
- session.entropy_file“”
- session.entropy_length“0”
- session.cache_limiter“nocache”
- session.cache_expire“180”
- session.use_trans_sid“0”PHP_INI_SYSTEM | PHP_INI_PERDIR

Podría observar, específicamente, que todos los datos de una sesión particular se almacenarán en un archivo, en el directorio especificado por el elemento session.save_path. Se creará un archivo para cada sesión (haya o no datos asociados con dicha sesión).

Bueno, es el momento de un ejemplo, phpsession.php, para almacenar el total de la compra hecha por el usuario en los datos de la sesión. Primero, se comienza con session_start:

```
<?php
  session_start();
    .
    .
    .
?>
```

Luego, almacene el total de la compra del usuario con la clave de sesión "purchase":

```
<?php
  session_start();
  $_SESSION['purchase'] = "39.25";
?>
```

Así luce, en phpsession.php:

```
<html>
  <head>
    <title>
      Almacenaje de datos en sesiones
    </title>
  </head>

  <body>
    <h1>
      Almacenaje de datos en sesiones
    </h1>

    <?php
      session_start();
      $_SESSION['purchase'] = "39.25";
    ?>

    Se almacenó su compra, $39.35.
    <br>
    Para leer su compra en una nueva página, <a href="phpsession2.php"> haga clic
      aquí</a>.
    </body>
</html>
```

Los resultados se encuentran en la Figura 11-17, donde el total de compra del usuario se almacenó en la sesión.

Este script, phpsession.php, tiene un hipervínculo a otra página, phpsession2.php, que lee los datos de sesión y muestra. En phpsession2.php, se comienza con session_start:

```
<?php
  session_start();
    .
    .
    .
  ?>
```

La información que busca está almacenada con la clave "compra" (pero antes de intentar mostrarla, debe revisar si hay una entrada en la matriz $_SESSION con esa clave para evitar errores:

```
<?php
  session_start();
```

FIGURA 11-17 Almacenaje de datos en una sesión

```
    if(isset($_SESSION["purchase"])){
      .
      .
      .
  }
  ?>
```

Si los datos que busca existen, puede mostrarlos de esta forma:

```
<?php
  session_start();

  if(isset($_SESSION["purchase"])){
    echo "Bienvenido. Ha comprado \$" . $_SESSION['purchase'] . " en mercancía.";
  }
  ?>
```

Así se ve en phpsession2.php:

```
<html>
  <head>
    <title>
      Recuperación de datos de sesiones
    </title>
  </head>

  <body>
    <h1>
      Recuperación de datos de sesiones
    </h1>
```

FIGURA 11-18 Recuperación de datos de sesiones

```
    <?php
      session_start();

      if(isset($_SESSION["purchase"])){
        echo "Bienvenido. Ha comprado \$" . $_SESSION['purchase'] . " en mercancía.";
      }
      ?>
  </body>
</html>
```

Y ver los resultados en la Figura 11-18, donde los datos almacenados en la sesión se recuperaron con éxito.

Cómo escribir un contador de visitas utilizando sesiones

Éste es otro ejemplo mostrando la forma en que las sesiones preservan datos entre accesos a la página: un contador de visitas a páginas Web indicando cuántas veces ha visitado el usuario una página Web (no utilice éste como contador de visitas real, pues sólo demuestra el funcionamiento de las sesiones —después de que el usuario esté inactivo, durante el periodo de extinción de la sesión, los datos serán eliminados).

Así comienza este ejemplo, phpcounter.php:

```
<?php
  session_start();
  .
  .
  .
?>
```

Si el usuario ha visitado esta página antes, ésta tendrá datos almacenados en la sesión, bajo la clave "count"; de tal modo, podrá comprobar si existen dichos datos:

```
<?php
  session_start();
  if (!isset($_SESSION['count'])) {
  .
  .
  .
  }
?>
```

Si no hay datos en la sesión bajo la clave "count", puede almacenar los datos en la sesión, estableciendo el valor en 0:

```
<?php
  session_start();
  if (!isset($_SESSION['count'])) {
    $_SESSION['count'] = 0;
  .
  .
  .
  }
?>
```

En caso contrario, incremente el valor almacenado en la sesión. Entonces la página mostrará ese valor:

```
<html>
  <head>
    <title>
      Contador de visitas de una sesión
    </title>
  </head>

  <body>
    <h1>
      Contador de visitas de una sesión
    </h1>
      Hola a todos. Usted ha visitado esta página
      <?php
        session_start();
        if (!isset($_SESSION['count'])) {
          $_SESSION['count'] = 0;
        } else {
          $_SESSION['count']++;
        }
        echo $_SESSION['count'];
```

FIGURA 11-19 Inicio del contador de visitas

```
      ?>
      veces antes.
    <body>
</html>
```

Y los resultados están en la Figura 11-19; el usuario ha navegado hasta la página y el contador de visitas se establece en 0.

Después de que el usuario carga la página de nuevo un par de veces, los resultados se aprecian en la Figura 11-20, donde se almacenaron los datos del contador de visitas en la sesión.

FIGURA 11-20 Uso del contador de visitas

12 CAPÍTULO

Ajax

Ajax es un tema de actualidad y la base de lo que se ha denominado Web 2.0. Empleando técnicas Ajax, puede crear aplicaciones basadas en la Web, con aspecto de aplicaciones de escritorio. La diferencia principal radica en que las aplicaciones Ajax no actualizan la pantalla completa del navegador cada vez que el usuario hace algo, un recurso muy propio de la Web. Usando Ajax puede comunicarse con el servidor tras el telón, descargar datos y mostrarlos en una sección específica de la página Web, sin cargar de nuevo la página completa.

Ajax (cuyas siglas significan Asynchronous JavaScript and XML, JavaScript y XML asíncrono) se basa en el uso de JavaScript en el navegador. El lenguaje más común en el servidor para usar con Ajax es PHP, así que esta tecnología merece nuestra atención.

Comience a utilizar Ajax

Éste es un ejemplo de Ajax para echar a andar las cosas —hay mucha tecnología basada en el cliente (es decir, basada en el navegador) que aprender aquí—. Este ejemplo se llamará index.html y aparece en la Figura 12-1.

Este sencillo ejemplo sólo captura el contenido de un archivo de texto del servidor, data.txt, cuando hace clic en el botón. La diferencia entre esta aplicación y una página Web ordinaria es que al hacer clic en el botón, la página no parpadea y se muestra de nuevo —todo lo que sucede es que el texto"El mensaje capturado aparecerá aquí.", se reemplaza por el contenido de data.txt, tomado del servidor—"Hola de Ajax.", como se aprecia en la Figura 12-2.

Esta aplicación luce y trabaja como una utilidad de escritorio —cuando trabaja en un procesador de palabras, escribe un carácter, por ejemplo, la página completa no parpadea cuando muestra el nuevo carácter—. Ésa es la aportación de Ajax a la programación Web —no tiene que

FIGURA 12-1 Demostración de Ajax

cargar la página completa de nuevo, cuando el usuario hace algo y usted necesita interactuar con el servidor—. Toda la interacción con el servidor se puede llevar a cabo tras el telón, descargas y cargas de datos, sólo se actualizarán secciones específicas de la página Web, en vez de cargar la página Web completa de nuevo.

Daremos un vistazo al trabajo con Ajax, usándolo para capturar el contenido de un archivo de texto, antes de pasar al uso de Ajax con PHP, antes de ver cómo enviar datos a scripts PHP recurriendo a técnicas Ajax y antes de leer lo que le devuelven dichos scripts.

FIGURA 12-2 Captura de un mensaje utilizando Ajax

Redacción en Ajax

Así se ve nuestro primer ejemplo, index.html —dedicaremos algo de tiempo a entender cómo trabaja Ajax con este ejemplo:

```
<html>
  <head>
    <title>Demostración de Ajax</title>

    <script language = "javascript”>
      var XMLHttpRequestObject = false;

      if (window.XMLHttpRequest) {
        XMLHttpRequestObject = new XMLHttpRequest();
      } else if (window.ActiveXObject) {
        XMLHttpRequestObject = new
          ActiveXObject("Microsoft.XMLHTTP");
      }

      function getData(dataSource, divID)
      {
        if(XMLHttpRequestObject) {
          var obj = document.getElementById(divID);
          XMLHttpRequestObject.open("GET", dataSource);

          XMLHttpRequestObject.onreadystatechange = function()
          {
            if (XMLHttpRequestObject.readyState == 4 &&
              XMLHttpRequestObject.status == 200) {
                obj.innerHTML = XMLHttpRequestObject.responseText;
            }
          }

          XMLHttpRequestObject.send(null);
        }
      }
    </script>
  </head>

  <body>
    <H1>Demostración de Ajax</H1>

    <form>
      <input type = "button” value = "Capturar el mensaje"
        onclick = "getData('data.txt', 'targetDiv')">
    </form>

    <div id="targetDiv">
      <p>El mensaje capturado aparecerá aquí.</p>
    </div>

  </body>
</html>
```

Bueno, es momento de hacer a un lado este ejemplo —aquí aprenderá a escribir aplicaciones Ajax/PHP completas—. La presente acción ocurre cuando el usuario hace clic en el botón con la leyenda "Capturar el mensaje" y es así como se ve el botón en forma HTML en index.html:

```
<body>

    <H1>Demostración de Ajax</H1>

    <form>
      <input type = "button" value = "Capturar el mensaje"
        onclick = "getData('data.txt', 'targetDiv')">
    </form>

<body>
```

Este botón se conecta a una función JavaScript, getData y pasa dos argumentos a esa función —"data.txt", el nombre del archivo que desea capturar del servidor y "targetDiv", el nombre del elemento donde se mostrará el texto descargado. Ese elemento , mostrando "El mensaje capturado aparecerá aquí.", luce así index.html:

```
<body>

    <H1>Demostración de Ajax</H1>

    <form>
      <input type = "button" value = "Capturar el mensaje"
        onclick = "getData('data.txt', 'targetDiv')">
    </form>

    <div id="targetDiv">
      <p>El mensaje capturado aparecerá aquí.</p>
    </div>

  </body>
```

Cuando se descargue el texto en data.txt, se presentará en este elemento , targetDiv, como se ve en la Figura 12-2.

La acción real en esta aplicación se lleva a cabo en JavaScript, incluida la función getData.

Creación del objeto XMLHttpRequest

Ajax funciona en torno al objeto XMLHttpRequest disponible en la mayoría de navegadores modernos. Este objeto le permite conectarse con el servidor tras bambalinas, pasando datos a scripts PHP en el servidor y dando lectura a las respuestas de esos scripts.

JavaScript en este ejemplo comienza con la creación de un objeto XMLHttpRequest, que almacena en una variable llamada XMLHttpRequestObject. El código JavaScript, contenido en un elemento <script> de HTML, comienza por declarar dicha variable y la establece con valor falso (de tal modo, si no tenemos éxito al crear un objeto XMLHttpRequest, esta variable se quedará con el valor FALSE, cuyo código subsecuente puede probar):

```
<script language = "javascript">
  var XMLHttpRequestObject = false;
    .
    .
    .
</script>
```

Ahora debe crear el objeto XMLHttpRequest (proceso que difiere, dependiendo del navegador empleado). Puede comenzar creando este objeto en Netscape Navigator (versión 7.0 y posteriores), Apple Safari (versión 1.2 y posteriores) y Firefox, a través de window.XMLHttpRequest, si existe tal objeto, que puede probar de la siguiente manera:

```
<script language = "javascript">
  var XMLHttpRequestObject = false;

  if (window.XMLHttpRequest) {
    .
    .
    .
  }
</script>
```

Mientras exista window.XMLHttpRequest, es posible crear un nuevo objeto XMLHttpRequest de esta forma, utilizando la expresión new XMLHttpRequest( ):

```
<script language = "javascript">
  var XMLHttpRequestObject = false;

  if (window.XMLHttpRequest) {
    XMLHttpRequestObject = new XMLHttpRequest();
  }
</script>
```

Eso se encarga de Firefox, Navigator y Safari. Ahora tiene un objeto XMLHttpRequest, listo para ser usado en esos navegadores.

¿Qué hay de Internet Explorer? En éste se crean objetos XMLHttpRequest como objeto ActiveX; de modo que primero pruebe si puede crear objetos ActiveX:

```
<script language = "javascript">
  var XMLHttpRequestObject = false;

  if (window.XMLHttpRequest) {
    XMLHttpRequestObject = new XMLHttpRequest();
  } else if (window.ActiveXObject) {
    .
    .
    .
  }
</script>
```

Si es posible crear objetos ActiveX, entonces puede crear un objeto new XMLHttpRequest usando la expresión ActiveXObject("Microsoft.XMLHTTP") en Internet Explorer:

```
<script language = "javascript">
  var XMLHttpRequestObject = false;

  if (window.XMLHttpRequest) {
    XMLHttpRequestObject = new XMLHttpRequest();
  } else if (window.ActiveXObject) {
    XMLHttpRequestObject = new
      ActiveXObject("Microsoft.XMLHTTP");
  }
</script>
```

En este punto, entonces, ha creado un objeto XMLHttpRequest para todos los navegadores principales. Propiedades y métodos principales de este objeto son los mismos en los navegadores (puede ver todas las propiedades y métodos de este objeto por navegador, en las Tablas 12-1 a 12-6).

Propiedad	Descripción
onreadystatechange	Contiene el nombre del manejador de eventos al que se debe llamar cuando cambia el valor de la propiedad readyState. Lectura/escritura.
reaqdyState	Contiene el estado de la solicitud. Sólo lectura.
responseBody	Contiene un cuerpo de respuesta, es una forma en la que se pueden devolver respuestas HTTP. Sólo lectura.
responseStream	Contiene un flujo de respuesta, un flujo binario hacia el servidor. Sólo lectura.
responseText	Contiene el cuerpo de respuesta como cadena. Sólo lectura.
responseXML	Contiene el cuerpo de respuesta como XML. Sólo lectura.
status	Contiene el código de estado HTTP que devuelve una solicitud. Sólo lectura.
statusText	Contiene el texto del estado de respuesta HTTP. Sólo lectura.

TABLA 12-1 Propiedades del objeto XMLHttpRequest para Internet Explorer

Método	Descripción
abort	Aborta la solicitud HTTP.
getAllResponseHeaders	Devuelve todos los encabezados HTTP.
getResponseHeader	Devuelve el valor de un encabezado HTTP.
open	Abre una solicitud al servidor.
send	Envía una solicitud HTTP al servidor.
setRequestHeader	Establece nombre y valor de un encabezado HTTP.

TABLA 12-2 Métodos del objeto XMLHttpRequest para Internet Explorer

Propiedad	Descripción
channel	Contiene el canal usado para realizar la solicitud. Sólo lectura.
readyState	Contiene el estado de la solicitud. Sólo lectura.
responseText	Contiene el cuerpo de la respuesta como cadena. Sólo lectura.
responseXML	Contiene el cuerpo de la respuesta como XML. Sólo lectura.
status	Contiene el código de estado HTTP que devuelve una solicitud. Sólo lectura.
statusText	Contiene el texto del estado de repuesta HTTP. Sólo lectura.

Tabla 12-3 Propiedades del objeto XMLHttpRequest para Mozilla, Firefox y NetScape Navigator

Método	Descripción
abort	Aborta la solicitud HTTP.
getAllResponseHeaders	Devuelve todos los encabezados HTTP.
getResponseHeader	Devuelve el valor de un encabezado HTTP.
openRequest	Método nativo (no script) para abrir una solicitud.
overrideMimeType	Reemplaza el tipo MIME que devuelve el servidor.

Tabla 12-4 Métodos del objeto XMLHttpRequest para Mozilla, Firefox y NetScape Navigator

Propiedad	Descripción
onreadystatechange	Contiene el nombre del manejador de eventos al que se debe llamar cuando cambie el valor de la propiedad readyState. Lectura/escritura.
readyState	Contiene el estado de la solicitud. Sólo lectura.
responseText	Contiene el cuerpo de la respuesta como cadena. Sólo lectura.
responseXML	Contiene el cuerpo de la respuesta como XML. Sólo lectura.
status	Contiene el código de estado HTTP que devuelve una solicitud. Sólo lectura.
statusText	Contiene el texto del estado de repuesta HTTP. Sólo lectura.

Tabla 12-5 Propiedades del objeto XMLHttpRequest para Apple Safari

Método	Descripción
abort	Aborta la solicitud HTTP.
getAllResponseHeaders	Devuelve todos los encabezados HTTP.
getResponseHeader	Devuelve el valor de un encabezado HTTP.
open	Abre una solicitud al servidor.
send	Envía una solicitud HTTP al servidor.
setRequestHeader	Establece nombre y valor de un encabezado HTTP.

Tabla 12-6 Métodos del objeto XMLHttpRequest para Apple Safari

Bueno, ahora tiene un objeto XMLHttpRequest —el siguiente paso es abrirlo, esto lo prepara para interactuar con el servidor.

Cómo abrir el objeto XMLHttpRequest

Para trabajar con un objeto XMLHttpRequest y conectarse al servidor, primero debe abrir ese objeto. Eso sucede en el código de este ejemplo, en la función getData, a la que se pasa el nombre del archivo que ha de capturarse del servidor (como argumento dataSource) y el nombre del elemento para mostrar el texto capturado (como argumento divID):

```
<script language = "javascript">
  var XMLHttpRequestObject = false;

  if (window.XMLHttpRequest) {
    XMLHttpRequestObject = new XMLHttpRequest();
  } else if (window.ActiveXObject) {
    XMLHttpRequestObject = new
      ActiveXObject("Microsoft.XMLHTTP");
  }

  function getData(dataSource, divID)
  {
    .
    .
    .
  }
</script>
```

Ahora puede comprobar si el código para crear el objeto XMLHttpRequest ha tenido éxito en su tarea encomendada. Ese código de creación, fuera de cualquier función de JavaScript, se ejecuta apenas se carga la página; así puede probar si ha creado un objeto XMLHttpRequest, de la siguiente forma en la función getData:

```
<script language = "javascript">
  var XMLHttpRequestObject = false;

  if (window.XMLHttpRequest) {
    XMLHttpRequestObject = new XMLHttpRequest();
  } else if (window.ActiveXObject) {
    XMLHttpRequestObject = new
      ActiveXObject("Microsoft.XMLHTTP");
  }

  function getData(dataSource, divID)
  {
    if(XMLHttpRequestObject) {
    .
    .
    .
    }
  }
</script>
```

Es el momento de abrir XMLHttpRequest, que lo configura para usarse con el servidor (no conecta el objeto con el servidor); así se utiliza ese método:

```
open("method", "URL"[, asyncFlag[, "userName"[, "password"]]])
```

Éste es el significado de los parámetros:

- **method** Es el método HTTP empleado para abrir la conexión, como GET, POST, PUT, HEAD o PROPFIND.
- **URL** La dirección URL solicitada.
- **asyncFlag** Valor booleano que indica si la llamada es asíncrona. El valor predeterminado es verdadero.
- **userName** Nombre del usuario.
- **password** Contraseña.

Así se abre el objeto XMLHttpRequest en este ejemplo, index.html, configurándolo para usar el método GET para conectarse con el servidor y descargar data.txt (que se pasa a la función getData, en el argumento dataSource):

```
<script language = "javascript">
  var XMLHttpRequestObject = false;

  if (window.XMLHttpRequest) {
    XMLHttpRequestObject = new XMLHttpRequest();
  } else if (window.ActiveXObject) {
    XMLHttpRequestObject = new
      ActiveXObject("Microsoft.XMLHTTP");
  }

  function getData(dataSource, divID)
  {
    if(XMLHttpRequestObject) {

      XMLHttpRequestObject.open("GET", dataSource);
    .
    .
    .
  }
</script>
```

Ha configurado el objeto XMLHttpRequest, que lo deja listo para trabajar con el servidor. A continuación, debe hacer provisiones para manejar los datos descargados del servidor.

Manejo de datos descargados

Ajax realiza su trabajo *de forma asíncrona*, significa que no espera el regreso de los datos del servidor antes de permitir el navegador atienda otras tareas. Específicamente, significa que debe crear una función *callback*. Se llama a esa función callback cuando se descargan los datos, de modo que coloque el código manejando la descarga en esa función callback.

La función callback se conecta al objeto XMLHttpRequest mediante la propiedad onreadystatechange de ese objeto. Se llamará una función *asignada* a esa propiedad, cuando ocurran eventos de descarga de datos.

Así se asigna una función a la propiedad onreadystatechange, del objeto XMLHttpRequest:

```
<script language = "javascript">
  var XMLHttpRequestObject = false;

  if (window.XMLHttpRequest) {
    XMLHttpRequestObject = new XMLHttpRequest();
  } else if (window.ActiveXObject) {
    XMLHttpRequestObject = new
      ActiveXObject("Microsoft.XMLHTTP");
  }

  function getData(dataSource, divID)
  {
    if(XMLHttpRequestObject) {
      XMLHttpRequestObject.open("GET", dataSource);

      XMLHttpRequestObject.onreadystatechange = function()
      {
      .
      .
      .
      }

    }
  }
</script>
```

Cuando se llama a esta función, el estado de descarga de sus datos ha cambiado y puede comprobarse empleando dos propiedades más del objeto XMLHttpRequest (readyState y status). La propiedad readyState indica cómo procede la descarga de datos. Éstos son los posibles valores para esta propiedad (un valor de 4 es lo que usted desea ver, ya que significa que los datos se han descargado completamente):

- 0 no inicializado
- 1 cargando
- 2 cargado
- 3 interactivo
- 4 completo

La propiedad status contiene el estado real de la descarga. En realidad es el código de estado HTTP normal, obtenido cuando se descargan páginas Web (por ejemplo, si no se encontraron los datos que usted busca, obtendrá el valor 404 en la propiedad status). Éstos son algunos de los posibles valores (observe que deseará ver un valor 200 aquí, significa que la descarga se completó normalmente):

- **200** Bien
- **201** Creado
- **204** Sin contenido
- **205** Reiniciar contenido
- **206** Contenido parcial
- **400** Solicitud incorrecta
- **401** No autorizado
- **403** Prohibido
- **404** No encontrado
- **405** Método no permitido
- **406** No aceptable
- **407** Se requiere autenticación del proxy
- **408** Tiempo agotado de la solicitud
- **411** Longitud requerida
- **413** Entidad solicitada demasiado larga
- **414** URL solicitada demasiado larga
- **415** Tipo de medio sin soporte
- **500** Error interno del servidor
- **501** No implementado
- **502** Gateway incorrecto
- **503** Servicio no disponible
- **504** Tiempo agotado del gateway
- **505** Versión de HTTP sin soporte

Entonces, en la función conectada a la propiedad onreadystatechange puede comprobar si sus datos se han descargado, verificando si la propiedad readyState es igual a 4:

```
<script language = "javascript">
  function getData(dataSource, divID)
  {
    if(XMLHttpRequestObject) {

      XMLHttpRequestObject.open("GET", dataSource);

      XMLHttpRequestObject.onreadystatechange = function()
      {
        if (XMLHttpRequestObject.readyState == 4 &&
        .
        .
        .
      }
    }
  }
</script>
```

Y puede comprobar si todo ha salido bien, verificando que la propiedad status tiene el valor 200:

```
<script language = "javascript">
  function getData(dataSource, divID)
  {
    if(XMLHttpRequestObject) {
      var obj = document.getElementById(divID);
      XMLHttpRequestObject.open("GET", dataSource);
      XMLHttpRequestObject.onreadystatechange = function()
      {
        if (XMLHttpRequestObject.readyState == 4 &&
          XMLHttpRequestObject.status == 200) {
          .
          .
          .
        }
      }
    }
  }
</script>
```

Si recibió los datos del servidor, puede mostrarlos en el elemento targetDiv, de la página Web, con sólo un poco de código Dynamic HTML sin actualizar la página. Así se define targetDiv en la página Web:

```
<body>

    <H1>Demostración de Ajax</H1>
    <form>
      <input type = "button" value = "Capturar el mensaje"
        onclick = "getData('data.txt', 'targetDiv')">
    </form>
    <div id="targetDiv">
      <p>El mensaje capturado aparecerá aquí.</p>
    </div>
  </body>
```

Puede conseguir un objeto JavaScript correspondiente a ese elemento de la siguiente forma:

```
<script language = "javascript">
  function getData(dataSource, divID)
  {
    if(XMLHttpRequestObject) {
      var obj = document.getElementById(divID);
      XMLHttpRequestObject.open("GET", dataSource);

      XMLHttpRequestObject.onreadystatechange = function()
      {
        if (XMLHttpRequestObject.readyState == 4 &&
          XMLHttpRequestObject.status == 200) {
```

```
            .
            .
            .
          }
        }
        XMLHttpRequestObject.send(null);
      }
    }
</script>
```

Y, cuando los datos se han descargado correctamente, puede mostrarlos en el elemento —al usar objetos XMLHttpRequest, su texto se descarga a la propiedad responseText, de modo que ésta es la forma en que maneja esa descarga en el ejemplo siguiente:

```
<script language = "javascript">
  function getData(dataSource, divID)
  {
    if(XMLHttpRequestObject) {
      var obj = document.getElementById(divID);
      XMLHttpRequestObject.open("GET", dataSource);

      XMLHttpRequestObject.onreadystatechange = function()
      {
        if (XMLHttpRequestObject.readyState == 4 &&
          XMLHttpRequestObject.status == 200) {
            obj.innerHTML = XMLHttpRequestObject.responseText;
        }
      }
      XMLHttpRequestObject.send(null);
    }
  }
</script>
```

Inicio de la descarga

Bueno, ya tiene todo listo para iniciar la descarga —ha abierto el objeto XMLHttpRequest para configurarlo y también configurado la función callback—. Ahora debe iniciar la conexión al servidor y todo el proceso de descarga.

Eso se hace con el método send de XMLHttpRequest. Cuando usa el método GET, se pasa un valor null al método send de esta forma:

```
<script language = "javascript">
  var XMLHttpRequestObject = false;
  if (window.XMLHttpRequest) {
    XMLHttpRequestObject = new XMLHttpRequest();
  } else if (window.ActiveXObject) {
    XMLHttpRequestObject = new
      ActiveXObject("Microsoft.XMLHTTP");
  }
```

```
    function getData(dataSource, divID)
    {
      if(XMLHttpRequestObject) {
        var obj = document.getElementById(divID);
        XMLHttpRequestObject.open("GET", dataSource);

        XMLHttpRequestObject.onreadystatechange = function()
        {
          if (XMLHttpRequestObject.readyState == 4 &&
            XMLHttpRequestObject.status == 200) {
              obj.innerHTML = XMLHttpRequestObject.responseText;
          }
        }

        XMLHttpRequestObject.send(null);
      }
    }
  </script>
```

Eso lo conecta con el servidor e inicia la descarga. Como verá, cuando utiliza el método POST, pasa los datos que desea enviar al servidor, si los hay, al método send.

Eso completa este ejemplo —ha inicializado el objeto XMLHttpRequest, ha configurado una función callback para manejar la descarga de datos y ha iniciado el acceso a los datos.

Creación de objetos XMLHttpRequest

Hasta ahora ha creado objetos XMLHttpRequest de esta manera:

```
<script language = "javascript">
  var XMLHttpRequestObject = false;

  if (window.XMLHttpRequest) {
    XMLHttpRequestObject = new XMLHttpRequest();
  } else if (window.ActiveXObject) {
    XMLHttpRequestObject = new
      ActiveXObject("Microsoft.XMLHTTP");
  }
</script>
```

Sin embargo, algunos navegadores dan soporte a diferentes versiones de objetos XMLHttpRequest y podría percatarse que desea usar una versión más avanzada (observe que el código anterior funcionará bien para nosotros en este libro). Internet Explorer, por ejemplo, da soporte a estas versiones más recientes: MSXML2.XMLHTTP, MSXML2.XMLHTTP3.0, MSXML2.XMLHTTP4.0 o bien MSXML2.XMLHTTP.5.0.

Así se crea un objeto XMLHttpRequest, mediante MSXML2.XMLHTTP, por ejemplo. JavaScript soporta una construcción try/catch, que puede utilizar para el manejo de errores, de modo que podría crear un objeto MSXML2.XMLHTTP recurriendo a un bloque try de esta forma:

```
<script language = "javascript">
  var XMLHttpRequestObject = false;
```

```
  try {
    XMLHttpRequestObject = new ActiveXObject("MSXML2.XMLHTTP");
  }
    .
    .
    .
</script>
```

Si hubo un error (el navegador no pudo crear un objeto MSXML2.XMLHTTP), puede capturar el error en un bloque catch e intentarlo de nuevo, creando un objeto Microsoft.XMLHTTP estándar:

```
<script language = "javascript">
  var XMLHttpRequestObject = false;

  try {
    XMLHttpRequestObject = new ActiveXObject("MSXML2.XMLHTTP");
  } catch (exception1) {
    try {
      XMLHttpRequestObject = new
          ActiveXObject("Microsoft.XMLHTTP");
    }
    .
    .
    .
  }
</script>
```

Y si eso no funcionara, puede cerciorarse de que XMLHttpRequestObject se haya establecido en falso:

```
<script language = "javascript">
  var XMLHttpRequestObject = false;

  try {
    XMLHttpRequestObject = new ActiveXObject("MSXML2.XMLHTTP");
  } catch (exception1) {
    try {
      XMLHttpRequestObject = new
          ActiveXObject("Microsoft.XMLHTTP");
    } catch (exception2) {
      XMLHttpRequestObject = false;
    }
  }
</script>
```

Si XMLHttpRequestObject se queda con FALSE, tras todo ese código, entonces no usa Internet Explorer y puede crear un objeto XMLHttpRequest para otros navegadores de esta forma:

```
<script language = "javascript">
  var XMLHttpRequestObject = false;
```

```
    try {
      XMLHttpRequestObject = new ActiveXObject("MSXML2.XMLHTTP");
    } catch (exception1) {
      try {
        XMLHttpRequestObject = new
            ActiveXObject("Microsoft.XMLHTTP");
      } catch (exception2) {
        XMLHttpRequestObject = false;
      }
    }

    if (!XMLHttpRequestObject && window.XMLHttpRequest) {
      XMLHttpRequestObject = new XMLHttpRequest();
    }
  </script>
```

Este código creará, si puede, un objeto MSXML2.XMLHTTP XMLHttpRequest en Internet Explorer o Microsoft.XMLHTTP, si no puede.

Hasta ahora ha descargado tan sólo un archivo de texto, pero Ajax se emplea más a menudo con PHP en el servidor; así que veamos eso ahora.

Ajax con algo de PHP

Suponga que tuviera un script de PHP llamado data.php, para reproducir texto como éste:

```
<?php
    echo 'Este texto se capturó del servidor con Ajax y PHP.';
?>
```

Puede descargar el texto mostrado usando Ajax, de esta forma, en una nueva versión de index.html, index2.html:

```
<html>
  <head>
    <title>Demostración de Ajax y PHP</title>

    <script language = "javascript">
      var XMLHttpRequestObject = false;

      if (window.XMLHttpRequest) {
        XMLHttpRequestObject = new XMLHttpRequest();
      } else if (window.ActiveXObject) {
        XMLHttpRequestObject = new
          ActiveXObject("Microsoft.XMLHTTP");
      }

      function getData(dataSource, divID)
      {
        if(XMLHttpRequestObject) {
          var obj = document.getElementById(divID);
          XMLHttpRequestObject.open("GET", dataSource);
```

```
          XMLHttpRequestObject.onreadystatechange = function()
          {
            if (XMLHttpRequestObject.readyState == 4 &&
              XMLHttpRequestObject.status == 200) {
                obj.innerHTML = XMLHttpRequestObject.responseText;
            }
          }

          XMLHttpRequestObject.send(null);
        }
      }
    </script>
  </head>
  <body>

    <H1>Demostración de Ajax y PHP</H1>

    <form>
      <input type = "button" value = "Capturar el mensaje"
        onclick = "getData('data.php', 'targetDiv')">
    </form>

    <div id="targetDiv">
      <p>El mensaje capturado aparecerá aquí.</p>
    </div>

  </body>
</html>
```

Paso de datos al servidor con GET

¿Qué le parece si ahora envía datos al servidor? Debe codificar usted mismo los datos enviados al servidor en aplicaciones Ajax —no puede confiar que lo haga automáticamente una forma HTML, como un envío de datos usando controles HTML.

