GUÍA DE APRENDIZAJE

Larry Ullman

PHP

ABARCA VERSIONES 3 y 4

Prentice Hall

005.13
V.14

Guía de aprendizaje
PHP

CONSULTORES EDITORIALES:

SEBASTIÁN DORMIDO BENCOMO
Departamento de Informática y Automática
UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

LUIS JOYANES AGUILAR
Departamento de Lenguajes, Sistemas Informáticos e Ingeniería del Software
UNIVERSIDAD PONTIFICIA DE SALAMANCA en Madrid

Guía de aprendizaje

PHP

Larry Ullman

Traducción
KME Sistemas, S.L.

Prentice Hall

Madrid - México - Santafé de Bogotá - Buenos Aires - Caracas - Lima
Montevideo - San Juan - San José - Santiago
São Paulo - White Plains

Datos de catalogación bibliográfica

LARRY ULLMAN
GUÍA DE APRENDIZAJE PHP
PEARSON EDUCACIÓN, S.A., Madrid, 2001

ISBN: 84-205-3231-2
Materia: Informática 681.3

Formato 170 × 240 Páginas: 296

Universidad Ingeniería de Aragua
REGISTRO: 00312
U.B.A.
Nucleo Puerto Ordaz

Larry Ullman
GUÍA DE APRENDIZAJE PHP

No está permitida la reproducción total o parcial de esta obra
ni su tratamiento o transmisión por cualquier medio o método
sin autorización escrita de la Editorial.

DERECHOS RESERVADOS
© 2001 respecto a la primera edición en español por:
PEARSON EDUCACIÓN, S.A.
Núñez de Balboa, 120
28006 Madrid

ISBN: 84-205-3231-2
Depósito Legal: M. 27.304-2001
PRENTICE HALL es un sello editorial autorizado de PEARSON EDUCACIÓN

Traducido de: Visual Quickstart Guide PHP for the World Wide Web
Copyright© 2001 by Larry Ullman
ISBN 0-201-72787-0

Editores de la edición en español: Alejandro Domínguez
Félix Fuentes
Eva María López
Cubierta: Mario Guindel y Yann Boix
Composición: COPIBOOK, S. L.
Impreso por: ELECE, Industria Gráfica, S.L.

IMPRESO EN ESPAÑA - PRINTED IN SPAIN

Este libro ha sido impreso con papel y tintas ecológicos

Para Jessica, Gina y Rich,
con gratitud por todo
su amor y apoyo.

Agradecimientos

Muchas, muchas gracias a todo el personal de Peachpit Press por su ayuda y duro trabajo, especialmente: Marjorie Baer, Nancy Davis, Rebecca Gulick, Lisa Brazieal, Nancy Aldrich-Ruenzel y Kate Reber.

Mi reconocimiento a Bruce Bowden por sus acertadísimas y embriagadoras enmiendas.

Gracias a Rasmus Lerdorf y a la comunidad PHP completa por desarrollar, mejorar y dar soporte a esta útil tecnología.

Índice de contenido

Introducción . . . . . **XI**

Capítulo 1: **Cómo empezar con PHP** **1**
Sintaxis básicas 2
Envío de datos al navegador. 3
Comprobación del *script* 5
Envío de texto al navegador. 7
Envío de HTML al navegador 9
Uso de espacios en blanco en PHP (y HTML). . . . 10
Adición de comentarios a los *scripts* 13

Capítulo 2: **Variables** **17**
¿Qué son las variables?. 18
Sintaxis de una variable 19
Tipos de variables 20
Asignación de valores a las variables 25
Variables predefinidas 26

Capítulo 3: **Formularios HTML y PHP** **27**
Creación de un formulario simple 28
Uso de Get o Post 31
Recepción de datos de un formulario en PHP . . . 33
Entrada manual de datos 35

Capítulo 4: **Utilización de números** **39**
Suma, resta, multiplicación y división 40
Formateo de números 43
Incremento y disminución de un número 45
Utilización de varios operadores 46
Utilización de funciones matemáticas internas. . . 48

Capítulo 5: **Utilización de cadenas** **51**
Recorte de cadenas 52
Conexión de cadenas (concatenación). 54
Codificación y descodificación de cadenas. 56
Encriptado y desencriptado de cadenas 59
Sustracción de partes de una cadena 62

Capítulo 6: **Estructuras de control** **67**
La condición if. 68
Más operadores 7

Lógicos 74
Utilización de else 77
Utilización de elseif 79
La condicional switch 82
El bucle while 88
El bucle for 92

Capítulo 7: Uso de los arrays 95
Creación de un array 97
Adición de elementos a un array 99
Acceso a los elementos del array 103
Clasificación de los arrays 106
Transformaciones entre cadenas y arrays 109
Creación de un array desde un formulario 113
Creación de arrays multidimensionales 115

Capítulo 8: Expresiones regulares 117
¿Qué son las expresiones regulares? 118
Definición de un patrón sencillo 119
Metacaracteres 120
Comparación de patrones 122
Definición de patrones más complejos 126
Clases 127
Ejemplos de patrones 128
Referencia regresiva 129
Patrones de búsqueda y sustitución 130

Capítulo 9: Creación de funciones 133
Creación y uso de funciones simples 134
Creación y llamada a funciones que admiten argumentos 138
Creación y uso de funciones que devuelven un valor 141
Variables y funciones 146

Capítulo 10: Archivos y directorios 155
Permisos de archivo 156
Escritura en archivos 158
Lectura desde archivos 163
Directorios 168
Manipulación de cargas de archivos 173
Renombrar y borrar archivos y directorios 177

Capítulo 11: Bases de datos 183
Conexión y creación de una base de datos 185
Creación de una tabla 189
Envío de datos a una base de datos 193
Recuperación de datos de una base de datos 197

Capítulo 12: *Cookies* 201
Creación y lectura de *cookies* 203
Adición de parámetros a una *cookie* 207
Eliminación de una *cookie* 210

Capítulo 13: **Creación de aplicaciones web** 213
Uso de include() y require() 214
Determinación de la fecha y la hora 219
Utilización de las cabeceras HTTP 225
Envío de un mensaje de correo electrónico 229

Capítulo 14: **Depuración** 233
Errores comunes 234
Informes de errores y registro 237
Depuración del trabajo 241
Uso de la instrucción die 244

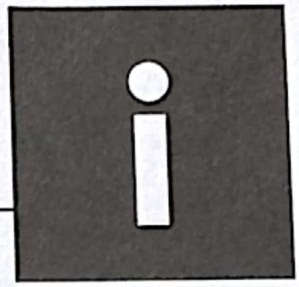
Apéndice A: **Instalación y configuración** 249
Instalación de PHP y Apache en un servidor Linux 250
Instalación de PHP y Apache en un servidor Windows 2000 253
Configuración 254

Apéndice B: **Seguridad** 225
Criptografía y SSL 256
Escritura de código PHP seguro 257
Recursos de seguridad 258

Apéndice C: **Recursos sobre PHP** 259
El manual de PHP 260
Sitios web y grupos de noticias 261
Recursos de bases de datos 263
Materias avanzadas 264
Tablas 266

Índice alfabético 269

INTRODUCCIÓN



La World Wide Web puede ser un lugar misterioso. Se crean e implementa tecnologías nuevas, a menudo con muy poca o ninguna documentación y soporte. Es fácil agobiarse al enfrentarse con acrónimos que representan sistemas que cambian prácticamente a diario.

Un ejemplo de esto es el crecimiento que ha experimentado el software de código fuente abierto: programas gratuitos disponibles para su distribución y mejora. El modelo de código fuente abierto más popular es el sistema operativo Unix y, en particular, Linux. Sin embargo, mientras estos programas de código fuente abierto crean productos estables y muy útiles, por desgracia pueden ser difíciles de aprender. La ausencia de manuales para principiantes y guías sencillas con instrucciones paso a paso, impiden que estas tecnologías sean tan populares como merecen. PHP, un lenguaje de *scripting* web de código fuente abierto, es otra herramienta útil, a pesar de su valor y sencillez de uso en general, que ha permanecido relegada hasta ahora.

Este libro pretende ayudarle en el aprendizaje de PHP, proporcionándole una sólida comprensión de sus fundamentos e indicándole dónde puede encontrar más información. Más que como una guía de referencia de programación, este libro le proporciona el conocimiento que necesita para empezar a crear sitios web dinámicos y aplicaciones web utilizando PHP.

¿Qué es PHP?

PHP era el acrónimo de *Personal Home Page* cuando fue creado en 1994 por Rasmus Lerdorf para hacer un seguimiento de los visitantes de su currículum *online*. Como su utilidad y sus capacidades comenzaron a crecer (a la vez que empezaba a ser utilizado en más situaciones profesionales), su significado cambió por el de *PHP: Hypertext Preprocessor* (PHP: preprocesador de hipertexto). (Básicamente, esta definición viene a significar que PHP manipula los datos antes de que estos se conviertan en HTML, el lenguaje de marcado de hipertexto.)

De acuerdo con el sitio oficial de PHP, que puede encontrar en www.php.net (véase la Figura i.1), PHP es un lenguaje de *scripting* del lado del servidor, de plataforma cruzada e incrustado en HTML. Esto puede sonar algo pretencioso, pero es bastante significativo y cierto cuando lo analizamos por partes.

En primer lugar, "del lado del servidor" significa que todo el código PHP se produce en el servidor (en oposición a cuando nos referimos a "del lado del cliente", que es la computadora con la que se visualiza el sitio web). Un servidor es, simplemente, una computadora especial que alberga las páginas que vemos cuando acudimos a una dirección desde nuestro navegador (por ejemplo, Netscape Navigator o Internet Explorer). Posteriormente en esta introducción explicaremos más detalladamente este proceso (consulte la sección "¿Cómo funciona PHP?").

Cuando declaramos que PHP es de "plataforma cruzada" estamos indicando que podemos utilizarlo en computadoras que ejecutan Unix, Windows NT, Macintosh, OS/2 y otros sistemas operativos. De nuevo, estamos hablando del sistema operativo del servidor, no del cliente. PHP no sólo puede funcionar en casi cualquier sistema operativo, sino que, a diferencia de otros lenguajes de programación, puede trasladar su trabajo de una plataforma a otra con muy pocos o ningún cambio en absoluto. Como es lógico, también puede desarrollar PHP en cualquier sistema operativo, al igual que sucede con HTML.

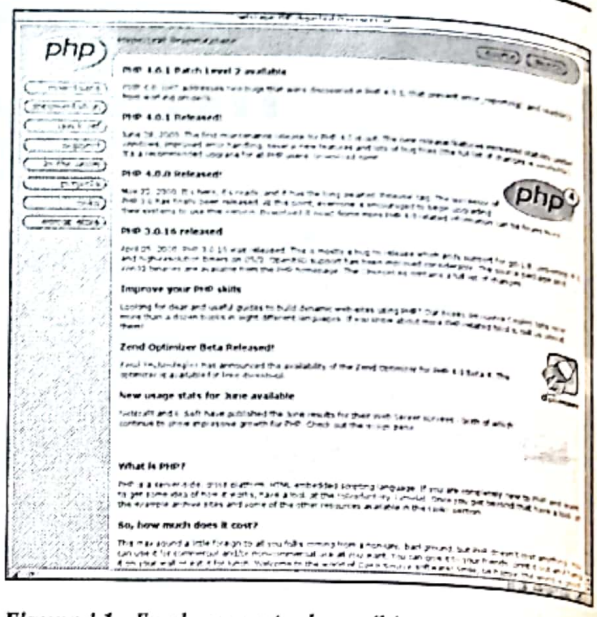


Figura i.1. *En el momento de escribir este libro, éste es el aspecto actual del sitio web oficial de PHP, ubicado en www.php.net. Naturalmente, éste debería ser el primer sitio al que tendría que acudir en relación a cualesquiera cuestiones y curiosidades relacionadas con PHP. Contiene la última edición del manual de PHP con los comentarios añadidos por otros usuarios, así como enlaces a otros muchos recursos.*

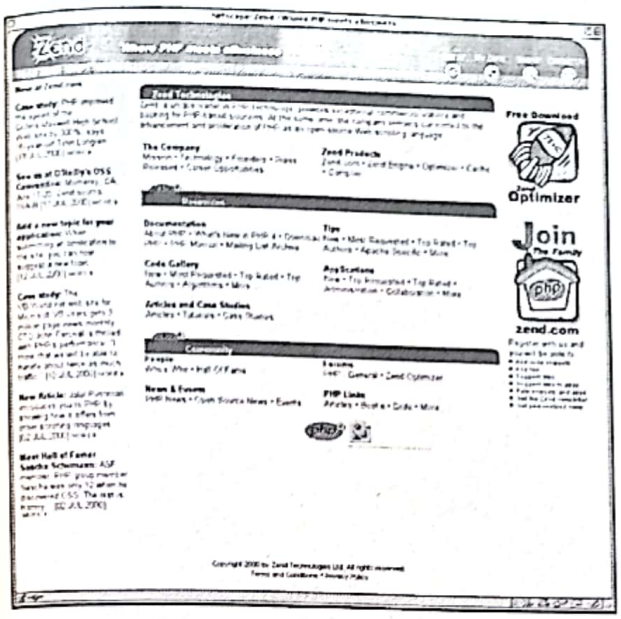


Figura i.2. Página principal de Zend, creadores de la programación que es el corazón de PHP 4. Aquí puede encontrar información detallada sobre la última versión de PHP.

Al decir que PHP está "incrustado en HTML" queremos decir que puede colocar el código PHP en su página HTML: HTML es el código con el que se construyen todas las páginas web. Por consiguiente, la programación PHP actual sólo puede ser ligeramente más complicada que la codificación manual en HTML.

Por último, PHP es un lenguaje de *scripting*, en oposición a un lenguaje de programación. Esto significa que PHP está diseñado para hacer algo después de que se produce un evento; por ejemplo, cuando un usuario envía un formulario o acude a un URL (*Uniform Resource Locator*, Localizador de recursos uniforme; el término técnico que utilizamos para referirnos a una dirección web). Los lenguajes de programación como Java, C y Perl se utilizan para crear aplicaciones únicas, que pueden o no estar involucradas con la Web. El ejemplo más popular de lenguaje de *scripting* es JavaScript, que normalmente manipula los eventos que se producen dentro del navegador, aspecto en el que es similar a PHP, aunque se trata de una tecnología del lado del cliente. Otra forma de referirse a los distintos tipos de lenguajes es utilizar el término "interpretado" para los lenguajes como PHP y JavaScript, que no pueden actuar por sí mismos, y de "compilado" para los lenguajes como C y Java, que sí pueden.

PHP va actualmente por la versión 4, aunque, como esta versión es relativamente nueva, muchos servidores continúan utilizando la versión 3.x. Este libro está dedicado especialmente a PHP 4, pero las diferencias entre las dos versiones son insignificantes en cuanto a la programación general se refiere. La mayoría de las ventajas de la versión nueva se encuentran en su rendimiento mejorado. Puede encontrar más información sobre PHP 4.0 en www.zend.com, la mente que está detrás del núcleo de PHP 4.0 (véase la Figura i.2).

¿Por qué utilizar PHP?

Simplemente porque PHP es mejor, más rápido y fácil de aprender que las alternativas. Al diseñar sitios web, las principales alternativas a PHP son: simplemente HTML; los *scripts* CGI (*Common Gateway Interface*, Interfaz de pasarela común), normalmente, pero no necesariamente, escritos en Perl; ASP (*Active Server Pages*, Páginas de servidor activas); y JSP (*Java Server Pages*, Páginas de servidor Java). JavaScript no es realmente una alternativa a PHP (o viceversa), al ser JavaScript una tecnología del lado del cliente y al no poder utilizarse para crear páginas HTML de la misma forma a como las crean PHP o CGI.

La ventaja que PHP tiene sobre el HTML básico es que este último es un sistema limitado que no permite ninguna flexibilidad o sensibilidad. Los visitantes que acceden a las páginas HTML ven páginas simples sin ningún nivel de personalización o comportamiento dinámico. Con PHP puede crear páginas originales y excitantes en base a cualesquiera factores que quiera considerar (por ejemplo, la hora del día o el sistema operativo del usuario). PHP también puede interactuar con bases de datos y archivos, manipular correo electrónico y hacer otras muchas cosas que HTML no puede.

Los *webmasters* saben de antiguo que HTML, por sí solo, no produce sitios web atractivos y duraderos. Con este fin, las tecnologías del lado del servidor, como los *scripts* CGI, han ido ganando mayor popularidad. Estos sistemas permiten a los diseñadores de páginas web crear aplicaciones web generadas dinámicamente, teniendo en cuenta cualquier elemento que el programador desee. A menudo, en la manipulación de bases de datos, estos avanzados sitios pueden, incluso, actualizarse y mantenerse más fácilmente que las páginas HTML estáticas.

Por tanto, la pregunta es la siguiente: ¿por qué un diseñador web debería utilizar PHP en lugar de CGI, ASP o JSP para crear un sitio web dinámico? En primer lugar, PHP es más rápido de programar y más rápido de ejecutar que los *scripts* CGI. No entraremos en demasiados detalles sobre la ejecución de estos distintos procesos (ni abriremos un debate sobre el asunto), pero sí en los suficientes para decir que, en comparación con otros lenguajes de programación, PHP es más fácil de aprender y utilizar. Las personas (quizá como usted) sin ninguna experiencia en programación podrán escribir *scripts* PHP con facilidad después de leer este libro.