En este ejemplo, vamos a enviar datos al servidor, con el nombre de parámetro"data". Si este contiene"l" (observe que se trata de la cadena"l", no el número 1, pues todos los datos enviados a scripts de PHP desde páginas Web son texto), el script, choosem.php, devuelve este mensaje:

```
<?php
  if ($_REQUEST["data"] == "1") {
    echo 'Envió al servidor un valor de 1';
  }
    .
    .
    .
?>
```

Por otra parte, si el dato enviado es“2”, el script choosem.php reproduce este mensaje:

```
<?php
  if ($_REQUEST["data"] == "1") {
    echo 'Envió al servidor un valor de 1';
  }
  if ($_REQUEST["data"] == "2") {
    echo 'Envió al servidor un valor de 2';
  }
?>
```

Cuando usa el método GET para capturar datos del servidor, los datos son enviados desde las páginas Web de regreso al servidor, mediante la codificación de direcciones URL, significando que los datos se anexan a la dirección URL leída desde el servidor. Por ejemplo, si usara el método GET y tuviese una página Web estándar con un campo de texto llamado“a”, conteniendo el número 5, un campo de texto llamado“b”conteniendo el número 6 y un campo de texto llamado“c”, cuyo contenido fuese“Éste es el momento”, todos los datos se codificarían y agregarían a la dirección URL a que tiene acceso. Los nombres de los campos de texto, a, b y c, son parámetros enviados al servidor y, el texto en cada campo de texto, son datos asignados a cada parámetro.

Cuando los datos se codifican en URL, se agrega un signo de interrogación (?) al final de la dirección URL y los datos, en formato name=data, se agregan tras el signo de interrogación. Los espacios en el texto se convierten en un signo más (+) y los pares de elementos name=data se separan con símbolos ampersand (&). Así, para codificar datos de los campos de texto“a”,“b”y“c” y enviarlos a http://www.servidor.com/usuario/nombredelscript, utilizaría esta dirección URL:

```
http://www.servidor.com/usuario/nombredelscript?a=5&b=6&c=Éste+es+el+momento
```

Esto implica que para enviar el parámetro data conteniendo“1”a choosem.php, puede usar una dirección URL como ésta:

```
choosem.php?data=1
```

Puede modificar la página de Ajax vista en este capítulo, para interactuar con choosem.php si coloca dos botones —el primero envía el parámetro data con“1”, mediante codificación URL:

```
<form>
  <input type = "button" value = "Capturar el mensaje 1"
    onclick = "getData('choosem.php?data=1', 'targetDiv')">
    .
    .
    .
</form>
```

Y el segundo botón envía el valor“2”a choosem.php:

```
<form>
  <input type = "button" value = "Capturar el mensaje 1"
    onclick = "getData('choosem.php?data=1', 'targetDiv')">
```

```
    <input type = "button" value = "Capturar el mensaje 2"
      onclick = "getData('choosem.php?data=2', 'targetDiv')">
  </form>
```

Así se ve todo esto en index3.html:

```
<html>
  <head>
    <title>Envío de datos de Ajax con GET</title>

    <script language = "javascript">
      var XMLHttpRequestObject = false;

      if (window.XMLHttpRequest) {
        XMLHttpRequestObject = new XMLHttpRequest();
      } else if (window.ActiveXObject) {
        XMLHttpRequestObject = new
          ActiveXObject("Microsoft.XMLHTTP");
      }

      function getData(dataSource, divID)
      {
        if(XMLHttpRequestObject) {
          var obj = document.getElementById(divID);
          XMLHttpRequestObject.open("GET", dataSource);

          XMLHttpRequestObject.onreadystatechange = function()
          {
            if (XMLHttpRequestObject.readyState == 4 &&
              XMLHttpRequestObject.status == 200) {
                obj.innerHTML = XMLHttpRequestObject.responseText;
            }
          }

          XMLHttpRequestObject.send(null);
        }
      }
    </script>
  </head>

  <body>

    <h1>Envío de datos de Ajax con GET</h1>

    <form>
      <input type = "button" value = "Capturar el mensaje 1"
        onclick = "getData('choosem.php?data=1', 'targetDiv')">
      <input type = "button" value = "Capturar el mensaje 2"
        onclick = "getData('choosem.php?data=2', 'targetDiv')">
    </form>
```

FIGURA 12-3 Envío de "1" al servidor con GET

```
    <div id="targetDiv">
      <p>El mensaje capturado aparecerá aquí.</p>
    </div>

  </body>
</html>
```

El resultado se aprecia en la Figura 12-3, donde el usuario hizo clic en el botón"Capturar el mensaje 1". En la Figura 12-4, el usuario hizo clic en el botón"Capturar el mensaje 2".

FIGURA 12-4 Envío de "2" al servidor con GET

Paso de datos al servidor con POST

Hemos escrito choosem.php de esta forma, usando $_REQUEST en lugar de $_GET o $_POST, para leer datos enviados a este script:

```
<?php
  if ($_REQUEST["data"] == "1") {
    echo 'Envió al servidor un valor de 1';
  }
  if ($_REQUEST["data"] == "2") {
    echo 'Envió al servidor un valor de 2';
  }
?>
```

Significa que choosem.php está configurado para usarse con el método POST, no sólo GET. ¿Pero cómo se envían datos a scripts de PHP utilizando POST?

Primero, puesto que ya no envía datos codificados en URL, debe cambiar ambos botones de la forma de esto:

```
<form>
  <input type = "button" value = "Capturar el mensaje 1"
    onclick = "getData('choosem.php?data=1', 'targetDiv')">
  <input type = "button" value = "Capturar el mensaje 2"
    onclick = "getData('choosem.php?data=2', 'targetDiv')">
</form>
```

a esto, donde los datos ("1" o "2") se envían a la función getData, como tercer argumento de la función:

```
<form>
  <input type = "button" value = "Capturar el mensaje 1"
    onclick = "getData('choosem.php?data=1', 'targetDiv', 1)">
  <input type = "button" value = "Capturar el mensaje 2"
    onclick = "getData('choosem.php?data=2', 'targetDiv', 2)">
</form>
```

Ahora debe modificar la función getData para tomar tres argumentos:

```
function getData(dataSource, divID, data)
{
  .
  .
  .
}
```

Y también cambiar esta línea, especificando el método GET:

```
function getData(dataSource, divID)
{
  if(XMLHttpRequestObject) {
    var obj = document.getElementById(divID);
    XMLHttpRequestObject.open("GET", dataSource);
```

```
  .
  .
  .
}
```

para utilizar el método POST en su lugar:

```
function getData(dataSource, divID)
{
  if(XMLHttpRequestObject) {
    var obj = document.getElementById(divID);
    XMLHttpRequestObject.open("POST", dataSource);
  .
  .
  .
}
```

Sin embargo, hay más que cambiar en el método POST que sólo eso —también debe indicar al código para el servidor que va a codificar datos bajo la forma que POST procesa tales datos; ello significa que debe incluir esta línea de código:

```
function getData(dataSource, divID)
{
  if(XMLHttpRequestObject) {
    var obj = document.getElementById(divID);
    XMLHttpRequestObject.open("POST", dataSource);
    XMLHttpRequestObject.setRequestHeader('Content-Type',
      'application/x-www-form-urlencoded');
  .
  .
  .
  }
}
```

Por último, usará el método send para enviar datos al servidor. Envía al servidor una cadena de datos como el nombre que anexaría al final de la dirección URL con el método GET; pero al usar POST, se pasa esa cadena al método send:

```
function getData(dataSource, divID, data)
      {
        if(XMLHttpRequestObject) {
          var obj = document.getElementById(divID);
          XMLHttpRequestObject.open("POST", dataSource);
          XMLHttpRequestObject.setRequestHeader('Content-Type',
            'application/x-www-form-urlencoded');

          XMLHttpRequestObject.onreadystatechange = function()
          {
            if (XMLHttpRequestObject.readyState == 4 &&
              XMLHttpRequestObject.status == 200) {
                obj.innerHTML = XMLHttpRequestObject.responseText;
            }
          }
```

```
            XMLHttpRequestObject.send("data=" + data);
          }
        }
```

Así luce en index4.html:

```
<html>
  <head>
    <title>Envío de datos de Ajax con POST</title>

    <script language = "javascript">
      var XMLHttpRequestObject = false;

      if (window.XMLHttpRequest) {
        XMLHttpRequestObject = new XMLHttpRequest();
      } else if (window.ActiveXObject) {
        XMLHttpRequestObject = new
          ActiveXObject("Microsoft.XMLHTTP");
      }

      function getData(dataSource, divID, data)
      {
        if(XMLHttpRequestObject) {
          var obj = document.getElementById(divID);
          XMLHttpRequestObject.open("POST", dataSource);
          XMLHttpRequestObject.setRequestHeader('Content-Type',
            'application/x-www-form-urlencoded');

          XMLHttpRequestObject.onreadystatechange = function()
          {
            if (XMLHttpRequestObject.readyState == 4 &&
              XMLHttpRequestObject.status == 200) {
                obj.innerHTML = XMLHttpRequestObject.responseText;
            }
          }

          XMLHttpRequestObject.send("data=" + data);
        }
      }
    </script>
  </head>

  <body>

    <h1>Envío de datos de Ajax con POST</h1>

    <form>
      <input type = "button" value = "Capturar el mensaje 1"
        onclick = "getData('choosem.php', 'targetDiv', 1)">
      <input type = "button" value = "Capturar el mensaje 2"
        onclick = "getData('choosem.php', 'targetDiv', 2)">
    </form>
```

FIGURA 12-5 Envío de "1" al servidor con POST

```
    <div id="targetDiv">
      <p>El mensaje capturado aparecerá aquí.</p>
    </div>

  </body>
</html>
```

El resultado se encuentra en la Figura 12-5, donde el usuario hizo clic en el botón "Capturar el mensaje 1", enviando "1" al servidor por medio de POST.

Hasta ahora, simplemente ha descargado texto ordinario, usando la propiedad responseText, del objeto XMLHttpRequest. Pero, el acrónimo Ajax ¿acaso no significa Asynchronous JavaScript and XML? ¿Qué tal si trabajamos con un poco de XML? Eso veremos a continuación.

Manejo de XML

Suponga que desea configurar una tienda en línea para venta de diferentes artículos (un libro de PHP, un televisor, una radio, etcétera). Podría almacenar los artículos para venta en diferentes archivos XML (products1.xml, products2.xml, etcétera), y luego descargar su contenido usando Ajax.

Por ejemplo, así se vería products1.xml. Comienza con una declaración XML, como deben tener todos los archivos XML:

```
<?xml version = "1.0" ?>
        .
        .
        .
```

Esta declaración XML dice que el documento es XML y está escrito en versión 1.0. Después, todos los documentos XML necesitarán un elemento documento —es decir, un elemento XML conteniendo todos los otros elementos del documento—. Usted crea sus propios elementos en

XML, de tal modo podría utilizar un elemento documento llamado <items> para almacenar los artículos a la venta, de la siguiente forma:

```
<?xml version = "1.0" ?>
<items>
        .
        .
        .
<items>
```

Y almacenar los artículos reales para venta dentro de elementos <items> así:

```
<?xml version = "1.0" ?>
<items>
  <item>Libro de PHP</item>
  <item>Televisor</item>
  <item>Radio</item>
</items>
```

Eso completa products1.xml. También podría tener otras listas de elementos para venta, como en products2.xml, que muestra Soda, Queso y Salami:

```
<?xml version = "1.0" ?>
<items>
  <item>Soda</item>
  <item>Queso</item>
  <item>Salami</item>
</items>
```

Bueno, ahora tiene los dos archivos XML almacenando los artículos para venta. ¿Qué le parece si descargamos esos archivos XML utilizando Ajax? Pondremos esto a trabajar en una página llamada store.html.

Muestre dos botones en la página store.html (una para la primera lista de productos y otra para la segunda lista de productos). Por ejemplo, el primer botón descargaría productr1.xml, llamando a la función de JavaScript llamada getproducts1:

```
<form>
  <input type = "button" value = "Seleccionar products1"
    onclick = "getproducts1()">
    .
    .
    .
</form>
```

Y el segundo botón descargará la segunda lista de artículos, products2.xml, llamando a otra función de JavaScript llamada getproducts2:

```
<form>
  <input type = "button" value = "Seleccionar products1"
    onclick = "getproducts1()">
  <input type = "button" value = "Seleccionar products2"
    onclick = "getproducts2()">
</form>
```

Además, podríamos presentar los artículos descargados para venta en un control de lista desplegable <select> llamado productsList:

```
<form>
  <select size="1" id="productsList">
    <option>Seleccione un artículo</options>
  <select>
  <br>
  <br>
  <input type = "button" value = "Seleccionar products1"
    onclick = "getproducts1()">
  <input type = "button" value = "Seleccionar products2"
    onclick = "getproducts2()">
</form>
```

Y cuando el usuario selecciona un artículo mostrado, usted podría indicar cuál eligió en la página, que haremos en una función de JavaScript llamada setproducts, a la cual se invoca cuando el usuario hace una selección en el control <select>:

```
<form>
  <select size="1" id="productsList">
    onchange="setproducts()">
    <option>Seleccione un artículo</options>
  <select>
  <br>
  <br>
  <input type = "button" value = "Seleccionar products1"
    onclick = "getproducts1()">
  <input type = "button" value = "Seleccionar products2"
    onclick = "getproducts2()">
</form>
```

Bueno, eso configura los controles en la página. Es momento de agregar un poco de Ajax, creando un objeto XMLHttpRequest que usaremos para comunicarnos con el servidor en JavaScript:

```
<html>
  <head>

    <title>Uso de Ajax con XML</title>

    <script language = "javascript">

      var XMLHttpRequestObject = false;

      if (window.XMLHttpRequest) {
        XMLHttpRequestObject = new XMLHttpRequest();
      } else if (window.ActiveXObject) {
        XMLHttpRequestObject = new ActiveXObject("Microsoft.XMLHTTP");
      }
        .
        .
        .
```

Vamos a usar una función llamada getproducts1 para capturar products1.xml y mostrar los artículos contenidos en ese documento XML en el control <select>. La función getproducts1 comprueba si existe el objeto XMLHttpRequest:

```
function getproducts1()
{
  if(XMLHttpRequestObject) {
  .
  .
  .
  }
}
```

Si existe el objeto XMLHttpRequest, puede abrirlo, que señala desea capturar products1.xml:

```
function getproducts1()
{
  if(XMLHttpRequestObject) {
    XMLHttpRequestObject.open("GET", "products1.xml");
    .
    .
    .
  }
}
```

Ahora puede descargar aquí el contenido XML como hizo antes con el texto —con excepción de que esta vez los datos descargados son XML, esto implica que usa la propiedad responseXML del objeto XMLHttpRequest, no la responseText:

```
function getproducts1()
{
  if(XMLHttpRequestObject) {
    XMLHttpRequestObject.open("GET", "products1.xml");

    XMLHttpRequestObject.onreadystatechange = function()
    {
      if (XMLHttpRequestObject.readyState == 4 &&
        XMLHttpRequestObject.status == 200) {
      var xmlDocument = XMLHttpRequestObject.responseXML;
      .
      .
      .
      }
    }
  }
}
```

La propiedad responseXML aloja sus datos XML bajo la forma de un objeto de documento XML de JavaScript. Puede extraer los elementos <item> de ese objeto de documento, usando el

método getElementsByTagName, que devuelve una matriz de elementos <item>, para ser almacenada en una variable llamada productos:

```
<html>
  <head>

    <title>Uso de Ajax con XML</title>

    <script language = "javascript">

      var products;
        .
        .
        .
      function getproducts1()
      {
        if(XMLHttpRequestObject) {
          XMLHttpRequestObject.open("GET", "products1.xml");

          XMLHttpRequestObject.onreadystatechange = function()
          {
            if (XMLHttpRequestObject.readyState == 4 &&
              XMLHttpRequestObject.status == 200) {
            var xmlDocument = XMLHttpRequestObject.responseXML;
            products = xmlDocument.getElementsByTagName("item");
            .
            .
            .
            }
          }
        }
      }
```

Muy bien, ahora la variable products aloja una matriz de elementos <item>. Puede llamar a una nueva función, listproducts, para ser escrita y mostrar el texto en esos elementos del control <select> en la página:

```
      function getproducts1()
      {
        if(XMLHttpRequestObject) {
          XMLHttpRequestObject.open("GET", "products1.xml");

          XMLHttpRequestObject.onreadystatechange = function()
          {
            if (XMLHttpRequestObject.readyState == 4 &&
              XMLHttpRequestObject.status == 200) {
            var xmlDocument = XMLHttpRequestObject.responseXML;
            products = xmlDocument.getElementsByTagName("item");
            listproducts();
            }
        }
      }
```

Y el último paso consiste en enviar un valor null al objeto XMLHttpRequest para iniciar la descarga:

```
function getproducts1()
{
  if(XMLHttpRequestObject) {
    XMLHttpRequestObject.open("GET", "products1.xml");

    XMLHttpRequestObject.onreadystatechange = function()
    {
      if (XMLHttpRequestObject.readyState == 4 &&
        XMLHttpRequestObject.status == 200) {
      var xmlDocument = XMLHttpRequestObject.responseXML;
      products = xmlDocument.getElementsByTagName("item");
      listproducts();
      }
    }

    XMLHttpRequestObject.send(null);
  }
}
```

Eso completa la función getproducts1; la función getproducts2 descarga el documento products2.xml:

```
function getproducts2()
{
  if(XMLHttpRequestObject) {
    XMLHttpRequestObject.open("GET", "products2.xml");

    XMLHttpRequestObject.onreadystatechange = function()
    {
      if (XMLHttpRequestObject.readyState == 4 &&
        XMLHttpRequestObject.status == 200) {
      var xmlDocument = XMLHttpRequestObject.responseXML;
      products = xmlDocument.getElementsByTagName("item");
      listproducts();
      }
    }

    XMLHttpRequestObject.send(null);
  }
}
```

Excelente. Ahora puede escribir la función listproducts para crear una lista de los productos a la venta, almacenados en la matriz products. La matriz products aloja estos elementos:

```
<item>Libro de PHP</item>
<item>Televisor</item>
<item>Radio</item>
```

Puede acceder a ellos como products[0], products[1], products[2]. Pero no es el final de la historia —¿cómo se accede al texto contenido en cada elemento?— Ése es uno de los trucos de

trabajar con XML y JavaScript —para acceder al texto contenido en cada elemento XML, por ejemplo products[0], debe usar la expresión products[0].firstChild—. Ésta devuelve el primer nodo de texto contenido en el elemento y dicho nodo aloja el texto deseado. Para tener acceso al texto contenido en el nodo de texto, usted usaría la expresión products[0].firstChild.data. Luego, aquí puede poblar el control <select> con elementos <option>, cuyas leyendas son los productos en venta:

```
function listproducts ()
{
  var loopIndex;
  var selectControl = document.getElementById('productsList');

  for (loopIndex = 0; loopIndex < products.length; loopIndex++ )
  {
      selectControl.options[loopIndex] = new
         Option(products[loopIndex].firstChild.data);
  }
}
```

Casi terminamos —todo lo que resta por hacer es escribir la función setproducts, a la que se llama cuando el usuario elige el control <select>, mostrando los artículos en venta—. Puede presentar el artículo elegido por el usuario en un elemento , llamado targetDiv, de la siguiente forma en setproducts:

```
function setproducts()
{
  document.getElementById('targetDiv').innerHTML =
    "You selected " + products[document.getElementById
      ('productsList').selectedIndex].firstChild.data;
}
```

Como puede apreciar, se requiere poco esfuerzo adicional para manejar XML en Ajax. Puede ver store.html en la Figura 12-6.

FIGURA 12-6 store.html

FIGURA 12-7 Descarga de productos utilizando XML

Cuando el usuario hace clic en el botón "Seleccionar products2", esos productos se descargan con XML y Ajax, para mostrarse en el control <select>, como se aprecia en la Figura 12-7.

Y cuando el usuario elige un artículo, éste se muestra en la página, como se ve en la Figura 12-8.

Eso le indica cómo manejar XML con Ajax. Ahora es momento de agregar PHP a esta imagen.

FIGURA 12-8 Selección de un producto

Manejo de XML con PHP

Ahora convertiremos store.html en un nuevo ejemplo, store2.html, que leerá su código XML de un script PHP, products.php. El script products.php es el responsable de devolver el primer documento XML:

```
<?xml version = "1.0" ?>
<items>
<item>Libro de PHP</item>
<item>Televisor</item>
<item>Radio</item>
<items>
```

y también el segundo:

```
<?xml version = "1.0" ?>
<items>
  <item>Soda</item>
  <item>Queso</item>
  <item>Salami</item>
<items>
```

Para indicar qué documento XML desea, puede enviar un parámetro, items, a products.php. Cuando items sea igual a"1", el script devolverá el primer documento XML; cuando items sea igual a"2", el script devolverá el segundo documento XML.

Escribiremos products.php primero —que da lugar a la pregunta: ¿cómo se devuelve exactamente un documento XML de un script PHP?— La forma de hacerlo consiste en establecer el encabezado Content-type de los datos enviados al navegador a"text/xml", mediante la función header de PHP:

```
<?php
header("Content-type: text/xml");
        .
        .
        .
?>
```

Este paso necesario hace saber al navegador que viene código XML. Ahora puede leer el parámetro items y cargar una matriz llamada $items en concordancia:

```
<?php
header("Content-type: text/xml");

if ($_REQUEST["items"] == "1")
  $items = array('Libro de PHP', 'Televisor', 'Radio');
if ($_REQUEST["items"] == "2")
  $items = array('Soda', 'Queso', 'Salami');
        .
        .
        .
?>
```

Ahora debe ensamblar un documento XML apropiado para enviarlo al navegador, comenzando con la declaración XML:

```
<?php
header("Content-type: text/xml");

if ($_REQUEST["items"] == "1")
  $items = array('Libro de PHP', 'Televisor', 'Radio');
if ($_REQUEST["items"] == "2")
  $items = array('Soda', 'Queso', 'Salami');

echo '<?xml version="1.0" ?>;
        .
        .
        .
?>
```

Y enviar el elemento documento, que es <items>;

```
<?php
header("Content-type: text/xml");

if ($_REQUEST["items"] == "1")
  $items = array('Libro de PHP', 'Televisor', 'Radio');
if ($_REQUEST["items"] == "2")
  $items = array('Soda', 'Queso', 'Salami');

echo '<?xml version="1.0" ?>;

echo '<items>';
        .
        .
        .
echo '</items>';
?>
```

Ahora puede recorrer los artículos en venta, creando un elemento <item> para cada uno, de la siguiente forma:

```
<?php
header("Content-type: text/xml");

if ($_REQUEST["items"] == "1")
  $items = array('Libro de PHP', 'Televisor', 'Radio');
if ($_REQUEST["items"] == "2")
  $items = array('Soda', 'Queso', 'Salami');

echo '<?xml version="1.0" ?>';

echo '<items>';
foreach ($items as $value)
{
  echo '<item>';
  echo $value;
  echo '</item>';
}
echo '</items>';
?>
```

Las modificaciones en store2.html son relativamente menores; todo lo que tiene que hacer es asegurarse de enviar el valor correcto para el parámetro items al script products.php ("1" o "2"). Ese cambio se hace en las funciones getproducts1 y getproducts2 (por ejemplo, así es como se envía un valor de"1"a products.php en la función getproducts1:

```
function getproducts1()
{
  if(XMLHttpRequestObject) {
    XMLHttpRequestObject.open("GET", "products.php?items=1");

    XMLHttpRequestObject.onreadystatechange = function()
    {
      if (XMLHttpRequestObject.readyState == 4 &&
        XMLHttpRequestObject.status == 200) {
      var xmlDocument = XMLHttpRequestObject.responseXML;
      products = xmlDocument.getElementsByTagName("item");
      listproducts();
      }
    }

    XMLHttpRequestObject.send(null);
  }
}
```

Y eso es todo lo que necesita (el resto de store2.html es igual a store.html). Puede ver store2.html en acción en la Figura 12-9.

Formidable.

FIGURA 12-9 Selección de un producto en store2.html

13 CAPÍTULO

Ajax avanzado

El capítulo anterior puso la bola a rodar con Ajax y PHP, este capítulo se centra en algunas otras facultades requeridas para usar Ajax. Por ejemplo, en el capítulo anterior se usó un objeto XMLHttpRequest para establecer comunicación con el servidor. Pero, ¿qué pasa si el usuario hace clic en un botón por segunda vez, antes de completar la primera operación de Ajax? El código utilizará de forma inocente el mismo objeto XMLHttpRequest para realizar otra solicitud al servidor —antes de que regrese la primera solicitud.

No es un gran problema con los ejemplos vistos, pues descargan datos estáticos; pero, ¿qué sucedería si se conectara a una base de datos cambiante todo el tiempo? O, ¿qué pasaría de tener 20 botones habilitados con Ajax en su página Web, pero sólo usara un objeto XMLHttpRequest para capturar sus datos? Claramente, debe crear y usar un objeto XMLHttpRequest para cada solicitud al servidor.

Profundizaremos en esto aquí también. Por ejemplo, Ajax está restringido a la captura de datos en el mismo servidor del que proviene la misma página Web habilitada con Ajax. Es decir, si su página Ajax es http://www.myajax.com, entonces tendrá problemas de acceso a datos desde cualquier otro dominio que no sea http://www.myajax.com utilizando Ajax. Analizaremos tal comportamiento y cómo rodear el problema en este capítulo.

Existen otros temas que se abordarán aquí también. Por ejemplo, además de descargar texto o XML mediante Ajax, una acción común realizada con Ajax es descargar código JavaScript ejecutable en el navegador. Así se conecta a una API (interfaz de programación de aplicaciones) Ajax de Google —Google Suggest, a revisarse también.

Además de todo eso, verá cómo usar Ajax con solicitudes de encabezados HTTP, que le permitirá comprobar si existen archivos en el servidor, cuál es su longitud, fecha de creación y más. También verá cómo anular el alojamiento en caché de navegadores como Internet Explorer y muchos otros temas que irá conociendo conforme profundice en Ajax.

Manejo de solicitudes concurrentes de Ajax con múltiples objetos XMLHttpRequest

Como recordará del capítulo anterior, sólo creamos un objeto XMLHttpRequest, aunque algunos de nuestros ejemplos, store.html —mostrado a continuación— dan al usuario la opción de hacer clic en dos botones. Eso significa que el usuario podría, en teoría, hacer clic en un segundo botón antes de haber completado su primera solicitud en Ajax, y ése es un problema. Así se ve

esto en store.html —observe que sólo hay un objeto XMLHttpRequest y, ese mismo objeto se usa sin importar en qué botón haga clic el usuario:

```
<html>
  <head>

    <title>Uso de Ajax con XML</title>

    <script language = "javascript">

      var products;

      var XMLHttpRequestObject = false;

      if (window.XMLHttpRequest) {
        XMLHttpRequestObject = new XMLHttpRequest();
      } else if (window.ActiveXObject) {
        XMLHttpRequestObject = new ActiveXObject("Microsoft.XMLHTTP");
      }

      function getproducts1()
      {
        if(XMLHttpRequestObject) {
          XMLHttpRequestObject.open("GET", "products1.xml?g=y");

          XMLHttpRequestObject.onreadystatechange = function()
          {
            if (XMLHttpRequestObject.readyState == 4 &&
              XMLHttpRequestObject.status == 200) {
            var xmlDocument = XMLHttpRequestObject.responseXML;
            products = xmlDocument.getElementsByTagName("item");
            listproducts();
            }
          }

          XMLHttpRequestObject.send(null);
        }
      }

      function getproducts2()
      {
        if(XMLHttpRequestObject) {
          XMLHttpRequestObject.open("GET", "products2.xml?b=t");

          XMLHttpRequestObject.onreadystatechange = function()
          {
            if (XMLHttpRequestObject.readyState == 4 &&
              XMLHttpRequestObject.status == 200) {
            var xmlDocument = XMLHttpRequestObject.responseXML;
            products = xmlDocument.getElementsByTagName("item");
            listproducts();
            }
          }
```

```
          XMLHttpRequestObject.send(null);
        }
      }

      function listproducts ()
      {
        var loopIndex;
        var selectControl = document.getElementById('productsList');

        for (loopIndex = 0; loopIndex < products.length; loopIndex++ )
        {
            selectControl.options[loopIndex] = new
               Option(products[loopIndex].firstChild.data);
        }
    }

    function setproducts()
    {
      document.getElementById('targetDiv').innerHTML =
        "You selected " + products[document.getElementById
          ('productsList').selectedIndex].firstChild.data;
    }

    </script>
  </head>

  <body>

    <h1>Uso de Ajax con XML</h1>

    <form>
      <select size="1" id="productsList"
        onchange="setproducts()">
        <option>Seleccione un artículo</option>
      </select>
      <br>
      <br>
      <input type = "button" value = "Seleccionar products 1"
        onclick = "getproducts1()">
      <input type = "button" value = "Seleccionar products 2"
        onclick = "getproducts2()">
    </form>

    <div id="targetDiv" width =100 height=100>
      Su selección aparecerá aquí.
    </div>
  </body>
</html>
```

Debemos resolver este problema común en aplicaciones Ajax. Saber cómo manejarlo es decisivo para la programación en Ajax. Existen varias soluciones que usted verá aquí. La primera de éstas consiste en usar múltiples objetos XMLHttpRequest.

Así se modifica store.html para utilizar dos objetos XMLHttpRequest. El proceso no es complicado: todo lo que debe hacer es crear dos objetos XMLHttpRequest

—XMLHttpRequestObject y XMLHttpRequestObject2—, luego utilizar un objeto en las funciones conectándose al servidor —getproducts1 y getproducts2:

```
<html>
  <head>

    <title> Uso de dos objetos XMLHttpRequest</title>

    <script language = "javascript">

      var products;

      var XMLHttpRequestObject = false;
      var XMLHttpRequestObject2 = false;

      if (window.XMLHttpRequest) {
        XMLHttpRequestObject = new XMLHttpRequest();
      } else if (window.ActiveXObject) {
        XMLHttpRequestObject = new ActiveXObject("Microsoft.XMLHTTP");
      }

      if (window.XMLHttpRequest) {
        XMLHttpRequestObject2 = new XMLHttpRequest();
      } else if (window.ActiveXObject) {
        XMLHttpRequestObject2 = new ActiveXObject("Microsoft.XMLHTTP");

      function getproducts1()
      {

        if(XMLHttpRequestObject) {
          XMLHttpRequestObject.open("GET", "products1.xml");

          XMLHttpRequestObject.onreadystatechange = function()
          {
            if (XMLHttpRequestObject.readyState == 4 &&
              XMLHttpRequestObject.status == 200) {
            var xmlDocument = XMLHttpRequestObject.responseXML;
            products = xmlDocument.getElementsByTagName("item");
            listproducts();
            }
          }

          XMLHttpRequestObject.send(null);
        }
      }

      function getproducts2()
      {
        if(XMLHttpRequestObject2) {
          XMLHttpRequestObject2.open("GET", "products2.xml");

          XMLHttpRequestObject2.onreadystatechange = function()
          {
            if (XMLHttpRequestObject2.readyState == 4 &&
```

```
            XMLHttpRequestObject2.status == 200) {
          var xmlDocument = XMLHttpRequestObject2.responseXML;
          products = xmlDocument.getElementsByTagName("item");
          listproducts();
          }
        }

        XMLHttpRequestObject2.send(null);
      }
    }

    function listproducts ()
    {
      var loopIndex;
      var selectControl = document.getElementById('productsList');

      for (loopIndex = 0; loopIndex < products.length; loopIndex++ )
      {
          selectControl.options[loopIndex] = new
             Option(products[loopIndex].firstChild.data);
      }
  }

  function setproducts()
  {
    document.getElementById('targetDiv').innerHTML =
      "Usted seleccionó " + products[document.getElementById
        ('productsList').selectedIndex].firstChild.data;
  }

  </script>
</head>
<body>

  <h1>Uso de dos objetos XMLHttpRequest</h1>
  <form>
    <select size="1" id="productsList"
      onchange="setproducts()">
      <option>Seleccione un artículo</option>
    </select>
    <br>
    <br>
    <input type = "button" value = "Seleccionar products1"
      onclick = "getproducts1()">
    <input type = "button" value = "Seleccionar products2"
      onclick = "getproducts2()">
  </form>

  <div id="targetDiv" width =100 height=100>
    Su selección aparecerá aquí.
  </div>
</body>
</html>
```

FIGURA 13-1 Uso de dos objetos XMLHttpRequest

Puede ver los resultados en la Figura 13-1, cada uno de los dos botones emplea un objeto XMLHttpRequest diferente.

Este enfoque es una mejora, pero también tiene sus detalles ¿qué pasaría de necesitar 1 000 de estos objetos; le agradaría redactar código para crear explícitamente cada uno de ellos? Difícilmente. Podría crear una matriz de objetos XMLHttpRequest en su lugar, que veremos a continuación.

Manejo de solicitudes concurrentes de Ajax con una matriz XMLHttpRequest

Para manejar el caso donde podría necesitar docenas de objetos XMLHttpRequest para procesar docenas de solicitudes, podría almacenar esos objetos en una matriz, XMLHttpRequestObjects:

```
var XMLHttpRequestObjects = new Array();
```

Y, luego, cuando necesite un nuevo objeto XMLHttpRequest, podría crear uno, agregándolo a la matriz con el método push, de JavaScript:

```
var XMLHttpRequestObjects = new Array();
function getproducts1()
{
  if (window.XMLHttpRequest) {
    XMLHttpRequestObjects.push(new XMLHttpRequest());
  } else if (window.ActiveXObject) {
  XMLHttpRequestObjects.push(new ActiveXObject("Microsoft.XMLHTTP"));
  }
  .
  .
  .
```

Esta página nueva, habilitada con Ajax, array.html, pone dicha premisa en acción, creando el ejemplo de la tienda mediante uso de una matriz de objetos XMLHttpRequest, en caso de tener múltiples llamadas concurrentes de Ajax:

```
<html>
  <head>

    <title>Uso de matrices XMLHttpRequest</title>

    <script language = "javascript">

      var products;
      var index = 0;
      var XMLHttpRequestObjects = new Array();

      function getproducts1()
      {
        if (window.XMLHttpRequest) {
          XMLHttpRequestObjects.push(new XMLHttpRequest());
        } else if (window.ActiveXObject) {
        XMLHttpRequestObjects.push(new ActiveXObject("Microsoft.XMLHTTP"));
        }

        index = XMLHttpRequestObjects.length - 1;

        if(XMLHttpRequestObjects[index]) {

          XMLHttpRequestObjects[index].open("GET", "products1.xml");
          XMLHttpRequestObjects[index].onreadystatechange = function()
          {
            if (XMLHttpRequestObjects[index].readyState == 4 &&
              XMLHttpRequestObjects[index].status == 200) {
            var xmlDocument = XMLHttpRequestObjects[index].responseXML;
            products = xmlDocument.getElementsByTagName("item");
            listproducts();
            }
          }

          XMLHttpRequestObjects[index].send(null);
        }
      }
      function getproducts2()
      {
        if (window.XMLHttpRequest) {
          XMLHttpRequestObjects.push(new XMLHttpRequest());
        } else if (window.ActiveXObject) {
        XMLHttpRequestObjects.push(new
          ActiveXObject("Microsoft.XMLHTTP"));
        }

        index = XMLHttpRequestObjects.length - 1;
```

```
      if(XMLHttpRequestObjects[index]) {
        XMLHttpRequestObjects[index].open("GET", "products2.xml");

        XMLHttpRequestObjects[index].onreadystatechange = function()
        {
          if (XMLHttpRequestObjects[index].readyState == 4 &&
            XMLHttpRequestObjects[index].status == 200) {
          var xmlDocument = XMLHttpRequestObjects[index].responseXML;
          products = xmlDocument.getElementsByTagName("item");
          listproducts();
          }
        }

        XMLHttpRequestObjects[index].send(null);
      }
    }

    function listproducts ()
    {
      var loopIndex;
      var selectControl = document.getElementById('productsList');

      for (loopIndex = 0; loopIndex < products.length; loopIndex++ )
      {
          selectControl.options[loopIndex] = new
             Option(products[loopIndex].firstChild.data);
      }
  }

  function setproducts()
  {
    document.getElementById('targetDiv').innerHTML =
      "Usted seleccionó " + products[document.getElementById
        ('productsList').selectedIndex].firstChild.data;
  }

  </script>
</head>

<body>

  <h1>Uso de matrices XMLHttpRequest</h1>

  <form>
    <select size="1" id="productsList"
      onchange="setproducts()">
      <option>Seleccione un artículo</option>
    </select>
    <br>
    <br>
    <input type = "button" value = "Seleccionar products1"
      onclick = "getproducts1()">
    <input type = "button" value = "Seleccionar products2"
      onclick = "getproducts2()">
```

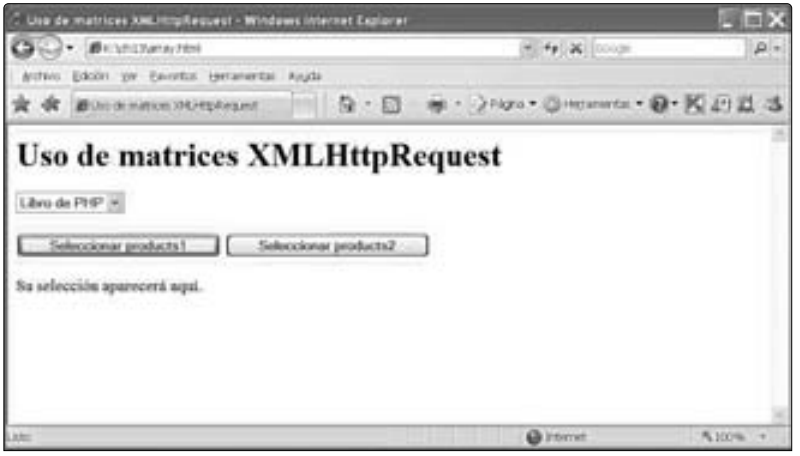

FIGURA 13-2 Uso de una matriz XMLHttpRequest

```
    </form>

    <div id="targetDiv" width =100 height=100>
      Su selección aparecerá aquí.
    </div>
  </body>
</html>
```

Puede apreciar los resultados en la Figura 13-2, donde cada botón usa un objeto XMLHttpRequest diferente de la matriz.

Esta técnica funciona, pero puede terminar con matrices grandes de objetos XMLHttpRequest y dependerá de usted eliminar objetos innecesarios. Ambas técnicas observadas —múltiples objetos XMLHttpRequest y matrices de objetos XMLHttpRequest— funcionan, y las verá en uso; pero quizá la mejor solución al manejo de solicitudes Ajax concurrentes sea manipular funciones internas de JavaScript, que veremos a continuación.

Manejo de solicitudes concurrentes de Ajax con funciones internas de JavaScript

¿Qué es una función interna? Es una función contenida dentro de otra función, de esta forma:

```
function outer(data)
{
  var variable1 = data;

  function inner(variable2)
  {
    alert(variable1 + variable2)
  }
}
```

Ahora suponga que llama a la función outer con un valor de 4 en esta forma: outer(4). Eso establece la variable variable1, de esta función, en 4. Resulta que la función inner accede los datos de la función outer —incluso después de finalizar la llamada a la función outer—. De modo que si llamara la función inner, pasando un valor de 5, eso establecería variable2, en la función inner, en 5 —y variable1 se quedaría en 4. De ese modo, el resultado de llamar a la función inner sería 4 + 5 = 9, el valor que mostraría la función alert de JavaScript en este caso.

Ésta es la buena parte: cada vez que llama la función outer, se crea una *nueva* copia de esa función, significa que un nuevo valor se almacenará como variable1.Y la función inner tendrá acceso a ese valor. De tal modo, si deja de pensar en términos de variable1 y lo hace en términos de la variable XMLHttpRequestObject, verá que cada ocasión que llame a una función de esta forma, JavaScript creará una nueva copia de la función con un nuevo objeto XMLHttpRequest, y ese estará disponible para cualquier función inner.

Eso es lo que usted desea en este caso, pues el código que ha redactado usa ya una función inner anónima, conectada a la propiedad onreadystatechange, de la función getData. Para hacer que esto funcione como en el ejemplo anterior, donde crea un objeto nuevo XMLHttpRequest, siempre que llama a las funciones getproducts1 o getproducts2, todo lo que debe hacer es colocar el código creado y llenar la variable XMLHttpRequestObject *dentro de las funciones getproducts1 y getproducts2*. Una vez que haya hecho eso, se creará un nuevo objeto XMLHttpRequest siempre que llame a la función, y la función inner anónima usará ese nuevo objeto de manera automática. Eso significa que la solución completa al problema de múltiples solicitudes concurrentes se ve así en inner.html:

```
<html>
  <head>

    <title>Uso de Ajax con funciones internas</title>

    <script language = "javascript">

      var products;

      function getproducts1()
      {
        var XMLHttpRequestObject = false;
        if (window.XMLHttpRequest) {
          XMLHttpRequestObject = new XMLHttpRequest();
        } else if (window.ActiveXObject) {
          XMLHttpRequestObject = new ActiveXObject("Microsoft.XMLHTTP");
        }

        if(XMLHttpRequestObject) {
          XMLHttpRequestObject.open("GET", "products1.xml");

          XMLHttpRequestObject.onreadystatechange = function()
          {
            if (XMLHttpRequestObject.readyState == 4 &&
              XMLHttpRequestObject.status == 200) {
            var xmlDocument = XMLHttpRequestObject.responseXML;
            products = xmlDocument.getElementsByTagName("item");
            listproducts();
```

```
        }
      }
      XMLHttpRequestObject.send(null);
    }
  }
  function getproducts2()
  {
    var XMLHttpRequestObject = false;
    if (window.XMLHttpRequest) {
      XMLHttpRequestObject = new XMLHttpRequest();
    } else if (window.ActiveXObject) {
      XMLHttpRequestObject = new ActiveXObject("Microsoft.XMLHTTP");
    }
    if(XMLHttpRequestObject) {
      XMLHttpRequestObject.open("GET", "products2.xml");

      XMLHttpRequestObject.onreadystatechange = function()
      {
        if (XMLHttpRequestObject.readyState == 4 &&
          XMLHttpRequestObject.status == 200) {
        var xmlDocument = XMLHttpRequestObject.responseXML;
        products = xmlDocument.getElementsByTagName("item");
        listproducts();
        }
      }
      XMLHttpRequestObject.send(null);
    }
  }
  function listproducts ()
  {
    var loopIndex;
    var selectControl = document.getElementById('productsList');
    for (loopIndex = 0; loopIndex < products.length; loopIndex++ )
    {
        selectControl.options[loopIndex] = new
           Option(products[loopIndex].firstChild.data);
    }
}

function setproducts()
{
  document.getElementById('targetDiv').innerHTML =
    "Usted seleccionó " + products[document.getElementById
      ('productsList').selectedIndex].firstChild.data;
}
</script>
```

```
  </head>

  <body>

    <h1>Uso de Ajax con funciones internas</h1>

    <form>
      <select size="1" id="productsList"
        onchange="setproducts()">
        <option>Seleccione un artículo</option>
      </select>
      <br>
      <br>
      <input type = "button" value = "Seleccionar products1"
        onclick = "getproducts1()">
      <input type = "button" value = "Seleccionar products2"
        onclick = "getproducts2()">
    </form>

    <div id="targetDiv" width =100 height=100>
      Su selección aparecerá aquí.
    </div>
  </body>
</html>
```

Puede ver los resultados en la Figura 13-3, el ejemplo usa funciones inner para crear automáticamente un nuevo objeto XMLHttpRequest en cada solicitud Ajax.

Además de descargar texto y código XML en Ajax, también puede descargar otro tipo de datos. Verá cómo descargar JavaScript (técnica de uso común) en este capítulo, además de imágenes.

FIGURA 13-3 Uso de funciones internas (inner)

NOTA *¿Imágenes? ¿Acaso Ajax no se limita a descargar datos de texto como XML? Resulta que cuando descarga imágenes, en realidad está descargando el nombre de la imagen y se apoya en Dynamic HTML (o HTML dinámico) en el navegador para que en realidad éste descargue la imagen.*

Descarga de imágenes por medio de Ajax

Este ejemplo muestra cómo descargar imágenes usando Ajax, image.html. Se basa en un script de PHP, imageName.php, al que puede enviar un parámetro llamado"image". Ése toma los valores"1"o"2"y devolverá el nombre de una imagen diferente, Image1.jpg, o Image2.jpg, según sea el caso:

```
<?php
  if ($_REQUEST["image"] == "1"){
    echo "Image1.jpg";
  }
  if ($_REQUEST["image"] == "2"){
    echo "Image2.jpg";
  }
?>
```

Y agregaría dos botones a la página,"Mostrar imagen 1"y"Mostrar imagen 2". Cada botón está conectado a la función getData de JavaScript, que hemos usado para descargar datos mediante Ajax. El primer botón llama a imageName.php con el parámetro image, establecido en"1" y el segundo botón llama a imageName.php, con el parámetro image establecido en"2":

```
<form>
  <input type = "button" value = "Mostrar imagen 1"
    onclick = "getData('imageName.php?image=1', callback)">
  <input type = "button" value = "Mostrar imagen 2"
    onclick = "getData('imageName.php?image=2', callback)">
</form>
```

Así, ¿cómo utiliza el nombre de la imagen descargada para forzar al navegador a descargar una imagen? Se hace escribiendo un elemento <img> en la página Web. Por esa razón, image.html incluye un elemento , targetDiv:

```
<div id="targetDiv">
  <p>La imagen capturada aparecerá aquí.</p>
</div>
```

En JavaScript hay una función llamada callback a la que se pasa el nombre de archivo de la imagen que desea descargar. Ésta escribe el nuevo elemento <img>, en el elemento targetDiv de esta forma:

```
function callback(text)
{
  document.getElementById("targetDiv").innerHTML =
    "<img src= " + text + ">";
}
```

Cuando aparece el nuevo elemento <img> en la página, el navegador descarga la imagen requerida. Así se ve en image.html:

```
<html>
  <head>
    <title>Descarga de imágenes con Ajax</title>

    <script language = "javascript">

      function getData(dataSource, callback)
      {
        var XMLHttpRequestObject = false;

        if (window.XMLHttpRequest) {
          XMLHttpRequestObject = new XMLHttpRequest();
        } else if (window.ActiveXObject) {
          XMLHttpRequestObject = new
            ActiveXObject("Microsoft.XMLHTTP");
        }

        if(XMLHttpRequestObject) {
          XMLHttpRequestObject.open("GET", dataSource);

          XMLHttpRequestObject.onreadystatechange = function()
          {
            if (XMLHttpRequestObject.readyState == 4 &&
              XMLHttpRequestObject.status == 200) {
                callback(XMLHttpRequestObject.responseText);
                delete XMLHttpRequestObject;
                XMLHttpRequestObject = null;

            }
          }

          XMLHttpRequestObject.send(null);
        }
      }

      function callback(text)
      {
        document.getElementById("targetDiv").innerHTML =
          "<img src= " + text + ">";
      }

    </script>
  </head>

  <body>

    <H1>Descarga de imágenes con Ajax</H1>

    <form>
      <input type = "button" value = "Mostrar imagen 1"
        onclick =
```

FIGURA 13-4 Descarga de imágenes con Ajax

```
          "getData('imageName.php?image=1', callback)">
      <input type = "button" value = "Mostrar imagen 2"
        onclick =
          "getData('imageName.php?image=2', callback)">
    </form>

    <div id="targetDiv">
      <p>La imagen capturada aparecerá aquí.</p>
    </div>

  </body>
</html>
```

Puede apreciar los resultados en la Figura 13-4, el usuario ha hecho clic en el botón Mostrar imagen 1 y la aplicación ha cargado Image1.jpg. Hacer clic en Mostrar imagen 2 presentará Image2.jpg.

Ahora descarga imágenes utilizando Ajax y PHP. No está mal. Demos un vistazo al manejo de descargas de JavaScript.

Descarga de JavaScript con Ajax

A veces, cuando usa Ajax, el sitio al que uno se conecta devolverá JavaScript, y bien vale la pena saber cómo manejarlo. Un ejemplo notable de esto es la conexión a Google Suggest, que busca coincidencias con palabras clave escritas por el usuario; daremos un vistazo a Google Suggest en este capítulo.

Puede enviar cualquier tipo de código JavaScript del servidor a JavaScript, habilitado con Ajax. Este ejemplo es breve, javascript.html y javascript.php. En javascript.html hay una función

de JavaScript llamada display y javascript.php simplemente reproduce esa llamada a función, como instrucción de JavaScript:

```
<?php
  echo 'display()';
?>
```

Cuando el código en javascript.html lea esta instrucción de JavaScript,"display( )", ejecutará esa instrucción, que llamará la función display, para mostrar un mensaje.

Javascript.html comienza con un botón llamando la función getData, que hemos usado para descargar datos de Ajax, pasándole el nombre del script PHP para el acceso, javascript.php:

```
<form>
  <input type = "button" value = "Descargar JavaScript"
    onclick = "getData('javascript.php')">
</form>
```

Ésta es la clave —cuando descarga JavaScript desde un script PHP, puede usar la función eval de JavaScript para ejecutar ese código de JavaScript—. Ese código de JavaScript se descargará en la propiedad responseText, del objeto XMLHttpRequest; de modo que puede ejecutarlo de esta forma en getData:

```
function getData(dataSource)
{
  if(XMLHttpRequestObject) {

    XMLHttpRequestObject.open("GET", dataSource);

    XMLHttpRequestObject.onreadystatechange = function()
    {
      if (XMLHttpRequestObject.readyState == 4 &&
        XMLHttpRequestObject.status == 200) {
          eval(XMLHttpRequestObject.responseText);
      }
    }

    XMLHttpRequestObject.send(null);
  }
```

Por último, necesita escribir una función display en javascript.html a la que se llame cuando ejecute el código JavaScript descargado de javascript.php. Así se ve en nuestra aplicación Ajax completa, javascript.html, donde la función display muestra el mensaje"Acierto"en la página:

```
<html>
  <head>
    <title>Descarga de JavaScript con Ajax</title>

    <script language = "javascript">
      var XMLHttpRequestObject = false;
```

```
    if (window.XMLHttpRequest) {
      XMLHttpRequestObject = new XMLHttpRequest();
    } else if (window.ActiveXObject) {
      XMLHttpRequestObject = new ActiveXObject("Microsoft.XMLHTTP");
    }

    function getData(dataSource)
    {
      if(XMLHttpRequestObject) {
        XMLHttpRequestObject.open("GET", dataSource);

        XMLHttpRequestObject.onreadystatechange = function()
        {
          if (XMLHttpRequestObject.readyState == 4 &&
            XMLHttpRequestObject.status == 200) {
              eval(XMLHttpRequestObject.responseText);
          }
        }

        XMLHttpRequestObject.send(null);
      }
    }

    function display()
    {
      var targetDiv = document.getElementById("targetDiv");
      targetDiv.innerHTML = "Acierto";
    }
  </script>
</head>

<body>

  <h1>Descarga de JavaScript con Ajax</h1>

  <form>
    <input type = "button" value = "Descargar JavaScript"
      onclick = "getData('javascript.php')">
  </form>

  <div id="targetDiv">
    <p>Los datos se mostrarán aquí.</p>
  </div>

</body>
</html>
```

Los resultados están en la Figura 13-5, donde se ve javascript.html.

Cuando el usuario hace clic en el botón, se descarga el código JavaScript de javascript.php y se ejecuta, da un mensaje de acierto, como se aprecia en la Figura 13-6.

FIGURA 13-5 javascript.html

Conexión a Google Suggest

Observe la Figura 13-7, mostrando la página de Google Suggest, que encontrará en http://www.google.com/webhp?complete=1&hl=en.

Ahora escriba algo en el campo de texto y Google buscará lo que usted desea conforme escribe, proveyéndole el cuadro desplegable de la Figura 13-8.

Es un primer ejemplo de Ajax —los elementos del cuadro de lista desplegable se capturaron tras bambalinas usando Ajax y luego se mostraron en la página.

FIGURA 13-6 Descarga y ejecución de JavaScript

FIGURA 13-7 Google Suggest

FIGURA 13-8 Uso de Google Suggest

Así, ¿cómo puede conectarse a Google Suggest? Suponga que ha almacenado el término de búsqueda parcial en una variable llamada searchTerm —entonces podría conectarse a Google Suggest en esta dirección URL:

```
http://www.google.com/webhp?complete/search?hl=es&js=true&qu= + searchTerm;
```

¿Cómo se comunica Google Suggest con usted? Le devolverá código JavaScript llamando a una función de nombre sendRPCDone. Éstos son los parámetros enviados a esa función:

```
sendRPCDone(unusedVariable, searchTerm, arrayTerm, arrayResults, unusedArray)
```

¿Cómo se ve en realidad la llamada JavaScript recibida de Google Suggest? Bueno, si busca "ajax", éste es el tipo de código JavaScript que recibirá de Google:

```
sendRPCDone(frameElement, "ajax", new Array("ajax", "ajax amsterdam",

"ajax fc", "ajax ontario", "ajax grips", "ajax football club", "ajax public
library", "ajax football", "ajax soccer", "ajax pickering transit"), new
Array("3,840,000 results", "502,000" results, "710,000" results, "275,000 results",
"8,860 results", "573,000 results", "40,500 results", "454,000 results", "437,000
results", "10,700 results"), new Array("");
```

Parta de allí, escriba su propia función sendRPCDone, que mostrará los resultados devueltos por Google Suggest.

Vamos a escribir un ejemplo, google.html, que se conectará a Google Suggest utilizando Ajax y PHP. ¿Por qué necesita PHP aquí? Eso tiene que ver con un problema de Ajax —si usa Ajax, no puede acceder dominios excepto del que proviene la página, sin que el navegador muestre mensajes. Analizaremos esto a fondo más adelante.

Puede ver nuestra versión de una página de Google Suggest en la Figura 13-9. Se ve y actúa como la página real, salvo que la crearemos nosotros mismos, poniendo a Ajax a trabajar con un poco de ayuda de PHP.

El ejemplo google.html comienza con un campo de texto llamando una función de JavaScript, cuyo nombre es connectGoogleSuggest, cada vez que libera una tecla pulsada, ligándola al siguiente evento onkeyup del campo de texto:

```
<body>
  <h1>Conectando con Google Suggest</h1>

  Buscar <input id = "textField" type = "text"
    name = "textField" onkeyup = "connectGoogleSuggest(event)">
    .
    .
    .
</body>
```

En la función connectGoogleSuggest, puede comprobar si hay texto que enviar a Google Suggest, y de haberlo, envíe dicho texto a google.php como el parámetro qu (de query, consulta). Puede obtener el texto para enviar utilizando la propiedad value del campo de texto:

```
function connectGoogleSuggest(keyEvent)
{
  var input = document.getElementById("textField");
```

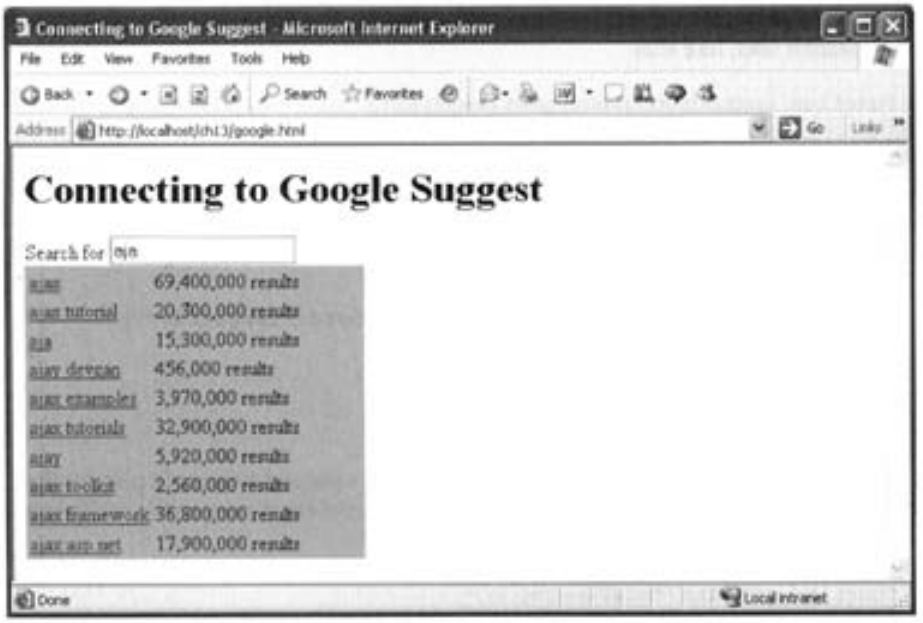

FIGURA 13-9 google.html

```
    if (input.value) {
      getData("google.php?qu=" + input.value);
    }
    .
    .
    .
}
```

Por otra parte, si no hay qué enviar, el usuario ha eliminado presumiblemente todo el texto del campo de texto, en cuyo caso deseará despejar la lista desplegable de coincidencias de Google Suggest. Esas coincidencias se mostrarán en realidad en una lista desplegable, en la práctica una tabla HTML, ingresada en un elemento , targetDiv, justo bajo el campo de texto:

```
<body>
  <h1>Conectando con Google Suggest</h1>

  Buscar <input id = "textField" type = "text"
    name = "textField" onkeyup = "connectGoogleSuggest(event)">

    <div id = "targetDiv"><div></div></div>

</body>
```

En la función connectGoogleSuggest, puede aclarar la lista desplegable si el usuario ha eliminado todo el texto de búsqueda, de esta forma:

```
function connectGoogleSuggest(keyEvent)
{
  var input = document.getElementById("textField");
  if (input.value) {
    getData("google.php?qu=" + input.value);
  }
  else {
    var targetDiv = document.getElementById("targetDiv");
    targetDiv.innerHTML = "<div></div>";
  }
}
```

De modo que, cuando el usuario ingresa texto para búsquedas, éste se pasa a google.php por medio de la función getData —y cuando Google Suggest devuelva JavaScript para ejecutarlo, ejecutaremos ese código JavaScript utilizando la función eval de JavaScript:

```
function getData(dataSource)
{
  var XMLHttpRequestObject = false;

  if (window.XMLHttpRequest) {
    XMLHttpRequestObject = new XMLHttpRequest();
  } else if (window.ActiveXObject) {
    XMLHttpRequestObject = new ActiveXObject("Microsoft.XMLHTTP");
  }

  if(XMLHttpRequestObject) {
    XMLHttpRequestObject.open("GET", dataSource);

    XMLHttpRequestObject.onreadystatechange = function()
    {
      if (XMLHttpRequestObject.readyState == 4 &&
        XMLHttpRequestObject.status == 200) {
          eval(XMLHttpRequestObject.responseText);
      }
    }

    XMLHttpRequestObject.send(null);
  }
}
```

Bueno, es tiempo de que google.php tome control. Este script le conecta con Google Suggest y descarga JavaScript para ejecutar la aplicación.

Ésta es la clave para tal aplicación: ¿por qué no conectarse directamente a Google Suggest desde google.html? ¿Por qué debe usar a google.php como intermediario? Necesita google.php por el problema mencionado en Ajax: utilizando Ajax, no puede acceder otro dominio excepto

del que procede su página, sin ver diálogos de advertencia del navegador. Esos diálogos de advertencia aparecen por seguridad —el navegador asume que su página hace algo indebido.

Para evitar esos diálogos de advertencia y conectarse a otros dominios utilizando Ajax, necesita relevar su solicitud mediante código para el servidor. De esa forma, el navegador sólo lo percibe accediendo un script en el mismo dominio que su página principal. Pero ese script puede conectarse a otro sitio y el navegador ni se enterará de ello.

La dirección URL que desea para acceder Google Suggest es http:///www.google.com/complete/search, que le enviará el término de búsqueda escrito parcialmente por el usuario, empleando el parámetro qu.Y podrá abrir esa dirección URL desde PHP, usando fopen. Así, ésta es la forma de enviar a Google Suggest el término de búsqueda parcialmente tecleado por el usuario, que consigue un manejador de archivo representando la respuesta en google.php:

```
<?php
  $filehandle =
    fopen("http://www.google.com/complete/search?hl=es&js=true&qu=" . $_GET["qu"],
"r");
        .
        .
        .
?>
```

Ahora tiene un manejador de archivo representando la respuesta JavaScript de Google Suggest, al término de la búsqueda escrita parcialmente enviada por usted. Puede recorrer esa respuesta en un ciclo while y feof:

```
<?php
  $filehandle =
    fopen("http://www.google.com/complete/search?hl=es&js=true&qu=" . $_GET["qu"],
"r");
  while (!feof($filehandle)){
        .
        .
        .
?>
```

Asimismo, puede descargar y reproducir con echo, la respuesta de Google Suggest a su consulta, línea por línea, de esta forma:

```
<?php
  $filehandle =
    fopen("http://www.google.com/complete/search?hl=es&js=true&qu=" . $_GET["qu"],
"r");

  while (!feof($filehandle)){
    $download = fgets($filehandle);
    echo $download;
  }
?>
```

Lo que resta por hacer es cerrar la conexión con Google Suggest:

```
<?php
  $filehandle =
    fopen("http://www.google.com/complete/search?hl=es&js=true&qu=" . $_GET["qu"],
"r");

  while (!feof($filehandle)){
    $download = fgets($filehandle);
    echo $download;
  }

fclose($filehandle);
?>
```

Bueno, eso recurre a PHP para resolver el problema de la falta de acceso a otros dominios, salvo el principal, de su página con Ajax. Este script se conecta a Google Suggest y recibe la respuesta de Google según haya escrito el usuario.