En comparación, ASP requiere un cierto conocimiento de VBScript, y CGI requiere Perl (o C), y ambos son dos de los lenguajes más completos, y, por consiguiente, más difíciles de aprender.

En segundo lugar, PHP se escribió específicamente para la creación de páginas web dinámicas, mientras que Perl (y VBScript y Java) no, de lo que se infiere que PHP puede realizar ciertas tareas más rápida y fácilmente que sus alternativas.

Sin embargo, nos gustaría dejar claro que mientras sugerimos que PHP es definitivamente mejor para ciertas cosas (especialmente para aquellas para las que fue creado) que CGI o ASP, PHP no es mejor lenguaje de programación que Java o Perl, en tanto en cuanto no puede hacer cosas que estos dos lenguajes sí pueden realizar.

Nuestro último argumento para convencerle de que aprenda PHP es que una vez que lo haya conseguido, y teniendo en cuenta que su popularidad sigue creciendo (ya está utilizándose en casi tres millones de sitios web), se encontrará en una posición muy avanzada en la curva de aprendizaje de esta última y "siguiente gran cosa" del mundo de la tecnología Internet.

¿Cómo funciona PHP?

PHP es un lenguaje del lado del servidor, lo que significa que el código que escribe en PHP reside en una computadora *host* que sirve páginas web a los navegadores.

Cuando un visitante acude a un sitio web (por ejemplo, www.DMCinsights.com), su proveedor de servicios de Internet dirige su solicitud al servidor que aloja la información de www.DMCinsights.com.

El servidor lee el código PHP y lo procesa de acuerdo con la dirección que ha escrito. En este ejemplo, el código PHP le dice al servidor que envíe los datos de la página web apropiada a su navegador en forma de código HTML (véase la Figura i.3). En resumen, PHP crea sobre la marcha una página HTML basándose en los parámetros de nuestra elección, y el servidor, por tanto, no contiene páginas HTML estáticas.

Esto difiere de un sitio generado con HTML en que cuando se realiza una solicitud, el servidor simplemente envía los datos HTML al navegador sin que se produzca ninguna interpretación por el lado del servidor (véase la Figura i.4). Para el navegador del usuario puede o no haber una diferencia obvia en el aspecto de home.html y home.php, pero el cómo llegamos a este punto se verá críticamente alterado. La mayor diferencia al utilizar PHP es que puede tener al servidor generando automáticamente el código HTML. Por ejemplo, podría presentarse información diferente si es lunes a si es martes, o si el usuario ha visitado o no la página con anterioridad. La creación dinámica de páginas web es lo que distingue los sitios estáticos y menos atractivos de los más interesantes y, por tanto, más visitados e interactivos.

La principal diferencia entre utilizar PHP y utilizar el más rígido HTML es que cada cosa que hace PHP, la hace en el servidor y después envía la información apropiada al navegador. Este libro abarca cómo utilizar PHP para enviar los datos correctos al navegador.

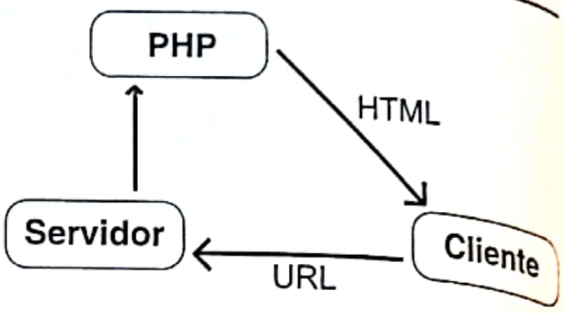


Figura i.3. *Este gráfico muestra cómo funciona el proceso entre un cliente, el servidor y el módulo PHP (una aplicación añadida al servidor para incrementar su funcionalidad) para enviar de regreso código HTML al navegador (aunque en términos muy simples). Todas las tecnologías del lado del servidor (ASP, por ejemplo) utilizan algún tipo de módulo de terceros en el servidor para procesar los datos que se enviarán de regreso al cliente.*

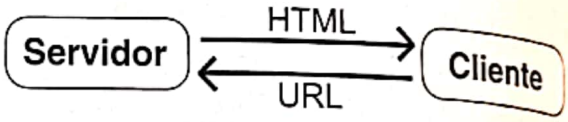


Figura i.4. *Compare esta relación directa de cómo un servidor funciona con HTML básico con la relación que aparece en la Figura i.3. Esto explica por qué puede ver las páginas HTML en el navegador de su computadora sin que sea preciso "servirlas"; pero a las páginas generadas dinámicamente se accede a través de un servidor que manipula el proceso.*

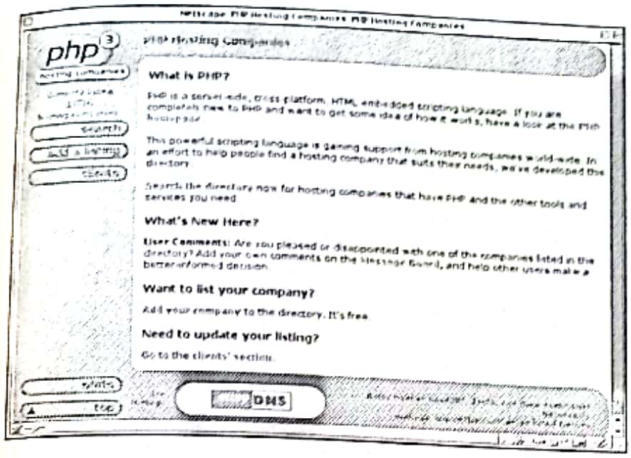


Figura i.5. La información relativa a la obra buscada. Desde la página principal de PHP (véase la Figura i.1), o directamente, puede acceder a host.php.net, un sitio donde aparecen los ISP que ofrecen PHP en sus servidores.

¿Qué necesita?

El requisito más importante para trabajar con PHP (puesto que se trata de un lenguaje de *scripting* del lado del servidor) es acceder al servidor con PHP. Considerando la popularidad de PHP, lo más probable es que su ISP (*Internet Service Provider*, Proveedor de servicios de Internet) o *host* web tenga disponible para usted esta opción en sus servidores. Para asegurarse, póngase en contacto con ellos para saber qué tecnología soportan. En el momento de escribir este libro, más de mil ISP y *hosts* web proporcionan soporte PHP (véase la Figura i.5).

La otra opción que tiene es instalar PHP en su propio servidor (normalmente una computadora con Windows NT o Linux, con la aplicación de servidor web necesaria instalada). (Por ejemplo, Apache es una aplicación de servidor web gratuita para sistemas NT y Unix, o puede utilizar Personal Web Sharing con Windows.) En el Apéndice A encontrará una breve descripción de cómo instalar PHP. Si depende de la tarea de utilizar su propio servidor PHP instalado, puede servirle de consuelo saber que PHP está disponible gratuitamente en el sitio web de PHP (www.php.net) y que se entrega en paquetes de fácil instalación.

El segundo requisito seguramente lo tiene cubierto: debe disponer de algún tipo de editor de textos. El Bloc de notas, WordPad y SimpleText o cualquier otra aplicación *freeware* es suficiente para sus propósitos, mientras que BBEdit, WordPerfect, Word y otras aplicaciones comerciales le ofrecen más funciones de las que necesitará. Si está acostumbrado a utilizar editores con una interfaz gráfica, interfaz también conocida como WYSIWYG (*What You See Is What You Get*, lo que ve es lo que obtiene), como Dreamweaver o GoLive, puede consultar el manual para saber cómo programar con esa aplicación.

En tercer lugar, necesitará algún método de cargar en el servidor los *scripts* que escribe en el editor de textos. Si está trabajando directamente en su propio servidor, simplemente guarde los *scripts* en el directorio adecuado. Sin embargo, si está utilizando un servidor remoto con su ISP o *host* web, necesitará un programa FTP (*File Transfer Protocol*, Protocolo de transferencia de archivos) para enviar el *script* al servidor. Otra opción es hacer *telnet* al servidor remoto y utilizar un editor como Vi o Pico para escribir *scripts* directamente en esta máquina.

Este libro sólo asume que tiene un conocimiento básico de HTML, aunque cuanto más cómodo se sienta con la manipulación de HTML sin la ayuda de una aplicación de creación web como Dreamweaver, GoLive, FrontPage o PageMill, más sencilla le resultará la transición a PHP. Todo programador acudirá a una referencia de HTML en algún momento, con independencia de lo que sepa, por lo que le instamos a que tenga a mano un buen libro de HTML. Una introducción semejante a HTML la constituye el libro **Guía de aprendizaje HTML 4**, también editada por Prentice Hall.

Si bien es cierto que no es necesaria experiencia en programación, sí puede acelerar su aprendizaje el hecho de tenerla, a la vez que le ayudará a ver las numerosas similitudes entre, por ejemplo, Perl y PHP o JavaScript y PHP.

Script i.1. Los mejores editores de textos visualizan automáticamente los números de línea mientras trabaja. ¡Usted nunca debe incluir estos números!

```
script
1  <?php
2  print ("Hello, World!");
3  ?>
```

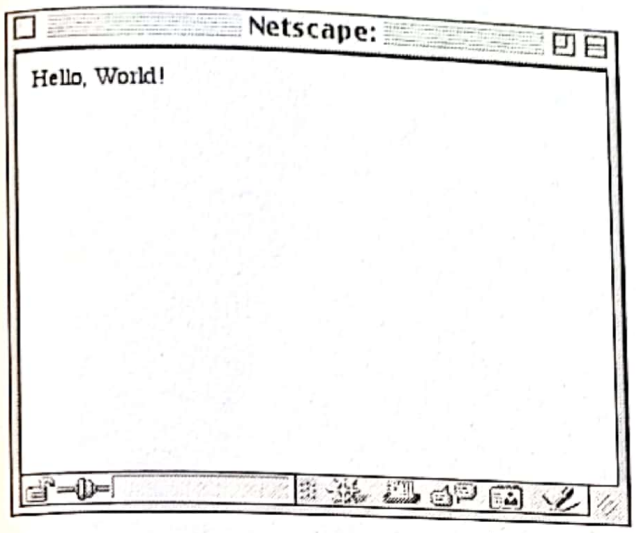


Figura i.6. Esto es lo que verá en la ventana del navegador. No hay diferencia si utiliza Netscape Navigator o Internet Explorer (o cualquier otro navegador) en Macintosh, Windows, Linux o cualquier otro sistema operativo para los propósitos de este libro.

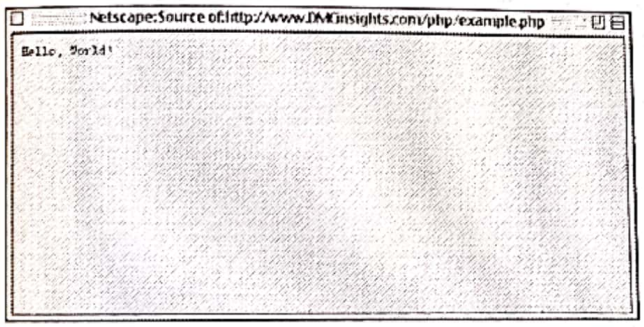


Figura i.7. Seleccione Origen de página en el menú Ver de Netscape, o Código fuente en el menú Ver de Internet Explorer para ver el código HTML que recibe el navegador. En este caso, sólo recibe el texto "Hello, World!".

Acerca de este libro

Con este libro intentamos acercarle a los fundamentos de la programación con PHP y a algunas de las funciones más avanzadas que acabará utilizando, sin abrumarle con los detalles. Hemos utilizado las siguientes convenciones para ello:

Las instrucciones paso a paso le indicarán qué código tiene que añadir y en qué lugar. El texto específico que debe escribir aparece con un tipo de letra distinto para distinguirlo del cuerpo de texto normal. Por ejemplo:

```
<?php print ("Hello, World!"); ?>
```

El código PHP también se escribirá en forma de *script* completo con las líneas numeradas a modo de referencia (véase el *Script* i.1). Usted no debe insertar estos números, o el *script* no funcionará. Le recomendamos utilizar un editor de textos que visualice automáticamente los números de línea, lo que le ayudará a depurar el código (consulte el Capítulo 14).

En estos bloques de código también aparece resaltado en negrita las partes específicas objeto de explicación en las instrucciones paso a paso, para que preste atención al material nuevo o relevante. Igualmente, donde es apropiado, también se mostrará el aspecto de la ventana del navegador (véase la Figura i.6). Una última cosa que también verá en este libro es el propio código fuente HTML (véase la Figura i.7), al que puede acceder seleccionando Ver, Origen de página en Netscape Navigator, o seleccionando Ver, Código fuente en Internet Explorer. La diferencia es significativa entre estas dos particulares figuras, entendiendo que la Figura i.7 muestra el texto que recibe el navegador y la Figura i.6 muestra cómo el navegador interpreta ese texto. Utilizando PHP crearemos el texto que se envía al navegador.

Puesto que la columna de este libro es más estrecha que la pantalla de un editor de textos convencional, algunas líneas de código PHP aquí impresas aparecen divididas, cuando en un editor de textos aparecerán en una sola línea. Una pequeña flecha gris indica cuándo se produce este tipo de ruptura. Por ejemplo:

```
<HTML><Head><TITLE>First PHP Script
→ </TITLE></HEAD><BODY>
```

En sus *scripts* debe utilizar una línea, o se producirán errores cuando los ejecute. (La flecha gris no se utiliza en los *scripts* que están numerados.)

Mientras le demostramos las nuevas funciones y técnicas, le explicaremos el por qué y el cómo de las cosas. Tenemos la esperanza de que si lee mientras prueba una función lo entenderá mejor. Si aun así algo queda confuso, este libro contiene referencias a sitios donde puede encontrar respuestas a las preguntas que se pueda hacer (consulte el Apéndice C).

Debido a la naturaleza de cómo trabaja PHP, debe entender que existen tres vistas esenciales de cada *script*: el propio código PHP, el código que se envía al navegador (principalmente HTML), y lo que el navegador muestra al usuario final. Mostraremos estas tres vistas siempre que sea posible, dependiendo de las restricciones de espacio, dando prioridad al propio PHP.

Código fuente del libro

En el "Área de descarga" de la sección de "Informática" del sitio web de Prentice Hall (www.pearsoneducacion.com) encontrará un archivo en formato zip con el código fuente de los *scripts* del libro. Sin embargo, a pesar de la comodidad que esto puede suponer, le animamos a que escriba los *scripts* con el objetivo de que se familiarice con la estructura y sintaxis de PHP.

CÓMO EMPEZAR CON PHP

1

Siempre que se aborda un nuevo lenguaje de programación debería comenzarse por la comprensión de la sintaxis que se va a emplear, y esto será lo que trataremos en este capítulo. Primero abordaremos lo fundamental, pero también incluiremos recomendaciones que mejorarán el trabajo con el tiempo. Los fallos al seguir las reglas de la sintaxis es la causa más común de los errores en los códigos.

Al final de este capítulo habrá escrito y ejecutado con éxito los primeros *scripts* de PHP.

Sintaxis básica

Para crear la primera página de PHP, comenzará igual que si estuviese creando un documento HTML.

Existen dos diferencias básicas entre un documento HTML estándar y uno de PHP. La primera es que los *scripts* de PHP se deben guardar con la extensión .php (por ejemplo, index.php). La segunda es que puede envolver el código PHP con las etiquetas <?PHP y ?>, para indicar que es PHP, en oposición a HTML.

Cómo crear un script PHP nuevo

1. Abra un editor de textos, como SimpleText, WordPad o cualquier otra aplicación de su agrado.
2. Selecccione Archivo, Nuevo, para crear un documento en blanco.
3. Escriba <HTML> <HEAD> <TITLE>First script → PHP</TITLE> </HEAD> <BODY> (véase el *Script* 1.1).

   Puede colocar cada elemento o grupo de elementos en la misma línea para ordenarla.
4. Escriba <?PHP en la línea.
5. Pulse Intro para crear una línea nueva y escriba ?>.
6. Escriba </BODY></HTML>.
7. Seleccione Archivo, Guardar como. En el cuadro de diálogo que aparecerá, seleccione el formato Sólo texto (o ASCII).
8. Elija la ubicación donde le gustaría guardar el *script*.
9. Guarde el *script* como first.php.

Script 1.1. *Ésta es la estructura más básica de un documento HTML, con las etiquetas PHP insertadas en la sección del cuerpo. Todos los scripts PHP deben usar algún formulario de etiquetas PHP, para indicar al servidor que debe procesarlo como PHP. No hay nada en el interior que limite el trato como PHP, mientras que todo el exterior se envía al navegador como HTML estándar.*

script

```
1  <HTML>
2  <HEAD>
3  <TITLE>First PHP Script</TITLE>
4  </HEAD>
5  <BODY>
6  <?PHP
7  ?>
8  </BODY>
9  </HTML>
```

- Verifique con el ISP las extensiones de archivo que puede utilzar para los documentos PHP. En este libro emplearemos .php, aunque el usuario también puede usar .phtml. Los servidores que aún funcionan con la versión 3 de PHP utilizan normalmente .php3 como extensión predeterminada. Una extensión de archivo le indica al servidor cómo tiene que tratar el archivo: file.php pasará por el módulo PHP, file.asp se procesa como ASP y file.html es un documento HTML estático.
- También puede comprobar con el ISP si las etiquetas cortas (usando <? y ?> en lugar de <?PHP y ?>) o las etiquetas ASP (<% y %>) son aceptadas. Los programas como Macromedia Dreamweaver trabajan mejor con las páginas PHP si emplea las etiquetas ASP.