Esa respuesta aparece en forma de JavaScript, como ya se vio, de esta manera:

```
sendRPCDone(frameElement, "ajax", new Array("ajax", "ajax amsterdam",
"ajax fc", "ajax ontario", "ajax grips", "ajax football club", "ajax public
library", "ajax football", "ajax soccer", "ajax pickering transit"), new
Array("3,840,000 results", "502,000" results, "710,000" results, "275,000 results",
"8,860 results", "573,000 results", "40,500 results", "454,000 results", "437,000
results", "10,700 results"), new Array("");
```

Y la página google.html ejecuta dicho código JavaScript con la función eval:

```
XMLHttpRequestObject.onreadystatechange = function()
{
  if (XMLHttpRequestObject.readyState == 4 &&
    XMLHttpRequestObject.status == 200) {
      eval(XMLHttpRequestObject.responseText);
  }
}
```

Es su responsabilidad escribir sendRPCDone. Así define Google los parámetros de esta función:

```
function sendRPCDone(unusedVariable, searchTerm, arrayTerm,
  arrayResults, unusedArray)
{
  .
  .
  .
}
```

Éstos son los parámetros:

- **unusedVariable** Una variable no utilizada
- **searchTerm** Término de búsqueda del usuario
- **arrayTerm** Matriz de términos coincidentes
- **arrayResults** Número de coincidencias con cada término de la matriz
- **unusedArray** Una matriz no utilizada

Esta información se mostrará en una tabla HTML en el elemento targetDiv del cuerpo de la página:

```
<body>
  <h1>Conectando con Google Suggest</h1>

  Buscar <input id = "textField" type = "text"
    name = "textField" onkeyup = "connectGoogleSuggest(event)">

    <div id = "targetDiv"><div></div></div>

</body>
```

El objetivo de la función sendRPCDone es crear la tabla HTML, en una variable llamada data para mostrar los resultados y exhibirla:

```
function sendRPCDone(unusedVariable, searchTerm, arrayTerm,
  arrayResults, unusedArray)
{
  var data = "<table>";
  .
  .
  .
  data += "</table>";

  var targetDiv = document.getElementById("targetDiv");

  targetDiv.innerHTML = data;
}
```

Puede recorrer los resultados de Google de la siguiente forma:

```
function sendRPCDone(unusedVariable, searchTerm, arrayTerm,
  arrayResults, unusedArray)
{
  var data = "<table>";
  var loopIndex;

  if (arrayResults.length != 0) {
    for (var loopIndex = 0; loopIndex < arrayResults.length;
      loopIndex++) {
      .
      .
```

```
        .
      }
    }
    data += "</table>";

    var targetDiv = document.getElementById("targetDiv");

    targetDiv.innerHTML = data;
  }
```

En cada línea de la tabla HTML mostrará un hipervínculo al término de búsqueda —en caso de que el usuario decida que ése es el término que desea buscar— y el número de coincidencias encontradas por Google para ese término:

```
function sendRPCDone(unusedVariable, searchTerm, arrayTerm,
  arrayResults, unusedArray)
{
  var data = "<table>";
  var loopIndex;

  if (arrayResults.length != 0) {
    for (var loopIndex = 0; loopIndex < arrayResults.length;
      loopIndex++) {
      data += "<tr><td>" +
      "<a href='http://www.google.com/search?q=" +
      arrayTerm[loopIndex] + "'>" + arrayTerm[loopIndex] +
      '</a></td><td>' + arrayResults[loopIndex] + "</td></tr>";
    }
  }

  data += "</table>";

  var targetDiv = document.getElementById("targetDiv");

  targetDiv.innerHTML = data;
}
```

También puede imprimir estilo al elemento targetDiv, con un color de fondo como éste, en un elemento <style> para resaltarlo:

```
<head>
  <title>Conectando con Google Suggest</title>
  <style>
  #targetDiv {
    background-color: #FFAAAA;
    width: 40%;
  }
</style>
    .
    .
    .
</head>
```

Y eso completa google.html —con esta página y google.php se ha conectado a Google Suggest, descargado JavaScript y ejecutado ese código JavaScript—. Éste es el archivo google.html completo:

```
<html>
  <head>
    <title>Conectando con Google Suggest</title>
    <style>
    #targetDiv {
      background-color: #FFAAAA;
      width: 40%;
    }
    </style>

    <script language = "javascript">
      function getData(dataSource)
      {
        var XMLHttpRequestObject = false;

        if (window.XMLHttpRequest) {
          XMLHttpRequestObject = new XMLHttpRequest();
        } else if (window.ActiveXObject) {
          XMLHttpRequestObject = new ActiveXObject("Microsoft.XMLHTTP");
        }

        if(XMLHttpRequestObject) {
          XMLHttpRequestObject.open("GET", dataSource);

          XMLHttpRequestObject.onreadystatechange = function()
          {
            if (XMLHttpRequestObject.readyState == 4 &&
              XMLHttpRequestObject.status == 200) {
                eval(XMLHttpRequestObject.responseText);
            }
          }

          XMLHttpRequestObject.send(null);
        }
      }

      function connectGoogleSuggest(keyEvent)
      {
        var input = document.getElementById("textField");

        if (input.value) {
          getData("google.php?qu=" + input.value);
        }
        else {

          var targetDiv = document.getElementById("targetDiv");
          targetDiv.innerHTML = "<div></div>";
        }
      }
```

```
      function sendRPCDone(unusedVariable, searchTerm, arrayTerm,
        arrayResults, unusedArray)
      {
        var data = "<table>";
        var loopIndex;

        if (arrayResults.length != 0) {
          for (var loopIndex = 0; loopIndex < arrayResults.length;
            loopIndex++) {
            data += "<tr><td>" +
            "<a href='http://www.google.com/search?q=" +
            arrayTerm[loopIndex] + "'>" + arrayTerm[loopIndex] +
            '</a></td><td>' + arrayResults[loopIndex] + "</td></tr>";
          }
        }

        data += "</table>";

        var targetDiv = document.getElementById("targetDiv");

        targetDiv.innerHTML = data;
      }
    </script>
  </head>

  <body>
    <H1>Conectando con Google Suggest</H1>

    Buscar <input id = "textField" type = "text"
      name = "textField" onkeyup = "connectGoogleSuggest(event)">

      <div id = "targetDiv"><div></div></div>

  </body>
</html>
```

Uno de los aspectos clave aquí fue rodear la restricción de Ajax de no poder conectar con otros dominios.

Conexión a otros dominios utilizando Ajax

Saber cómo conectarse a otros dominios usando Ajax es una habilidad valiosa, como se ve en el ejemplo de Google Suggest. La clave está en relevar la solicitud a otro dominio usando un script PHP en el servidor.

Puede adaptar google.php para realizar esta acción. Usted abre el otro dominio con fopen, después lee la respuesta de ese dominio, línea por línea, con fgets y feof. Cuando todo está listo, cierra la URL ajena:

```
<?php
  $filehandle =
    fopen("http://www.google.com/complete/search?hl=es&js=true&qu=" . $_GET["qu"],
      "r");
```

```
  while (!feof($filehandle)){
    $download = fgets($filehandle);
    echo $download;
  }

  fclose($filehandle);
?>
```

Inicio de sesión con Ajax y PHP

Ésta es otra tarea para la que se usa Ajax a menudo y PHP tiene que ver también: comprobar inicios de sesión. Por ejemplo, puede verificar el inicio de sesión de un usuario discretamente o permitirle elegir su nombre de usuario y contraseña. Este ejemplo, log.html, concede al usuario la elección de su nombre —y, si ya se encuentra ese nombre registrado, log.html informa al usuario de ese hecho—. Puede ver cómo funciona esto en la Figura 13-10 —la aplicación comprueba el nombre de usuario ingresado por la persona, mientras ésta lo teclea y si ya no está disponible, como aquí, le informa del hecho.

Este ejemplo lee claves conforme el usuario las escribe, llamando a una función JavaScript de nombre checker, comprobando el nuevo nombre de usuario:

```
<body>

  <H1>Elija un nombre de usuario</H1>

  Escriba un nombre de usuario: <input id = "textField" type = "text"
    name = "textField" onkeyup = "checker()">

    <div id = "targetDiv"><div></div></div>

</body>
```

FIGURA 13-10 Comprobación de un nombre de usuario

La función checker llama a getData con la URL log.php, enviando el nombre de usuario que éste escribe en log.php, mediante el uso del nombre de parámetro "qu":

```
function checker()
{
  var targetDiv = document.getElementById("targetDiv");
  targetDiv.innerHTML = "<div></div>";

  if (document.getElementById("textField").value) {
    getData("log.php?qu=" +
      document.getElementById("textField").value);
  }
}
```

La función getData se conecta al servidor y descarga los resultados:

```
function getData(dataSource)
{
  if(XMLHttpRequestObject) {
    XMLHttpRequestObject.open("GET", dataSource);

    XMLHttpRequestObject.onreadystatechange = function()
    {
      if (XMLHttpRequestObject.readyState == 4 &&
        XMLHttpRequestObject.status == 200) {
        .
        .
        .
      }
    }

    XMLHttpRequestObject.send(null);
  }
}
```

En este caso, el script PHP devolverá "OK" si el nombre de usuario elegido es correcto y "notOK", si ya está en uso. Si el nombre de usuario ya fue tomado, este ejemplo mostrará el mensaje "Lo sentimos, ese nombre de usuario no está disponible." en un elemento :

```
function getData(dataSource)
{
  if(XMLHttpRequestObject) {
    XMLHttpRequestObject.open("GET", dataSource);

    XMLHttpRequestObject.onreadystatechange = function()
    {
      if (XMLHttpRequestObject.readyState == 4 &&
        XMLHttpRequestObject.status == 200) {
          if(XMLHttpRequestObject.responseText == "notOK"){
      var targetDiv = document.getElementById("targetDiv");

      targetDiv.innerHTML =
        "<div>Lo sentimos, ese nombre de usuario no está disponible.</div>";
```

```
                }
            }
          }
          XMLHttpRequestObject.send(null);
        }
      }
```

El script PHP, log.php, comprueba el nombre de usuario que se envía usando el parámetro qu y devuelve el texto"notOK", en caso de que el nombre de usuario ya esté en uso:

```
<?php
    if ($_GET["qu"] == "steve"){
      echo "notOK";
    }
        .
        .
        .
?>
```

De lo contrario, devuelve"OK":

```
<?php
    if ($_GET["qu"] == "steve"){
      echo "notOK";
    }
    else {
      echo "OK";
    }
?>
```

Y eso es todo lo que necesita.

Obtención de datos con solicitudes de encabezados y Ajax

Puede usar también Ajax para comprobar archivos en el servidor antes de trabajar con ellos. Este ejemplo, head.html, lee los encabezados HTTP devueltos por el servidor para el archivo head.html mismo, haciendo una lista de ellos, como verá en la Figura 13-11.

Como se aprecia en la figura, cuando envía una solicitud de encabezado para un archivo, recibe de vuelta información como ésta, indicando servidor, última fecha de modificación del archivo y longitud:

```
Server: Microsoft-IIS/5.1 Date: Mon 28, May 2007 17:57:54 GMT Content-Type: text/
html Accept-Ranges: bytes Last-Modified: Mon, 28 May 2007 17:57:49 GMT ETag:
"3c058b451a1c71:a2f" Content-Length: 1266
```

Vale la pena observar cómo se trabaja con solicitudes de encabezados en Ajax, y este ejemplo, head.html, presenta su funcionamiento. El ejemplo comienza con el botón de la figura, ligado a la función getData:

```
<body>

  <h1>Obtención de información de encabezados</h1>
```

```
  <form>
    <input type = "button" value = "Obtener datos de encabezados"
      onclick = "getData('head.html', 'targetDiv')">
  </form>
  <div id="targetDiv">
    <div></div>
  </div>
</body>
```

En la función getData, usted especifica que desea información de encabezados especificando abrir el documento con el método HEAD (no GET ni POST):

```
function getData(dataSource, divID)
{
  if(XMLHttpRequestObject) {
    var targetDiv = document.getElementById(divID);
    XMLHttpRequestObject.open("HEAD", dataSource);
.
.
.
}
```

Y cuando se descarguen sus datos, puede recuperar los datos de los encabezados con el método getAllResponseHeaders del objeto XMLHttpRequest:

```
function getData(dataSource, divID)
{
  if(XMLHttpRequestObject) {
    var targetDiv = document.getElementById(divID);
    XMLHttpRequestObject.open("HEAD", dataSource);
    XMLHttpRequestObject.onreadystatechange = function()
    {
      if (XMLHttpRequestObject.readyState == 4 &&
        XMLHttpRequestObject.status == 200) {
          targetDiv.innerHTML =
            XMLHttpRequestObject.getAllResponseHeaders();
      }
    }
    XMLHttpRequestObject.send(null);
  }
}
```

Este código da los resultados en la Figura 13-11. Genial.

Puede utilizar el método getAllResponseHeaders para leer todos los encabezados del servidor, pero también emplear el método getResponseHeader para leer el valor de cualquier encabezado en particular. Este ejemplo, lastModified.html, comprueba su última fecha de modificación y la muestra, como se ve en la Figura 13-12

Esto se pone en acción llamando el método getResponseHeader, del objeto

FIGURA 13-11 Obtención de información acerca de encabezados

XMLHttpRequest de esta forma: getResponseHeader("Last-Modified"). Éste es el uso del método —pasa el nombre del encabezado de su interés—. Así es como lastModified.html pone esto a trabajar:

```
<html>
  <head>
    <title>Obtención de la última fecha de modificación de un documento</title>

    <script language = "javascript">
```

FIGURA 13-12 Comprobación de la última fecha de modificación de un documento

```
    var XMLHttpRequestObject = false;
    if (window.XMLHttpRequest) {
      XMLHttpRequestObject = new XMLHttpRequest();
    } else if (window.ActiveXObject) {
      XMLHttpRequestObject = new ActiveXObject("Microsoft.XMLHTTP");
    }

    function getData(dataSource, divID)
    {
      if(XMLHttpRequestObject) {
        var targetDiv = document.getElementById(divID);
        XMLHttpRequestObject.open("HEAD", dataSource);

        XMLHttpRequestObject.onreadystatechange = function()
        {
          if (XMLHttpRequestObject.readyState == 4 &&
            XMLHttpRequestObject.status == 200) {
             targetDiv.innerHTML =  "Este documento fue modificado por última vez el " +
                  XMLHttpRequestObject.getResponseHeader(
                     "Last-Modified");
          }
        }

        XMLHttpRequestObject.send(null);
      }
    }
  </script>
</head>

<body>

  <H1>Obtención de la última fecha de modificación de un documento</H1>

  <form>
    <input type = "button" value = "Obtener la fecha de este documento"
      onclick = "getData('lastModified.html', 'targetDiv')">
  </form>

  <div id="targetDiv">
    <div></div>
  </div>

</body>
</html>
```

CAPÍTULO 14

Trazo de imágenes en el servidor

Todo este capítulo trata sobre el uso de PHP para crear gráficos en el servidor. Puede trazar casi cualquier cosa y devolverla al navegador en diferentes formatos gráficos (JPEG, PNG, etc.) mediante PHP.

Para hacer que funcione, debe instalar el soporte GD2 para PHP. Ya que se hace de diferentes maneras, compruebe las instrucciones de instalación de PHP para su sistema operativo. Por ejemplo, podría tener que comprobar la casilla de selección CD2, cuando el programa de instalación pregunte qué opciones desea instalar.

¿Pero qué funciones gráficas están disponibles? Ésta es la lista:

Función	Descripción
gd_info	Recupera información acerca de la biblioteca GD
getimagesize	Devuelve el tamaño de una imagen
image_type_to_extension	Devuelve la extensión del archivo de la imagen
image_type_to_mime_type	Devuelve el Tipo Mime de imagen que detecta getimagesize, exif_read_data, exif_thumbnail, exif_imagetype
image2wbmp	Envía la imagen al navegador o archivo
imagealphablending	Establece la combinación de una imagen
imageantialias	Establece si se usarán funciones antialias o no
imagearc	Traza un arco
imagechar	Traza un carácter horizontalmente
imagecharup	Traza un carácter verticalmente
imagecolorallocate	Asigna color a una imagen
imagecolorallocatealpha	Asigna color a una imagen y la configuración alfa
imagecolorat	Devuelve el índice del color de un píxel
imagecolorclosest	Devuelve el índice del color más próximo al color dado
imagecolorclosestalpha	Devuelve el índice del color más próximo al color dado y configuración alfa

Función	Descripción
imagecolorclosestthwb	Devuelve el índice del color que tiene el matiz, cantidad de blanco y negro más próximos al color dado
imagecolordeallocate	Desestima un color
imagecolorexact	Devuelve el índice del color dado
imagecolorexactalpha	Devuelve el índice del color dado y la configuración alfa
imagecolormatch	Hace que los colores de la versión de la paleta de una imagen se aproximen más a la versión de color verdadero
imagecolorresolve	Devuelve el índice del color dado o su alternativa más próxima posible
imagecolorresolvealpha	Devuelve el índice del color dado y configuración alfa o su alternativa más próxima posible
imagecolorset	Establece el color del índice de paleta dado
imagecolorsforindex	Devuelve los colores de un índice
imagecolorstotal	Devuelve el número de colores de la paleta de una imagen
imagecolortransparent	Establece transparente un color
imageconvolution	Aplica una matriz de circunvolución de 3 x 3
imagecopy	Copia sólo parte de una imagen
imagecopymerge	Copia y combina parte de una imagen
imagecopymergegray	Copia y combina parte de una imagen con escala de grises
imagecopyresampled	Copia y cambia el tamaño de parte de una imagen, usando muestreo repetido
imagecopyresized	Copia y cambia el tamaño de parte de una imagen
imagecreate	Crea una nueva imagen
imagecreatefromgd	Crea una nueva imagen a partir de un archivo GD o URL
imagecreatefromgd2	Crea una nueva imagen a partir de un archivo GD2 o URL
imagecreatefromgd2part	Crea una nueva imagen a partir de una parte dada de un archivo GD2 o URL
imagecreatefromgif	Crea una nueva imagen a partir de un archivo GIF
imagecreatefromjpeg	Crea una nueva imagen a partir de un archivo JPEG
imagecreatefrompng	Crea una nueva imagen a partir de un archivo PNG
imagecreatefromstring	Crea una nueva imagen a partir del flujo de imágenes
imagecreatefromwbmp	Crea una nueva imagen a partir de un archivo WBMP
imagecreatefromxbm	Crea una nueva imagen a partir de un archivo XBM
imagecreatefromxpm	Crea una nueva imagen a partir de un archivo XPM
imagecreatetruecolor	Crea una nueva imagen de color verdadero
imagedashedline	Traza una línea punteada
imagedestroy	Destruye una imagen
imageellipse	Traza una elipse

Función	Descripción
imagefill	Relleno por inundación de color
imagefilledarc	Traza una elipse parcial y la rellena
imagedfilledellipse	Traza una elipse rellena
imagefilledpolygon	Traza un polígono relleno
imagefilledrectangle	Traza un rectángulo relleno
imagefilltoborder	Relleno por inundación con un color específico
imagefilter	Aplica un filtro a una imagen
imagefontheight	Devuelve la altura de la fuente
imagefontwidthh	Devuelve el ancho de la fuente
imageftbbox	Proporciona el cuadro limítrofe de texto
imagefttext	Escribe texto en la imagen
imagegammacorrect	Aplica una corrección gamma
imagegd	Envía una imagen GD al navegador o archivo
imagegd2	Envía una imagen GD2 al navegador o archivo
imagegif	Envía una imagen al navegador o archivo
imageinterlace	Habilita o deshabilita operaciones de entrelazado
imageistruecolor	Determina si una imagen es de color verdadero
imagejpeg	Envía una imagen al navegador o archivo
imagelayereffect	Establece la bandera de combinación alfa para utilizar efectos por capas
imageline	Traza una línea
imageloadfont	Carga una nueva fuente
imagepalettecopy	Copia la paleta de una imagen
imagepng	Envía una imagen PNG al navegador o un archivo
imagepolygon	Traza un polígono
imagepsbbox	Especifica el cuadro de texto limítrofe usando fuentes PostScript Type1
imagepsencodefont	Cambia la codificación de caracteres de una fuente
imagepsextendfont	Extiende o condensa una fuente
imagepsfreefont	Libera la memoria usada actualmente por una fuente PostScript Type1
imagepsloadfont	Inclina una fuente
imagepstext	Traza texto mediante fuentes PostScript Type1
imagerectangle	Traza un rectángulo
imagerotate	Gira una imagen a un ángulo dado

Función	Descripción
imagesavealpha	Establece la bandera para guardar información completa del canal alfa cuando se guardan imágenes PNG
imagesetbrush	Establece la imagen del pincel para el trazo de líneas
imagesetpixel	Establece un píxel
imagesetstyle	Establece el estilo para el trazo de líneas
imagesetthickness	Establece el grosor para el trazo de líneas
imagesettile	Establece la imagen en mosaico para relleno
imagestring	Traza una cadena horizontalmente
imagestringup	Traza una cadena verticalmente
imagesx	Devuelve el ancho de la imagen
imagesy	Devuelve el alto de la imagen
imagetruecolortopalette	Convierte una imagen de color verdadero en una de paleta de colores
imagettfbbox	Proporciona el cuadro limítrofe de un texto usando fuentes TrueType
imagettftext	Escribe texto en la imagen usando fuentes TrueType
imagetypes	Devuelve los tipos de imágenes admitidos
imagewbmp	Envía una imagen al navegador o archivo
imagexbm	Envía una imagen XBM al navegador o archivo
iptcembed	Incorpora datos IPTC binarios en una imagen JPEG
iptcparse	Analiza la estructura de un bloque IPTC binario
jpeg2wbmp	Convierte un archivo de imagen JPEG en WBMP
png2wbmp	Convierte un archivo de imagen PNG en WBMP

Bueno, comencemos con la creación de una imagen simple.

Creación de una imagen

Para crear una imagen con la cual trabajar en la memoria, se comienza con la función imagecreate de GD2:

```
imagecreate(x_size, y_size)
```

Los parámetros *x_size* y *y_size* están en píxeles.

Así se crea una primera imagen:

```
$image_height = 100;
$image_width = 300;

$image = imagecreate($image_width, $image_height);
        .
        .
        .
```

Después, para establecer los colores a usarse en la imagen, se usa la función imagecolorallocate:

```
imagecolorallocate (image, red, green, blue)
```

Debe pasar a esta función la imagen con que está trabajando, además de los componentes rojo, verde y azul (red, green, blue) como valores de 0 a 255. Por ejemplo, si desea rojo sólido, pasaría a imagecolorallocate un valor rojo de 255, mientras verde y azul serían de 0.

La primera vez que usted llama a imagecolorallocate, esta función establece el color de fondo; las llamadas subsecuentes establecen diferentes colores para dibujar. Así se establece el color de fondo en gris claro (rojo = 200, verde = 200, azul = 200):

```
$image = imagecreate($image_width, $image_height);

$back_color = imagecolorallocate($image, 200, 200, 200);
        .
        .
        .
```

Para devolver una imagen JPEG al navegador, debe indicarle que está haciendo eso con la función header, para establecer el tipo de imagen y luego enviarla con la función imagejpeg:

```
$image_height = 100;
$image_width = 300;

$image = imagecreate($image_width, $image_height);

$back_color = imagecolorallocate($image, 200, 200, 200);

header('Content-Type: image/jpeg');
imagejpeg($image);
        .
        .
        .
```

Éstas son algunas funciones para crear imágenes para diversos formatos que puede usar:

- **imagegif** Envía una imagen GIF al navegador o un archivo
- **imagejpeg** Envía una imagen JPEG al navegador o un archivo
- **imagewbmp** Envía una imagen WBMP al navegador o un archivo
- **imagepng** Envía una imagen PNG al navegador o un archivo

Después de enviar la imagen, puede destruir el objeto de imagen con la función imagedestroy; todo esto se muestra en phpimage.php:

```
<?php
  $image_height = 100;
  $image_width = 300;

  $image = imagecreate($image_width, $image_height);

  $back_color = imagecolorallocate($image, 200, 200, 200);
```

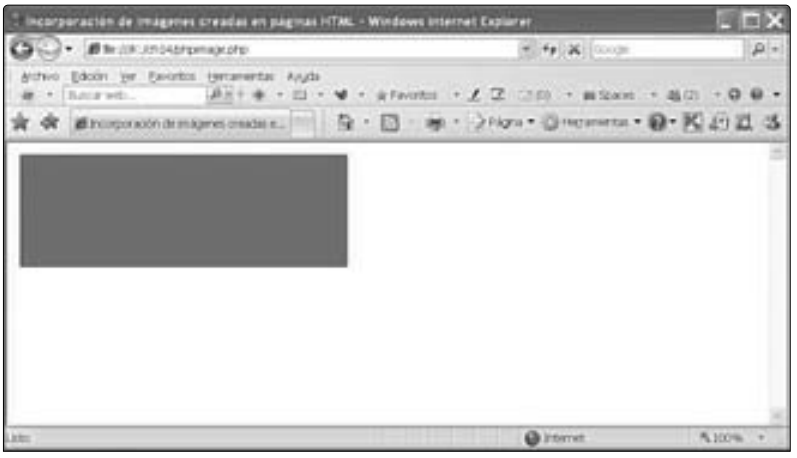

FIGURA 14-1 Creación de la imagen de un rectángulo en el servidor

```
  header('Content-Type: image/jpeg');

  imagejpeg($image);
  imagedestroy($image);

?>
```

Puede ver los resultados en la Figura 14-1. Todo lo que aparece es un rectángulo en el fondo de este ejemplo. Sin embargo, hemos logrado progresar (es una imagen real la que ve en la figura).

Por otra parte, es una página Web rara que muestra simplemente una imagen. En general, dicha imagen aparece en una página HTML.

Presentación de imágenes en páginas HTML

La imagen creada en phpimage.php es JPEG estándar; así que no hay razón por la que no pueda incorporarla en una página Web. ¿Cómo haría eso? Con un elemento <img> estándar, desde luego. Por ejemplo, si tuviera una imagen JPEG en el servidor, image.jpg, podría mostrarla de esta forma en una página Web:

```
<img src="image.jpg">
```

De la misma forma, puede dar el nombre del script que genera una imagen JPEG, phpimage.php, como el atributo src de esta manera:

```
<img src="phpimage.php">
```

FIGURA 14-2 Cómo mostrar la imagen de un rectángulo en una página Web

Así se ve una página Web, phpimage.html, mostrando el rectángulo en blanco creado por phpimage.php:

```
<html>
  <head>
    <title>
      Incorporación de imágenes creadas en páginas HTML
    </title>
  </head>

  <body>
    <h1>
      Incorporación de imágenes creadas en páginas HTML
    </h1>
      Ésta es una imagen en blanco que se crea en el servidor:
      <br>
      <img src="phpimage.php">
  </body>
</html>
```

Puede ver los resultados en la Figura 14-2, donde aparece la imagen JPEG creada por phpimage.php, incorporada en una página HTML. No está mal.

Bueno, eso crea una imagen básica y la muestra. ¿Qué tal si trazamos algunos gráficos reales?

Trazo de líneas

Puede trazar líneas con la función imageline:

```
imageline(image, x1, y1, x2, y2, color)
```

Esta función traza una línea de (*x1*, *y1*) a (*x2*, *y2*) (el extremo superior izquierdo de la imagen es (0, 0) y todas las medidas están en píxeles) en la imagen *image*, usando el color de trazo *color*.

Es momento de trazar líneas en un nuevo script, phpline.php. Este ejemplo comienza con la creación del mismo fondo en la imagen ya vista:

```
<?php
  $image_height = 100;
  $image_width = 300;
  $image = imagecreate($image_width, $image_height);
  $back_color = imagecolorallocate($image, 200, 200, 200);
       .
       .
       .
?>
```

Ahora estableceremos el color de trazo para las líneas. En este caso usaremos negro, cuyos valores son rojo, verde y azul en 0:

```
<?php
    $image_height = 100;
    $image_width = 300;
    $image = imagecreate($image_width, $image_height);
    $back_color = imagecolorallocate($image, 200, 200, 200);
    $draw_color = imagecolorallocate($image, 0, 0, 0);
        .
        .
        .
?>
```

Luego trace líneas con imageline:

```
<?php
    $image_height = 100;
    $image_width = 300;

    $image = imagecreate($image_width, $image_height);

    $back_color = imagecolorallocate($image, 200, 200, 200);

    $draw_color = imagecolorallocate($image, 0, 0, 0);

    imageline($image, 40, 20, 90, 80, $draw_color);

    imageline($image, 30, 90, 250, 10, $draw_color);

    imageline($image, 110, 20, 140, 90, $draw_color);
        .
```

```
        .
        .
?>
```

Y envíe la imagen al navegador, para luego destruirla de esta forma:

```
<?php
    $image_height = 100;
    $image_width = 300;

    $image = imagecreate($image_width, $image_height);

    $back_color = imagecolorallocate($image, 200, 200, 200);

    $draw_color = imagecolorallocate($image, 0, 0, 0);

    imageline($image, 40, 20, 90, 80, $draw_color);

    imageline($image, 30, 90, 250, 10, $draw_color);

    imageline($image, 110, 20, 140, 90, $draw_color);

    header('Content-Type: image/jpeg');

    imagejpeg($image);

    imagedestroy($image);
?>
```

Así luce la página Web, phpline.html, mostrando las líneas creadas por phpline.php:

```
<html>
  <head>
    <title>
      Trazo de líneas
    </title>
  </head>

  <body>
    <h1>
      Trazo de líneas
    </h1>
      Estas líneas se trazaron en el servidor:
      <br>
      <img src="phpline.php">
  </body>
</html>
```

Puede ver los resultados en la Figura 14-3, las líneas creadas por phpline.php aparecen en una página HTML. Genial.

Puede ajustar los parámetros de líneas como éstas; por ejemplo, trazar líneas más gruesas y eso veremos a continuación.

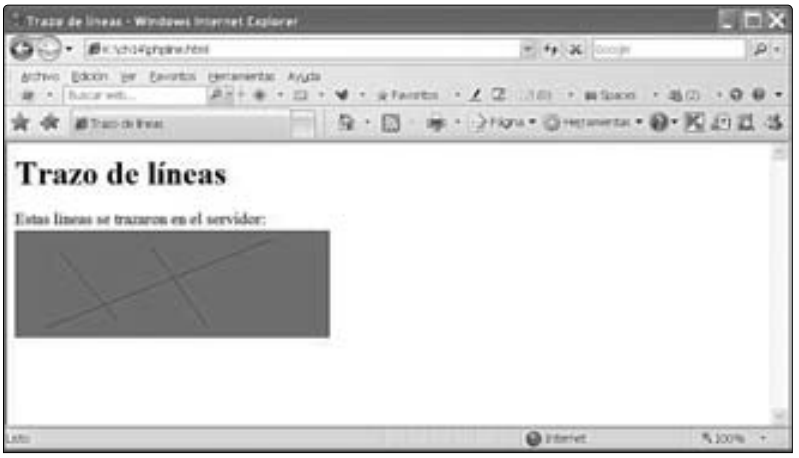

FIGURA 14-3 Cómo mostrar líneas en una página Web

Cómo establecer el grosor de la línea

Puede establecer el grosor del trazo cuando crea una imagen, mediante la función imagesetthickness:

```
imagesetthickness(image, thickness)
```

Cuando llama a esta función y le pasa una imagen, establece ancho del trazo de esa imagen, en píxeles. Por ejemplo, puede definir el grosor de líneas en phpline.php a 6 píxeles (en lugar del valor predeterminado de 1 píxel), de esta forma en phpthickline.php:

```
<?php
    $image_height = 100;
    $image_width = 300;

    $image = imagecreate($image_width, $image_height);
    imagesetthickness($image, 6);

    $back_color = imagecolorallocate($image, 200, 200, 200);

    $draw_color = imagecolorallocate($image, 0, 0, 0);

    imageline($image, 40, 20, 90, 80, $draw_color);
    .
    .
    .
    imagedestroy($image);
?>
```

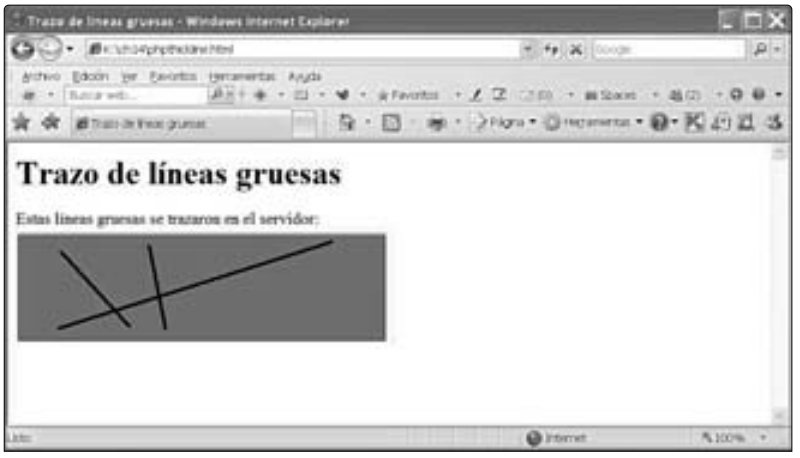

FIGURA 14-4 Cómo mostrar líneas gruesas en una página Web

Así vería la página Web phpthickline.html, mostrando las líneas más gruesas creadas en phpthickline.php:

```
<html>
  <head>
    <title>
      Trazo de líneas gruesas
    </title>
  </head>

  <body>
    <h1>
      Trazo de líneas gruesas
    </h1>
      Estas líneas gruesas se trazaron en el servidor:
      <br>
      <img src="phpthickline.php">
  </body>
</html>
```

Y puede ver los resultados en la Figura 14-4, las líneas gruesas aparecen en una página HTML.

Trazo de rectángulos

Puede dibujar muchas figuras usando sólo líneas, pero hay más funciones de GD a considerar. Una de ellas, imagerectangle, traza rectángulos:

```
imagerectangle(image, x1, y1, x2, y2, color)
```

Esta función crea un rectángulo de color *color* en la imagen *image*, a partir de la coordenada superior izquierda *x1, y1*, que termina en la coordenada inferior derecha *x2, y2*.