Script 1.2. Ya que lo hemos guardado en un archivo diferente, también cambiaremos la línea del título HTML cuando le añadamos la función phpinfo().

```
script
1  <HTML>
2  <HEAD>
3  <TITLE>Test PHP Script</TITLE>
4  </HEAD>
5  <BODY>
6  <?PHP
7  phpinfo();
8  ?>
9  </BODY>
10 </HTML>
```

Envío de datos al navegador

Ahora que ya ha creado el primer *script* de PHP, puede hacer algo más. Como dijimos en la introducción, PHP indica al servidor qué datos debe enviar al navegador web del usuario. Al arrancar, el usuario empleará la función phpinfo() para crear los datos. Cuando se llama a esta función, se envía una tabla de información al navegador web, detallando las especificaciones de la instalación PHP en ese servidor en concreto.

Cómo añadir la función phpinfo() al script

1. Abra el *script* first.php en el editor de textos, si aún no lo está.
2. Coloque el cursor entre las etiquetas PHP (<?PHP y ?>), y cree una línea nueva pulsando Intro.
3. Escriba phpinfo();.
4. Cambie el título de la página reemplazando First por Test en la línea 3 del HTML (véase el *Script* 1.2).
5. Guarde el *script* como test.php.

Cada instrucción del código PHP debe finalizar con un punto y coma (;).Olvidar esto es un error común. Puede colocar varias instrucciones en una línea, pero siempre separadas por puntos y comas. Sin embargo, no lo recomendamos por motivos de claridad.

Una instrucción PHP es una línea de código ejecutable, como print() o phpinfo(). El punto y coma que concluye dichas líneas indica a PHP que continuúe y ejecute el comando. Las líneas de comentarios, las etiquetas PHP, las estructuras de control (condicionales, bucles, etc.), y otras construcciones que trataremos en este libro no necesitan un punto y coma. Cada uno de estos aspectos de PHP no hace nada más que dictar las circunstancias para las instrucciones. Es como decir: la etiqueta PHP sólo indica que se sigue el código PHP; los caracteres de comentario explican el texto, etc. Así, en general, un punto y coma finaliza una acción específica, mientras que no será necesario para las estructuras condicionales.

- Para bien o para mal, PHP es bastante liberal y poco sensible al empleo de mayúsculas o minúsculas; así, funciones integradas como PHPinfo() y PHPINFO() darán los mismos resultados. Más adelante (por ejemplo, en el Capítulo 2), verá ejemplos de casos en los que las mayúsculas o minúsculas serán el motivo de las diferencias. HTML, por el contrario, es completamente sensible a mayúsculas y minúsculas.
- Phpinfo() es un ejemplo de una función integrada estándar de PHP. Para aprender más sobre las funciones que puede crear el usuario, consulte el Capítulo 9.
- Encontrará muy útil tener a mano una copia del archivo test.php. Puede emplearla para comprobar las capacidades PHP de un nuevo servidor o para ver las funciones que soporta, como bases de datos, creación de GIF, etc. También puede utilizarlo para experimentar con extensiones diferentes y aprender cuáles son las que el servidor procesa correctamente y cuáles no.

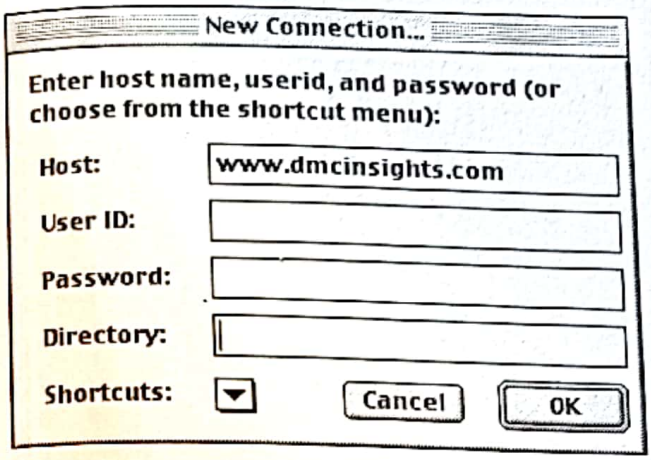


Figura 1.1. Fetch es una popular aplicación FTP para Macintosh, aunque hay otras. Introduzca su nombre de usuario y contraseña que el ISP o el host web le asignaron. Si conoce el directorio donde están almacenados los archivos, también puede introducirlo.

Comprobación del *script*

A diferencia de HTML, que puede probarse en la computadora del usuario empleando un navegador web, para ver el resultado de salida del *script* PHP, hay que guardarlo en un servidor que permita PHP. Si trabaja directamente en un servidor, el *script* ya se encontrará ahí cuando lo guarde. Si crea el *script* empleando un editor de textos en la computadora, necesitará usar FTP (Protocolo de transferencia de archivos) para colocarlo en el servidor. Su ISP o el *host* web deberían proporcionarle acceso a FTP. También necesitará una aplicación FTP, como Fetch (para Macintosh) o WS_FTP (para Windows).

Cómo cargar por FTP el *script* en el servidor

1. Abra la aplicaión FTP.
2. Conéctese al servidor; utilice la dirección, nombre de usuario y contraseña proporcionados por el ISP o el *host* web (véase la Figura 1.1).
3. Busque el directorio de ubicación adecuado para las páginas HTML (por ejemplo, www/ o htdocs/).
4. Guarde el *script* (test.php) en el servidor. (Como norma, la mayoría de aplicaciones FTP guardan las páginas transferidas al servidor con el mismo nombre de archivo que el empleado por el usuario en la computadora para dichos archivos. Si la aplicación FTP le da la opción de especificar el nombre de archivo, use test.php.)

Cómo probar el *script* en el navegador

1. Abra el navegador web.
2. Introduzca el URL del sitio donde ha guardado el *script* (en este caso, http://www.DMCinsights.com/php).
3. Añada /test.php al URL.
4. Pulse Intro. La página se cargará en la ventana del navegador (véase la Figura 1.2).

La función phpinfo() muestra la información del sistema por el módulo PHP que está instalado en el servidor. Es muy manejable para comprobar la nueva instalación PHP, determinando cuáles son las extensiones PHP útiles y grabando las funciones que soporta.

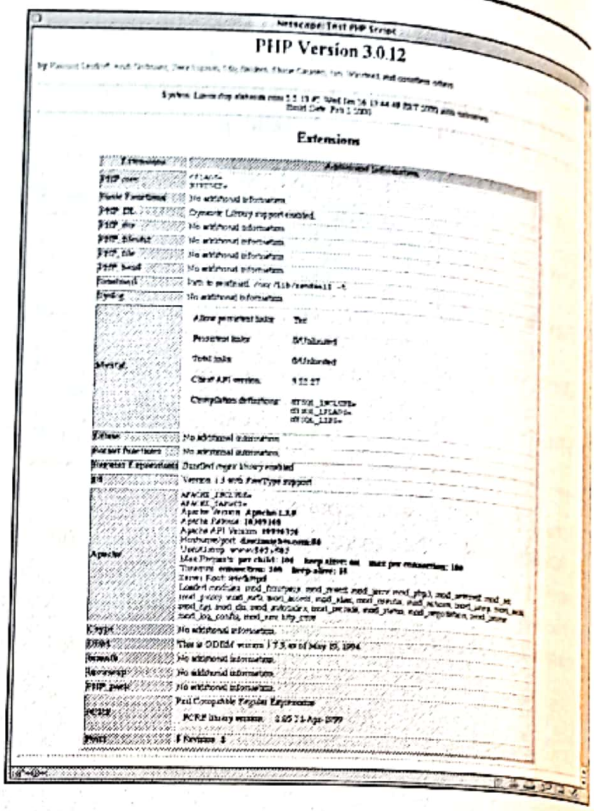


Figura 1.2. Si ve el texto phpinfo(), entonces es que PHP no está correctamente instalado, o la extensión que ha utilizado (.php en nuestro caso) no se ha tratado como un archivo PHP.

CONSEJO

- Algunos editores de textos, como BBEdit, tienen la capacidad FTP integrada, permitiéndole guardar los *scripts* directamente en el servidor.

Script 1.3. Al colocar la instrucción print entre parentésis PHP, tenemos al servidor enviando el saludo "Hello, world!" al navegador. Es lo mismo que si colocamos "Hello, world!" en el código HTML.

```
1  <HTML>
2  <HEAD>
3  <TITLE> First PHP
Script</TITLE>
4  </HEAD>
5  <BODY>
6  <?PHP
7  print("Hello, world!");
8  ?>
9  </BODY>
10 </HTML>
```

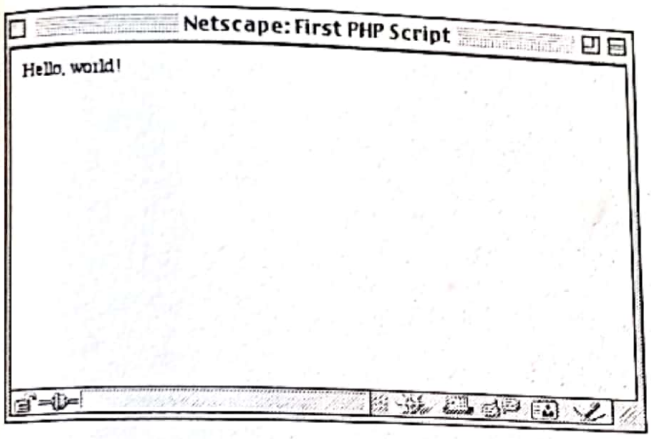


Figura 1.3. Si el script se ha ejecutado correctamente, su navegador debería parecerse a esto (no es muy interesante, pero funciona).

Envío de texto al navegador

PHP no le sería muy útil si todo lo que pudiese hacer fuese ver las funciones PHP que ha instalado. Lo empleará con mucha frecuencia para enviar información al navegador con el formulario de sólo texto y etiquetas HTML. Para hacerlo, tendrá que emplear la función print().

Cómo imprimir un mensaje sencillo

1. Abra first.php en el editor de textos.
2. Coloque el cursor entre los parentésis PHP y cree una línea nueva pulsando Intro.
3. Escriba print ("Hello, world!"); (véase el *Script* 1.3).

   Imprimir la frase "Hello, world!" es el primer paso en la mayoría de las aplicaciones que aborde. Aunque es una razón muy trivial para emplear PHP, la utilizaremos por motivos de demostración.
4. Guarde el *script*.
5. Descargue el *script* en el servidor y pruébelo en el navegador (véase la Figura 1.3).

CONSEJOS

- Existen funciones diferentes que puede emplear para enviar texto al navegador, como echo() y printf(). Echo es un sinónimo virtual de imprimir, por lo que no se va a tratar más en este libro. Trataremos sobre printf() en el Capítulo 13.
- Puede emplear print() con o sin los paréntesis, pero tiene que usar siempre las comillas; por ejemplo, puede escribir print "Hello, world!";. Aunque en este libro utilicemos paréntesis para separar las instrucciones, la mayoría de los programadores no los usan.
- La sugerencia es que busque un estilo que sea de su agrado, y lo siga posteriormente.

El fallo en el uso de unas comillas de apertura o de cierre, de un paréntesis o el descuido al incluir los puntos y comas necesarios en cada instrucción, son los errores más comunes cuando se emplea la instrucción print(). Si el *script* no se ejecuta de modo correcto, compruebe estos puntos.

Script 1.4. Al usar la función print, puede enviar etiquetas HTML junto con el texto al navegador, donde se aplicará el formato.

```
script
1  <HTML>
2  <HEAD>
3  <TITLE>First PHP Script</TITLE>
4  </HEAD>
5  <BODY>
6  <?PHP
7  print("<B><CENTER>Hello, world!
   </CENTER></B>");
8  ?>
9  </BODY>
10 </HTML>
```

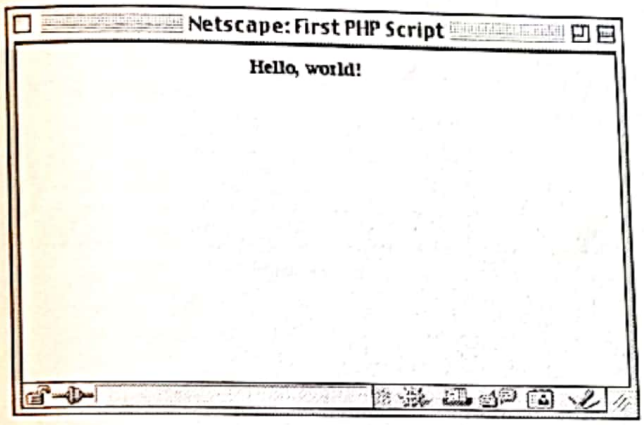


Figura 1.4. *La nueva versión de nuestro script, con un poco más de decoración y atractivo. Desde PHP puede enviarse cualquier etiqueta HTML; sólo debe asegurarse de seguir las convenciones HTML (por ejemplo, utilizar etiquetas de cierre).*

Envío de HTML al navegador

A los que primero descubrieron HTML, la visualización de texto simple en un navegador web no les agradará. De hecho, HTML se creó para hacer un texto plano más atractivo y útil. Ya que HTML trabaja añadiendo etiquetas al texto, puede emplear PHP para enviar etiquetas HTML junto con otros datos al navegador.

Cómo enviar HTML al navegador utilizando PHP

1. Abra el *script* first.php en el editor de textos.
2. Edite el texto "Hello, world!" en la línea 7, añadiendo etiquetas B y CENTER para que se lea: print("<B><CENTER>Hello, world!</CENTER> ↪ </B>"); (véase el *Script* 1.4).
3. Cargue el *script* en el servidor y actualice la página en el navegador (véase la Figura 1.4).

CONSEJO

- Las etiquetas HTML que necesitan comillas (por ejemplo, <FONT COLOR="#000000">) causarán problemas cuando se impriman en PHP, ya que la función print() también utiliza comillas. Para evitar complicaciones, emplee la barra inclinada inversa (\). En el ejemplo, la instrucción que usaría sería: print"<FONT COLOR=\"#000000\">";. Al evitar las comillas en la instrucción print(), PHP imprimirá la marca en lugar de interpretarla, como al principio y final de esta instrución. Trataremos más ejemplos de este tipo en el libro.

Uso de espacios en blanco en PHP (y HTML)

A los que comprendan el manejo del código HTML, los espacios en blanco (líneas de espacios en blanco, tabulaciones y otros espacios extra) del código les puede ayudar a mantener el desarrollo de una programación ordenada, mientras no afecten a lo que se visualiza en el navegador. Al insertar líneas de espacios en blanco entre partes del código, colocamos elementos anidados una tabulación por delante de su predecesor, y al espaciar el código, el *script* parece estar más organizado, facilitando la iniciación de la aplicación y la posterior depuración. Por tanto, animamos al uso juicioso de los espacios en blanco en el trabajo, para utilizarlo en PHP y en HTML.

Recuerde que este libro considera tres áreas del desarrollo web: los *scripts* de PHP, los datos que PHP envía al navegador web (mayormente en un formulario de HTML), y cómo un navegador web visualiza o interpreta dichos datos. Trataremos brevemente la emisión de espacios en blanco para cada uno de ellos.

Cuando programe PHP, deberá comprender que los espacios en blanco se ignoran generalmente (no universalmente). Cada línea de espacio en blanco (haya una o varias en una fila) es completamente irrelevante para el resultado final. Asimismo, las tabulaciones y los espacios normalmente son ilógicos en PHP.

El código HTML fuera de PHP (véase el *Script* 1.4, líneas 1 a 5), puede espaciarse mientras esté dentro de PHP, pero para dirigir el esquema de HTML que está generando PHP (véase el *Script* 1.4, línea 7), debe emplear carcteres especiales.

Script 1.5. Cuando añadimos líneas de espacios en blanco, no sólo no afectan a la apariencia de la página en el navegador, sino que hacen que el código sea más legible. Por cada \n que hemos incluido en la instrucción print, se ha insertado un salto de línea en la fuente HTML (no debe confundirse con <BR>, que crea un retorno de cómo HTML se visualizará en el navegador).

script

```
1  <HTML>
2  <HEAD>
3  <TITLE>First PHP Script</TITLE>
4  </HEAD>
5  <BODY>
6  <?PHP
7
8  print("<B><CENTER>Hello, world!
   </CENTER></B>\n");
9
10 ?>
11 </BODY>
12 </HTML>
```

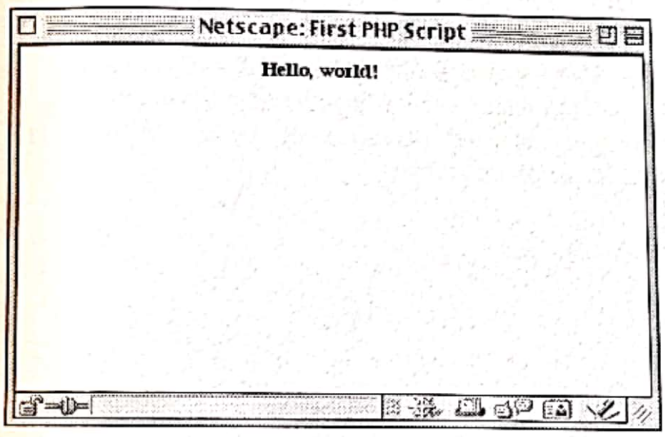


Figura 1.5. *Nuestra página sigue pareciendo la misma en el navegador con la línea nueva que sin ella, pues \n no es una etiqueta HTML y las líneas en blanco en el código PHP (líneas 7 y 9) no se envían al navegador.*

Cómo cambiar los espacios del PHP y los datos enviados al navegador

1. Abra first.php en el editor de textos.
2. Inserte una línea nueva antes y después del comando print, pulsando Intro en los lugares adecuados.

   Las líneas nuevas en PHP sólo sirven para añadir enfoque y claridad al *script*.

3. Al final del comando print (ahora en la línea 8), añada \n, entre comillas (véase el *Script* 1.5).

   La combinación con el carácter \n, envía un comando al navegador web para que dé comienzo la nueva línea de origen HTML. Piense que es el equivalente a pulsar la tecla Intro.

4. Guarde el *script*, cárguelo en el servidor y visualícelo después con el navegador (véase la Figura 1.5).

Aunque las series de pasos anteriores puedan hacer que los PHP y HTML sean más legibles, recuerde que los espacios en blanco adicionales no afectarán a la apariencia de la página en el navegador web (véase la Figura 1.5). Para hacerlo, debe emplear las etiquetas HTML (el código para crear un espacio en blanco en el navegador es   y el equivalente HTML de presionar Intro es <BR>).

CONSEJOS

- Una excepción a la regla de los espacios en blanco en PHP serían los espacios en blanco de la instrucción print, resultantes de espacios extra que han sido impresos por el navegador. No obstante, HTML también los ignorará.
- Para ver el código que se envía al navegador, use las funciones Ver código fuente u Origen de página integradas en todos los navegadores. Rápidamente sabrá la diferencia entre usar una línea nueva o no (véanse la Figuras 1.6 y 1.7). Puede que sea difícil apreciar los beneficios de los espacios en blanco en un *script* de doce líneas, y esta apreciación será incluso más complicada a medida que aumente el número de líneas.
- Existe una corriente de pensamiento que recomienda condensar el código HTML en un paquete lo más compacto posible, eliminando los espacios superfluos. La idea es aumentar la velocidad de descarga de la página al no transmitir espacios extra. Sin restar méritos a esta intención, hacer esto no sirve para practicar y aprender.

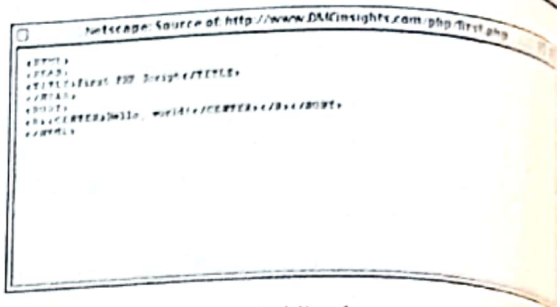


Figura 1.6. *Visualizar el código fuente de una página es una buena forma de determinar dónde reside un problema de formateo. Éste es el código fuente de nuestro script antes de añadir \n y las líneas en blanco.*

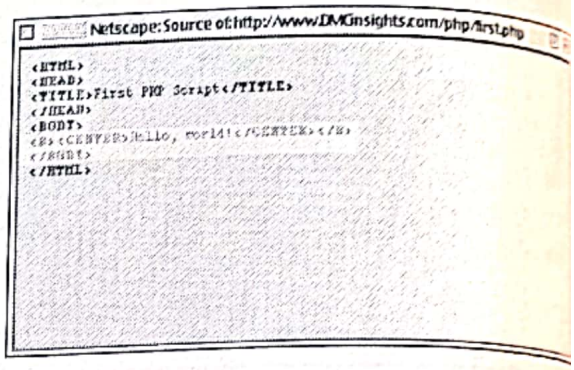


Figura 1.7. *El \n colocado en la instrucción de impresión separa la línea "Hello, world!" de las demás etiquetas HTML. Cuando se envía código HTML más complejo al navegador, el uso de líneas nuevas ayuda a simplificar la visualización del código.*

Script 1.6. Al colocar // o # delante de una línea de código, no será procesada por el servidor.

```
script
1  <HTML>
2  <HEAD>
3  <TITLE>First PHP Script</TITLE>
4  </HEAD>
5  <BODY>
6  <?PHP
7
8  // print("<B><CENTER>Hello, world!
   </CENTER></B>\n");
9
10 ?>
11 </BODY>
12 </HTML>
```

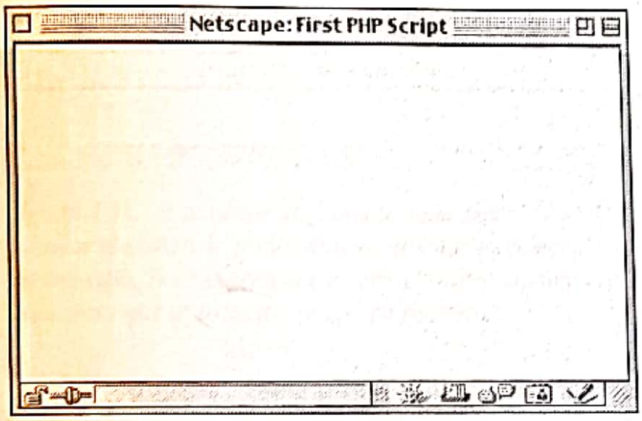


Figura 1.8. *Con el comando de impresión comentado, la página parece no tener dicho comando. Puesto que el servidor no procesó el comando, el navegador nunca recibirá el texto "Hello, world!".*

Adición de comentarios a los *scripts*

Los programadores a veces se dan cuenta de que tomar notas es un salvavidas, cuando meses después tienen que regresar inevitablemente a un proyecto para afinar, copiar o arreglar algún código. Añadir esas notas, o comentarios, al trabajo le ayudará a recordar lo que estaba pensando en ese momento, cosa que puede que no sea evidente con el paso del tiempo. La computadora ignora esos comentarios cuando procesa el *script*. PHP soporta tres métodos para añadir comentarios.

Existen dos modos de comentar una línea de código. Colocando // o # al principio de la línea que desea ignorar. También puede emplear // o # para dar comienzo a un comentario al final de una línea PHP como ésta:

```
print("Hola."); // Un saludo.
```

Cómo comentar una línea de código

1. Abra el *script* first.php en el editor de textos.
2. En la línea 8 y antes del comando print, escriba // o # (véase el *Script* 1.6).
3. Guarde el *script*, cárgelo en el servidor y visualize la página en el navegador web (véase la Figura 1.8).

   Si lo ha hecho bien, no debería ver nada en el navegador. No se alarme. No tiene ningún misterio. El navegador no ha impreso "Hello, world!", porque el PHP nunca se lo indicó, de acuerdo con el signo // o # que haya añadido.

Al utilizar /* antes y */ después de una parte de código, puede hacer que el servidor ignore totalmente una palabra o varias líneas.

Cómo comentar varias líneas de código

1. Abra el *script* first.php en el editor de textos.
2. Elimine el signo # o // que esté antes del comando print().
3. Escriba /* en cualquier parte antes del comando print() de la línea 8, pero después de la etiqueta inicial PHP (<?PHP) de la línea 6.
4. Escriba */ en cualquier parte después del comando print() de la línea 8 (por ejemplo, después del punto y coma), pero antes de la etiqueta PHP de cierre (?>) de la línea 10 (véase el *Script* 1.7).
5. Guarde el *script* en el servidor y visualize la página en el navegador web (véase la Figura 1.9).

Script 1.7. *Aunque pueda parecer superfluo el uso de /* */ para comentar una línea de código, no hay nada malo en hacerlo, y obtendremos los mismos resultados si utilizamos // o # (véanse la Figuras 1.8 y 1.9).*

```
script
1  <HTML>
2  <HEAD>
3  <TITLE>First PHP Script</TITLE>
4  </HEAD>
5  <BODY>
6  <?PHP
7  /*
8  print("<B><CENTER>Hello, world!
   </CENTER></B>\n");
9  */
10 ?>
11 </BODY>
12 </HTML>
```

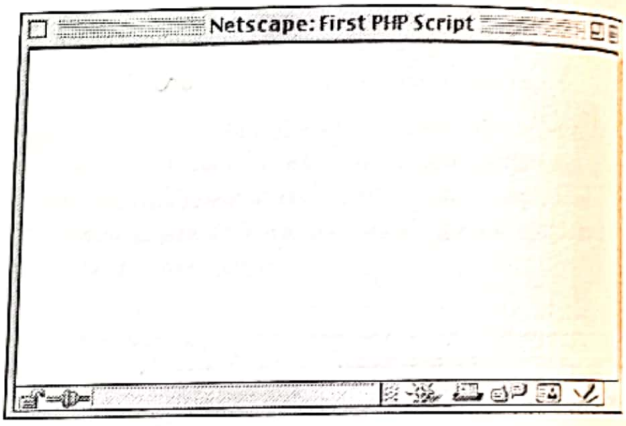


Figura 1.9. *No importa la convención que utilice para comentar el script si la utiliza correctamente: // y # para una línea y /* */ para cualquier número de líneas. (Ésta es la última imagen vacía que aparece en el libro, ¡lo prometemos!)*

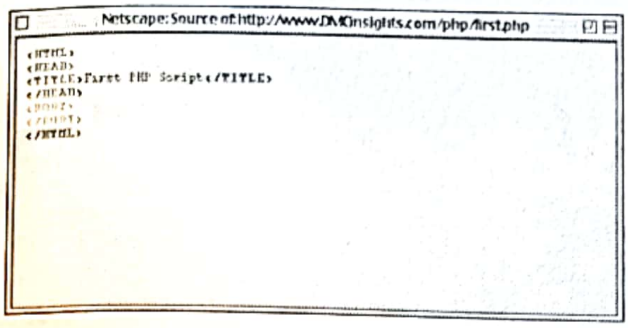


Figura 1.10. Compare este código con el anterior en el que estaba comentado el comando de impresión (véase la Figura 1.7) y verá que el código PHP nunca alcanza el navegador. El código HTML comentado (utilizando <!- - ->) todavía aparece en el código fuente, pero no es interpretado por el navegador.

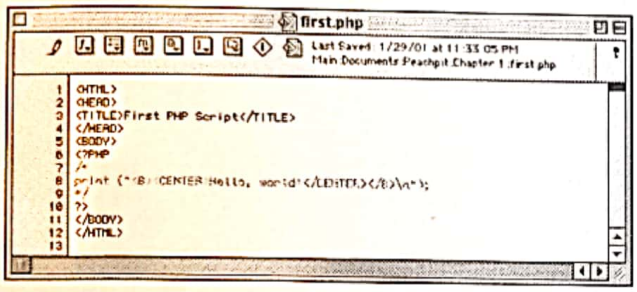


Figura 1.11. Si el editor de textos tiene la posibilidad de colorear el código, la programación se hace más sencilla. En este caso, BBEdit presenta en gris el código comentado, indicando que se trata de un código inactivo.

CONSEJOS

- Puede comentar una línea de código (como en el ejemplo) o varias, empleando el método /* y */. Con // o # sólo podrá anular una línea cada vez.
- Distintos programadores prefieren comentar el código de modos diferentes. Lo más importante es que el usuario encuentre un sistema que haga el trabajo, y que lo siga después. Todos aquellos que programen en JavaScript utilizarán normalmente // y /* */. Los programadores de Perl están más familizarizados con el método #.
- Fíjese que no puede utilizar los caracteres de comentario HTML (<!-- y -->) en PHP para comentar un código. Puede hacer que PHP imprima esos elemntos en un navegador, pero en ese caso, estaría creando un comentario que aparecería en el código de origen HTML de la computadora del cliente (pero no en la ventana del navegador). Los comentarios PHP nunca se habrían hecho, en lo que respecta a dicha computadora.
- Un texto que se comenta en PHP y que no se envía al navegador (véase la Figura 1.10), es un buen lugar para dejar notas que sólo podrá ver el programador.
- Los editores de textos más avanzados, como BBEdit, utilizarán colores para indicar qué código se ha comentado y cuál no (véase la Figura 1.11). Esto puede resultar muy útil cuando se trabaja con *scripts* largos.

VARIABLES 2

En el capítulo anterior ha utilizado PHP para enviar texto sencillo y código HTML a un navegador. En otras palabras, algo para lo que no necesitaba PHP. No se preocupe, en este libro le enseñaremos cómo emplear la instrucción print() en conjunción con otras funciones de PHP, para que haga cosas realmente útiles en el sitio web.

Para saltar de la creación de páginas estáticas y sencillas a las aplicaciones dinámicas y los sitios web interactivos, necesitará manipular datos. Lo que vamos a hacer es enseñarle a lograr esas variables. Las variables son un concepto importante y una herramienta esencial para PHP, además de para JavaScript, Java, Perl o para cualquier otro lenguaje de programación.

Las variables le permiten almacenar y manipular datos temporalmente, dando al lenguaje de programación su verdadero poder. Comprender lo que son, qué tipos lenguaje soportan y cómo se usan es básico para el trabajo. Este capítulo tratará los fundamentos de las variables que se usan en PHP, mientras que en los Capítulos del 4 al 6 veremos lo que puede hacer con los diferentes tipos de variables.

¿Qué son las variables?

Resulta mejor pensar en una variable como en un contenedor de datos. Una vez se han almacenado los datos en su interior (o, concretando, una vez que se le ha asignado un valor), esos datos/variables se pueden alterar, imprimir en el nevagador web (cuando decimos imprimir, queremos decir que se envían, pero es la instrucción print la que efectúa el envío, por lo que ambos términos son correctos), guardar en la base de datos, enviar por correo electrónico, etc.

Las variables son, por naturaleza, flexibles: en ellas puede introducir datos, recuperarlos (sin que afecte al valor), colocar nuevos datos en su interior, y continuar el ciclo hasta donde sea necesario. Pero las variables en PHP también son temporales: sólo existen, es decir, sólo tienen un valor, mientras se utilizan en un *script*. Cuando el usuario esté en una página nueva, esas variables ya no existirán, a no ser que las pase a la nueva página, lo que trataremos en el capítulo siguiente.

Script 2.1. Siempre resulta más adecuado incluir muchos comentarios que pocos, porque lo que parece obvio a la hora de iniciar la programación, puede no serlo tanto cuando tenga que volver a ella al cabo del tiempo.

```
script
1  <HTML>
2  <HEAD>
3  <TITLE>First PHP Script</TITLE>
4  </HEAD>
5  <BODY>
6  <?PHP
7  /* esta página imprime un simple saludo. */
8  print("Hello, World!");
9  ?>
10 </BODY>
11 </HTML>
```

Sintaxis de una variable

En PHP, todas las variables comienzan con el signo del dólar ($), seguido del nombre de la variable. Este nombre debe comenzar con una letra (A-Z, a-z) o con un subrayado (_), seguido por cualquier número de letras, subrayados o números, combinados o no. No puede incluir espacios en el nombre de una variable. En su lugar, normalmente se emplean los subrayados.

Recuerde que las variables permiten el uso de mayúsculas y minúsculas. Por tanto, $variable y $Variable son dos estructuras diferentes, aunque no tendría sentido usar dos variables con unos nombres tan parecidos. Ya verá como pronto asume el hábito de crear nombres de variables que tengan sentido por sí mismas, además de utilizar comentarios para indicar su propósito (véase el *Script* 2.1). Estos hábitos reducirán los errores y harán que la revisión del trabajo sea más fácil. Por ejemplo, $FirstName es mucho más útil que $FN, e incluir un comentario que detalle su propósito, es lo que hará que el trabajo sea lo suficientemente claro. De hecho, puede decidir si $first_name es un nombre de variable mejor que $FirstName, ya que no tiene que escribir letras mayúsculas y las palabras están separadas con claridad. No importa cómo decida el nombre de las variables, lo más importante es que recuerde que la convención empleada debe ser coherente. Esto le ayudará a evitar los errores triviales en la programación.

A diferencia de otros lenguajes, en PHP no tiene que declarar lo que es una variable (declarar una variable es asignarle un tipo), ni iniciarla antes de usarla (iniciarla es crearla). Con PHP, una variable existe y se define la primera vez que la usa.

Tipos de variables

En este libro abordaremos tres extensas categorías de tipos variables de PHP: números, cadenas y arrays.

Hablando técnicamente, PHP divide los números en dos tipos: "enteros" y "de punto flotante" (o "de punto flotante de doble precisión" o "dobles"). Pero para nuestros propósitos nos referiremos a ellos sólo como "números". Dado el modo en que PHP maneja las variables, no afectará a la programación agrupar las dos categorías en una sola.