Este ejemplo, phprectangle.php, traza varios rectángulos:

```
<?php

    $image_height = 100;

    $image_width = 300;

    $image = imagecreate($image_width, $image_height);

    $back_color = imagecolorallocate($image, 200, 200, 200);

    $draw_color = imagecolorallocate($image, 0, 0, 0);

    imagerectangle($image, 30, 20, 50, 90, $draw_color);

    imagerectangle($image, 70, 20, 140, 90, $draw_color);

    imagerectangle($image, 170, 20, 270, 90, $draw_color);

    imagerectangle($image, 10, 40, 280, 80, $draw_color);

    header('Content-Type: image/jpeg');

    imagejpeg($image);

    imagedestroy($image);
?>
```

Y así podría incorporar esos rectángulos en una página Web, phprectangle.html:

```
<html>
  <head>
    <title>
      Trazo de rectángulos
    </title>
  </head>
  <body>
    <h1>
      Trazo de rectángulos
    </h1>
      Estos rectángulos se trazaron en el servidor:
      <br>
      <img src="phprectangle.php">
  </body>
</html>
```

En los resultados de la Figura 14-5, aparecen los rectángulos:

FIGURA 14-5 Cómo mostrar rectángulos en una página Web

Trazo de elipses

¿Desea trazar circunferencias o elipses? Use la función imageellipse:

```
imageellipse(image, cx, cy, w, h, color)
```

Así trabaja. Esta función traza una elipse con centro en *cx*, *cy* en la imagen representada por *image*. Los valores *w* y *h* especifican ancho y alto de la elipse, respectivamente. El color de la elipse lo define *color*. Esta función es bastante similar a imagerectangle, pero en lugar de especificar las esquinas superior izquierda e inferior derecha, se define el centro de la elipse, además de su ancho y alto.

Este ejemplo, phpellipse.php, traza elipses:

```
<?php

    $image_height = 100;
    $image_width = 300;

    $image = imagecreate($image_width, $image_height);

    $back_color = imagecolorallocate($image, 200, 200, 200);

    $draw_color = imagecolorallocate($image, 0, 0, 0);

    imageellipse($image, 100, 40, 150, 50, $draw_color);

    imageellipse($image, 150, 50, 150, 50, $draw_color);

    imageellipse($image, 200, 60, 150, 50, $draw_color);

    header('Content-Type: image/jpeg');
```

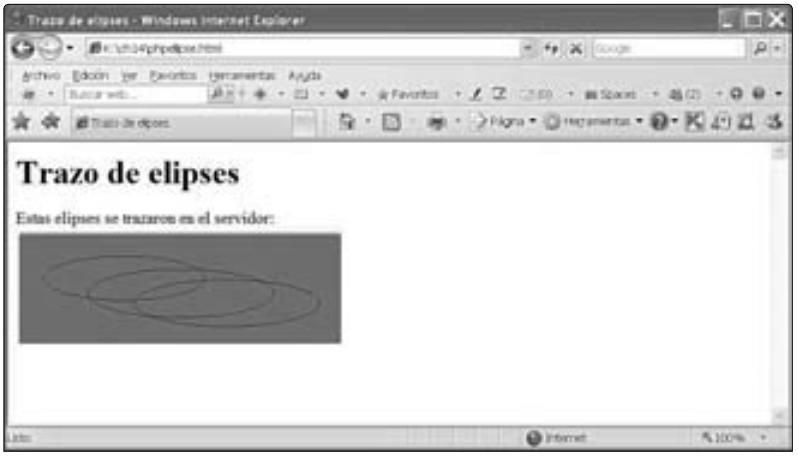

FIGURA 14-6 Cómo mostrar elipses en una página Web

```
    imagejpeg($image);
    imagedestroy($image);
?>
```

Y así se incorporan estas elipses en una página Web, phpellipse.html:

```
<html>
  <head>
    <title>
      Trazo de elipses
    </title>
  </head>

  <body>
    <h1>
      Trazo de elipses
    </h1>
      Estas elipses se trazaron en el servidor:
      <br>
      <img src="phpellipse.php">
  </body>
</html>
```

En los resultados de la Figura 14-6, aparecen las elipses.

Trazo de arcos

¿Qué tal si trazamos arcos? La función imagearc traza arcos, que incluyen circunferencias y elipses parciales, además de circunferencias y elipses completas:

```
imagearc(image, cx, cy, w, h, s, e, color)
```

Esta función sirve para trazar arcos con centro en *cx*, *cy* en la imagen representada por *image*. Los valores *w* y *h* especifican ancho y alto de la elipse, respectivamente; mientras los puntos inicial y final se especifican en grados, indicados por los argumentos *s* y *e* (aquí, 0º se localiza en la posición de las 3 en punto). El arco mismo se traza en el sentido del reloj.

Así se puede trazar una cara sonriente usando arcos:

```
imagearc($image, 150, 50, 50, 50, 30, 150, $drawing_color);
imagearc($image, 150, 50, 70, 70, 0, 360, $drawing_color);
imagearc($image, 135, 45, 20, 20, 190, -10, $drawing_color);
imagearc($image, 165, 45, 20, 20, 190, -10, $drawing_color);
imagearc($image, 135, 42, 10, 10, -10, 190, $drawing_color);
imagearc($image, 165, 42, 10, 10, -10, 190, $drawing_color);
```

Así como se ve en phparc.php:

```
<?php
    $image_height = 100;
    $image_width = 300;
    $image = imagecreate($image_width, $image_height);
    $backcolor = imagecolorallocate($image, 200, 200, 200);
    $drawing_color = imagecolorallocate($image, 0, 0, 0);
    imagearc($image, 150, 50, 50, 50, 30, 150, $drawing_color);
    imagearc($image, 150, 50, 70, 70, 0, 360, $drawing_color);
    imagearc($image, 135, 45, 20, 20, 190, -10, $drawing_color);
    imagearc($image, 165, 45, 20, 20, 190, -10, $drawing_color);
    imagearc($image, 135, 42, 10, 10, -10, 190, $drawing_color);
    imagearc($image, 165, 42, 10, 10, -10, 190, $drawing_color);
    header("Content-type: image/jpeg");
    imagejpeg($image);
    imagedestroy($image);
?>
```

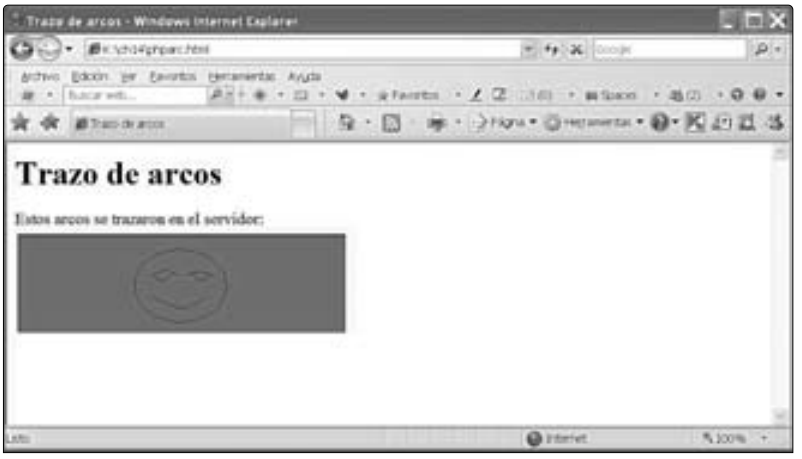

FIGURA 14-7 Cómo mostrar arcos en una página Web

En la página HTML phparc.html se muestran estos arcos:

```
<html>
  <head>
    <title>
      Trazo de arcos
    </title>
  </head>

  <body>
    <h1>
      Trazo de arcos
    </h1>
      Estos arcos se trazaron en el servidor:
      <br>
      <img src="phparc.php">
  </body>
</html>
```

La cara sonriente aparece en la Figura 14-7. Muy bonito.

Trazo de polígonos

Si desea trazar sus propias figuras, puede entrelazarlas con líneas múltiples, pero hay una forma más fácil; use la función imagepolygon para trazar un polígono con sólo pasarle una matriz de puntos. Así se utiliza esta función en general:

```
imagepolygon(image, points, num_points, color)
```

Esta función crea un polígono en una imagen. El parámetro *points* es una matriz conteniendo los vértices del polígono (*points*[0] = x0, *points*[1] = y0, *points*[2] = x1, *points*[3] = y1, etc.). El parámetro *num_points* aloja el número total de puntos en el polígono y *color* es el color para dibujar que desea usar.

Ésta es una buena función para trazar figuras complejas (todo lo que debe hacer es proporcionar los vértices del polígono en una matriz, como una matriz llamada $points):

```
$points = array(
        0  => 120, 1  => 60,
        2  => 130, 3  => 60,
        4  => 150, 5  => 80,
        6  => 170, 7  => 40,
        8  => 150, 9  => 40,
        10 => 110, 11 => 20,
        12 => 110, 13 => 90
);
```

Así se ve esa matriz en phppolygon.php, usando imagepolygon:

```
<?php
    $points = array(
        0  => 120, 1  => 60,
        2  => 130, 3  => 60,
        4  => 150, 5  => 80,
        6  => 170, 7  => 40,
        8  => 150, 9  => 40,
        10 => 110, 11 => 20,
        12 => 110, 13 => 90
    );

    $image_height = 100;
    $image_width = 300;

    $image = imagecreate($image_width, $image_height);
    $back_color = imagecolorallocate($image, 200, 200, 200);

    $draw_color = imagecolorallocate($image, 0, 0, 0);

    imagepolygon($image, $points, 7, $draw_color );

    header('Content-type: image/jpeg');

    imagejpeg($image);

    imagedestroy($image);
?>
```

Así se ve la página HTML phppolygon.html, mostrando este polígono:

```
<html>
  <head>
    <title>
```

FIGURA 14-8 Cómo mostrar un polígono en una página Web

```
      Trazo de polígonos
    </title>
  </head>

  <body>
    <h1>
      Trazo de polígonos
    </h1>
      Este polígono se trazó en el servidor:
      <br>
      <img src="phppolygon.php">
  </body>
</html>
```

El polígono trazado en este ejemplo se aprecia en la Figura 14-8.

Relleno de figuras

Además de simplemente dibujar el contorno de figuras, también puede rellenarlas con color usando diversas funciones que trazan figuras rellenas, como éstas:

- **imagefilledarc** Traza una elipse parcial y la rellena
- **imagefilledellipse** Traza una elipse rellena
- **imagefilledpolygon** Traza un polígono relleno
- **imagefilledrectangle** Traza un rectángulo relleno

Por ejemplo, dé un vistazo a imagefilledrectangle:

```
imagefilledrectangle(image, x1, y1, x2, y2, color)
```

Esta función crea un rectángulo relleno del color *color*, en la imagen *image*, a partir de la coordenada superior izquierda *x1*, *y1* y terminando en la coordenada inferior derecha *x2*, *y2*.

Este ejemplo pone a trabajar imagefilledrectangle, que modifica nuestro ejemplo anterior phprectangle.php en phpfilledrectangle.php. La diferencia radica en que esta versión rellena su rectángulo con color (rojo en este caso). Así se ve el código:

```
<?php

    $image_height = 100;
    $image_width = 300;

    $image = imagecreate($image_width, $image_height);

    $back_color = imagecolorallocate($image, 200, 200, 200);

    $draw_color = imagecolorallocate($image, 255, 0, 0);

    imagefilledrectangle($image, 30, 20, 50, 90, $draw_color);

    imagefilledrectangle($image, 70, 20, 140, 90, $draw_color);

    imagefilledrectangle($image, 170, 20, 270, 90, $draw_color);

    header('Content-Type: image/jpeg');

    imagejpeg($image);

    imagedestroy($image);
?>
```

Ésta es la página HTML, phpfilledrectangle.html, mostrando dichos rectángulos rellenos:

```
<html>
  <head>
    <title>
      Trazo de rectángulos rellenos
    </title>
  </head>

  <body>
    <h1>
      Trazo de rectángulos rellenos
    </h1>
      Estos rectángulos rellenos se trazaron en el servidor:
      <br>
      <img src="phpfilledrectangle.php">
  </body>
</html>
```

Los rectángulos rellenos trazados para este ejemplo aparecen en la Figura 14-9. Aquí están en blanco y negro, pero en la pantalla son rojos. Formidable.

FIGURA 14-9 Cómo mostrar rectángulos rellenos en una página Web

Trazo de píxeles individuales

¿Desea mayor poder gráfico? Puede establecer píxeles individuales usando imagesetpixel:

```
imagesetpixel(image, x, y, color)
```

Como sería de esperar, esta función traza un píxel en *x, y* en la imagen *image*, de color *color*.

Este ejemplo, phppixel.php, traza una línea punteada usando imagesetpixel. Para delinearla, el ejemplo simplemente usa un ciclo for:

```
for($loop_index = 50; $loop_index < 270; $loop_index += 3){

    imagesetpixel($image, $loop_index, $loop_index / 3, $drawing_color);
}
```

Así se ve este ciclo for en phppixel.php:

```
<?php
    $image_height = 100;
    $image_width = 300;

    $image = imagecreate($image_width, $image_height);

    $back_color = imagecolorallocate($image, 200, 200, 200);

    $drawing_color = imagecolorallocate($image, 0, 0, 0);

    for($loop_index = 50; $loop_index < 270; $loop_index += 3){

        imagesetpixel($image, $loop_index, $loop_index / 3, $drawing_color);

    }
```

```
    header("Content-type: image/jpeg");
    imagejpeg($image);

    imagedestroy($image);

?>
```

Y ésta es la página HTML, phppixel.html, mostrando estos píxeles:

```
<html>
  <head>
    <title>
      Trazo de píxeles
    </title>
  </head>

  <body>
    <h1>
      Trazo de píxeles
    </h1>
      Estos píxeles se trazaron en el servidor:
      <br>
      <img src="phppixel.php">
  </body>
</html>
```

Los píxeles que traza este ejemplo se muestran en la Figura 14-10.

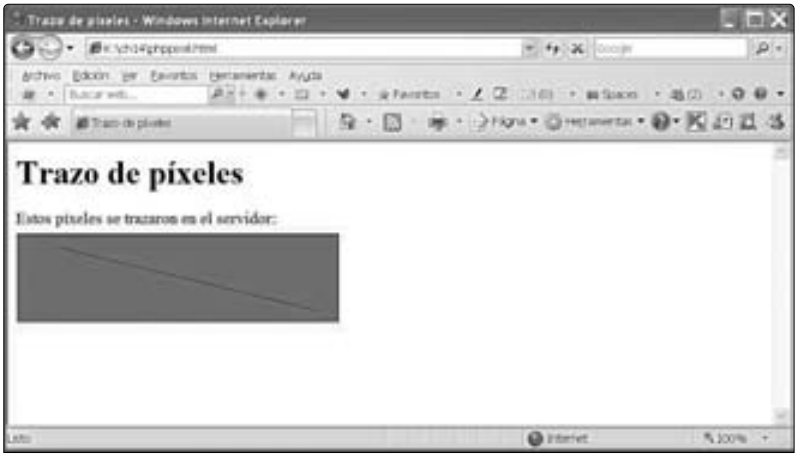

FIGURA 14-10 Cómo mostrar píxeles en una página Web

Trazo de texto

¿Qué tal si ahora trazamos texto? Existen varias funciones para trazar texto, como imagestring:

```
imagestring(image, font, x, y, s, color)
```

Esta función traza la cadena *s* especificada por la imagen *image*, con la esquina superior izquierda en las coordenadas *x*, *y* en el color *color*. El paquete gráfico incluye fuentes integradas (si *font* es 1, 2, 3, 4 o 5, se usa una fuente integrada). También puede registrar sus propias fuentes con el paquete GD2, mediante la función imageloadfont. Observe que como trabajamos con gráficos, el texto se traza como imagen, no como texto editable, mismo que aparecería en un campo de texto.

Suponga que quiere mostrar un texto centrado en una imagen, en el ejemplo phptext.php. Usaría el número de fuente 4, mostrando el texto"Sin problemas.":

```
<?php
    $font_number = 4;
    $text = "Sin problemas.";
        .
        .
        .
?>
```

¿Cómo ajustar el tamaño de la imagen para centrar este texto? Puede hacer la imagen dos veces más ancha que el texto, con la función imagefontwidth, para determinar el ancho de cada carácter de su fuente y multiplicando por 2 * strlen($text), para obtener el ancho que debe tener la imagen:

```
<?php
    $font_number = 4;
    $text = "Sin problemas.";
    $width = 2 * strlen($text) * imagefontwidth($font_number);
        .
        .
        .
?>
```

Asimismo, hacer la imagen tres veces más alta que el texto —con la función imagefontheight):

```
<?php
    $font_number = 4;
    $text = "Sin problemas.";
```

```
    $width = 2 * strlen($text) * imagefontwidth($font_number);
    $height = 3 * imagefontheight($font_number);
        .
        .
        .
?>
```

Después, crear la imagen, así como colores de fondo y trazo:

```
<?php
    $font_number = 4;
    $text = "Sin problemas.";
    $width = 2 * strlen($text) * imagefontwidth($font_number);
    $height = 3 * imagefontheight($font_number);
    $image = imagecreate($width, $height);
    $back_color = imagecolorallocate($image, 200, 200, 200);
    $drawing_color = imagecolorallocate($image, 0,   0,   0);
        .
        .
        .
?>
```

Luego, determinar las posiciones *x* y *y* en que iniciar el texto de modo que aparezca centrado en la imagen:

```
<?php
    $font_number = 4;
    $text = "Sin problemas.";
    $width = 2 * strlen($text) * imagefontwidth($font_number);
    $height = 3 * imagefontheight($font_number);
    $image = imagecreate($width, $height);
    $back_color = imagecolorallocate($image, 200, 200, 200);
    $drawing_color = imagecolorallocate($image, 0,   0,   0);
    $x_position = ($width - (strlen($text) * imagefontwidth($font_number)))/ 2;
    $y_position = ($height - imagefontheight($font_number)) / 2;
```

```
        .
        .
        .
?>
```

Tras ello, trazar el texto, de la siguiente forma en phptext.php:

```
<?php

    $font_number = 5;

    $text = "Sin problemas.";

    $width = 2 * strlen($text) * imagefontwidth($font_number);

    $height = 3 * imagefontheight($font_number);

    $image = imagecreate($width, $height);

    $back_color = imagecolorallocate($image, 200, 200, 200);

    $drawing_color = imagecolorallocate($image, 0,   0,   0);

    $x_position = ($width - (strlen($text) * imagefontwidth($font_number)))/ 2;

    $y_position = ($height - imagefontheight($font_number)) / 2;

    imagestring($image, $font_number, $x_position, $y_position, $text,
        $drawing_color);

    header('Content-Type: image/jpeg');

    imagejpeg($image);

    imagedestroy($image);
?>
```

Y ésta es la página HTML, phptext.html, mostrando el texto nuevo:

```
<html>
  <head>
    <title>
      Trazo de texto
    </title>
  </head>

  <body>
    <h1>
      Trazo de texto
    </h1>
      Este texto se trazó en el servidor:
```

FIGURA 14-11 Cómo mostrar texto centrado en una página Web

```
      <br>
      <img src="phptext.php">
  </body>
</html>
```

El texto centrado, que se trazó en este ejemplo, aparece en la Figura 14-11.

Trazo de texto vertical

Puede trazar texto con imagestring horizontalmente, de cualquier forma). ¿Qué le parece si trazamos texto verticalmente, como cuando desea rotular el eje "y" de una gráfica? Puede usar la función imagestringup:

```
imagestringup(image, font, x, y, s, color)
```

Esta función traza la cadena *s* verticalmente en la imagen especificada por *image*, en las coordenadas *x*, *y* en el color *color*. Si *font* es 1, 2, 3, 4 o 5, se usa una fuente integrada. También puede registrar sus propias fuentes con el paquete GD2 mediante la función imageloadfont.

Este ejemplo, phpverticaltext.php, traza texto vertical. Comienza seleccionando un número de fuente y determinando el texto que se mostrará:

```
<?php

    $font_number = 4;
    $text = "Sin problemas.";
        .
        .
        .
?>
```

Luego debe determinar las dimensiones de la imagen en que mostrará el texto. Por ejemplo, hacer que el ancho de la imagen sea tres veces mayor que el alto de la fuente seleccionada y que el alto de la imagen sea dos veces mayor respecto al ancho del texto a mostrar, además de seleccionar un color de dibujo rojo:

```
<?php

    $font_number = 4;
    $text = "Sin problemas.";

    $width = 3 * imagefontheight($font_number);

    $height = 2 * strlen($text) * imagefontwidth($font_number);

    $image = imagecreate($width, $height);

    $back_color = imagecolorallocate($image, 200, 200, 200);

    $drawing_color = imagecolorallocate($image, 255,   0,   0);
        .
        .
        .
?>
```

Eso crea la imagen y elige el color de dibujo. Puede crear la imagen con un poco de aritmética y trazar el texto usando la función imagestringup de esta forma:

```
<?php

    $font_number = 4;
    $text = "Sin problemas.";

    $width = 3 * imagefontheight($font_number);

    $height = 2 * strlen($text) * imagefontwidth($font_number);

    $image = imagecreate($width, $height);

    $back_color = imagecolorallocate($image, 200, 200, 200);

    $drawing_color = imagecolorallocate($image, 255,   0,   0);

    $x_position = ($width - imagefontheight($font_number)) / 2;

    $y_position = ($height + (strlen($text) * imagefontwidth($font_number)))/2;

    imagestringup($image, $font_number, $x_position, $y_position, $text,
        $drawing_color);
        .
        .
        .
?>
```

Y enviar la nueva imagen al navegador como en phpverticaltext.php:

```
<?php

    $font_number = 4;
    $text = "Sin problemas.";

    $width = 3 * imagefontheight($font_number);

    $height = 2 * strlen($text) * imagefontwidth($font_number);

    $image = imagecreate($width, $height);

    $back_color = imagecolorallocate($image, 200, 200, 200);

    $drawing_color = imagecolorallocate($image, 255,   0,   0);

    $x_position = ($width - imagefontheight($font_number)) / 2;

    $y_position = ($height + (strlen($text) * imagefontwidth($font_number)))/2;

    imagestringup($image, $font_number, $x_position, $y_position, $text,
        $drawing_color);

    header('Content-Type: image/jpeg');

    imagejpeg($image);

    imagedestroy($image);
?>
```

Eso completa phpverticaltext.php; ésta es la página HTML, phpverticaltext.html, presentando texto:

```
<html>
  <head>
    <title>
      Trazo de texto vertical
    </title>
  </head>

  <body>
    <h1>
      Trazo de texto vertical
    </h1>
      Este texto se trazó en el servidor:
      <br>
      <img src="phpverticaltext.php">
  </body>
</html>
```

El texto vertical trazado en este ejemplo aparece en la Figura 14-12. Genial.

FIGURA 14-12 Cómo mostrar texto vertical en una página Web

Trabajo con archivos de imágenes

Puede crear objetos gráficos de imágenes a partir de archivos de imágenes usando estas funciones:

- **imagecreatefromgif** Crea una nueva imagen a partir de un archivo GIF o una URL
- **imagecreatefromjpeg** Crea una nueva imagen a partir de un archivo JPEG o una URL
- **imagecreatefrompng** Crea una nueva imagen a partir de un archivo PNG o una URL
- **imagecreatefromwbmp** Crea una nueva imagen a partir de un archivo WBMP o una URL
- **imagecreatefromxbm** Crea una nueva imagen a partir de un archivo XBM o una URL
- **imagecreatefromxpm** Crea una nueva imagen a partir de un archivo XPM o una URL

Es sensacional cuando desea incorporar imágenes en páginas Web, pero también desea agregarles algo "de su cosecha", como un aviso de derechos de autor. Puede agregar una imagen de su creación, o un logotipo, dentro de otra imagen. Como ejemplo, usaremos la función imagecreatefromjpeg, cargando una imagen JPEG existente, image.jpg, a la que se añadirá una cara sonriente y un borde de imagen. Así se usa esta función:

```
imagecreatefromjpeg (string filename)
```

Ésta devuelve un identificador de imagen representando la obtenida del nombre de archivo dado, que devuelve una cadena vacía si falla.

Vamos a modificar la imagen JPEG en la Figura 14-3, image.jpg, agregando la cara sonriente que creó antes en el capítulo, además de un borde alrededor de la imagen completa.

FIGURA 14-13 Imagen JPEG que se modificará

El archivo PHP, phpjpg.php, comienza como sería de esperar, creando una imagen en la memoria a partir del archivo image.jpg:

```
<?php
    $image = imagecreatefromjpeg ("image.jpg");
        .
        .
        .
?>
```

Eso carga la imagen, image.jpg, en el objeto $image (en la práctica, no olvide agregar código de manejo de errores, en caso de que PHP no encuentre su archivo de imagen). Puede trazar un borde dentro de esta imagen (pero sería de ayuda si conociera sus dimensiones).

Puede hallarlas con las funciones imagesx e imagesy, que devuelven el tamaño *x* y *y* de una imagen.Y usar la función imagerectangle para crear el borde; así se traza ese borde, 10 píxeles dentro de la imagen:

```
<?php
    $image = imagecreatefromjpeg ("image.jpg");
    $back_color = imagecolorallocate($image, 200, 200, 200);
    $draw_color = imagecolorallocate($image, 0, 0, 0);
    imagerectangle($image, 10, 10, imagesx($image) - 10, imagesy($image) - 10,
        $draw_color);
        .
        .
        .
?>
```

Y se agrega la cara sonriente a la imagen, también de esta forma en phpjpg.php:

```
<?php
    $image = imagecreatefromjpeg ("image.jpg");
    $back_color = imagecolorallocate($image, 200, 200, 200);
```

```
    $draw_color = imagecolorallocate($image, 0, 0, 0);
    imagerectangle($image, 10, 10, imagesx($image) - 10, imagesy($image) - 10,
        $draw_color);
    imagearc($image, 50, 50, 50, 50, 30, 150, $drawing_color);
    imagearc($image, 50, 50, 70, 70, 0, 360, $drawing_color);
    imagearc($image, 35, 45, 20, 20, 190, -10, $drawing_color);
    imagearc($image, 65, 45, 20, 20, 190, -10, $drawing_color);
    imagearc($image, 35, 42, 10, 10, -10, 190, $drawing_color);
    imagearc($image, 65, 42, 10, 10, -10, 190, $drawing_color);
    header('Content-Type: image/jpeg');
    imagejpeg($image);
    imagedestroy($image);
?>
```

Ésta es la página HTML, phpjpg.html, mostrando la imagen modificada:

```
<html>
  <head>
    <title>
      Uso de imágenes
    </title>
  </head>
  <body>
    <h1>
      Uso de imágenes
    </h1>
      Esta imagen se modificó en el servidor:
      <br>
      <img src="phpjpg.php">
  </body>
</html>
```

Y el resultado se ve en la Figura 14-14, se modificó la imagen JPEG original, como usted quería.

FIGURA 14-14 Cómo mostrar una imagen modificada en una página Web

Colocación de imágenes en mosaico

También puede usar una imagen para colocarla en mosaico en otra imagen, apareciendo repetidamente en el fondo, con la función imagesettile. Así se usa esta función:

```
imagesettile(image, tile)
```

Así se establece la imagen de mosaico que usarán todas las funciones para relleno de regiones (como imagefilledrectangle e imagefilledpolygon) cuando rellena con el color especial IMG_COLOR_TILED.

¿Qué tal un ejemplo? Este script, phptile.php, comienza creando una imagen a colocarse en mosaico. Usaremos imagecreatetruecolor para dar a la imagen creada toda la gama de colores posibles, en vez de imagecreate, que podría restringir inadvertidamente el número de colores disponibles en los mosaicos (cuando tenga dudas respecto a la restricción de colores, debe usar imagecreatetruecolor en lugar de la función más simple imagecreate):

```
<?php
    $image_width = 300;
    $image_height = 200;

    $image = imagecreatetruecolor($image_width, $image_height);
        .
        .
        .
?>
```

Ahora crearemos el mosaico, $tile, dentro de la imagen almacenada en $image. En este caso, colocaremos en mosaico la imagen de la cara sonriente, leyendo primero image.jpg con imagecreatefromjpeg:

```
<?php
    $image_width = 300;
    $image_height = 200;

    $image = imagecreatetruecolor($image_width, $image_height);

    $tile = imagecreatefromjpeg ('image.jpg');
        .
        .
        .
?>
```

Puede agregar la cara sonriente a este mosaico con imagearc:

```
<?php
    $image_width = 300;
    $image_height = 200;

    $image = imagecreatetruecolor($image_width, $image_height);

    $tile = imagecreatefromjpeg (‘image.jpg’);

    imagearc($tile, 50, 50, 50, 50, 30, 150, $drawing_color);

    imagearc($tile, 50, 50, 70, 70, 0, 360, $drawing_color);

    imagearc($tile, 35, 45, 20, 20, 190, -10, $drawing_color);

    imagearc($tile, 65, 45, 20, 20, 190, -10, $drawing_color);

    imagearc($tile, 35, 42, 10, 10, -10, 190, $drawing_color);

    imagearc($tile, 65, 42, 10, 10, -10, 190, $drawing_color);
        .
        .
        .
?>
```

Eso completa la creación de la imagen en mosaico, luego se establecería $tile como mosaico para $image con imagesettile:

```
<?php
    $image_width = 300;
    $image_height = 200;

    $image = imagecreatetruecolor($image_width, $image_height);

    $tile = imagecreatefromjpeg ('image.jpg');
```

```
        .
        .
        .
    imagearc($tile, 65, 42, 10, 10, -10, 190, $drawing_color);

    imagesettile($image, $tile);
        .
        .
        .
?>
```

Ahora puede trazar un rectángulo relleno, que usará este mosaico si establece el estilo de relleno a IMG_COLOR_TILED:

```
<?php
    $image_width = 300;
    $image_height = 200;

    $image = imagecreatetruecolor($image_width, $image_height);

    $tile = imagecreatefromjpeg ('image.jpg');
        .
        .
        .
    imagearc($tile, 65, 42, 10, 10, -10, 190, $drawing_color);

    imagesettile($image, $tile);

    imagefilledrectangle ($image, 0, 0, $image_width, $image_height,
        IMG_COLOR_TILED);
        .
        .
        .
?>
```

Todo lo que resta por hacer es mostrar la imagen y luego destruirla en phptile.php:

```
<?php
    $image_width = 300;
    $image_height = 200;

    $image = imagecreatetruecolor($image_width, $image_height);

    $tile = imagecreatefromjpeg ('image.jpg');
        .
        .
        .
    imagearc($tile, 65, 42, 10, 10, -10, 190, $drawing_color);

    imagesettile($image, $tile);

    imagefilledrectangle ($image, 0, 0, $image_width, $image_height,
        IMG_COLOR_TILED);
```

```
    Header("Content-type: image/jpeg");

    imagejpeg($image);

    imagedestroy ($image);
    imagedestroy ($tile);
?>
```

Y ésta es la página HTML, phptile.html, que muestra la imagen en mosaico:

```
<html>
  <head>
    <title>
      Colocación de imágenes en mosaico
    </title>
  </head>

  <body>
    <h1>
      Colocación de imágenes en mosaico
    </h1>
      Esta imagen se colocó en mosaico en el servidor:
      <br>
      <img src="phptile.php">
  </body>
</html>
```

Y puede ver el resultado en la Figura 14-15, donde aparece la imagen en mosaico.

FIGURA 14-15 Cómo mostrar una imagen en mosaico en una página Web

Copia de imágenes

Disponemos de más poder para manipular gráficos; por ejemplo, la función imagecopy le permite copiar toda, o parte de, una imagen:

```
imagecopy (dest_image, src_image, dest_x, dest_y, src_x, src_y, src_w, src_h)
```

Esta función copia una parte de *src_image* en *dest_image* comenzando en las coordenadas *x, y src_x, src_y* con un ancho de *src_w* y un alto de *src_h*. La porción definida se copiará en las coordenadas *x, y, dest_x* y *dest_y*.