PHP también usa un tipo de variable al que hace referencia como "objeto", pero éste es un tema más complicado que trataremos en otro libro. Para aprender más sobre los objetos y cómo se emplean, consulte el Apéndice C. Cuando se sienta más cómodo con PHP, descubrirá que aprender más sobre los objetos va a mejorar la habilidad en la codificación, ya que la programación orientada a objetos (OOP), supone un ahorro de tiempo fabuloso.

Números

Hemos combinado los dos tipos de números (enteros y de punto flotante) en un único grupo para facilitar el aprendizaje. Trataremos las difrenecias entre ambos brevemente.

El primer tipo de números (enteros) son los números en sí mismos. Pueden ser positivos o negativos, pero no pueden incluir fracciones ni decimales. Los números que utilizan un punto decimal (aunque sea "1.0") son números de punto flotante. Debe emplear estos números al referirse a las fracciones, ya que el único modo de expresar una fracción en PHP es convertirla en su equivalente decimal. Por lo que "1 1/4" se escribiría como "1.25".

Algunos ejemplos de valores enteros son los siguientes:

```
1
1972
-1
```

A continuación tiene algunos ejemplos de valores de punto flotante:

```
1.0
19.72
-1.0
```

Ya que aquí nos referiremos a ambos números, cualquiera de los anteriores serían considerados como valores numéricos válidos.

Éstos son algunos ejemplos de valores numéricos no válidos:

```
1 1/4
1972a
02.23.72
```

La fracción no es válida porque contiene dos caracteres que no se pueden usar: el espacio y la barra inclinada inversa (/). El segundo elemento no es válido porque emplea números y letras, lo que es aceptable como nombre de variable, pero no como valor de una variable numérica. El tercer ejemplo no es válido porque emplea dos puntos decimales. Si necesitase referirse a alguno de esos valores por alguna otra razón que para efectuar cálculos, puede asignarlos como cadenas.

Cadenas

Una variable es un cadena si consta de caracteres (combinaciones de letras, números, símbolos o espacios) cerrados con dos comillas sencillas (') o dobles ("). Las cadenas pueden contener cualquier combianción de caracteres, incluyendo los nombres de otras variables.

Algunos ejemplos de valores de cadena válidos son éstos:

```
"Hello, World!"
"Hola, $FirstName!"
"1 1/4"
'¡ Hello, World! How are you today?'
"02.23.72"
"1972"
```

Observe que en el ejemplo anterior, tomó un número entero y lo transformó en una cadena situándolo entre comillas. Esencialmente, la cadena contiene los caracteres "1972" porque el número es igual a 1972. Es una pequeña distinción que no causa problemas en el código, mientras pueda efectuar cálculos matemáticos con la cadena "1972", así como con el número.

Algunos ejemplos de valores de cadena no válidos son los siguientes:

```
Hello, World!
"I said, "How are you today?""
```

El primer ejemplo no es válido porque no está dentro de comillas dobles o sencillas. El segundo es una trampa. Tendría problemas si asigna un valor a la cadena, porque una vez que PHP lea la segunda comilla, asume que la cadena ha finalizado en ese lugar, y el texto que sigue causará un error.

Entonces se preguntará, ¿cómo se usan unas comillas dentro de una cadena? Como comentamos en el Capítulo 1, cuando usamos la función print() para crear HTML, puede evitar las comillas colocando una barra inclinada inversa (\) en su lugar.

Para cambiar la cadena a "I said, \" How are you today?\"", tiene que decirle a PHP que incluya esas dos comillas como parte del valor de la cadena y no las trate como si fuesen indicadores de apertura o cierre de la cadena. Así que mientras cualquier combinación de caracteres puede ser incluida en una cadena, se deben evitar los caracteres especiales para que la impresión sea correcta. Además de las comillas dobles, también debería evitar el apóstrofe o comilla sencilla ('), la barra inclinada inversa (\) y el signo del dólar ($).

CONSEJOS

- El beneficio de usar comillas dobles en lugar de sencillas con las cadenas, es que el valor de la variable se imprimirá usando el primero, no el último. Si utiliza comillas sencillas, la línea print 'Hello, $FirstName!'; quedaría como Hello, $FirstName! y se imprimiría en lugar de, supongamos, Hello, Larry! (asumiendo que a $FirstName se le haya asignado el valor Larry). Si evita el signo del dólar dentro de las comillas dobles (print «Hello, \$FirstName!»;) imprimirá de nuevo la variable, pero no su valor (Hello, $FirstName!).

- En el Capítulo 1 vimos cómo se crea una línea nueva imprimiendo el carácter \n. Aunque unas comillas como secuencia de escape se imprimen como comillas, una secuencia de escape con la "n" imprime una línea nueva, con una "r" crea un retorno de carro y con una "t" inserta una tabulación en el código.

Arrays

Abordaremos los arrays más ampliamente en el Capítulo 6, ya que son más complicados que los números o que las cadenas, pero ahora haremos una breve presentación. Como quiera que una cadena o un número contiene al menos un valor, un array puede tener un número de valores asignado. Piense que un array es una lista de valores. En otras palabras, puede colocar varias cadenas y/o números en un array. Incluso puede colocar varios arrays dentro del mismo array.

CONSEJO

- Un array estándar PHP, compuesto por cadenas y números (también conocido como array "indexado" o "vectorial"), también se llama array en Perl. Para crear un array que conste de arrays, debe crear en PHP el equivalente de un *hash* de Perl, también conocido como array "asociativo" o "multidimensional". En PHP, nos referiremos a ambos tipos tan sólo como "arrays".

Arrays

Abordaremos los arrays más ampliamente en el Capítulo 6, ya que son más complicados que los números o que las cadenas, pero ahora haremos una breve presentación. Como quiera que una cadena o un número contiene al menos un valor, un array puede tener un número de valores asignado. Piense que un array es una lista de valores. En otras palabras, puede colocar varias cadenas y/o números en un array. Incluso puede colocar varios arrays dentro del mismo array.

- Un array estándar PHP, compuesto por cadenas y números (también conocido como array "indexado" o "vectorial"), también se llama array en Perl. Para crear un array que conste de arrays, debe crear en PHP el equivalente de un *hash* de Perl, también conocido como array "asociativo" o "multidimensional". En PHP, nos referiremos a ambos tipos tan sólo como "arrays".

Asignación de valores a las variables

Hemos mencionado al principio de este capítulo que no tiene que iniciar o declarar las variables. Entonces, ¿cómo se les da un valor en los *scripts*? Se asigna un valor a una variable sin importar el tipo de ésta, utilizando el signo igual (=). Por tanto, a este signo se le conoce como "operador de asignación", y asigna el valor que aparece a la derecha a la variable que se encuentra a la izquierda. Aprenderá más sobre los tipos de operadores en los capítulos siguientes.

Por ejemplo:

```
$number = 1;
$floating_number = 1.2;
$string = "Hello, World!";
```

Para bien o para mal, como los tipos de variables no están bloqueados (PHP es conocido como lenguaje "poco tipado", al igual que JavaScript), se pueden cambiar sobre la marcha:

```
$variable = 1;
$variable = "Greetings";
```

Si fuese a imprimir ahora el valor de $variable, se imprimiría Greetings.

- Aprenderá más cosas sobre la asignación de valores a arrays en el Capítulo 6.
- Puede establecer el tipo de variable indicándolo en el primer uso (un método similar a declararla, donde formalmente se establece el tipo). La sintaxis para hacerlo es la siguiente:

  ```
  $number = (integer) 5;
  $string = (string) " Hello, World!";
  ```

  Para ser honestos, hacer esto tiene muy poco mérito, pero es una opción si quiere mantener la coherencia con otros lenguajes de programación que utilice.

Variables predefinidas

Existen dos razones por las que deseará conocer el concepto de variables predefinidas: 1) tiene utilidades propias en programación, y 2) para no crear otra variable con el mismo nombre sin darse cuenta.

Las variables predefinidas son un tipo especial de variables que usan las apliaciones de los servidores web (por ejemplo, Apache), el sistema operativo de los servidores web (Solaris o Windows NT), o los propios módulos PHP. A las dos primeras categorías se las conoce como "variables de entorno". Existen algunas incongruencias entre los servidores y las variables predefinidas que existen, así que emplee el *script* test.php (tratado en el Capítulo 1), para ver cómo funciona en la computadora (véase la Figura 2.1). Ejemplos de variables de entorno de un servidor son: HOSTNAME (como el servidor se llama a sí mismo), y OSTYPE (sistema operativo que ejecuta el servidor).

Ejemplos de variables de entorno de Apache son: DOCUMENT_ROOT, que indica dónde se almacenan los archivos en el servidor, y HTTP_USER_AGENT, que detalla el navegador y la plataforma del usuario que está viendo la página.

La variable PHP_SELF es la que utiliza normalmente PHP y es igual al nombre de la página actual. En la Figura 2.1, la variable PHP_SELF indica que la página visitada es test.php.

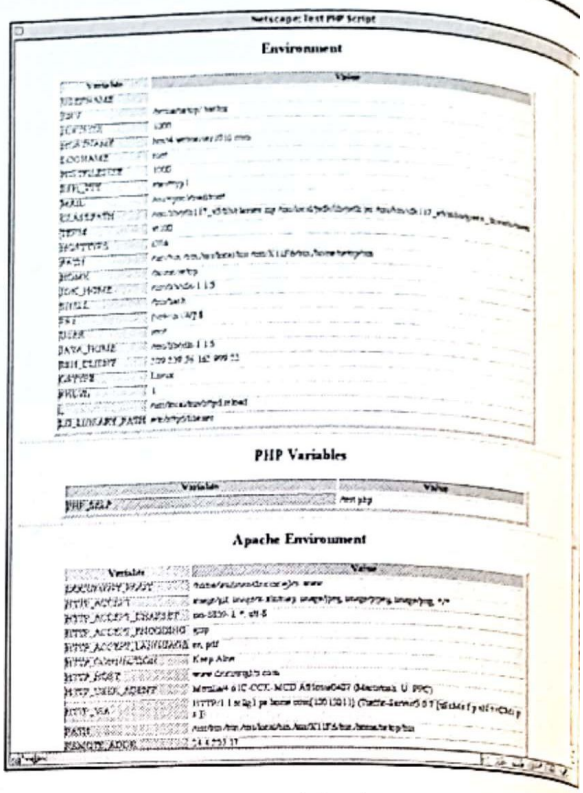


Figura 2.1. Otro gran uso del script test.php es comprobar qué variables PHP y de entorno existen y cuál es su configuración.

- Aunque no sea probable, la creación de una variable dentro de una página con el mismo nombre que la variable de entorno, provocaría unos resultados muy peculiares, así que ser consciente de los nombres de las variables facilitará la depuración. Como puede ver en la lista de variables de entorno que produjo el *script* test.php (véase la Figura 2.1), sencillamente al no usar letras mayúsculas en el nombre de una variable, se minimizaron los riesgos de cometer un error.

FORMULARIOS HTML Y PHP 3

Quizá el uso más común de las variables sea en conjunción con los formularios HTML. Los sitios web emplean formularios para registrar e iniciar la sesión de los usuarios, para recibir retroalimentación, para compras *online* y para otros muchos propósitos. Incluso los sitios más básicos encontrarán razones lógicas para incorporarlos.

Frecuentemente, los programadores crean los *scripts* CGI en Perl para manipular los datos que crean los formularios, pero pueden lograrse los mismos resultados más fácilmente si se usa PHP. A diferencia de los *scripts* CGI, donde tiene que escribir un segmento de código para que extraiga la información que envía el formulario, PHP tiene un maravilloso método de soporte integrado que recibe datos de un formulario HTML, sin necesidad de ningún análisis sintáctico.

En este capítulo trataremos sobre la creación básica de un formulario HTML y cómo se transmiten los datos a su *script* PHP. Los lectores nuevos en el tema de los formularios, desearán que tratemos en mayor profundidad los recursos HTML, teniendo en cuenta su importancia en el terreno del diseño de los sitios web.

Creación de un formulario simple

Va a crear un formulario HTML con una página de retroalimentación, que tenga su nombre y primer apellido, dirección de e-mail y comentarios. Recuerde esto para crear los campos necesarios.

Cómo crear un formulario HTML

1. Abra el editor de texto y cree un nuevo documento:

```
<HTML><HEAD><TITLE>HTML Form
→ </TITLE></HEAD><BODY></BODY>
→ </HTML>
```

2. Añada las etiquetas de apertura y cierre de <FORM> entre las etiquetas BODY:

```
<FORM ACTION="HandleForm.php">
→ </FORM>
```
(véase el *Script* 3.1).

Las etiquetas <FORM> indican dónde comienza y termina. Cualquiera de los elementos debe estar entre esas dos líneas. El atributo ACTION le indica al servidor la página (o *script*) que va a recibir los datos del formulario.

3. Guarde la página como form.html.

4. Después de abrir la línea FORM (línea 6) pero antes de la etiqueta de cierre, pulse Intro para crear una nueva línea.

5. Comience a añadir los campos a FORM, escriba:

```
First Name <INPUT TYPE=TEXT NAME=
→ "FirstName" SIZE=20><BR>
```

Siga una convención de nombres coherente para el formulario, dando a los campos un nombre lógico y descriptivo. Emplee letras, números y el subrayado (_) cuando nombre los campos. Mientras trabaja, tenga cuidado con los nombres usados para cada campo (véase el *Script* 3.2).

Script 3.1. Cada formulario HTML comienza y acaba con las etiquetas <FORM> </FORM>. Cuando codifique formularios manualmente, no se olvide de ellas. Asegúrese de dirigir el formulario al script adecuado con el atributo ACTION.

script

```
1  <HTML>
2  <HEAD>
3  <TITLE>HTML Form</TITLE>
4  </HEAD>
5  <BODY>
6  <FORM ACTION="HandleForm.php">
7  </FORM>
8  </BODY>
9  </HTML>
```

Script 3.2. Puede añadir cualquier combinación de tipos de entrada al formulario, sólo tiene que asegurarse de que todos los elementos están dentro de las etiquetas <FORM>, o no aparecerán. Una sugerencia de estilo sería aplicar esos elementos de entrada a una tabla, lo que daría al formulario una apariencia más profesional y manejable.

script

```
1  <HTML>
2  <HEAD>
3  <TITLE>HTML Form</TITLE>
4  </HEAD>
5  <BODY>
6  <FORM ACTION="HandleForm.php">
7  First Name <INPUT TYPE=TEXT
   NAME="FirstName" SIZE=20><BR>
8  Last Name <INPUT TYPE=TEXT
   NAME="LastName" SIZE=40><BR>
9  E-mail Address <INPUT TYPE=TEXT
   NAME="e-mail" SIZE=60><BR>
10 Comments <TEXTAREA NAME="Comments"
   FILAS=5 COLS=40></TEXTAREA><BR>
11 <INPUT TYPE=SUBMIT NAME=" SUBMIT "
   VALUE="Submit!">
12 </ FORM >
13 </BODY>
14 </HTML>
```

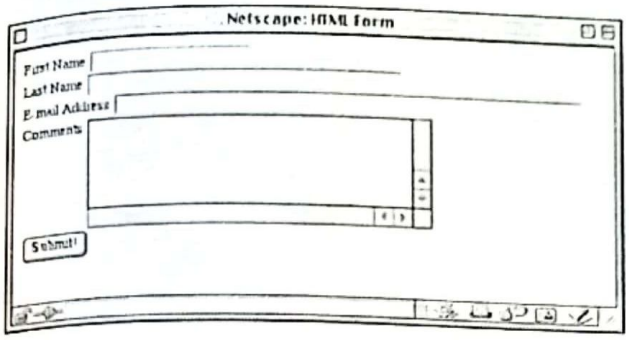


Figura 3.1. Si ha escrito correctamente el formulario, su aspecto debería parecerse a éste en el navegador. Asegúrese de que cierra el formulario e incluye el botón de envío.

```
Last Name <INPUT TYPE=TEXT NAME=
→ "LastName" SIZE=40><BR>
```

Añada las etiquetas <BR> para que el formulario aparezca más claramente cuando se visualice en el navegador web.

```
E-mail Address<INPUT TYPE=
→ TEXT NAME="e-mail" SIZE=60><BR>
Comments <TEXTAREA NAME="Comments"
→ ROWS=5 COLS=40></TEXTAREA><BR>
```

TEXTAREA ofrece al usuario más espacio para introducir sus comentarios que el campo TEXT. Sin embargo, con la entrada de TEXT puede limitar la cantidad de información que el usuario puede introducir, cosa que no podría hacer con TEXTAREA. Cuando cree el formulario, seleccione los tipos de entrada apropiados para la información que desea recuperar del usuario.

6. Escriba en líneas separadas

```
<INPUT TYPE=SUBMIT NAME="SUBMIT"
→ VALUE="Submit!">
```

El valor de un tipo SUBMIT, es lo que aparecerá en el botón del navegador web. También puede emplear "¡Ir!" o "Entrar", por ejemplo.

7. Guarde el *script*, cárgelo en el servidor y visualícelo en el navegador web (véase la Figura 3.1). (Al tratarse una página HTML, no un *script* PHP, puede visualizarlo directamente en el navegador web de la computadora.)

CONSEJOS

- En este ejemplo ha creado manualmente el formulario codificando HTML, pero también puede hacerlo mediante una aplicación para la creación de páginas web (como Dreamweaver o GoLive) si le resulta más cómodo.
- Fíjese que ha empleado la extensión HTML (.html) mientras creaba una página HTML estándar (no necesariamente una página PHP). Podría haber usado la extensión PHP y haber obtenido los mismos resultados, aunque actualmente no se utiliza PHP. Recuerde que en una página PHP, todo lo que no esté entre paréntesis PHP (<?php y ?>), se tratará como HTML.
- Aunque aquí no lo hagamos, recomendamos que en la mayoría de los formularios, y especialmente en los más complicados, se incluya un botón Reset, creado con el código:

```
<INPUT TYPE=RESET NAME=RESET VALUE=
→ "Reset">.
```

- Asegúrese de que el atributo ACTION señala correctamente un archivo existente del servidor, o no se procesará el resto del formulario. Por si sucediera esto, le indicamos que el archivo HandleForm.php está en el mismo directorio que la página form.html.

Uso de Get o Post

Un lector experimentado notaría que hemos olvidado algo en la etiqueta <FORM> inicial: añadir un atributo METHOD. Este atributo indica al servidor cómo transmitir los datos desde el formulario hasta el *script* de manipulación. Lo hemos omitido anteriormente porque este punto merece un tratamiento individual.

Existen dos elecciones en METHOD: GET o POST. Sospechamos que la mayoría de codificadores HTML no tienen muy clara esa distinción y cuándo utilzarla. De hecho, en la mayor parte, el usuario no notará la diferencia (especialmente la primera vez que los utilice), ya que generalmente le da los resultados apetecidos.

La diferencia entre GET y POST está en el modo de pasar la información del formulario al *script* de procesamiento. El método GET envía toda la información recogida como parte del URL. El método POST transmite la información de un modo invisible para el usuario. Por ejemplo, al enviar el formulario, si emplea el método GET, el URL resultante se parecería a:

```
http://www.DMCinsights.com/php/
→ HandleForm.php?FirstName=
→ Larry&LastName=Ullman.
```

Donde quiera que use el método POST, el usuario final sólo podrá ver:

```
http://www.DMCinsights.com/php/
→ HandleForm.php.
```

Una vez elegido el método que va a utilizar, debe tener en cuenta los siguientes factores:

1. Con el método GET, estará limitado a la cantidad de información que puede pasar.
2. El método GET envía públicamente la entrada al *script* de manipulación (lo que significa, por ejemplo, que una palabra clave introducida en un formulario puede visualizarse en el navegador web, ocasionando mayor riesgo en la seguridad).
3. Una página generada por un formulario que utilizó el método GET se puede marcar, cosa que no sucede en el caso de POST.

En este texto, usaremos POST para manipular formularios, principalmente para hacer una distinción con la técnica que utilizaremos cuando se trata de añadir capacidades al sitio web mediante el método GET (véase la sección "Entrada manual de datos", posteriormente en este capítulo). Cualquiera de estos métodos pasará con éxito los datos desde el formulario y la decisión final acerca del mejor método a utilizar debería basarse en las necesidades del formulario y en el hecho de que la página resultante pueda o no ser marcada.

Cómo añadir un método al *script*

1. Abra form.html en el editor de textos.
2. En la etiqueta inicial <FORM>, añada METHOD=POST (línea 6, *Script* 3.3).
3. Guarde el *script* y cárguelo en el servidor.
4. Visualice el código fuente de la página para asegurarse de que aparecen todos los elementos requeridos (véase la Figura 3.2).

Script 3.3. La utilización de cualquiera de los métodos GET o POST es una opción personal, pero recuerde que tiene que usar uno de ellos. Con la práctica, podrá decidir fácilmente cuál usar; más que un aspecto funcional, es una cuestión de sutileza.

script

```
1  <HTML>
2  <HEAD>
3  <TITLE>HTML Form</TITLE>
4  </HEAD>
5  <BODY>
6  <FORM ACTION="HandleForm.php" METHOD=POST>
7  First Name <INPUT TYPE=TEXT
   NAME="FirstName" SIZE=20><BR>
8  Last Name <INPUT TYPE=TEXT NAME="LastName"
   SIZE=40><BR>
9  E-mail Address<INPUT TYPE=TEXT
   NAME="e-mail" SIZE=60><BR>
10 Comments <TEXTAREA NAME="Comments" ROWS=5
   COLS=40></ TEXTAREA><BR>
11 <INPUT TYPE=SUBMIT NOMBRE="SUBMIT"
   VALUE="Submit!">
12 </FORM>
13 </BODY>
14 </HTML>
```

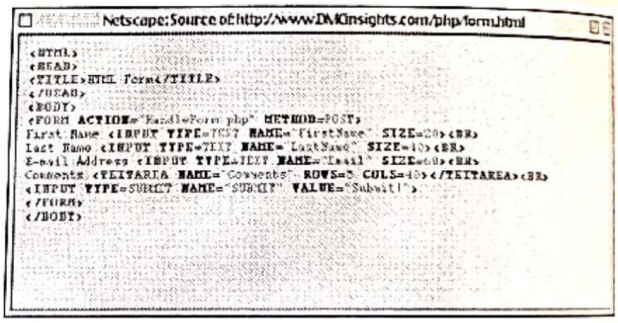


Figura 3.2. Visualizando el código fuente de una página puede determinar lo que funciona y lo que no funciona. Como en toda programación, cambios sutiles pueden marcar la diferencia.

Script 3.4. Con el valor del elemento NAME="Name" en el formulario HTML y añadiendo el signo del dólar, puede crear una variable que contenga el valor que introdujo el usuario en el correspondiente campo del formulario. Esto es válido, tanto si el tipo de entrada de HTML es TEXT, TEXTAREA o un menú SELECT. Ésta es una de las razones por las que PHP es muy bueno para el manejo de formularios HTML (comparado, por ejemplo, con los scripts CGI, que requieren un código de analizador sintáctico).

script

```
1  <HTML>
2  <HEAD>
3  <TITLE>Form Results</TITLE>
4  <BODY>
5  <?php
6  /* Esta página recibe y administra los
   datos generados por "form.html". */
7  print "Your first name is
   $FirstName.<BR>\n";
8  print "Your last name is $LastName.<BR>\n";
9  print "Your E-mail address is
   $Email.<BR>\n";
10 print "This is what you had to say:<BR>\n
   $Comments<BR>\n";
11 ?>
12 </BODY>
13 </HTML>
```

Recepción de datos de un formulario en PHP

Ya tiene creado el formulario. Ahora necesita escribir el *script* HandleForm.php, que recibirá y procesará los datos generados por la página form.html. Es aquí donde queda demostrada la simplicidad de PHP.

Para crear el *script* HandleForm.php:

1. Abra el editor de texto y cree un nuevo documento.

```
<HTML><HEAD><TITLE>Form Results
→ </TITLE><BODY><?php /* Esta página
→ recibe y administra los datos
→ generados por "form.html". */ ?>
→ </BODY></HTML>.
```

   Éste es el formulario estándar para una página PHP (véase el *Script* 3.4). Añadimos el comentario para definir el propósito del *script*. Incluso aunque la página form.html indique dónde fueron enviados los datos (a través del atributo ACTION), deberíamos hacer aquí un comentario indicando el proceso contrario.

2. Cree una nueva línea después del comentario, pero antes de la etiqueta de cierre PHP. Escriba:

```
print " Your first name is
→ $FistName.<BR>\n";.
print " Your last name is
→ $LastName.<BR>\n";
print " Your E-mail Address is
→ $Email.<BR>\n";
print " This is what you had to
→ say:<BR>\n $Comments<BR>\n";
```

3. Guarde el *script* como HandleForm.php.

4. Cargue el *script* en el servidor, asegurándose de que lo guarda en el mismo directorio que form.html.

5. Revise el *script* en el navegador web (véanse las Figuras 3.3 y 3.4).

El fin de este ejercicio es demostrarle cómo puede transferir datos fácilmente de un formulario HTML a una página PHP. El *script* PHP almacenará los datos en las variables correspondientes, de manera que $FirstName elejirá el valor que el usuario introdujo en el campo etiquetado FirstName (elija el nombre del campo en HTML, añada el signo del dólar, y así obtendrá la variable con el valor correspondiente).

La transferencia es automática y, al contrario de lo que ocurre en los *scripts* CGI, no necesitará ser analizado sintácticamente.

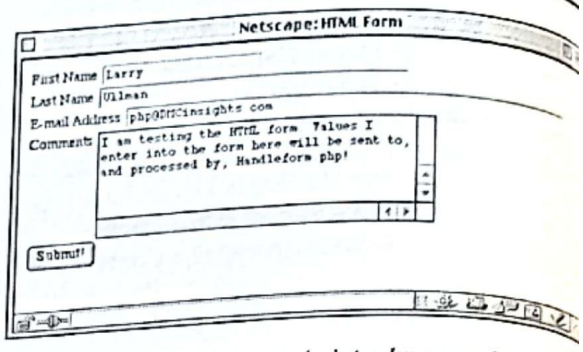


Figura 3.3. Lo que el usuario introduzca en el formulario HTML, el script HandleForm.php lo imprimirá así en el navegador (véase la Figura 3.4).

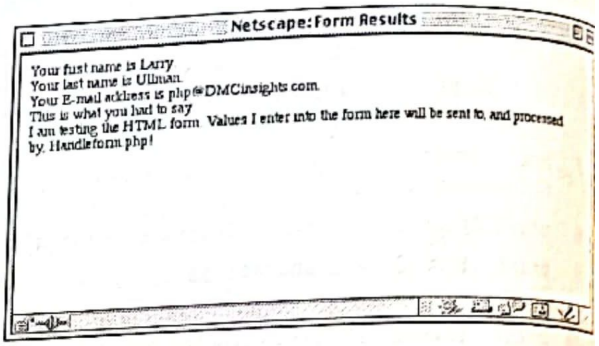


Figura 3.4. Ésta es otra aplicación de la instrucción print explicada en el Capítulo 1, pero constituye su primera página web generada dinámicamente. En capítulos posteriores veremos cómo manipular los datos recibidos, así como su envío por correo electrónico o su almacenamiento en una base de datos.

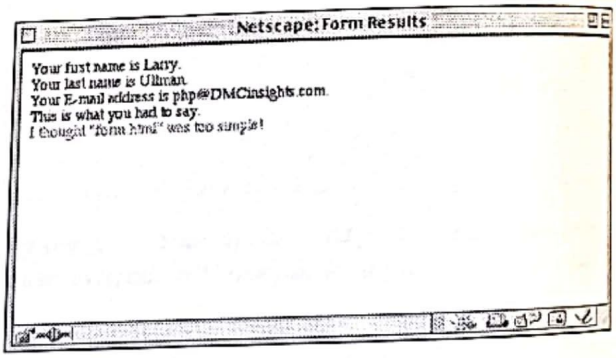


Figura 3.5. PHP crea automáticamente secuencias de escape con los caracteres especiales introducidos en HTML. Esto es útil cuando se envían datos de regreso al navegador (como en este ejemplo, donde las comillas interfieren con la instrucción print() si no se expresan como secuencias de escape) y cuando se introducen en una base de datos.

CONSEJOS

- Otras ventajas de la utilización de PHP para administrar formularios HTML es que los datos escapan automáticamente durante el tránsito. Por ejemplo, si introducimos "Pienso que "form.html" es demasiado simple" como comentario, la variable $Comments será igual a "Pienso que \"form.html\" es demasiado simple", de manera que puede imprimirse sin problemas (véase la Figura 3.5).
- Si quiere pasar un valor predeterminado al *script*, use el tipo de entrada HIDDEN en el formulario HTML. Por ejemplo, la línea <INPUT TYPE= → HIDDEN NAME="ThisPage" VALUE="form.html"> insertada entre las etiquetas del formulario crearían una variable en el *script* de manipulación denominada $ThisPage con el valor de "form.html". Igualmente, solicitando a PHP:

```
print ("<INPUT TYPE=
→ HIDDEN NAME=\"FirstName\"
→ VALUE=\"$FirstName\">");
```

  puede extender la vida de la variable $FirstName pasando su valor.
- Aunque puede que el mismo archivo despliegue un formulario y administre la salida del formulario usando PHP, esto puede hacer que los *scripts* sean demasiado complicados y difíciles de depurar. Para simplificarlo, usaremos un archivo aparte, acertadamente denominado HandleForm.php.

Script 3.5. En vez de codificar la página para que diga automáticamente "Hello, world!" o "Hello, Larry!", hemos dado dinamismo a la misma imprimiendo el valor de una variable. Ahora, esos valores han cambiado, así como la página resultante.

```
script
1  <HTML>
2  <HEAD>
3  <TITLE>Using Variables</TITLE>
4  <BODY>
5  <?php
6  /* Esta página puede tener numerosas líneas
   de código con la siguiente instrucción
   print. */
7  print "<H2> <CENTER> <B> Hello, $FirstName.
   </B> </CENTER> </H2> <BR>\n";
8  /* Esta página puede tener más código
   después de la instrucción print a la que
   nos referimos. */
9  ?>
10 </BODY>
11 </HTML>
```

Entrada manual de datos

Obviamente, no todos los datos derivan de los formularios HTML, y en ocasiones puede que desee introducir datos manualmente en el *script*. Si creó un *script* que imprimió un nombre de usuario en un saludo, puede establecer una variable que contenga el nombre separado del resto del saludo. De este modo, podrá cambiar fácilmente el nombre, sin necesidad de cambiar el saludo, la instruccción print o el *script*. Crearemos un *script* PHP para mostrarle el proceso.

Cómo crear un script PHP con un saludo

1. Cree un nuevo documento en el editor de textos (véase el *Script* 3.5).

```
<HTML><HEAD><TITLE>Using Variables
→ </TITLE><BODY><?php
/* Esta página puede tener numerosas
→ líneas de código con la siguiente
→ instrucción print. */
```

   Hemos añadido dos comentarios al *script* para representar la idea de que la línea de saludo puede ser parte de una página más complicada y más larga.

```
print "<H2> <CENTER> <B> Hello,
→ $FirstName. </B> </CENTER> </H2>
→ <BR>\n";
/* Esta página puede tener más
→ código después de la instrucción
→ print a la que nos referimos. */
?></BODY></HTML>
```

2. Guarde el documento como hello.php y cárguelo en el servidor.

Si quisiera visualizar ahora el *script*, sólo aparecerá "Hello, .", ya que la variable $FirstName no tiene valor. Existen dos formas de establecer este valor manualmente, sin necesidad de usar formularios. Utilice lo que ya sabe acerca del modo en que el método GET pasa datos a un *script*.

Cómo usar el método GET sin un formulario HTML

1. Visualice el *script* hello.php en el navegador web a través del URL apropiada (en este caso, [http://www.DMCinsights.com/php/hello.php]; véase la Figura 3.6).

2. Añada al URL el texto ?FirstName=Larry (puede usar su nombre, para asegurarse de que no hay espacios en la entrada).

   Como ya mencionamos al principio de este capítulo, cuando se envía una variable al *script* a través de un URL (por ejemplo, usando el método GET en un formulario HTML), se usa el formato de www.url.com/script.php?variable=value.

3. Vuelva a cargar la página en el navegador con el nuevo URL (véase la Figura 3.7).

   Si no ve el nombre impreso en el navegador, habrá cometido un error. Revíselo para asegurarse de que incluyó el signo de interrogación, que separa el archivo de ubicación de los datos. Después, asegúrese de que usó FirstName, ya que es lo que espera hello.php como variable (recuerde que firstname constituiría una variable diferente).

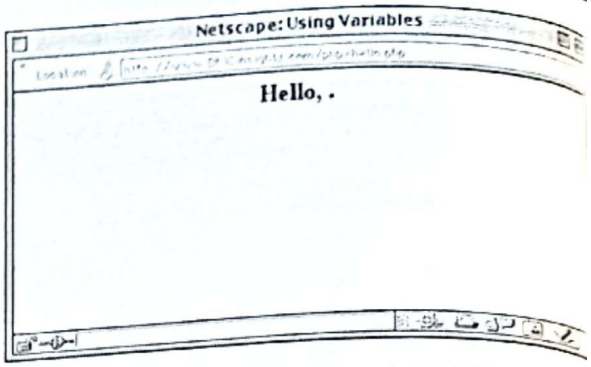


Figura 3.6. Sin asignar un valor a la variable $FirstName, el navegador imprime este torpe mensaje.

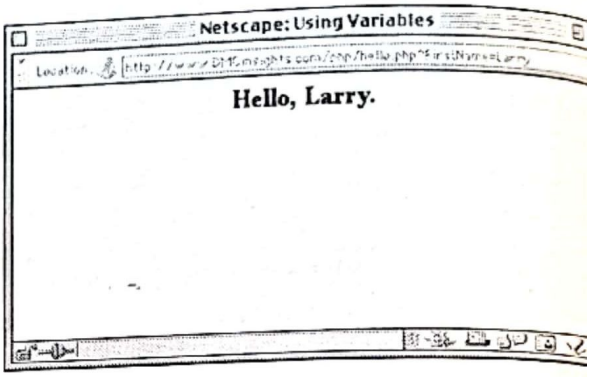


Figura 3.7. Asignando a $FirstName el valor Larry o cualquier otro nombre, ha creado una página dinámica que cambia para dirigirse personalmente a un usuario determinado.