Copiar imágenes le permite realizar toda clase de trucos (por ejemplo, éste es un script, phpflip.php, que voltea una imagen horizontalmente). Comienza leyendo image.jpg, agregándole una cara sonriente y luego creando una imagen en blanco, $image_new, del mismo tamaño:

```
<?php

    $image_original = imagecreatefromjpeg("image.jpg");

    imagearc($image_original, 50, 50, 50, 50, 30, 150, $drawing_color);

    imagearc($image_original, 50, 50, 70, 70, 0, 360, $drawing_color);

    imagearc($image_original, 35, 45, 20, 20, 190, -10, $drawing_color);

    imagearc($image_original, 65, 45, 20, 20, 190, -10, $drawing_color);

    imagearc($image_original, 35, 42, 10, 10, -10, 190, $drawing_color);

    imagearc($image_original, 65, 42, 10, 10, -10, 190, $drawing_color);

    $image_width = imagesx($image_original);

    $image_height = imagesy($image_original);

    $image_new = imagecreate($image_width, $image_height);
        .
        .
        .
?>
```

Es aquí donde puede voltear la imagen, lo cual hace el código encimando la imagen original y volteándola, píxel por píxel, en la imagen volteada:

```
    for ($col = 0 ; $col < $image_width ; $col++)
    {
        for ($row = 0 ; $row < $image_height ; $row++)
        {
            imagecopy($image_new, $image_original, $image_width - $col - 1,
                $row, $col, $row, 1, 1);
        }
    }
```

Ésta es la página HTML, phpflip.html, que muestra la imagen volteada:

```
<html>
  <head>
    <title>
      Voltear imágenes
    </title>
  </head>

  <body>
    <h1>
      Voltear imágenes
    </h1>
      Esta imagen se volteó en el servidor:
      <br>
      <img src="phpflip.php">
  </body>
</html>
```

Puede ver el resultado en la Figura 14-16, donde aparece la imagen volteada.

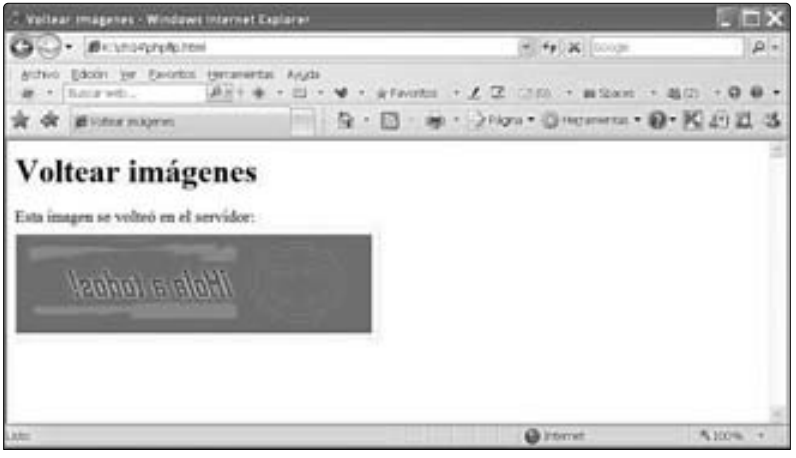

FIGURA 14-16 Cómo mostrar una imagen volteada en una página Web

CAPÍTULO 15

XML y RSS

En este capítulo daremos un vistazo a XML y RSS. Hay mucho soporte integrado para XML en PHP y aquí lo analizaremos.

Comenzaremos creando código XML en el servidor, enviándolo de regreso al navegador.

Creación de XML

¿Cómo se crea código XML en el servidor para enviarlo de vuelta al navegador? Como quizás recuerde, resolvimos este problema en el capítulo 12; creamos código XML y se envió al navegador. Daremos un vistazo aquí a este script, phpitems.php.

La clave al crear XML en PHP consiste en establecer el encabezado de tipo de contenido (Content-Type) a XML, como text/xml:

```
<?php
header("Content-type: text/xml");
        .
        .
        .
?>
```

En el presente ejemplo construye dos documentos XML, dependiendo de si el parámetro items pasado a éste contiene“1”o“2”. Esto se obtiene con“1”:

```
<?xml version="1.0" ?>
<items>
  <item>Libro de PHP</item>
  <item>Televisor</item>
  <item>Radio</item>
</items>
```

y esto con“2”:

```
<?xml version="1.0" ?>
<items>
  <item>Soda</item>
```

```
  <item>Queso</item>
  <item>Salami</item>
</items>
```

En phpitems.php, así se maneja el parámetro items (usando una de dos matrices de datos diferentes):

```
<?php
header("Content-type: text/xml");

if ($_REQUEST["items"] == "1")
  $items = array('Libro de PHP', 'Televisor', 'Radio');
if ($_REQUEST["items"] == "2")
  $items = array('Soda', 'Queso', 'Salami');
        .
        .
        .
?>
```

Ahora debe reproducir con echo la declaración XML del documento, <?xml version=”1.0”?>, requerida en todos los documentos XML (las versiones válidas, actualmente, son sólo 1.0 o 1.1):

```
<?php
header("Content-type: text/xml");

if ($_REQUEST["items"] == "1")
  $items = array(‘Libro de PHP’, 'Televisor', 'Radio');
if ($_REQUEST["items"] == "2")
  $items = array('Soda', 'Queso', 'alami');

echo '<?xml version="1.0" ?>';
        .
        .
        .
?>
```

El elemento documento, <items>, contiene todos los otros elementos, como es siempre el caso de los elementos documento, en documentos de XML:

```
<?php
header("Content-type: text/xml");

if ($_REQUEST["items"] == "1")
  $items = array('Libro de PHP', 'Televisor', 'Radio');
if ($_REQUEST["items"] == "2")
  $items = array('Soda', 'Queso', 'Salami');

echo '<?xml version="1.0" ?>';

echo '<items>';
```

```
        .
        .
        .
?>
```

Y está en libertad de reproducir los datos con echo en la matriz $items al navegador, en elementos <item> y luego finalizar el documento XML, cerrando el elemento documento con una etiqueta </items>:

```
<?php
header("Content-type: text/xml");

if ($_REQUEST["items"] == "1")
  $items = array('Libro de PHP', 'Televisor', 'Radio');
if ($_REQUEST["items"] == "2")
  $items = array('Soda', 'Queso', 'Salami');

echo '<?xml version="1.0" ?>';

echo '<items>';

foreach ($items as $value)
{
  echo '<item>';
  echo $value;
  echo '</item>';
}

echo '</items>';
?>
```

Puede observar este script pasándole un valor al parámetro items, como en esta URL: http://localhost/ch15/phpitems.php?items=1. Los resultados se observan en la Figura 15-1, donde se aprecia nuestro código XML.

Genial.

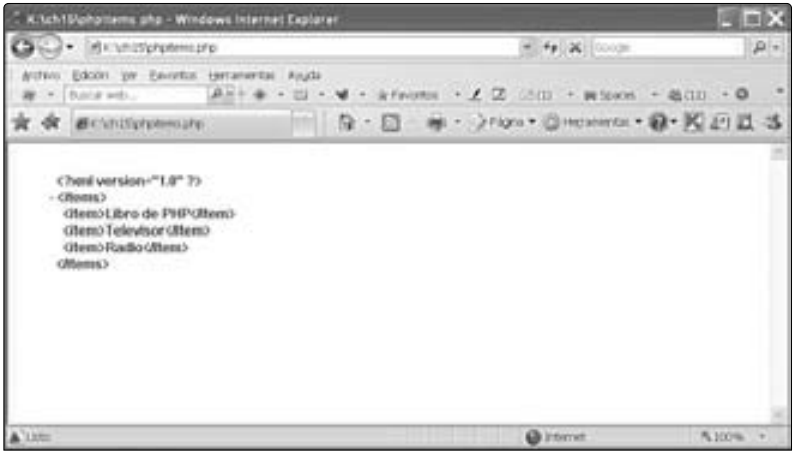

FIGURA 15-1 Creación de código XML en el servidor

Creación de RSS

Un dialecto XML de uso muy común en PHP es RSS (Really Simple Syndication), distribuye fuentes de noticias que la gente puede revisar a través de lectores RSS. Es algo que verá a menudo en PHP, pues usando PHP puede automatizar el proceso de creación RSS. Por ejemplo, redacte un script PHP para examinar su blog y crear una fuente RSS a partir de éste.

Este ejemplo crea una fuente RSS, phprss.php. Para simplificar las cosas, incorporaremos datos RSS en este script de ejemplo, pero por supuesto, podría examinar un blog o cualquier otro archivo para obtener artículos nuevos.

Este script debe proporcionar un título, URL y descripción para cada nuevo artículo; éste es el documento RSS que crearemos en el ejemplo, que se podrá leer con lectores RSS:

```
<?xml version="l.O" ?>
<rss version="2.0">
<channel>
  <title>Noticias muy importantes</title>
  <link>http://url</link>
  <description>Las noticias más importantes de todas partes</description>
  <language>en-us</language>
  <copyright>Copyright 2007</copyright>
  <webMaster>webmaster@url</webMaster>

  <item>
    <title>Nevadas en agosto</title>
    <link>http://url</link>
    <description>Nieva en agosto. ¿Sigue siendo éste el hemisferio norte?</
      description>
  </item>

  <item>
    <title>Los árboles son verdes</title>
    <link>http://url</link>
    <description>Científicos determinan que los árboles son verdes. Se sospecha que
      el cielo es azul.</description>
  </item>

  <item>
    <title>El mercado accionario a la alza y a la baja</title>
    <link>http://url</link>
    <description>Sí, el mercado accionario estuvo a la alza y a la baja hoy. Como
      siempre.</description>
  </item>
  </channel>
</rss>
```

Puede crear una matriz multidimensional, $items, alojando los datos en los elementos de noticias de phprss.php:

```
<?php
  $items[0] = array('Nevadas en agosto',
    'http://url',
    "Nieva en agosto. ¿Sigue siendo éste el hemisferio norte?"
  );

  $items[1] = array('Los árboles son verdes',
    'http://url',
    'Científicos determinan que los árboles son verdes. Se sospecha que el cielo es
      azul.'
  );

  $items[2] = array('El mercado accionario a la alza y a la baja',
    'http://url',
    'Sí, el mercado accionario estuvo a la alza y a la baja hoy. Como siempre.'
  );
        .
        .
        .
?>
```

También puede crear el documento con la función header para hacerlo documento XML, además de incluir declaraciones XML y RSS; en este caso, usaremos RSS versión 2.0, que es la versión actual:

```
<?php
  $items[0] = array('Nevadas en agosto',
    'http://url',
    "Nieva en agosto. ¿Sigue siendo éste el hemisferio norte?"
  );
        .
        .
        .
  header('Content-type: text/xml');

  echo '<?xml version="1.0" ?>';

  echo '<rss version="2.0">';
        .
        .
        .
?>
```

Y reproducir con echo los elementos <channel>, <title> y otros al inicio del documento:

```
<?php
  $items[0] = array('Nevadas en agosto',
    'http://url',
    "Nieva en agosto. ¿Sigue siendo éste el hemisferio norte?"
```

```
  );
        .
        .
        .
  header('Content-type: text/xml');

  echo '<?xml version="1.0" ?>';

  echo '<rss version="2.0">';

  echo '<channel>';
  echo '<title>Noticias muy importantes</title>';
  echo '<link>http://url</link>';
  echo '<description>Las noticias más importantes de todas partes</description>';
  echo '<language>en-us</language>';
  echo '<copyright>Copyright 2007</copyright>';
  echo '<webMaster>webmaster@url</webMaster>';
        .
        .
        .
?>
```

Por último, reproduzca con echo los datos reales de la fuente y termine el documento:

```
<?php
  $items[0] = array('Nevadas en agosto',
    'http://url',
    "Nieva en agosto. ¿Sigue siendo éste el hemisferio norte?"
  );

  $items[1] = array('Los árboles son verdes',
    'http://url',
    'Científicos determinan que los árboles son verdes. Se sospecha que el cielo es azul.'
  );
        .
        .
        .
echo '<webMaster>webmaster@url</webMaster>';

  foreach ($items as $data) {
    echo '<item>';
    echo "<title>{$data[0]}</title>";
    echo "<link>{$data[1]}</link>";
    echo "<description>{$data[2]}</description>";
    echo '</item>';

  }
```

```
  echo '</channel>';

  echo '</rss>';
?>
```

Vea el documento RSS creado por el ejemplo, phprss.php, en la Figura 15-2.

Y observe nuestra nueva fuente RSS, en un lector RSS, en la Figura 15-3.

Genial. Ahora puede crear fuentes RSS con código PHP para que los usuarios empleen lectores RSS.

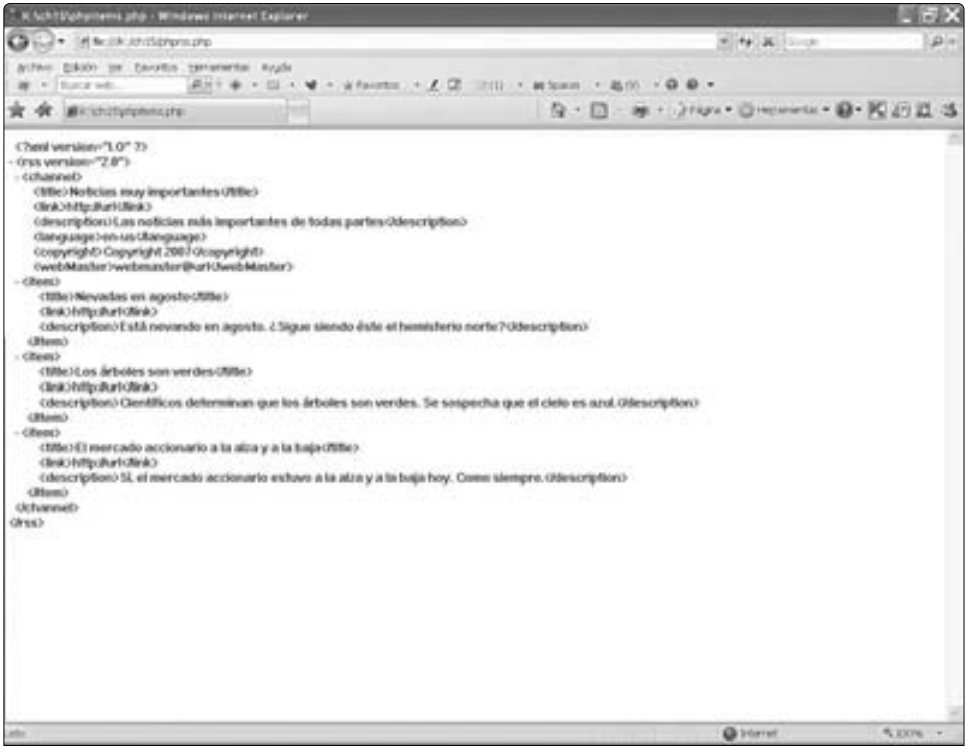

FIGURA 15-2 Creación de RSS en el servidor

FIGURA 15-3 Lectura de RSS en un lector RSS

Uso de las funciones SimpleXML

PHP 5 incluye una amplia biblioteca de funciones integradas para el manejo de código XML (se trata funciones SimpleXML, analizan XML, funciones DOM, funciones XMLReader, etc.). No tenemos espacio en este capítulo para analizar todo el soporte a XML en PHP, pero abarcaremos el de uso más popular, comenzando con SimpleXML. Aquí están ellas:

- **simplexml_import_dom** Devuelve un objeto SimpleXMLElement desde un nodo DOM
- **simplexml_load_file** Carga un archivo XML en un objeto
- **simplexml_load_string** Carga una cadena de XML en un objeto
- **SimpleXMLElement->_construct( )** Crea un nuevo objeto SimpleXMLElement
- **SimpleXMLElement->addAttribute( )** Agrega un atributo a un elemento SimpleXML
- **SimpleXMLElement->addChild( )** Agrega un elemento hijo al nodo XML
- **SimpleXMLElement->asXML( )** Devuelve una cadena XML basada en un elemento SimpleXML
- **SimpleXMLElement->attributes( )** Obtiene los elementos de un atributo
- **SimpleXMLElement->children( )** Obtiene los elementos hijos de un nodo dado

- **SimpleXMLElement->getDocNamespaces( )** Devuelve espacios en nombres declarados en el documento
- **SimpleXMLElement->getNamne( )** Obtiene el nombre de un elemento XML
- **SimpleXMLElement->getNamespaces( )** Devuelve los espacios en nombres utilizados en el documento
- **SimpleXMLElement->registerXPathNamespace( )** Crea un prefijo /ns para la siguiente consulta de XPath
- **SimpleXMLElement->->path( )** Ejecuta una consulta XPath en datos XML

Use estas funciones para analizar y editar documentos XML al instante. Éste es el documento XML en los siguientes ejemplos, event.xml. Presenta la lista de un evento, Premios Nacionales y sus asistentes:

```
<?xml version="1.0"?>
<events>
    <event type="fundraising">
       <event_title>Premios Nacionales</event_title>
       <event_number>3</event_number>
       <subject>Premios a Mascotas</subject>
       <date>5/5/2007</date>
       <people>
           <person attendance="present">
               <first_name>June</first_name>
               <last_name>Allyson</last_name>
           </person>
           <person attendance="absent">
               <first_name>Virgina</first_name>
               <last_name>Mayo</last_name>
           </person>
           <person attendance="present">
               <first_name>Jimmy</first_name>
               <last_name>Stewart</last_name>
           </person>
       </people>
   </event>
</events>
```

Cargue este documento XML en un objeto SimpleXML, mediante la función simplexml_load_file de esta forma:

```
<?php
  $xml = simplexml_load_file("event.xml");
         .
         .
         .
?>
```

Ahora el documento se carga en el objeto $xml. Aquí es donde la simplicidad de SimpleXML entra en acción. Cada elemento de event.xml (con excepción del elemento documento) se vuelve propiedad de ese objeto.

Por ejemplo, suponga que quiere extraer la fecha de event.xml:

```
<?xml version="1.0"?>
<events>
    <event type="fundraising">
       <event_title>Premios Nacionales</event_title>
       <event_number>3</event_number>
       <subject>Premios a Mascotas</subject>
       <date>5/5/2007</date>
         .
         .
         .
```

Para acceder al texto de esa fecha como $xml->event->date:

```
<?php
  $xml = simplexml_load_file("event.xml");

  echo "<h1>Extracción de datos simples de un documento XML</h1>";

  echo "La fecha del evento es " . $xml->event->date;
?>
```

Los resultados de este ejemplo aparecen en la Figura 15-4, la fecha se ha extraído correctamente de event.xml.

Las funciones SimpleXML pueden leer también datos XML de una cadena, si la pasa al constructor SimpleXMLElement. Por ejemplo, podría tener un script cargando el texto de event.xml en una cadena llamada $xmlstr:

```
<?php
$xmlstr = <<<XML
<?xml version="1.0"?>
<events>
    <event type="fundraising">
       <event_title>Premios Nacionales</event_title>
       <event_number>3</event_number>
       <subject>Premios a Mascotas</subject>
```

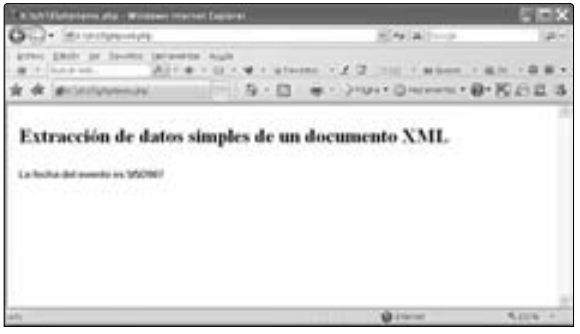

FIGURA 15-4 Extracción de datos de event.xml

```
        <date>5/5/2007</date>
        <people>
            <person attendance="present">
                <first_name>June</first_name>
                <last_name>Allyson</last_name>
            </person>
            <person attendance="absent">
                <first_name>Virgina</first_name>
                <last_name>Mayo</last_name>
            </person>
            <person attendance="present">
                <first_name>Jimmy</first_name>
                <last_name>Stewart</last_name>
            </person>
        </people>
    </event>
</events>
XML;
?>
```

Ahora puede incluir event.php en un nuevo script, phpxml2.php:

```
<?php
  include 'event.php';
        .
        .
        .
?>
```

Esto introduce $xmlstr en el código de phpxml2.php y pasa la cadena, incluido el texto XML que contiene, al constructor SimpleXMLElement:

```
<?php
  include 'event.php';

  $xml = new SimpleXMLElement($xmlstr);
        .
        .
        .
?>
```

Esto crea el mismo objeto en $xml que, si hubiera utilizado la función simplexml_load_file, pasa esa función a event.xml. Ahora puede extraer los datos del elemento <date> como antes:

```
<?php
  include 'event.php';

  $xml = new SimpleXMLElement($xmlstr);

  echo "<h1>Extracción de datos simples de una cadena XML</h1>";

  echo $xml->event->date;
?>
```

Este script, phpxml2.php, da los mismos resultados que phpxml.php.

¿Cómo maneja el caso de un documento con más de un mismo elemento? Por ejemplo, ¿qué sucedería si deseara extraer nombre y apellido de cada persona de estos tres elementos <person>?

```
<?xml version="1.0"?>
<events>
    <event type="fundraising">
       <event_title>Premios Nacionales</event_title>
       <event_number>3</event_number>
       <subject>Premios a Mascotas</subject>
       <date>5/5/2007</date>
       <people>
           <person attendance="present">
               <first_name>June</first_name>
               <last_name>Allyson</last_name>
           </person>
           <person attendance="absent">
               <first_name>Virgina</first_name>
               <last_name>Mayo</last_name>
           </person>
           <person attendance="present">
               <first_name>Jimmy</first_name>
               <last_name>Stewart</last_name>
           </person>
       </people>
   </event>
</events>
```

No puede usar simplemente la terminología $people->person —¿cómo sabría SimpleXML qué elemento <person> busca?— Resulta que SimpleXML maneja esto permitiéndole usar terminología de matrices, como la siguiente: $people->person[0],$people->person[1],$people->person[2], etc., que facilita las cosas.

También puede recorrer en ciclo múltiples elementos del mismo nombre utilizando foreach. En este ejemplo haremos exactamente eso para extraer nombres de todos los asistentes al evento en phpparse.php. Este script comienza cargando el archivo event.xml:

```
<?php
  $xml = simplexml_load_file("event.xml");
         .
         .
         .
?>
```

Puede obtener un objeto correspondiente al elemento <event>:

```
<?php
  $xml = simplexml_load_file("event.xml");

  echo "<h1>Análisis de datos de un documento XML</h1>";

  echo "Estas personas asistieron al evento: <br>";
```

```
  $event = $xml->event;
        .
        .
        .
?>
```

Obtendrá un objeto correspondiente al elemento <people> de esta forma:

```
<?php
  $xml = simplexml_load_file("event.xml");

  echo "<h1>Análisis de datos de un documento XML</h1>";

  echo "Estas personas asistieron al evento: <br>";

  $event = $xml->event;

  $people = $event->people;
        .
        .
        .
?>
```

Puede reproducir con echo las personas en una lista desordenada; de modo que rodee el código para presentar las personas con <ul> y :

```
<?php
  $xml = simplexml_load_file("event.xml");

  echo "<h1>Análisis de datos de un documento XML</h1>";

  echo "Estas personas asistieron al evento: <br>";

  $event = $xml->event;

  $people = $event->people;

  echo "<ul>";
        .
        .
        .
  echo "</ul>";
?>
```

Y recorrer todos los elementos <person> con un ciclo foreach, accediendo fácilmente el texto de los elementos <first_name> y <last_name> la persona de esta forma (es el código HTML para un espacio sin cambio de línea):

```
<?php
  $xml = simplexml_load_file("event.xml");

  echo "<h1>Análisis de datos de un documento XML</h1>";
```

FIGURA 15-5 Extracción de nombres de event.xml

```
  echo "Estas personas asistieron al evento: <br>";

  $event = $xml->event;

  $people = $event->people;

  echo "<ul>";

  foreach ($people->person as $person){
    echo "<li>", $person->first_name, ' ', $person->last_name , "</li>";

  }

  echo "</ul>";
?>
```

Y verá los resultados de este ejemplo en la Figura 15-5, se han extraído los nombres de las personas de event.xml.

Extracción de atributos

¿Que le parece si extraemos atributos de un documento con SimpleXML? Resulta fácil —sólo se emplea la notación de matriz.

Por ejemplo, ¿qué pasaría si extrajera el valor del atributo de asistencia de cada elemento <person>?

```
<?xml version="1.0"?>
<events>
    <event type="fundraising">
       <event_title>Premios Nacionales</event_title>
       <event_number>3</event_number>
       <subject>Premios a Mascotas</subject>
       <date>5/5/2007</date>
```

```
        <people>
            <person attendance="present">
                <first_name>June</first_name>
                <last_name>Allyson</last_name>
            </person>
            <person attendance="absent">
                <first_name>Virgina</first_name>
                <last_name>Mayo</last_name>
            </person>
            <person attendance="present">
                <first_name>Jimmy</first_name>
                <last_name>Stewart</last_name>
            </person>
        </people>
    </event>
</events>
```

Puede extraer el valor del atributo de asistencia de un elemento <person> de la siguiente forma: $person['attendance'], donde $person aloja el elemento <person>. Este ejemplo, phpattribute.php, extrae los atributos de asistencia y muestra. Como en el ejemplo anterior, phpparse.php, puede presentar cada persona (en este caso, la asistencia de cada persona) usando una lista HTML en desorden:

```
<?php
  $xml = simplexml_load_file("event.xml");
  echo "<h1>Análisis de atributos de un documento XML</h1>";
  echo "Estas personas asistieron al evento: <br>";
  $event = $xml->event;
  $people = $event->people;
  echo "<ul>";
        .
        .
        .
  echo "</ul>";
?>
```

Y, así, mostrar la asistencia de todas las personas:

```
<?php
  $xml = simplexml_load_file("event.xml");
  echo "<h1>Análisis de atributos de un documento XML</h1>";
  echo "Estas personas asistieron al evento: <br>";
  $event = $xml->event;
  $people = $event->people;
  echo "<ul>";
```

FIGURA 15-6 Extracción de nombres de event.xml

```
  foreach ($people->person as $person){
    echo "<li>", $person->first_name, ' ', $person->last_name, " was ",
     $person['attendance'], "</li>";
  }
  echo "</ul>";
?>
```

Vea los resultados en la Figura 15-6, se han extraído los atributos de asistencia de todas las personas de event.xml. Genial.

Uso de XPath

XPath es una especificación XML de un lenguaje especial para realizar búsquedas en documentos XML. Una exposición completa de XPath escapa al contenido de este libro. Brevemente, para usar XPath, considere un documento XML como árbol de nodos; éstos son los tipos de nodos legales:

- elemento
- atributo
- texto
- sección CDATA
- referencia de entidad
- entidad
- instrucción de procesamiento

Cuando ejecuta una expresión XPath en un nodo, este se conoce como nodo de contexto. Éstos son ejemplos de XPath (suponga que tiene este documento XML):

```
<?xml version="l.O" encoding ="UTF-8"?>
<states>
    <state>
        <name>California</name>
        <population units="people">33871648</population><!--2000 census-->
        <capital>Sacramento</capital>
        <bird>Quail</bird>
        <flower>Golden Poppy</flower>
        <area units="square miles">155959</area>
    </state>

    <state>
        <name>Massachusetts</name>
        <population units="people">6349097</population><!--2000 census-->
        <capital>Boston</capital>
        <bird>Chickadee</bird>
        <flower>Mayflower</flower>
        <area units="square miles">7840</area>
    </state>

    <state>
        <name>New York</name>
        <population units="people">18976457</population><!--2000 census-->
        <capital>Albany</capital>
        <bird>Bluebird</bird>
        <flower>Rose</flower>
        <area units="square miles">47214</area>
    </state>

</states>
```

Entonces, estas expresiones XPath producen los siguientes resultados:

- ***** Empata todos los hijos del elemento del nodo de contexto.
- ***/*/state** Empata todos los nietos <state> del nodo de contexto.
- **.** Empata el nodo de contexto.
- **..** Empata el padre del nodo de contexto.
- **../@units** Empata el atributo units del padre del nodo de contexto.
- **.//state** Empata todos los descendientes del elemento <state> del nodo de contexto.
- **//state** Empata todos los descendientes <state> del nodo raíz.
- **//state/name** Empata todos los elementos <name> con un padre <state>.
- **/states/state[4]/name[3]** Empata el tercer elemento <name> del cuarto elemento <state> del elemento <states>.

- **@*** Empata todos los atributos del nodo de contexto.
- **@units** Empata el atributo units del nodo de contexto.
- **state** Empata los hijos del elemento <state> del nodo de contexto.
- **state[@nickname and @units]** Empata todos los hijos <state> del nodo de contexto con atributo nickname y atributo units.
- **state[@units = "people"]** Empata todos los hijos <state> del nodo de contexto con atributo units con el valor"people".
- **state[7]** Empata el séptimo hijo <state> del nodo de contexto.
- **state[7][@units = "people"]** Empata el séptimo hijo <state> del nodo de contexto, si ese hijo tiene un atributo units con valor"people".
- **state[last()]** Empata el último hijo <state> del nodo de contexto.
- **state[name]** Empata los hijos <state> del nodo de contexto con hijos <name>.
- **state[name="Massachusetts"]** Empata los nodos del hijo <state> del nodo de contexto con| hijos <name>, cuyo valor de texto es"Massachusetts".
- **states//state** Empata todos los descendientes del elemento <state> de los hijos del elemento <state> del nodo de contexto.
- **text()** Empata todos los nodos de texto hijos del nodo de contexto.

Observemos esto en código. Si desea crear una matriz de todos los elementos <person> —sin importar dónde se encuentran en el documento event.xml—, puede usar la expresión XPath //person. Entonces, así se recorren todos los elementos <person> de event.xml —sin dirigirse a esos elementos <person>, pasando por elementos <event> y <people>:

```
<?php
  $xml = simplexml_load_file("event.xml");

  echo "<h1>Uso de XPath en un documento XML</h1>";

  echo "Estas personas asistieron al evento: <br>";

  echo "<ul>";

  foreach ($xml->xpath('//person') as $person){
        .
        .
        .
  }

  echo "</ul>";
?>
```

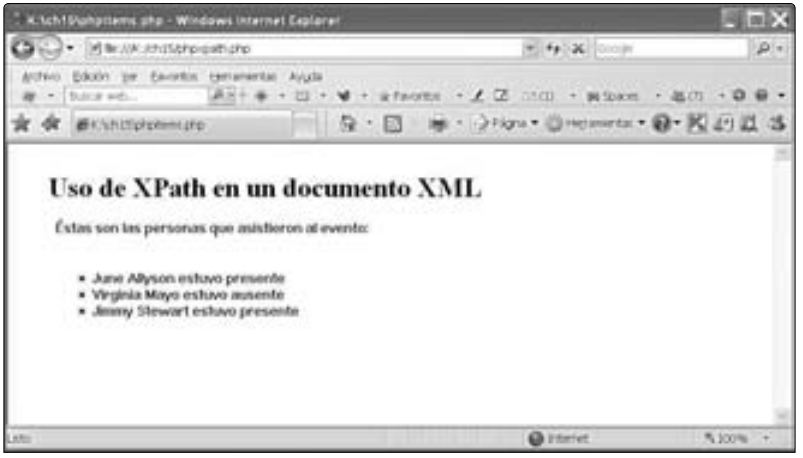

FIGURA 15-7 Uso de XPath en event.xml

Y lista cada persona y asistencia al evento:

```
<?php
  $xml = simplexml_load_file("event.xml");

  echo "<h1>Uso de XPath en un documento XML</h1>";

  echo "Éstas personas asistieron al evento: <br>";

  echo "<ul>";

  foreach ($xml->xpath('//person') as $person){
    echo "<li>", $person->first_name, ' ', $person->last_name, " estuvo ",
$person['attendance'], "</li>";
  }

  echo "</ul>";
?>
```

Vea los resultados en la Figura 15-7, la expresión XPath logró extraer correctamente todos los elementos <person>.

Modificación de elementos y atributos XML

Modifique también elementos y atributos usando SimpleXML —todo lo que debe hacer es asignar un nuevo valor al elemento o atributo—. Este ejemplo, phpmodify.php, asigna el nombre "Myrna Loy" a cada elemento <person> de event.xml:

```
<?php
  $xml = simplexml_load_file("event.xml");

  echo "<h1>Modificación de elementos en un documento XML</h1>";
```

```
  echo "Estas personas asistieron al evento: <br>";

  $event = $xml->event;

  $people = $event->people;

  foreach ($people->person as $person){
    $person->first_name = "Myrna";
    $person->last_name = "Loy";
  }
        .
        .
        .
?>
```

Entonces, el ejemplo presenta el texto en cada elemento <person> —que será Myrna Loy, tres veces:

```
<?php
  $xml = simplexml_load_file("event.xml");

  echo "<h1>Modificación de elementos en un documento XML</h1>";

  echo "Estas personas asistieron al evento: <br>";

  $event = $xml->event;

  $people = $event->people;

  foreach ($people->person as $person){
    $person->first_name = "Myrna";
    $person->last_name = "Loy";
  }

  echo "<ul>";

  foreach ($people->person as $person){
    echo "<li>", $person->first_name, ' ', $person->last_name, " estuvo ",
      $person['attendance'], "</li>";
  }

  echo "</ul>";
?>
```

Vea los resultados en la Figura 15-8, en realidad cambió todo elemento <person> para contener el nombre Myrna Loy.

Como puede apreciar, es posible que todos los elementos contengan Myrna Loy, pero no es la mejor manera de indicar que ella asistió al evento. ¿Qué le parece si agregamos un elemento Myrna Loy, en lugar de sobreescribir todos los demás?

FIGURA 15-8 Modificación de elementos en event.xml

Adición de nuevos elementos y atributos

Puede crear nuevos elementos con el paquete SimpleXML en PHP. Por ejemplo, agregue un elemento a event.xml para Myrna Loy:

```
<?xml version="1.0"?>
<events>
    <event type="fundraising">
       <event_title>Premios Nacionales</event_title>
       <event_number>3</event_number>
       <subject>Premios a Mascotas</subject>
       <date>5/5/2007</date>
       <people>
           <person attendance="present">
               <first_name>June</first_name>
               <last_name>Allyson</last_name>
           </person>
           <person attendance="absent">
               <first_name>Virgina</first_name>
               <last_name>Mayo</last_name>
           </person>
           <person attendance="present">
               <first_name>Jimmy</first_name>
               <last_name>Stewart</last_name>
           </person>
           <person attendance="present">
               <first_name>Myrna</first_name>
               <last_name>Loy</last_name>
```

```
            </person>
        </people>
    </event>
</events>
```

Este ejemplo, phpadd.php, inicia obteniendo un objeto correspondiente a elemento <people>:

```
<?php
  $xml = simplexml_load_file("event.xml");

  echo "<h1>Adición de elementos a un documento XML</h1>";

  echo "Estas personas asistieron al evento: <br>";

  $event = $xml->event;

  $people = $event->people;
        .
        .
        .
?>
```

Agregue un elemento <person>, como hijo del elemento <people> actual, con el método addChild:

```
<?php
  $xml = simplexml_load_file("event.xml");

  echo "<h1>Adición de elementos a un documento XML</h1>";
  echo "Estas personas asistieron al evento: <br>";

  $event = $xml->event;

  $people = $event->people;

  $person = $people->addChild('person');
        .
        .
        .
?>
```

Añada elementos hijo <first_name> y <last_name>, al nuevo elemento <person>, especificando los datos de texto para cada nuevo elemento hijo:

```
<?php
  $xml = simplexml_load_file("event.xml");

  echo "<h1>Adición de elementos a un documento XML</h1>";
```

```
  echo "Estas personas asistieron al evento: <br>";

  $event = $xml->event;

  $people = $event->people;

  $person = $people->addChild('person');

  $person->addChild('first_name', 'Myrna');
  $person->addChild('last_name', 'Loy');
        .
        .
        .
?>
```

Por último, agregue un nuevo atributo attendance al elemento <person> con el método addAttribute y luego muestre todos los elementos:

```
<?php
  $xml = simplexml_load_file("event.xml");

  echo "<h1>Adición de elementos a un documento XML</h1>";

  echo "Estas personas asistieron al evento: <br>";

  $event = $xml->event;

  $people = $event->people;

  $person = $people->addChild('person');

  $person->addChild('first_name', 'Myrna');
  $person->addChild('last_name', 'Loy');

  $person->AddAttribute('attendance', 'present');

  echo "<ul>";

  foreach ($people->person as $person){
    echo "<li>", $person->first_name, ' ', $person->last_name, " estuvo ",
      $person['attendance'], "</li>";
  }

  echo "</ul>";
?>
```

Vea los resultados en la Figura 15-9, se agregó el nuevo elemento <person> Myrna Loy.

FIGURA 15-9 Adición de elementos a event.xml

Envío de XML al navegador

¿Qué sucede si, tras hacer sus modificaciones a un documento XML, desea enviar el documento XML modificado a un navegador? Se requerirá trabajo para descifrar el documento XML completo a partir del objeto SimpleXML y enviarlo, elemento por elemento, atributo por atributo, de vuelta al navegador.

Por fortuna, existe una forma más fácil (usar el método asXML para reproducir todo el objeto SimpleXML como documento XML). Este ejemplo, phpasxml.php, lee event.xml, lo modifica y envía de vuelta al navegador mediante asXML:

```
<?php
  $xml = simplexml_load_file("event.xml");

  $event = $xml->event;

  $people = $event->people;

  $person = $people->addChild('person');

  $person->addChild('first_name', 'Myrna');

  $person->addChild('last_name', 'Loy');

  $person->AddAttribute('attendance', 'present');

  header("Content-type: text/xml");
  echo $xml->asXML();

?>
```

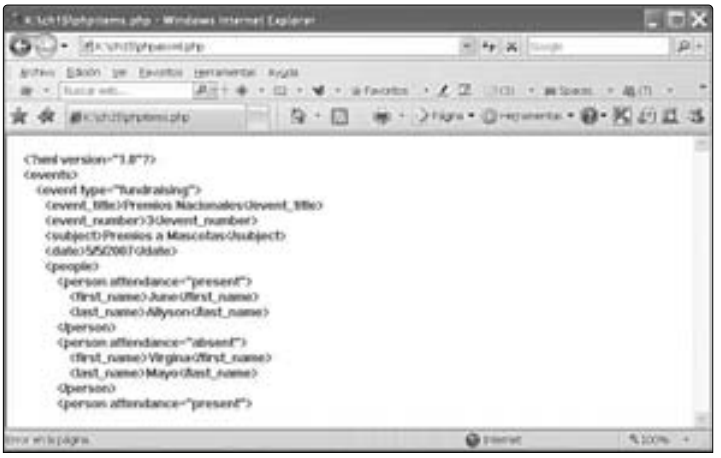

FIGURA 15-10 Envío de XML a navegadores

Vea los resultados en la Figura 15-10, este script realmente envió el documento XML completo al navegador. Muy útil.

Interacción con otros paquetes XML de PHP

SimpleXML es sólo uno de los paquetes XML con soporte en PHP. Puede conectar SimpleXML a algunos de estos otros paquetes, como DOM (Document Object Model). Este ejemplo, phpsimpledom.php, indica cómo importar un objeto DOM a SimpleXML mediante la función simplexml_import_dom.

Este ejemplo comienza creando un nuevo objeto de documento DOM XML, $dom:

```
<?php
  $dom = new domDocument;
         .
         .
         .
?>
```

Luego carga una cadena de texto XML en el objeto DOM, usando el método loadXML:

```
<?php
  $dom = new domDocument;
```

```
  $dom->loadXML (
    '<people><person><first_name>Myrna</first_name></person></people>');
        .
        .
        .
?>
```

Verifique si el objeto DOM fue creado correctamente:

```
<?php
  $dom = new domDocument;

  $dom->loadXML (
    '<people><person><first_name>Myrna</first_name></person></people>');

  if (!$dom) {
     echo 'Error';
     exit;
  }
        .
        .
        .
?>
```

Eso creará el objeto DOM, con sus propios métodos y propiedades. Convierta eso en un objeto SimpleXML con la función simplexml_import_dom y luego secargue el objeto como ML, de vuelta al navegador:

```
<?php
  $dom = new domDocument;

  $dom->loadXML(
    '<people><person><first_name>Myrna</first_name></person></people>');

  if (!$dom) {
     echo 'Error';
     exit;
  }

  $xml = simplexml_import_dom($dom);

  header("Content-type: text/xml");

  echo $xml->asXML();
?>
```

Vea los resultados en la Figura 15-11, este script convirtió un objeto DOM XML de PHP en SimpleXML.

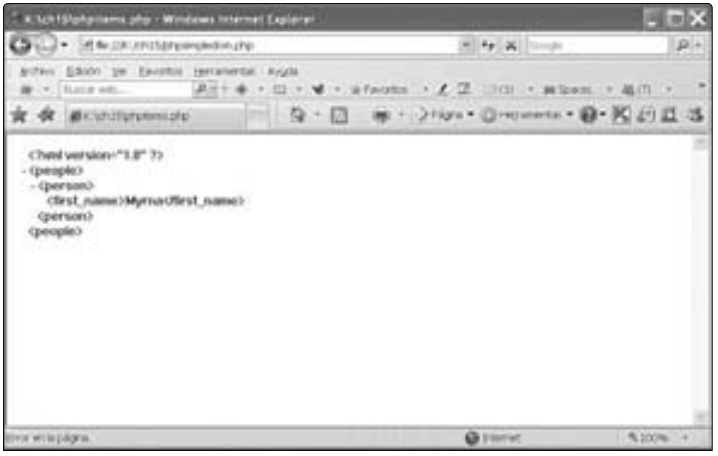

FIGURA 15-11 Conversión de un objeto DOM de PHP en objeto SimpleXML

Análisis de datos con las funciones analizadoras XML

Use las funciones analizadoras XML de PHP para revisar documentos XML; estas funciones actúan de forma muy similar a una función analizadora XML SAX (Simple API for XML), si está familiarizado con SAX. Éstas son las funciones analizadoras XML:

- **xml_error_string** Devuelve la cadena de error del analizador XML
- **xml_get_current_byte_index** Devuelve el índice de byte actual del analizador XML
- **xml_get_current_column_number** Devuelve el número de columna actual del analizador XML
- **xml_get_current_line_number** Devuelve el número de línea actual del analizador XML
- **xml_get_error_code** Devuelve el código de error del analizador XML
- **xml_parse_into_struct** Analiza los datos XML en una matriz
- **xml_parse** Inicia el análisis de un documento XML
- **xml_parser_create_ns** Crea un analizador XML con soporte para espacio en el nombre (namespace)
- **xml_parser_create** Crea un objeto analizador XML
- **xml_parser_free** Libera un analizador XML en la memoria
- **xml_parser_get_option** Devuelve opciones del analizador XML
- **xml_parser_set_option** Establece opciones en el analizador XML
- **xml_set_character_data_handler** Registra el manejador de datos de caracteres
- **xml_set_default_handler** Registra el manejador predeterminado
- **xml_set_element_handler** Registra los manejadores de elementos inicial y final

- **xml_set_end_namespace_decl_handler** Registra el manejador de declaraciones de espacio en nombre final
- **xml_set_external_entity_ref_handler** Registra el manejador de referencia de entidades externas
- **xml_set_notation_decl_handler** Registra el manejador de declaraciones de notación
- **xml_set_object** Usa el analizador XML en un objeto
- **xml_set_processing_instruction_handler** Registra el manejador de instrucciones de procesamiento (PI)
- **xml_set_start_namespace_decl_handler** Registra el manejador de declaraciones de espacio para nombres de inicio
- **xml_set_unparsed_entity_decl_handler** Registra el manejador de declaraciones de entidad no analizadas

Este ejemplo indica cómo echar a andar estas funciones, phpxmlparser.php. Analizará event.xml para mostrar cómo funcionan las cosas, pero puede usarlo para analizar cualquier documento XML y cuando sepa cómo emplearlo en documentos XML, entenderá cómo extraer los datos que quiera.

Al usar funciones analizadoras XML, los datos XML se revisan y pasan a una serie de funciones de llamada. Por ejemplo, cuando el analizador XML ve el inicio de un elemento XML, pasa el elemento a la función callback, creada para manejar inicios de elementos. Cuando el analizador encuentra texto, lo envía a la función de manejo de texto y así sucesivamente.

Este ejemplo, phpxmlparser.php, comienza almacenando el nombre del archivo que analizará en $file y usted modifica esta línea para analizar cualquier documento XML:

```
<?php
  $file = "event.xml";
        .
        .
        .
?>
```

Una cosa que no manejan las funciones analizadoras XML son declaraciones XML al principio de cada documento XML; de modo que imprimiremos una declaración XML estándar (la documentación de PHP respecto a funciones analizadoras XML, señala que puede utilizar un manejador predeterminado —pronto lo veremos— para leer declaraciones XML, pero de hecho no funciona):

```
<?php
  $file = "event.xml";

  echo "<h1>Análisis de XML con funciones analizadoras XML</h1>";
  echo "Éste es el archivo analizado: <br><br>";

  echo "&lt;?xml version='1.0'?&gt; <br>";
        .
        .
        .
?>
```

Después, se crea un objeto analizador XML, con la función xml_parser_create:

```
<?php
  $file = "event.xml";

  echo "<h1>Análisis de XML con funciones analizadoras XML</h1>";
  echo "Éste es el archivo analizado: <br><br>";

  echo "&lt;?xml version='1.0'?&gt; <br>";

  $xml_parser_object = xml_parser_create();
        .
        .
        .
?>
```

Luego debe registrar sus funciones de llamada con el objeto analizador XML. Estas funciones manejarán elementos, atributos, texto, etc., encoentrados por el analizador conforme lee el documento XML.

Por ejemplo, para establecer funciones que manejarán inicio y final de elementos, se emplea la función xml_set_element_handler. Se pasa el objeto analizador XML a esta función, además de la función callback para manejar el inicio de elementos, startElement, así como la función para manejar el final de elementos, endElement:

```
<?php
  $file = "event.xml";

  echo "<h1>Análisis de XML con funciones analizadoras XML</h1>";
  echo "Éste es el archivo analizado: <br><br>";

  echo "&lt;?xml version='1.0'?&gt; <br>";

  $xml_parser_object = xml_parser_create();

  xml_set_element_handler($xml_parser_object, "startElement", "endElement");
        .
        .
        .
?>
```

Para procesar los datos de texto contenidos en los elementos, se registra una función callback con el analizador XML, usando la función xml_set_character_data_handler y emplearemos una función llamada text.Y para manipular cualquier dato que el analizador XML encuentre y no haya establecido manejador, registre un manejador predeterminado con la función xml_set_default_handler. Éste será para nosotros una función llamada defaultHandler:

```
<?php
  $file = "event.xml";

  echo "<h1>Análisis de XML con funciones analizadoras XML</h1>";
  echo "Éste es el archivo analizado: <br><br>";

  echo "&lt;?xml version='1.0'?&gt; <br>";
```

```
  $xml_parser_object = xml_parser_create();
  xml_set_element_handler($xml_parser_object, "startElement", "endElement");
  xml_set_character_data_handler($xml_parser_object, "text");
  xml_set_default_handler($xml_parser_object, "defaultHandler");
        .
        .
        .
?>
```

Es su responsabilidad abrir el documento XML y enviarlo al analizador, de modo que inicia el proceso abriendo el documento XML y consiguiendo un manejador de archivos para él, $handle:

```
<?php
  $file = "event.xml";
        .
        .
        .
  xml_set_default_handler($xml_parser_object, "defaultHandler");
  if (!($handle = fopen($file, "r"))) {
    die("Error.");
  }
        .
        .
        .
?>
```

Ahora puede leer datos XML, 1024 bytes a la vez, con fread:

```
<?php
  $file = "event.xml";
        .
        .
        .
  xml_set_default_handler($xml_parser_object, "defaultHandler");
  if (!($handle = fopen($file, "r"))) {
    die("Error.");
  }
  while ($xml_data = fread($handle, 1024)) {
        .
        .
        .
    }
  }
        .
        .
        .
?>
```

Se pasan los datos XML a la función xml_parse del analizador XML, pasándole el objeto analizador XML, datos de texto que han de analizarse y un argumento TRUE/FALSE, indicando si ha pasado o no todos los datos XML a xml_parse. Determine si ha pasado todos los datos a xml_parse con feof; de ese modo, se pasan los datos XML al analizador:

```
<?php
  $file = "event.xml";
        .
        .
        .
  xml_set_default_handler($xml_parser_object, "defaultHandler");

  if (!($handle = fopen($file, "r"))) {
    die("Error.");
  }

  while ($xml_data = fread($handle, 1024)) {
    if (!xml_parse($xml_parser_object, $xml_data, feof($handle))) {
        .
        .
        .
    }
  }
?>
```

Si xml_parse devuelve un valor de FALSE, hubo un error. Para establecer cuál fue el error con la función xml_error_string, a la que se pasa el código de error (y obtendrá el código de error, enviando el objeto analizador XML a la función xml_get_error_code). También puede recuperar el número de línea del documento XML en que ocurrió el error con la función xml_get_current_line_number y mostrarlo también:

```
<?php
  $file = "event.xml";
        .
        .
        .
  xml_set_default_handler($xml_parser_object, "defaultHandler");

  if (!($handle = fopen($file, "r"))) {
    die("Error.");
  }

  while ($xml_data = fread($handle, 1024)) {
    if (!xml_parse($xml_parser_object, $xml_data, feof($handle))) {
      die(sprintf("Error %s on line %d",
        xml_error_string(xml_get_error_code($xml_parser_object)),
        xml_get_current_line_number($xml_parser_object)));
    }
  }
        .
        .
        .
?>
```

En este punto, el analizador invocará sus funciones de llamada. Todo lo que resta por hacer —además de escribir dichas funciones de llamada— es terminar con el analizador llamando a xml_parser_free y cerrar el documento XML con fclose:

```
<?php
  $file = "event.xml";
        .
        .
        .
  while ($xml_data = fread($handle, 1024)) {
    if (!xml_parse($xml_parser_object, $xml_data, feof($handle))) {
      die(sprintf("Error %s on line %d",
        xml_error_string(xml_get_error_code($xml_parser_object)),
        xml_get_current_line_number($xml_parser_object)));
    }
  }

  xml_parser_free($xml_parser_object);

  fclose($handle);
        .
        .
        .
?>
```

Comenzaremos con la función defaultHandler, que maneja cualquier dato XML sin soporte de otros manejadores que pudo crear. Hemos registrado una función manejadora predeterminada llamada defaultHandler con el analizador XML; a la función defaultHandler se le pasa el objeto analizador y texto XML no manejado; de modo que reproduciremos con echo ese texto en el navegador:

```
<?php
  $file = "event.xml";
        .
        .
        .
  function defaultHandler($parser, $text)
  {
    echo $text, "<br>";
  }
        .
        .
        .
?>
```

Eso controlará cualquier dato XML sin manipulación de otros manejadores. A continuación puede crear los manejadores startElement y endElement. El manejador startElement se invocará cuando el analizador encuentre la etiqueta de apertura de un elemento XML. Objeto analizador, nombre del elemento y combinación contenida por sus atributos se pasan a startElement.

Vamos a sangrar el texto XML analizado y controlaremos el nivel de sangría con una variable llamada $indent. Lo primero por hacer en startElement es reproducir con echo el espacio de sangría —dos espacios sin cambio de línea por cada nivel de sangría:

```
<?php
  $file = "event.xml";
  $indent = 0;
        .
        .
        .
  function startElement($parser, $name, $attrs)
  {
    global $indent;
    for ($i = 0; $i < $indent; $i++) {
        echo "    ";
    }
        .
        .
        .
  }
?>
```

El analizador XML convierte todos los nombres de elementos a mayúsculas, pero en realidad aparecen en minúsculas en event.xml; así que convertiremos el nombre de cada elemento de vuelta a minúsculas con strtolower:

```
<?php
  $file = "event.xml";
  $indent = 0;
        .
        .
        .
  function startElement($parser, $name, $attrs)
  {
    global $indent;
    for ($i = 0; $i < $indent; $i++) {
        echo "    ";
    }
    $lower = strtolower($name);
        .
        .
        .
  }
?>
```

Los atributos del elemento XML actual, si los tiene, se pasan a startElement, en una combinación que hemos llamado $attrs, donde las claves son nombres de atributos y los valores, valo-

res de los atributos. Puede recorrer esa lista combinada de la siguiente forma, ensamblando una cadena de texto alojando atributos y valores de esta forma:"attr1 = value1 attr2 = value2...":

```
<?php
  $file = "event.xml";
  $indent = 0;
        .
        .
        .
  function startElement($parser, $name, $attrs)
  {
    global $indent;
    for ($i = 0; $i < $indent; $i++) {
        echo "    ";
    }
    $lower = strtolower($name);
    $att_string = "";
    foreach ($attrs as $key => $value) {
      $att_string .= strtolower($key) . " = \"" . $value . "\"";
    }
        .
        .
        .
  }
?>
```

Todo lo que resta por hacer es reproducir con echo la etiqueta de apertura del elemento actual —completo con atributos y valores— en el navegador y luego incrementar el nivel de sangría de cualquier elemento anidado:

```
<?php
  $file = "event.xml";
  $indent = 0;
        .
        .
        .
  function startElement($parser, $name, $attrs)
  {
    global $indent;
    for ($i = 0; $i < $indent; $i++) {
        echo "    ";
    }
    $lower = strtolower($name);
    $att_string = "";
    foreach ($attrs as $key => $value) {
      $att_string .= strtolower($key) . " = \"" . $value . "\"";
    }
    echo "&lt;${lower} ${att_string}&gt; <br>";
    $indent++;
  }
?>
```

Eso maneja la etiqueta de inicio de los elementos —¿qué hay de la etiqueta final?— Ésa la maneja una función que hemos denominado endElement, a la que se llama cuando el analizador encuentra la etiqueta final de un elemento. Primero mostramos la cadena de sangría en endElement:

```
<?php
  function endElement($parser, $name)
  {
    global $indent;
    $indent--;
    for ($i = 0; $i < $indent; $i++) {
        echo "    ";
    }
        .
        .
        .
  }
?>
```

Luego, mostramos la etiqueta de cierre del elemento, después de reconvertir primero el nombre del elemento a minúsculas:

```
<?php
  function endElement($parser, $name)
  {
    global $indent;
    $indent--;
    for ($i = 0; $i < $indent; $i++) {
        echo "    ";
    }
    $lower = strtolower($name);
    echo "&lt;/${lower}&gt; <br>";
  }
?>
```

Bueno, hemos manejado etiquetas de apertura y cierre en elementos XML. La tarea final en este ejemplo consiste en manejar el texto contenido en todos y cada uno de los elementos, que se hace en la función llamada text. A ésta se le pasa el objeto analizador y texto encontrado por el analizador:

```
<?php
  function text($parser, $text)
  {
        .
        .
        .
?>
```

Al método de texto se le pasa todo el texto que no sea de marcado en el documento XML —incluido el espacio en blanco empleado como sangría en ese documento—. Por ejemplo, eche un vistazo a event.xml:

```
<?xml version="1.0"?>
<events>
    <event type="fundraising">
       <event_title>Premios Nacionales</event_title>
       <event_number>3</event_number>
       <subject>Premios a Mascotas</subject>
       <date>5/5/2007</date>
       <people>
           <person attendance="present">
               <first_name>June</first_name>
               <last_name>Allyson</last_name>
           </person>
            .
            .
            .
</events>
```

Observe el espacio en blanco entre la etiqueta <events> y la etiqueta <event> (ese espacio se reporta como texto a nuestra función text). Como en el texto entre las etiquetas <event> y <event_title>, y así sucesivamente. No deseamos mostrar este espacio en blanco en nuestra versión analizada de este archivo (nosotros nos encargamos de la sangría). Para evitar responder a texto que es puramente espacio en blanco, puede utilizar una expresión regular. En particular, puede insistir en que debe aparecer al menos un carácter que no sea espacio en blanco (lo cual puede comprobar con la expresión regular"/S") en el texto que se nos pasa antes de que lo mostremos de esta forma:

```
<?php
  function text($parser, $text)
  {
    if(preg_match("/\S/", $text)){
        .
        .
        .
    }
  }
?>
```

Ahora puede reproducir con echo la cadena de sangría y también el texto que no es espacio en blanco que el analizador encontró:

```
<?php
  function text($parser, $text)
  {
    global $indent;
    if(preg_match("/\S/", $text)){
```

```
    for ($i = 0; $i < $indent; $i++) {
        echo "    ";
    }
      echo "$text <br>";
    }
  }
?>
```

Bueno, eso completa el analizador XML. Puede verlo en acción en la Figura 15-12, donde aparece la versión analizada completa de event.xml. Excelente.

Eso pone fin a este ejemplo y también a nuestro libro sobre PHP. ¡Feliz programación!

FIGURA 15-12 Análisis de event.xml

Índice analítico

Símbolos

carácter de control \n, 17
matriz $items, 541
operador $a & $b, 52
operador $a ^ $b, 52
operador $a | $b, 52
operador $a << $b, 52
operador $a >> $b, 52
operador %=, 47
operador &=, 47
operador *=, 47
operador .=, 47
operador /=, 47
operador ^=, 47
operador |=, 47
operador ~$a, 52
operador +=, 47
operador <<=, 47
operador -=, 47
operador >>=, 47

A

acceso
  a propiedades y métodos, 253-257
  base de datos MySQL
    cierre de la conexión, 376-377
    cómo mostrar datos de una tabla, 374-376
    conexión a la base de datos, 372
    conexión al servidor, 371-372
    descripción, 370-371
    lectura de tablas, 372-374
  miembro de clase, 264-266
actualización, 377-380
Adabas, 361
Ajax (Asynchronous JavaScript and XML), 433-500
  conexión a otros dominios, 494-496
  descarga
    de imágenes, 479-481
    de JavaScript, 481-484
  descripción, 433-434
  diálogos de advertencia, 489
  escritura, 435-436
  Google Suggest, 484-494
  inicio de descargas, 445-447
  inicio de sesión, 495-497
  manejo de datos descargados, 441-445
  método GET, 450-453
  método POST, 453-456
  objetos XMLHttpRequest, 436-441, 446-448
  PHP y, 448-449
  solicitudes concurrentes
    funciones inner de JavaScript, 475-479
    matrices XMLHttpRequest, 472-475
    múltiples objetos XMLHttpRequest, 467-472
  solicitudes de encabezados, 497-500
  XML, 456-466
almacenaje de datos, 27-30, 425-429
análisis de archivos, 338-341
anulación de variables, 30
apertura
  de archivos, 319-322
  de objetos XMLHttpRequest, 440-441
API (interface de programación de aplicaciones), Google Ajax, 467
apuntadores de matriz, 116-117
apuntadores de archivos, 343
arcos, 514-416
archivos, 319-359
  análisis, 338-341
  anexión a, 352-354

apertura, 319-322
binarios, 348-352
bloqueo, 357-359
carga, 187-191, 408-411
cierre, 323-324
copia, 343-344
descarga, 406-408
descripción, 319
eliminación, 345-346, 411-413
escritura
archivos binarios, 348-352
file_put_contents, 355-356
de cadenas en, 346-348
establecer la ubicación del apuntador, 343
función file_exists, 332-334
función stat, 341-342
imágenes, 528-531
de inclusión, 157-158
inicialización, 339-341
lectura
archivos binarios, 335-337, 348-352
carácter por carácter, 325-327
contenido completo de, 328-329
en matrices, 330-332
texto de, 322-323
recorrido en ciclo por el contenido, 322
tamaño de, 334-335
archivos adjuntos, en correo electrónico, 421-425
archivos binarios, 335-337, 348-352
archivos de inclusión, 157-158
áreas de texto, 167-170
argumento pos, 406, 409
argumentos, 132-135
argumentos predeterminados, 132-133
Asynchronous JavaScript and XML. *Vea* Ajax
atributos de acción, 162
atributos y funciones SimpleXML
adición, 557-560
extracción, 550-552
modificación, 555-557

B

bandera 't', 320
bandera de traducción de modo de texto ('t'), 320
bases de datos, 361-394
descripción, 361-362
MySQL
acceso a, 370-377
actualización, 377-380
clasificación de datos, 393-394
clasificación de datos en, 368-370
creación de, 389-392
descripción, 364-367
eliminación de registros, 383-385
inserción de elementos de datos, 380-382
tablas, 367-368, 385-388
SQL, 362-364
bases de datos MySQL
acceso a
cierre de la conexión, 376-377
conexión a una base de datos, 372
conexión a un servidor, 371-372
descripción, 370-371
muestra de datos en una tabla, 374-376
lectura de tablas, 372-374
actualización, 377-380
clasificación de datos, 393-394
creación, 389-392
descripción, 364-367
eliminación de registros, 383-385
entrada de datos, 368-370
inserción de nuevos elementos de datos, 380-382
tablas, 367-368, 385-388
bloqueo de archivos, 357-359
botón de reinicio, 163
botones, 191-201
descripción, 191-192
Enviar, como botones HTML, 195-201
persistencia de datos, 192-195
de radio, 172-175
de reinicio, 163
botones Enviar, 195-201
botones de radio, 172-175

C

cadenas, 81-91
conversión a y de, 87-88
descripción, 81
escritura en archivos, 346-348
formateo, 88-91
función explode, 107
función implode, 106-107

funciones, 81-86
interpolación, 31-33
cadenas de texto. *Vea* cadenas
campos, definidos, 362
campos de texto, 164-167
caracteres, 320, 326
caracteres de control, 17
caracteres de final de línea, 320
caracteres de nueva línea, 326
carga de archivos, 187-191, 408-411
carga automática de clases, 277-279
CGI (Common Gateway Interface), 20
ciclos, 69-80
descripción, 41
do...while, 74-76
for, 69-72
foreach, 76-77, 548-549
manejo de matrices con, 99-104
matrices multidimensionales en, 114-115
omisión de iteraciones, 78-79
sintaxis alterna, 80
terminación prematura, 77-78
while, 72-74
ciclos do...while, 74-76
ciclos for, 69-72, 99-100
ciclos foreach, 76-77, 101-103, 548-549
ciclos while, 72-74, 103-104
cierre de archivos, 323-324
clases. *Vea también* métodos estáticos
abstractas, 294-297
carga automática de, 277-279
constantes, 306-308
creación de, 246-249
herencia
acceso protegido, 264-266
constructores y, 266-267
descripción, 262-264
métodos de clase de base, 267-271
ReflectionFunction, 315
clases abstractas, 294-297
clase DataHandler, 301
clase ReflectionFunction, 315
clasificación
datos, 393-394
matrices, 109-110
CLI (intérprete de la línea de comandos), 20
cláusula AND, 364
cláusula IN, 363
cláusula NOT, 364
cláusula OR, 364
cláusulas, 363-364
cláusula WHERE, 363
codificación de URL, 450
comando CREATE DATABASE, 366
comando SHOW DATABASES;, 366
combinación de matrices, 117-119
comentarios, 24-25
comentarios anidados, 24
comentarios en líneas múltiples, 24
comentarios en una línea, 24-25
Common Gateway Interface (CGI), 20
comparaciones, matriz de, 112
concatenación, 18
conexiones
base de datos, 371-372, 376-377
Google Suggest, 484-494
cookies, 395-402
descripción de las, 395
eliminación de, 400-402
establecimiento de, 395-397
lectura de, 397-398
tiempo de caducidad de las, 399-400
constante _CLASS_, 36
constante DEFAULT_INCLUDE_PATH, 36
constante _FILE_, 36
constante _FUNCTION_, 36
constante _LINE_, 36
constante _METHOD_, 36
constante PHP_OS, 36
constante PHP_VERSION, 36
constante SEEK_CUR, 343
constante SEEK_END, 343
constante SEEK_SET, 343
constantes
clase, 306-308
PHP, 36-37
constructores, 257-260, 266-267
contadores, de ciclos, 71
contadores de visitas, 429-431
control de flujo, 63-80
ciclos do...while, 74-76
ciclos for, 69-72
ciclo foreach, 76-77
ciclos while, 72-74
descripción, 41
instrucción else, 63-64
instrucción elseif, 65-66
instrucción if, 55-58

instrucción switch, 67-69
omisión de iteraciones, 78-79
sintaxis alterna, 80
terminación prematura de ciclos, 77-78
controles
contraseña, 179-182
ocultos, 182-184
controles de contraseñas, 179-182
control de selección múltiple, 176
controles ocultos, 182-184
conversión, cadena de, 87-88, 106-107
copia, de imagen, 535-536
correo electrónico, 416-425
adición de archivos adjuntos, 421-425
avanzado, 418-421
descripción, 416-418
correos electrónicos avanzados, 418-421
cuadros
de alerta de JavaScript, 239
de lista, 175-179
de selección, 170-172
imágenes de, 506
cuadros de lista, 175-179
cuadros de selección, 170-172