Script 3.6. La línea $FirstName = "Jude"; asigna el valor "Jude" a la variable "FirstName" para esta página.

```
script
1  <HTML>
2  <HEAD>
3  <TITLE>Using Variables</TITLE>
4  <BODY>
5  <?php
6  $ FirstName = "Jude";
7  /* Esta página puede tener numerosas líneas
   de código con la siguiente instrucción
   print. */
8  print "<H2> <CENTER> <B> Hello, $FirstName.
   </B> </CENTER> </H2> <BR>\n";
9  /* Esta página puede tener mucho código
   después de la instrucción print en la que
   estamos centrados. */
10 ?>
11 </BODY>
12 </HTML>
```

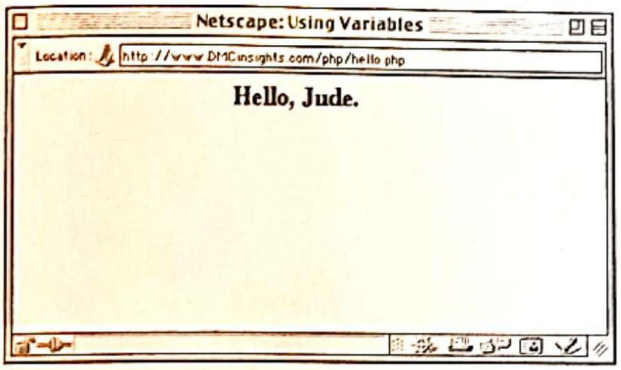


*Figura 3.8. Estableciendo el valor de $FirstName a Jude, obtendremos el mismo efecto que cambiando la línea de print por print "<H2> <CENTER> <B> Hello, Jude. </B> </CENTER> </H2>
\n";, pero nos resultará más fácil cambiar el valor si utilizamos una variable.*

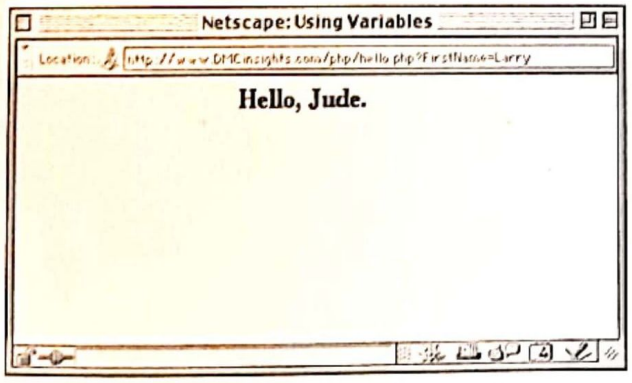


Figura 3.9. Puesto que la propia página contiene una línea que asigna el valor "Jude" a $FirstName, no importa lo que pase al script a través de un URL, siempre se imprimirá "Jude".

El segundo método para prefijar un valor variable es asignándolo directamente en el *script*.

Cómo asignar un valor a una variable

1. Abra hello.php en el editor de textos.
2. En la línea anterior a la instrucción print, añada $FirstName = "Jude"; (véase el *Script* 3.6).
3. Guarde la página, cárgela en el servidor y visualícela en el navegador web (véase la Figura 3.8).
4. Ahora vea la misma página utilizando la versión añadida del URL al igual que hizo en el último ejemplo (por ejemplo, http://
   → www.DMCinsights.com/php/
   → hello.php?FirstName=Larry)
   (véase la Figura 3.9).

   Esto demuestra un punto importante, qué es lo que sucede cuando a una variable se le asigna un valor varias veces: el último valor asignado es lo que la variable transforma en "valor"; por tanto, en nuestro caso, se imprimirá hello.php. Tenga cuidado de no sustituir las variables del programa sin darse cuenta.

El concepto para preestablecer valores de variables es espacialmente útil cuando el programa se hace cada vez más complicado. Si utiliza una variable de fácil edición, nunca tendrá que sortear múltiples líneas de código para reemplazar los valores.

Asimismo, añadiendo una variable al URL, puede enlazar una página con otra que haya sido generada dinámicamente (como hello.php), codificando los enlaces de la página web para incluir valores de variables donde fuese necesario. Por ejemplo, un formulario de un sitio web puede tomar el nombre del usuario y pasarlo a través de las páginas enlazadas con el código <A HREF ="hello.php? FirstName=Larry"> hello.php </A>.

CONSEJOS

- Si quiere usar el método GET para enviar datos al *script*, puede pasar múltiples valores separando los pares variable=value (por ejemplo, FirstName=Larry) con lel signo (&).

  Un URL añadido puede ser hello.php?FirstName=Larry&LastName=Ullman.

- Los espacios dentro de los valores que se pasan como parte de un URL, se deberían reemplazar con un signo más (+). En el Capítulo 5 trataremos sobre cómo PHP puede preparar automáticamente una cadena de texto para que pase como parte de un URL.

- Aunque este ejemplo (ajustar el valor del nombre de una persona) puede que no sea muy práctico, crear una variable $email en la parte superior del *script* podría ayudarle a cambiar después la dirección de dicho e-mail sin tener que revisar las líneas del código para encontrarla.

  De hecho, en el Capítulo 11, crearemos el hábito de establecer los valores de acceso a las bases de datos a variables al principio de los *scripts*, para que puedan modificarse con facilidad sin cambiar cada uno de los ejemplo donde se usaron dichos valores.

UTILIZACIÓN DE NÚMEROS 4

En el Capítulo 2 tratamos de forma superficial los distintos tipos de variables, cómo asignarles valores y cómo se utilizan normalmente. En este capítulo trabajará específicamente con variables numéricas, tanto enteras (es decir, números enteros) como de punto flotante (decimales). Al final de la sección mostraremos algunas funciones útiles para trabajar con números.

Suma, resta, multiplicación y división

Como aprendió en el colegio, las matemáticas más básicas implican sumas, restas, multiplicaciones y divisiones. Para demostrar estos principios, creará un *script* PHP que calcule el coste total de las ventas de varios dispositivos. Este *script* podría ser la base de un formulario de compras, un elemento muy práctico de las páginas web.

Cómo crear la calculadora de costes de ventas

1. Abra el editor de textos y cree un nuevo documento (véase el *Script* 4.1).

```
<HTML><HEAD><TITLE>Using Numbers
→ </TITLE></HEAD><BODY><?php?>
→ </BODY></HTML>
```

2. Dentro de las etiquetas PHP, añada

```
$Cost = 2000.00; $Tax = 0.06;.
```

Ha configurado manualmente el coste del dispositivo a 2000 dólares. Observe que no se utiliza el signo del dólar ni la coma en el valor de la variable. También ha configurado de forma manual los impuestos de ventas a un 6%, siendo "0.06" la forma apropiada de escribir un porcentaje decimal. Ambos números son de punto flotante.

```
$TotalCost = $Cost * $Quantity;
```

El asterisco (*) se utiliza para indicar una multiplicación en PHP. El valor de $Quantity pasará al *script* como si lo hiciera en un formulario de compras en un sitio web. Puede utilizar las técnicas mostradas en el Capítulo 3 para crear un formulario donde el usuario introduzca la cantidad, pero aquí sólo añadirá el valor al URL.

Script 4.1. Aunque los cálculos sean sencillos en sí mismos, debería tener la libertad de añadir algún otro comentario si lo creyera conveniente para aclarar el proceso. Si desea mejorar las técnicas HTML, cree un formulario que tome información del usuario (incluyendo cantidad y descuento) para que después pase a este script.

script

```
1  <HTML>
2  <HEAD>
3  <TITLE>Using Numbers</TITLE>
4  </HEAD>
5  <BODY>
6  <?php
7  /* $Quantity se debe pasar a esta página
   desde un formulario o mediante un URL.
   $Discount es opcional. */
8  $Cost = 2000.00;
9  $Tax = 0.06;
10 $TotalCost = $Cost * $Quantity;
11 $Tax = $Tax + 1; // $Tax tiene ahora el
   valor de 1.06.
12 $TotalCost = $TotalCost - $Discount;
13 $TotalCost = $TotalCost * $Tax;
14 $Payments = $TotalCost / 12;
15 // Imprimir los resultados.
16 print ("You requested to purchase $Quantity
   widget(s) at \$$Cost each.\n<P>");
17 print ("The total with tax, minus your
   \$$Discount, comes to \$$TotalCost.
   \n<P>");
18 print ("You may purchase the widget(s)in
   12 monthly installments of \$$Payments
   each.\n<P>");
19 ?>
20 </BODY>
21 </HTML>
```

```
$Tax = $Tax + 1; // $Tax tiene ahora
→ el valor de 1.06.
```

Lógicamente el signo más (+) se emplea para realizar la suma. Puede calcular cuánto costará algo con las tasas incluidas añadiendo 1 al porcentaje y multiplicando después esa nueva tasa por el total. Se ha añadido un comentario para aclarar la operación (el comentario se puede situar después de la línea, en la siguiente línea u omitirlo). Una de las razones por las que hemos agrupado ambos tipos de números en una categoría de variable es que, como puede ver aquí, puede realizar cálculos con tipos de variables mixtos sin tener consecuencias desafortunadas.

```
$TotalCost = $TotalCost - $Discount;
```

Al mostrar una sustracción utilizando el signo menos (-), supondrá que puede aplicarse un descuento, que también se añadirá al URL (o se introducirá en un formulario).

```
$TotalCost = $TotalCost * $Tax;
```

En realidad, puede realizar un cálculo en una variable para determinar el valor (de hecho, es bastante común), pero observe que el valor original de la variable ha desaparecido. De modo que aquí, el valor original de $TotalCost se sustituye por el valor de $TotalCost multiplicado por $Tax.

```
$Payments = $TotalCost / 12;
```

Como ejemplo de división, supongamos que el dispositivo se financia a un año. Por tanto, ha dividido el total, incluyendo las tasas y cualquier descuento aplicable entre 12 meses para obtener el pago mensual.

```
// Imprimir los resultados.
```

Este comentario diferencia dónde terminan los cálculos y dónde empieza la salida al navegador.

```
print ("You requested to purchase
→ $Quantity widget(s) at \$$Cost
→ each.\n<P>");

print ("The total with tax, minus
→ your \$$Discount, comes to
→ \$$TotalCost.\n<P>"); print ("You
→ may purchase the widget(s)in 12
→ monthly installments of $$Payments
→ each.\n<P>");
```

3. Guarde el *script* como numbers.php y cárguelo en el servidor.

   Pruebe el *script* en el navegador web, asegurándose de que asigna valores a "Quantity" y "Discount" (véase la Figura 4.1).

4. Puede probar con estos valores (incluso omitiendo completamente las variables como se ve en la Figura 4.2) para ver si la calculadora trabaja eficazmente.

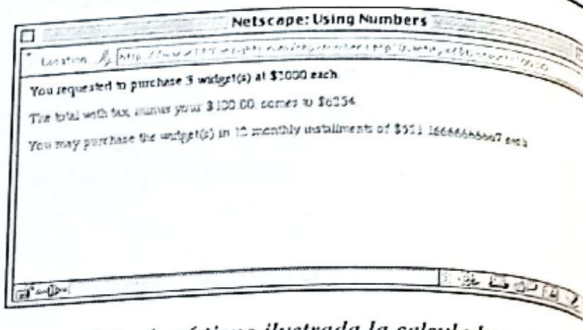


Figura 4.1. Aquí tiene ilustrada la calculadora. Asegúrese de añadir valores para "Quantity" y "Discount" al URL o cree un formulario para pasar estos valores, o verá unos resultados como los de la Figura 4.2.

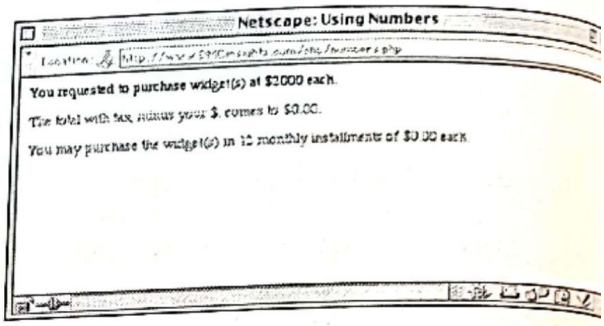


Figura 4.2. Si no se reciben los valores necesarios, el script seguirá realizando los cálculos asignados, pero los resultados serán erróneos.

CONSEJOS

- Como observará, la calculadora se encontrará con números que no correspondan a valores verdaderos de dólares (véase la Figura 4.1). En la siguiente sección aprenderá a ajustar los números convenientemente.
- Si quisiera imprimir el valor total, antes de la tasa y antes del descuento (o ambos), tendría dos formas de hacerlo. Una sería insertar las instrcciones print() apropiadas después de que se hubiera determinado el valor correcto pero antes de que se hubiera cambiado la variable $TotalCost. El segundo método sería utilizar nuevas variables que contuvieran los valores de los cálculos consiguientes (por ejemplo, $TotalWithTax y $TotalLessDiscount).
- Hay dos métodos que podría haber utilizado para imprimir un número como 2000 dólares. Uno es omitir el signo del dólar, como hizo aquí, y el otro es dejar un espacio entre el signo de dólar y el nombre de la variable, lo que también crearía un espacio en el navegador. No puede emplear $$Variable, ya que la combinación de dos signos de dólar juntos creará un tipo de variable demasiado compleja para los propósitos de este libro.

Script 4.2. La función printf() imprimirá los números formateados de acuerdo con ciertas especificaciones, creando números más lógicos.

```
script
1  <HTML>
2  <HEAD>
3  <TITLE>Using Numbers</TITLE>
4  </HEAD>
5  <BODY>
6  <?php
7  /* $Quantity se debe pasar a esta página
   desde un formulario o mediante un URL.
   $Discount es opcional. */
8  $Cost = 2000.00;
9  $Tax = 0.06;
10 $TotalCost = $Cost * $Quantity;
11 $Tax = $Tax + 1; // $Tax tiene ahora el
   valor de 1.06.
12 $TotalCost = $TotalCost - $Discount;
13 $TotalCost = $TotalCost * $Tax;
14 $Payments = $TotalCost / 12;
15 // Imprimir los resultados.
16 print ("You requested to purchase $Quantity
   widget(s) at \$$Cost each.\n<P>");
17 print ("The total with tax, minus your
   \$$Discount, comes to $");
18 printf ("%01.2f", $TotalCost);
19 print (".\n<P>You may purchase the
   widget(s)in 12 monthly installments of $");
20 printf ("%01.2f", $Payments);
21 print (" each.\n<P>");
22 ?>
23 </BODY>
24 </HTML>
```

Formateo de números

Aunque la calculadora está encaminada a ser práctica, todavía tiene un verdadero problema: no puede pedir a alguien que haga un pago mensual de 521,16666666667 dólares.

Para crear un número más normal como pago mensual, hará del coste total un número fácilmente divisible. Para hacer esto, utilice la función printf(), que imprime un número formateado de acuerdo a las especificaciones.

Para utilizar printf(), debe darle un formato y una cadena (o número). Por ejemplo, para imprimir en el formulario una variable de cantidad que sea un número de punto flotante con dos dígitos a la derecha del decimal (por ejemplo, 1.02), codificaría:

```
printf ("%01.2f", $Amount);
```

El segmento "%01.2f" le dice a PHP que imprima $Amount utilizando 0 para rellenar los espacios extra, con al menos un dígito a la izquierda del decimal y con dos dígitos a la derecha del decimal. Hay usos más complicados de printf(), sin embargo esto será suficiente para crear cantidades verdaderas.

Cómo utilizar printf()

1. Abra numbers.php en el editor de textos.
2. Cambie la línea 17 para que se lea (véase el *Script* 4.2):

```
print ("The total with tax, minus
→ your \$$Discount, comes to $");
→ printf ("%01.2f", $TotalCost);
```

Para utilizar printf(), deberá separar la mayoría de las instrucciones print() de la que manipula el formato.

3. Cambie la línea 18 para que se lea:

```
print (".\n<P>You may purchase the
→ widget(s)in 12 monthly
→ installments of $");
printf ("%01.2f", $Payments);
print (" each.\n<P>");
```

   Hará lo mismo a la línea 18 que a la línea 17. Una línea original del código se convertirá en tres pero ofrecerá mejores resultados.

4. Guarde el archivo, cárguelo en el servidor y pruébelo en el navegador (véase la Figura 4.3).

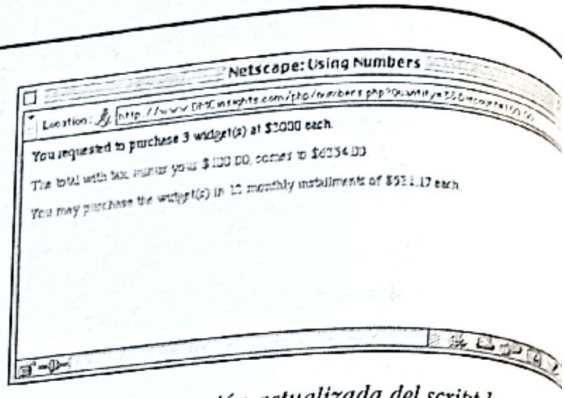


Figura 4.3. La versión actualizada del script le proporciona unos números para los pagos mensuales mejores que antes de incorporar la función printf() (véa la Figura 4.1).