D

datos de formas, 212-220
manejo en una página, 218-220
matrices personalizadas, 215-218
vaciado, 212-215
datos de usuario persistentes, 234-237
datos globales, acceso a, 145-147
datos modificados, en matrices, 95-97
datos en el nivel de script, 145
datos requeridos, 223-234
ingresados por el usuario, 223-226
números, 227-230
texto, 230-234
dBase, 361
descarga
con Ajax
funciones callback, 441-445
imágenes, 479-481
JavaScript, 481-484
con FTP, 406-408
disminución de valores, 48-50
división de matrices, 117-119

E

elemento targetDiv, 444-445
elementos
adición con funciones SimpleXML, 557-560
eliminación en matrices, 97-99
documento, 456
modificación con funciones SimpleXML, 555-557
targetDiv, 444-445
elementos de documentos, 456
eliminación
de cookies, 400-402
de elementos en matrices, 97-99
de archivos, 345-346
con FTP, 411-413
de registros de MySQL, 383-385
elipses, 513-514
Empress, 361
encabezados cc, 418-421
encabezados HTTP, 205-206, 209-212
entorno de desarrollo, 6-7
entornos de desarrollo integrado (IDEs), 6-7
entrada en bases de datos, 368-370
envío de correo electrónico, 416-421
errores, devolución de funciones, 158-160
escritura
Ajax, 435-436
archivos
binarios, 348-352
cadenas en, 346-348
file_put_contents, 355-356
en archivos, 346-348
contadores de visitas usando sesiones, 429-431
especificadores, 89
especificadores de alineación, 89
especificadores de ancho, 89
especificadores de precisión, 89
expresiones regulares, 231
eXtended Markup Language (XML). *Vea* XML
extracción de atributos, 550-552

F

File Transfer Protocol. *Vea* FTP
FilePro, 361
Firefox, 437, 439
formateo de cadenas, 88-91

formato de datos de tabla DECIMAL(totaldigits, decimalplaces), 367
formato de datos de tabla INT, 367
formato de datos de tabla VARCHAR(*length*), 367
formato de tabla de datos DATETIME, 367
formatos de datos, 367
FrontBase, 361
FTP (File Transfer Protocol), 402-415
  carga
    archivos, 408-411
    páginas PHP con, 7
  creación y eliminación de directorios, 414-415
  descarga de archivos, 406-408
  descripción, 402-405
  eliminación de archivos, 411-413
  permisos en scripts, 5
función abs, 43
función acos, 43
función acosh, 43
función addcslashes, 82
función addslashes, 82
función array, 105
función array_change_key_case, 104
función array_chunk, 104
función array_combine, 104
función array_count_values, 104
función array_diff, 104
función array_diff_assoc, 104
función array_diff_uassoc, 104
función array_fil, 104
función array_filter, 104
función array_flip, 104, 119
función array_intersect, 104
función array_intersect_assoc, 104
función array_key_exists, 105
función array_keys, 105
función array_map, 105
función array_merge, 105
función array_merge_recursive, 105
función array_multisort, 105
función array_pad, 105
función array_pop, 105
función array_push, 105
función array_rand, 105
función array_reduce, 105
función array_reverse, 105
función array_search, 105
función array_shift, 105
función array_slice, 105, 117
función array_splice, 105
función array_sum, 105, 119
función array_udiff, 105
función array_udiff_assoc, 105
función array_udiff_uassoc, 105
función array_unique, 105, 120
función array_unshift, 105
función array_values, 105
función array_walk, 105
función arsort, 105
función asin, 43
función asinh, 43
función asort, 105, 110
función atan, 43
función atan2, 43
función atanh, 43
función _autoload, 277
función base_convert, 43
función bin2hex, 82
función bindec, 43
función ceil, 43
función check_data, 222, 235
función chop, 82
función chr, 82
función chunk_split, 82
función compact, 105
función connectGoogleSuggest, 486
función convert_cyr_string, 82
función copy, 343-344
función cos, 43
función cosh, 43
función count, 99, 105
función count_chars, 82
función crc32, 82
función crypt, 82
función current, 105
función decbin, 43
función dechex, 43
función decoct, 43
función defaultHandler, 568
función define, 35
función deg2rad, 43
función die, 158
función display, 482
función each, 103, 106
función echo, 82
función end, 106
función endElement, 571
función exp, 43

función explode, 82, 107
función expm1, 43
función extract, 106-108
función fclose, 323-324
función feof, 189, 322
función fgetc, 325-327
función fgets, 190, 322-323
función file, 330-332
función file_exists, 332-334
función file_get_contents, 328-329
función file_put_contents, 355-356
función filesize, 334-335
función flock, 357-359
función floor, 43
función fmod, 43
función fopen, 189, 319-322
función fprintf, 82
función fread, 335-337, 348-352
función fscanf, 82
función fseek, 343
función ftp_alloc, 402
función ftp_cdup, 402
función ftp_chdir, 402
función ftp_chmod, 402
función ftp_close, 402
función ftp_connect, 402
función ftp_delete, 402
función ftp_exec, 402
función ftp_fget, 402
función ftp_fput, 402
función ftp_get, 402
función ftp_get_option, 402
función ftp_login, 402
función ftp_mdtm, 402
función ftp_mkdir, 402
función ftp_nb_continue, 402
función ftp_nb_fget, 403
función ftp_nb_fput, 403
función ftp_nb_get, 403
función ftp_nb_put, 403
función ftp_nlist, 403
función ftp_put, 403
función ftp_pwd, 403
función ftp_quit, 403
función ftp_raw, 403
función ftp_rawlist, 403
función ftp_rename, 403
función ftp_rmdir, 403
función ftp_set_option, 403
función ftp_site, 403
función ftp_size, 403
función ftp_ssl_connect, 403
función ftp_systype, 403
función func_get_arg, 134
función func_get_args, 134
función func_num_args, 134
función fwrite, 346-348, 352-354
función gd_info, 501
función get_html_translation_table, 82
función getData, 436, 498
función getimagesize, 501
función getrandmax, 43
función hebrev, 82
función hebrevc, 82
función hexdec, 43
función html_entity_decode, 82
función htmlentities, 82, 242
función htmlspecialchars, 82
función hypot, 43
función image_type_to_extension, 501
función image_type_to_mime_type, 501
función image2wbmp, 501
función imagealphablending, 501
función imageantialias, 501
función imagearc, 501, 514-516
función imagechar, 501
función imagecharup, 501
función imagecolorallocate, 501, 505
función imagecolorallocatealpha, 501
función imagecolorat, 501
función imagecolorclosest, 501
función imagecolorclosestalpha, 501
función imagecolorclosesthwb, 502
función imagecolordeallocate, 502
función imagecolorexact, 502
función imagecolorexactalpha, 502
función imagecolormatch, 502
función imagecolorresolve, 502
función imagecolorresolvealpha, 502
función imagecolorset, 502
función imagecolorsforindex, 502
función imagecolorstotal, 502
función imagecolortransparent, 502
función imageconvolution, 502
función imagecopy, 502, 535
función imagecopymergegray, 502
función imagecopyresampled, 502
función imagecopyresized, 502

función imagecreate, 502, 504
función imagecreatefromgd, 502
función imagecreatefromgd2, 502
función imagecreatefromgd2part, 502
función imagecreatefromgif, 502, 528
función imagecreatefromjpeg, 502, 528
función imagecreatefrompng, 502, 528
función imagecreatefromstring, 502
función imagecreatefromwbmp, 502, 528
función imagecreatefromxbm, 502, 528
función imagecreatefromxpm, 502, 528
función imagecreatetruecolor, 502, 531
función imagedashedline, 502
función imagedestroy, 502, 505
función imageellipse, 502, 513-514
función imagefill, 503
función imagefilledarc, 503, 518
función imagefilledellipse, 503, 518
función imagefilledpolygon, 503, 517
función imagefilledrectangle, 503, 518
función imagetoborder, 503
función imagefilter, 503
función imagefontheight, 503
función imagefontwidth, 503, 522
función imageftbbox, 503
función imagefttext, 503
función imagegammacorrect, 503
función imagegd, 503
función imagegd2, 503
función imagegif, 503, 505
función imageinterlace, 503
función imageistruecolor, 503
función imagejpeg, 503, 505
función imagelayereffect, 503
función imageline, 503, 507-509
función imageloadfont, 503, 522
función imagepalettecopy, 503
función imagepng, 503, 505
función imagepolygon, 503, 516-518
función imagepsbbox, 503
función imagepsencodefont, 503
función imagepsextendfont, 503
función imagepsfreefont, 503
función imagepsloadfont, 503
función imagepsslantfont, 503
función imagepstext, 503
función imagerectangle, 503, 511-512, 529
función imagerotate, 503
función imagesavealpha, 504
función imagesetbrush, 504
función imagesetpixel, 504
método imagesetpixel, 520
función imagesetstyle, 504
función imagesetthickness, 504, 510-511
función imagesettile, 504, 531
función imagestring, 504, 522
función imagestringup, 504, 525-526
función imagesx, 504
función imagesy, 504
función imagetruecolorpalette, 504
función imagettfbbox, 504
función imagettftext, 504
función imagetypes, 504
función imagewbmp, 504
función imagexbm, 504
función implode, 82, 106-107
función in_array, 106
función iptcembed, 504
función iptcparse, 504
función is_finite, 43
función is_infinite, 43
función is_nan, 43
función join, 82
función jpeg2wbmp, 504
función key, 106
función krsort, 106, 110
función lcg_value, 43
función levenshtein, 82
función list, 103, 106, 108
función localeconv, 82
función log, 44
función log10, 44
función log1p, 44
función ltrim, 82
función mail, 416
función manejadora startElement, 568
función max, 44
función md5, 82
función md5_file, 82
función metaphone, 82
función min, 44
función money_format, 82
función mt_getrandmax, 44
función mt_rand, 44
función mt_srand, 44
función mysql_affected_rows, 370
función mysql_change_user, 370
función mysql_client_encoding, 370

función mysql_close, 370, 376
función mysql_connect, 370, 371
función mysql_create_db, 370
función mysql_data_seek, 370
función mysql_db_name, 370
función mysql_db_query, 370
función mysql_drop_db, 370
función mysql_error, 370
función mysql_fetch_array, 370, 375
función mysql_fetch_assoc, 370
función mysql_fetch_row, 371
función mysql_field_len, 371
función mysql_field_name, 371
función mysql_field_seek, 371
función mysql_field_table, 371
función mysql_field_type, 371
función mysql_get_server_info, 371
función mysql_info, 371
función mysql_list_dbs, 371
función mysql_list_fields, 371
función mysql_list_tables, 371
función mysql_num_fields, 371
función mysql_num_rows, 371
función mysql_pconnect, 371
función mysql_query, 363, 371, 372
función mysql_result, 371
función mysql_select_db, 371, 372
función mysql_tablename, 371
función natcasesort, 106
función natsort, 106, 110
función next, 106
función nl_langinfo, 82
función nl2br, 82
función number_format, 82, 91
función octdec, 44
función ord, 82
función pack, 348-349
función parse_ini_file, 339-341
función parse_str, 83
función phpinfo( ), 8
función pi, 44
función png2wbmp, 504
función pos, 106
función pow, 44
función prev, 106
función print, 83
función print_r, 100-101
función printf, 83, 89
función quoted_printable_decode, 83
función quotemeta, 83
función rad2deg, 44
función rand, 44
función range, 106
función rename, 344
función reset, 106
función round, 44
función rsort, 106
función rtrim, 83
función scoper, 143
función sendRPCDone, 491
función setcookie, 395-396
función setlocale, 83
función setproducts, 458-459, 461
función sha1, 83
función sha1_file, 83
función show_errors, 225
función show_welcome, 235
función shuffle, 106
función similar_text, 83
función simplexml_import_dom, 544
función simplexml_load_file, 544
función simplexml_load_string, 544
función SimpleXMLElement->_construct( ), 544
función SimpleXMLElement->addAttribute( ), 544
función SimpleXMLElement->addChild( ), 544
función SimpleXMLElement->asXML( ), 544
función SimpleXMLElement->attributes( ), 544
función SimpleXMLElement->children( ), 544
función SimpleXMLElement->getDocNamespaces ( ), 544
función SimpleXMLElement->getName( ), 545
función SimpleXMLElement->getNamespaces (), 545
función SimpleXMLElement- >registerXPathNamespace( ), 545
función SimpleXMLElement->xpath( ), 545
función sin, 44
función sinh, 44
función sizeof, 106
función sort, 106, 108
función soundex, 83
función sprintf, 83, 89
función sqrt, 44
función srand, 44
función sscanf, 83
función stat, 341-342
función str_ireplace, 83
función str_pad, 83
función str_repeat, 83

función str_replace, 83
función str_rot13, 83
función str_shuffle, 83
función str_split, 83
función str_word_count, 83
función strcasecmp, 83
función strchr, 83
función strcmp, 83
función strcoll, 83
función strcspn, 83
función strip_tags, 83, 243
función stripcslashes, 83
función stripos, 83
función stripslashes, 83
función stristr, 83
función strlen, 83
función strnatcasecmp, 84
función strnatcmp, 84
función strncasecmp, 84
función strncmp, 84
función strpos, 84
función strrchr, 84
función strrev, 84
función strripos, 84
función strrpos, 84
función strspn, 84
función strstr, 84
función strtok, 84
función strtolower, 84
función strtoupper, 84
función strtr, 84
función substr, 84
función substr_compare, 84
función substr_count, 84
función substr_replace, 84
función tan, 44
función tanh, 44
función trim, 84
función uasort, 106
función ucfirst, 84
función ucwords, 84
función uksort, 106
función unlink, 345-346
función unpack, 349
función usort, 106
función vardump, 58
función vprintf, 84
función vsprintf, 84
función wordwrap, 84
función xml_error_string, 563
función xml_get_current_byte_index, 563
función xml_get_current_column_number, 563
función xml_get_current_line_number, 563, 567
función xml_get_error_code, 563
función xml_parse, 563, 567
función xml_parse_into_struct, 563
función xml_parser_create_function, 563
función xml_parser_create_ns, 563
función xml_parser_free, 563
función xml_parser_get_option, 563
función xml_parser_set_option, 563
función xml_set_character_data_handler, 563, 565
función xml_set_default_handler, 563
función xml_set_element_handler, 563
función xml_set_end_namespace_decl_handler, 564
función xml_set_external_entity_ref_handler, 564
función xml_set_notation_decl_handler, 564
función xml_set_object, 564
función xml_set_processing_instruction_handler, 564
función xml_set_start_namespace_decl_handler, 564
función xml_set_unparsed_entity_decl_handler, 564
funciones, 123-160. *Vea también funciones individuales por nombre*
  acceso a datos globales, 145-147
  de ámbito variable, 143-144
  analizador XML, 563-573
  anidación, 156-157
  archivos de inclusión, 157-158
  array, 104-106, 119-121
  de cadena, 81-86
  condicionales, 150-153
  copiar, 343-344
  creación, 123-125
  definidas, 8
  descripción, 123
  FTP, 402-403
  gráficas, 303-306
  inner, 475-479
  math, 43-44
  MySQL, 370-371
  paso
    de argumentos a, 132-135
    de datos a, 125-127
    de matrices a, 127-130
    por referencia, 130-131
  retorno
    de datos de, 135-137
    de errores de, 158-160

de listas de, 139-141
de matrices de, 137-139
de referencias de, 141-143
SimpleXML
adición de nuevos atributos con, 557-560
adición de nuevos elementos con, 557-560
extracción de atributos con, 550-552
modificación de atributos con, 555-557
modificación de elementos con, 555-557
variables, 153-155
variables estáticas, 147-150
funciones analizadoras de XML, 563-573
funciones anidadas, 156-157
funciones callback, 441-442, 479, 565
funciones cast, 39, 87
funciones condicionales, 150-153
funciones para crear imágenes, 505
funciones inner, JavaScript, 475-479
funciones math, 43-44
funciones multiparte, 187, 422
funciones SimpleXML
descripción, 544-550
extracción de atributos, 550-552
modificación de atributos, 555-557
modificación de elementos, 555-557
nuevos atributos, 557-560
nuevos elementos, 557-560

G

Google Suggest, 481, 484-494
gráficos, 501-536
arcos, 514-516
copia de imágenes, 535-536
creación de imágenes, 504-506
creación de texto, 522-525
descripción, 501
elipses, 513-514
funciones, 501-504
imágenes en mosaico, 531-534
líneas, 507-511
muestra de, en páginas HTML, 506-507
píxeles, 520-521
polígonos, 516-518
rectángulos, 511-513
relleno de figuras, 518-520
texto vertical, 525-528
trabajo con archivos de imagen, 528-531
grosor de línea, 510-511
Gutmans, Andi, 2

H

herencia
acceso protegido, 264-266
constructores y, 266-267
descripción, 262-264
métodos de clase de base, 267-271
miembros estáticos y, 291-293
herencia múltiple, 297
HTML
botones, 195-201
combinación de PHP y, 10-13
etiquetas, 241-243
impresión, 16-17
muestra de imágenes en, 506-507
Hyperwave Direct, 361

I

IBM DB2, 361
IDEs (entornos de desarrollo integrado), 6-7
imágenes
archivos de imagen, 528-531
copia, 535-536
cuadro, 506
descarga con Ajax, 479-481
en mosaico, 531-534
muestra de, en páginas HTML, 506-507
trazo, 504-506
imágenes en mosaico, 531-534
impresión, 14-17
incremento de valores, 48-50
Informix, 361
Ingres, 361
inicio de sesión con Ajax, 495-497
inserción de elementos en bases de datos, 380-382
instalación local, 5-6
instalación local de PHP, 5-6
instrucción continue, 78-79
instrucción else, 63-64
instrucción elseif, 65-66
instrucción if, 55-58
instrucción SQL CREATE, 385
instrucción SQL INSERT, 381

instrucción switch, 67-69
instrucciones
  break, 68, 77
  continue, 78
  echo, 14, 17-19
  else, 63-64
  elseif, 65-66
  if, 55-58
  sintaxis alterna, 80
  SQL CREATE, 385
  SQL INSERT, 381
  switch, 67-69
instrucciones break, 68, 77-78
instrucciones echo, 14, 17-19
instrucciones print, 19
InterBase, 361
interfaces
  API Ajax de Google, 467
  CGI, 20
  iDatabase, 297
  Iterator de PHP, 301
  OOP, 297-300
interface de programación de aplicaciones (API)
  Google Ajax, 467
interface iDatabase, 297
Internet Explorer, 437-438
interpolación de cadenas, 31-33
intérprete de la línea de comandos (CLI), 20
iteración de objetos, 301-303
iteraciones de ciclo omitidas, 78-79
iteraciones de ciclos, 78-79

J

JavaScript. *Vea también* Ajax
  cuadro de alerta, 239
  descarga con Ajax, 481-484
  función callback, 479
  función connectGoogleSuggest, 486
  función display, 482
  funciones inner, 475-479
  método push, 472
  páginas Web estáticas, 3
  validación en el cliente, 237
  variables, 27

L

lecturas
  lecturas binarias, 335-337
  cookies, 397-398
  archivos
    en matrices, 330-332
    binarios, 348-352
    carácter por carácter, 325-327
    función file_get_contents, 328-329
    texto de, 322-323
  tablas, 372-374
Lerdorf, Rasmus, 1
línea de comandos, 17, 20-24
líneas, 507-511
listas, retorno a funciones, 139-141
logotipos, 12-13

M

manejadores de archivos, 189
matrices, 92-121
  asociativas, 112
  clasificación, 109-110
  comparación de, 112
  construcción de, 92-95
  descripción, 81
  división y combinación, 117-119
  eliminación de elementos de, 97-99
  extracción de datos de, 107-108
  función explode, 107
  función implode, 106-107
  funciones, 104-106, 119-121
  lectura de archivos en, 330-332
  manejo con ciclos, 99-104
  modificación de datos en, 95-97
  multidimensionales, 112-115
  navegación en, 116-117
  operadores, 110-112
  paso a funciones, 127-130
  personalizadas, 215-218
  retorno a funciones, 137-139
  superglobales, 203
  voltear, 120
  XMLHttpRequest, 472-475
matrices asociativas, 112

matrices multidimensionales, 112-115
matrices personalizadas, 215-218
matriz superglobal, 203
método _clone, 312
método de apertura, 438
método GET, 165-166, 451-454
método getAllResponseHeaders, 438-439, 498
método getElementsByTagName, 460
método getResponseHeader, 438-439
método HEAD, 498
método imagesetpixel, 520
método loadXML, 561-562
método openRequest, 439
método overrideMimeType, 439
método POST, 166, 452-455
método public function current( ), 301
método public function key( ), 301
método public function next( ), 301
método public function rewind( ), 301
método public function valid( ), 301
método push, 472
método send, 438
método setRequestHeader, 438
métodos
  abort, 438-439
  accessor, 252
  addAttribute, 559
  addChild, 558
  anulación, 271-273
  de acceso, 253-257
  de apertura, 438
  de clase, 283
  de clase de base, 267-271
  estáticos, 281-291
    creación, 283-285
    descripción, 281-283
    paso de datos a, 285-286
    uso de propiedades en, 286-291
  GET, 165-1666, 449-452
  getAllResponseHeaders, 438-439, 498
  getElementsByTagName, 459
  getResponseHeader, 438-439
  HEAD, 498
  imagesetpixel, 520
  interfaz Iterator, 301
  loadXML, 561-562
  openRequest, 439
  overrideMimeType, 439
  POST, 166, 452-456
  public function current( ), 301
  public function key( ), 301
  public function next( ), 301
  public function rewind( ), 301
  public function valid( ), 301
  push, 472
  send, 438
  sobrecarga, 273-277
métodos de anulación, 271-273
métodos de clase de base, 267-271
métodos estáticos
  creación, 283-285
  descripción, 281-283
  paso de datos a, 285-286
  uso de propiedades en, 286-291
métodos de la interfaz Iterator, 301
métodos de sobrecarga, 273-277
Microsoft WordPad, 6
Mini SQL (mSQL), 1, 361
miembros estáticos, 291-293
modificaciones, con funciones SimpleXML, 555-557
modificador de acceso privado, 253-257
modificadores de acceso, 253
Mozilla, 439
mSQL (Mini SQL), 1, 361
MS-SQL, 361
múltiples objetos XMLHttpRequest, 467-472

N

navegación en matrices, 116-117
navegadores, 203-243
  validación de datos, 221-223, 237-240
  determinación del usuario, 206-209
  datos de forma, 221-220
    vaciado, 212-215
    manejo en una página, 218-220
    manejo con matrices personalizadas, 215-218
  etiquetas HTML en la entrada del usuario, 241-243
  encabezados HTTP, 205-206, 209-212
  descripción, 203
  datos de usuario persistentes, 234-237
  datos requeridos, 223-224
    ingresados por el usuario, 223-226
    números, 227-230
    texto, 230-234
  variables de servidor, 203-205
Navigator, 437, 439

Netscape Navigator, 437, 439
nodo de contexto, 553
notación de matriz, 550-552
números requeridos, 227-230
números variables, argumento, 133-135

O

objetos. *Vea también* programación orientada a objetos (POO)
  clonación, 312-315
  comparación, 304-305
  constructores, 257-260
  creación, 250-253
  destructores, 260-261
  inicialización con constructores, 257-260
  iteración, 301-303
  XMLHttpRequest
    apertura, 440-441
    creación, 436-440, 446-448
    solicitudes Ajax concurrentes con múltiples, 467-472
objetos clonados, 312-315
ODBC, 361
opción –v, 4
operador de ejecución, 52-53
operador de identidad, 304
operador de igualdad, 59
operador ternario, 66-67
operadores, 41-80. *Vea también* control de flujo
  asignación, 28, 46-48
  cadena, 50-51
  comparación, 59-60, 304
  descripción, 41
  ejecución, 52-53
  identidad, 304
  incremento y disminución de valores, 48-50
  lógicos, 61-62
  matemáticos, 41-46
  matriz, 110-112
  orientados a bits, 51-52
  precedencia de, 53-55
  ternarios, 66-67
operadores de asignación, 28, 46-48
operadores de cadena, 50-51
operadores de comparación, 59-60, 304-305
operadores de desplazamiento, 52
operadores lógicos, 61-62
operadores lógicos“and”, 62
operadores lógicos“or”, 62
operadores matemáticos, 41-46
operadores orientados a bits, 51-52
Oracle, 361
Ovrimos, 361

P

páginas, PHP, 8-10
páginas Web, 164-201
  cuadros de selección, 170-172
  cargas de archivos, 187-191
  manejo de datos de formas en, 218-220
  controles ocultos, 182-184
  mapas de imágenes, 184-186
  cuadros de lista, 175-179
  descripción, 164
  controles de contraseña, 179-182
  persistencia de datos de botones, 192-195
  botones de radio, 173-175
  configuración para PHP, 161-164
  botones Enviar como botones HTML, 195-201
  áreas de texto, 167-170
  campos de texto, 164-167
  creación, 8-9
  ejecución, 9-10
  solución de problemas, 9-10
palabra clave final, 308-312
  clone, 312
  estáticas, 149
  final, 308-312
  lista de, 37
parámetro num_points, 517
parámetro points, 517
parámetros port, 403
Personal Home Page (PHP), 1-39
  combinación de HTML y, 10-13
  comentarios, 24-25
  constantes, 36-37
  creación de páginas, 8-9
  descripción, 1-3
  documentos here, 19-20
  ejecución de páginas, 9-10
  entorno de desarrollo, 6-7
impresión
    HTML, 16-17
    texto, 14-15

instalación local, 5-6
instrucción echo, 17-19
línea de comandos, 20-24
soporte de PSIs, 4-6
tipos de datos internos, 37-39
variables, 26-35
descripción, 26-27
interpolación de cadenas, 31-33
alternas, 33-35
almacenaje de datos en, 27-30
PHP. *Vea* Personal Home Page
píxeles, 520-521
polígonos, 516-518
POO. *Vea* programación orientada a objetos
PostgreSQL, 361
precedencia, operador de, 53-55
presentaciones
de datos en tablas, 374-376
de imágenes, 506-507
programación orientada a objetos (POO), 245-279, 281-317
acceso a métodos, 253-257
acceso a propiedades, 253-257
clases
abstractas, 294-297
carga automática, 277-279
creación de constantes, 306-308
creación de, 246-249
clonación de objetos, 312-315
comparación de objetos, 304-305
constructores, 257-260, 266-267
descripción, 245-246, 281
destructores, 260-261
herencia
interfaces, 297-300
iteración, 301-303
métodos estáticos, 281-291
creación, 283-285
descripción, 281-283
paso de datos a, 285-286
uso de propiedades en, 286-291
métodos de sobrecarga, 273-277
métodos de anulación, 271-273
objetos, 250-253
palabra clave final, 308-312
métodos de clase de base, 267-271
descripción, 262-264
acceso protegido, 264-266
miembros estáticos y, 291-293
reflexión, 315-317
propiedad onreadystatechange, 438-439
propiedad readyState, 438-439, 442
propiedad responseBody, 438
propiedad responseStream, 438
propiedad responseText, 438-439
propiedad responseXML, 438-439, 459
propiedad status, 438-439, 442
propiedad statusText, 438-439
PSIs (proveedores de servicios de Internet), 4-6

R

Really Simple Syndication (RSS), 540-544
rectángulos, 511-513
redirección de navegadores, 209-212
referencias, 130-131, 141-143
reflexión, 315-317
registros, 362, 383-385
RSS (Really Simple Syndication), 540-544

S

Safari, 437, 439
script
ejecutable, 5
redirección, 211
script ejecutable, 5
servidores
método GET, 449-452
método POST, 452-455
sesiones, 425-431
almacenaje de datos en, 425-429
descripción, 395
escritura de contadores de visitas utilizando, 429-431
sintaxis alterna, 80
sintaxis parent::, 270
solicitudes Ajax concurrentes, 476-479
funciones inner de JavaScript, 475-479
matrices XMLHttpRequest, 472-475
múltiples objetos XMLHttpRequest, 467-472
Solid, 361
SQL (Structured Query Language), 362-364
Structured Query Language (SQL), 362-364

Suggest, 481, 484-494
Suraski, Zeev, 2
Sybase, 361

T

tablas
  creación, 367-368, 385-388
  lectura, 372-374
  muestra de datos, 374-376
terminación de ciclo, 77-78
texto
  impresión, 14-15
  requerido, 230-234
  trazo, 522-525
  vertical, 525-528
texto estático, 26
texto vertical, 525-528
tiempo de caducidad, de una cookie, 399-400
tipo de datos booleano, 37
tipo de datos de cadena, 37
tipo de datos de objeto, 37
tipo de datos de recurso, 37
tipo de datos entero, 38
tipo de datos flotante, 37
tipo de datos interno de matriz, 38
tipo de datos NULL, 37
tipo de nodo, XPath, 552
tipo Multipurpose Internet Mail Extension (MIME), 421
tipos de datos, 37-39
tipos de datos internos, 37-39
trazos. *Vea* gráficos

U

Unicode, 4
Unix, 361, 399

V

vaciado de datos de formas, 212-215
validación de datos, 221-223, 237-240
validación de datos en el cliente, 237-240
variable 'HTTP_ACCEPT', 205
variable 'HTTP_ACCEPT_CHARSET', 205
variable 'HTTP_ACCEPT_ENCODING', 205
variable 'HTTP_ACCEPT_LANGUAGE', 205
variable 'HTTP_CONNECTION', 205
variable 'HTTP_HOST', 205
variable 'HTTP_REFERER', 205
variable 'HTTP_USER_AGENT', 205
variable de servidor 'AUTH_TYPE', 204
variable de servidor 'DOCUMENT_ROOT', 204
variable de servidor 'GATEWAY_INTERFACE', 204
variable de servidor 'PATH_TRANSLATED', 204
variable de servidor 'PHP_AUTH_PW', 204
variable de servidor 'PHP_AUTH_USER', 204
variable de servidor 'PHP_SELF', 204
variable de servidor 'QUERY_STRING', 204
variable de servidor 'REMOTE_ADDR', 204
variable de servidor 'REMOTE_HOST', 204
variable de servidor 'REMOTE_PORT', 204
variable de servidor 'REQUEST_METHOD', 204
variable de servidor 'REQUEST_URL', 204
variable de servidor 'SCRIPT_FILENAME', 204
variable de servidor 'SCRIPT_NAME', 204
variable de servidor 'SERVER_ADMIN', 204
variable de servidor 'SERVER_NAME', 204
variable de servidor 'SERVER_PORT', 204
variable de servidor 'SERVER_PROTOCOL', 204
variable de servidor 'SERVER_SIGNATURE', 204
variable de servidor 'SERVER_SOFTWARE', 204
variables, 26-35
  almacenaje de datos en, 27-30
  alternas, 33-35
  ámbito de, 143-144
  descripción, 26-27
  estáticas, 147-150
  funciones, 153-155
  interpolación de cadenas, 31-33
  servidor, 203-205
variables de servidor, 203-205
variables de servidor HTTP, 205
variables estáticas, 147-150
Velocis, 361
voltear matrices, 120

W

WordPad, 6

X

XML, 537-573
  creación, 537-539
  declaraciones, 564
  descripción, 537
  envío al navegador, 560-561
  funciones Parser, 563-573
  funciones SimpleXML
    adición de nuevos elementos y atributos, 557-560
    descripción, 544-550
    extracción de atributos, 550-552
    modificación de elementos y atributos, 555-557
  interacción con otros paquetes, 561-563
  manejo, 456-466
  Really Simple Syndication, 540-544
  XPath, 552-555
XML (eXtended Markup Language). *Vea* XML
XMLHttpRequest
  apertura de objetos, 440-441
  creación de objetos, 436-440, 446-448
  matrices, 472-475
  método send, 445
  solicitudes Ajax concurrentes y múltiples objetos, 467-472
XPath, 552-555

Z

zcontext, 319