CONSEJO

- Una función paralela a printf() es la función sprintf(). Funciona de la misma forma exceptuando que printf() enviará los resultados al navegador mientras que sprintf() cambia una variable sin imprimirla.

```
$Amount = sprintf ("%01.2f",
→ $Amount);
```

Script 4.3. El script es ahora ligeramente más claro de lo que era antes de que utilizara el método incremental "++". El resultado matemático será el mismo a pesar de ello.

```
script
1  <HTML>
2  <HEAD>
3  <TITLE>Using Numbers</TITLE>
4  </HEAD>
5  <BODY>
6  ?php
7  /* $Quantity se debe pasar a esta página
   desde un formulario o mediante un URL.
   $Discount es opcional. */
8  $Cost = 2000.00;
9  $Tax = 0.06;
10 $TotalCost = $Cost * $Quantity;
11 $Tax++; // $Tax tiene ahora el valor de
   1.06.
12 $TotalCost = $TotalCost - $Discount;
13 $TotalCost = $TotalCost * $Tax;
14 $Payments = $TotalCost / 12;
15 // Imprimir los resultados.
16 print ("You requested to purchase $Quantity
   widget(s) at \$$Cost each.\n<P>");
17 print ("The total with tax, minus your
   \$$Discount, comes to $");
18 printf ("%01.2f", $TotalCost);
19 print (".\n<P>You may purchase the
   widget(s)in 12 monthly installments of $");
20 printf ("%01.2f", $Payments);
21 print (" each.\n<P>");
22 ?>
23 </BODY>
24 </HTML>
```

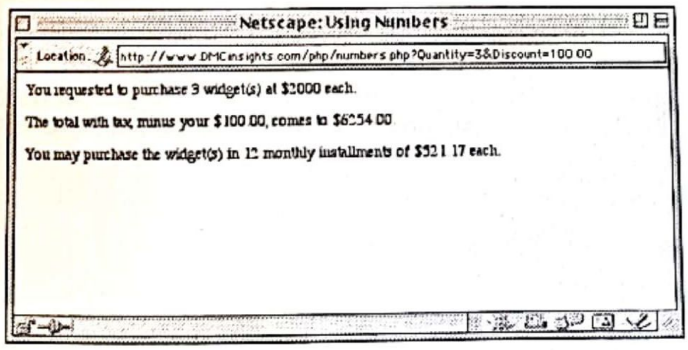


Figura 4.4. No afecta los cálculos el que utilice la versión larga o corta del incremento de una variable (compare los Scripts 4.2 y 4.3).

Incremento y disminución de un número

PHP, como Perl y la mayoría de los lenguajes de programación, incluye algunos accesos directos para que pueda evitar algunas construcciones feas como $Tax = $Tax + 1;. Cuando esté en una situación en la que necesita incrementar un valor en 1 (llamado ajuste "incremental") o disminuir el valor de la variable en 1 (ajuste "decremental") puede utilizar "++" o "-", según sea el caso.

Cómo incrementar el valor de una variable

1. Abra numbers.php en el editor de textos.
2. Cambie la línea 10 del *Script* 4.3 para que se lea:

   $Tax++;

3. Guarde el *script*, cárguelo en el servidor y pruébelo en el navegador (véase la Figura 4.4).

- Aunque la funcionalidad no será problema si codifica $Tax = $Tax + 1; o el abreviado $Tax++, el último (utilizando el operador de incremento) es el método más profesional y común.
- En el Capítulo 6 verá cómo se utiliza normalmente el operador de incremento para contar a través de una serie de acciones.
- Para disminuir una variable, sencillamente utilice el símbolo de sustracción (-) dos veces en lugar del signo (+), así: $Number-;

Utilización de varios operadores

Inevitablemente, después de discutir los distintos tipos de operadores matemáticos, se llega a la explicación de la precedencia. La precedencia se refiere al orden en que se ejecutarán una serie de cálculos. Por ejemplo, ¿cuál será el valor de la siguiente variable?

```
$Number = 10 - 4 / 2;
```

¿Será $Number igual a 3 (10 menos 4 son 6 entre 2 igual a 3) o será 8 (4 dividido entre 2 igual a 2 restado de 10 es igual a 8)? La respuesta es 8, porque la división tiene prioridad sobre la resta. En el Apéndice C se muestra una lista completa sobre la prioridad de los operadores para PHP (incluyendo aquellos que no hemos abordado todavía).

Sin embargo, en lugar de intentar memorizar una larga tabla de caracteres extraños, nuestra sugerencia es que englobe el concepto completo utilizando paréntesis. Los paréntesis siempre tienen prioridad sobre cualquier otro operador. Así:

```
$Number = (10 - 4) / 2;
$Number = 10 - (4 / 2);
```

En el primer ejemplo, $Number es ahora igual a 3 y en el segundo ejemplo, $Number es igual a 8. La utilización de paréntesis en los cálculos asegurará que nunca obtenga desde ellos resultados extraños debido a la prioridad.

Puede volver a escribir el *script*, combinando varias líneas en una mientras mantiene la exactitud utilizando paréntesis.

Script 4.4. En lugar de realizar los cálculos sobre varias líneas, lo ha reducido a una, sin afectar sus bases matemáticas. Con paréntesis, no necesita preocuparse de la prioridad.

```
script
1  <HTML>
2  <HEAD>
3  <TITLE>Using Numbers</TITLE>
4  </HEAD>
5  <BODY>
6  <?php
7  /* $Quantity se debe pasar a esta página
   desde un formulario o mediante un URL.
   $Discount es opcional. */
8  $Cost = 2000.00;
9  $Tax = 0.06;
10 $Tax++; // $Tax tiene ahora el valor de
   1.06.
11 $TotalCost = (($Cost * $Quantity) -
   $Discount) * $Tax;
12 $Payments = $TotalCost / 12;
13 // Imprimir los resultados.
14 print ("You requested to purchase $Quantity
   widget(s) at \$$Cost each.\n<P>");
15 print ("The total with tax, minus your
   \$$Discount, comes to $");
16 printf ("%01.2f", $TotalCost);
17 print (".\n<P>You may purchase the widget(s)
   in 12 monthly installments of $");
18 printf ("%01.2f", $Payments);
19 print (" each.\n<P>");
20 ?>
21 </BODY>
22 </HTML>
```

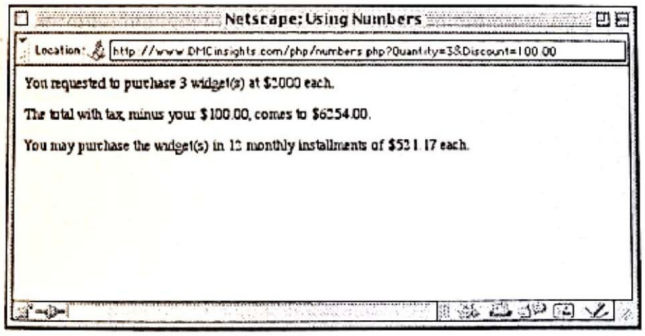


Figura 4.5. Aunque los cálculos se han condensado, las matemáticas todavía funcionan igual. Si ve resultados distintos o si obtiene un mensaje de error, compruebe los paréntesis.

Cómo utilizar paréntesis para establecer la prioridad

1. Abra numbers.php en el editor de textos (véase el *Script* 4.3).
2. Cambie la línea 13 para que se lea (véase el *Script* 4.4):

   ```
   $TotalCost = (($Cost * $Quantity) -
   → $Discount) * $Tax;
   ```

   No hay razón para no hacer todos los cálculos en un paso, mientras utilice los paréntesis para asegurar que la operación funciona correctamente. La otra opción es memorizar las reglas PHP de prioridad para múltiples operadores, pero el uso de paréntesis resulta mucho más fácil.
3. Borre las líneas 10 y 12.

   Ya que todos los cálculos se hacen en una línea, estas líneas que ya no son necesarias.
4. Guarde el *script*, cárguelo en el servidor y pruébelo en el navegador (véase la Figura 4.5).

CONSEJO

- Haga que la correspondencia de paréntesis sea coherente cuando cree las fórmulas (cada paréntesis abierto necesita uno de cierre).

Utilización de funciones matemáticas internas

PHP tiene funciones internas para trabajar con los datos matemáticos. Emplearemos dos de ellos aquí para mejorar el *script* numbers.php. Además puede dirigirse al manual de PHP, en concreto al capítulo de funciones matemáticas, para conocer las demás funciones (consulte el Apéndice C, respecto al manual de PHP).

Una función matemática interna que puede utilizar en el *script* de la calculadora es round(). Como su nombre indica, round() toma cualquier número y lo redondea al entero más cercano, utilizando convenciones estándar: cualquiera con .50 o mayor se redondea hacia arriba, cualquiera menor de .50 se redondea hacia abajo. Se supone que el redondeo afectará a un número decimal, si bien intentar redondear un número entero no causará problemas (sencillamente no hará nada, ya que redondear 2 dará como resultado 2). Algunos ejemplos son:

```
$Number = round(23.309); // $Number is
→ equal to 23.
$Number = round(23.51); // $Number is
→ equal to 24.
```

Puede redondear un número para que sólo tenga un punto decimal añadiendo un segundo parámetro a la ecuación.

```
$Number = round(23.51, 1); // $Number
→ is equal to 23.5.
```

PHP divide round() en dos funciones. La primera, ceil(), redondea un número al entero superior y la segunda, floor(), redondea un número al entero inferior.

Otra función de la que la página de la calculadora puede hacer buen uso es abs(), que devuelve el valor absoluto de un número. En caso de que no recuerde los valores absolutos, funcionan así:

```
$Number = abs(-23); // $Number is equal
→ to 23.
$Number = abs(23); // $Number is equal
→ to 23.
```

En términos comunes, el valor absoluto de un número es siempre un número positivo.

Las dos últimas funciones que presentaremos son srand() y rand(). La última es un generador de números aleatorios y la primera es una función que se utiliza para dotar de argumento a rand(). Debe emplear srand() antes de llamar a rand() para garantizar verdaderos números aleatorios. El manual de PHP recomienda el siguiente código:

```
srand ((double) microtime() * 1000000);
$RandomNumber = rand();
```

Es un hecho extraordinario que el manual no tenga respuesta para la pregunta de por qué debería utilizar esa línea exacta para argumentar el generador de números aleatorios, excepto para decir que los creadores de PHP 4 establecen que esto garantiza los números más aleatorios.

La función rand() también puede tomar un parámetro mínimo y uno máximo en caso de que quiera que el número generado se limite a un rango específico.

```
$RandomNumber = rand (0, 10);
```

De las funciones mencionadas aquí, incorporará abs() y round() a numbers.php para evitar entradas erróneas del usuario.

Cómo utilizar funciones matemáticas internas

1. Abra numbers.php en el editor de textos (véase el *Script* 4.4).

2. Después de la línea 9, añada (véase el *Script* 4.5):

```
$Quantity = abs($Quantity);
$Discount = abs($Discount);
```

Si el usuario intenta introducir una cantidad negativa o descuento, la página automáticamente supondrá que es una cantidad positiva y utilizará abs() para ajustarla a la forma correspondiente. También podría redondear el valor de $Quantity si no desea vender una parte de un dispositivo. Mientras vuelve a escribir este *script*, podría ser más cuidadoso e ir un paso más allá aplicando también la función abs() a $Payments.

3. Cambie la línea 14, donde se calcula la variable $Payments, a:

```
$Payments = round ($TotalCost, 2)
→ / 12;
```

Para incrementar la exactitud de la calculadora, los pagos mensuales se basarán en una cantidad real de dólares ($Totalcost redondea a dos posiciones decimales).

4. Guarde el *script*, cárguelo en el servidor y pruébelo en el navegador, utilizando un número negativo para la cantidad o descuento (véase la Figura 4.6).

Script 4.5. Al utilizar las funciones round() y abs() creará cantidades de pago lógicas. De hecho, cada vez que necesite asegurarse de que sean números enteros, utilice round(), y cada vez que necesite garantizar números positivos, utilice abs().

script

```
1  <HTML>
2  <HEAD>
3  <TITLE>Using Numbers</TITLE>
4  </HEAD>
5  <BODY>
6  <?php
7  /* $Quantity se debe pasar a esta página
   desde un formulario o mediante un URL.
   $Discount es opcional. */
8  $Cost = 2000.00;
9  $Tax = 0.06;
10 $Quantity = abs($Quantity);
11 $Discount = abs($Discount);
12 $Tax++; // $Tax tiene ahora el valor de
   1.06.
13 $TotalCost = (($Cost * $Quantity) - $Dis-
   count) * $Tax;
14 $Payments = round ($TotalCost, 2) / 12;
15 // Imprimir los resultados.
16 print ("You requested to purchase $Quantity
   widget(s) at \$$Cost each.\n<P>");
17 print ("The total with tax, minus your
   \$$Discount, comes to $");
18 printf ("%01.2f", $TotalCost);
19 print (".\n<P>You may purchase the
   widget(s)in 12 monthly installments of $");
20 printf ("%01.2f", $Payments);
21 print (" each.\n<P>");
22 ?>
23 </BODY>
24 </HTML>
```

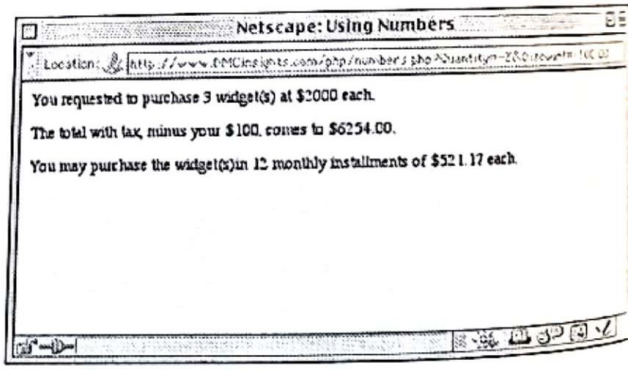


Figura 4.6. La incorporación de las funciones round() y abs() al script lo hacen más profesional, asegurando la coherencia y los resultados significativos.

UTILIZACIÓN DE CADENAS 5

Como se dijo en el Capítulo 2, la segunda categoría de variables son las cadenas (un conjunto de caracteres que van entre comillas dobles o sencillas).

Una cadena puede componerse de una letra, una palabra, una frase, un párrafo o incluso una mezcla sin sentido de letras, números y símbolos (que pueden representar una clave de acceso). Las cadenas son el tipo de variable más utilizado en PHP.

Claves de acceso, nombres, direcciones de correo electrónico, comentarios y entradas similares desde formularios HTML se convierten en cadenas en el *script* PHP como vio al utilizar las páginas form.html y HandleForm.php en el Capítulo 3.

En este capítulo trataremos las funciones internas más básicas que PHP tiene para trabajar con los datos de una cadena, tanto si la cadena se origina desde un formulario como si no. Presentaremos algunas técnicas habituales, como recortar cadenas, unir cadenas y extraer secciones de cadenas, aunque en capítulos posteriores se ilustrarán los mejores usos de las cadenas.

Recorte de cadenas

Tanto por un usuario que introduzca información descuidadamente como por un código HTML desordenado, es bastante común que se añadan espacios extra a una variable de cadena. Con el fin de tener claridad, integridad de datos y diseño web, merece la pena que borre esos espacios de las cadenas antes de utilizarlas. Los espacios adicionales enviados al navegador web podrían hacer que la página pareciera extraña y aquellos enviados a una base de datos o a una *cookie* podrían tener desafortunadas consecuencias posteriormente (por ejemplo, si una clave de acceso tiene un espacio superfluo puede no corresponder cuando se introduzca sin el espacio).

La función trim() elimina automáticamente los espacios extra al principio y al final de una cadena (pero no en el medio). El formato para usar trim() es:

```
$String = " extra space before and
→ after text ";
$String = trim($String);
// $String no es igual a "extra space
→ before and after text".
```

Utilizará el *script* HandleForm.php (véase el *Script* 3.4) del Capítulo 3, y lo modificará recordando esto.

Cómo recortar cadenas

1. Abra HandleForm.php en el editor de textos (véase el *Script* 5.1).
2. En el *Script* 5.1, después de la línea 6 (el comentario), añada las siguientes líneas (véase el *Script* 5.2):

   ```
   $FirstName = trim($FirstName);
   ```

   Recortando la variable $FirstName evita enviar una instrucción al navegador como "Your first name is Larry.", que contenga espacios extraños (véanse las Figuras 5.1 y 5.2).

Script 5.1. *El HandleForm.php original es bastante sencillo y, por tanto, carece de algunas medidas extra para manipular los datos recibidos.*

script

```
1  <HTML>
2  <HEAD>
3  <TITLE>Form Results</TITLE>
4  <BODY>
5  <?php
6  /* Esta página recibe y manipula los datos
   generados por "form.html". */
7  print "Your first name is
   $FirstName.<BR>\n";
8  print "Your last name is
   $LastName.<BR>\n";
9  print "Your E-mail address is
   $Email.<BR>\n";
10 print "This is what you had to say:<BR>\n
   $Comments<BR>\n";
11 ?>
12 </BODY>
13 </HTML>
```

Script 5.2. *Además de recortar los espacios en blanco extra de todos los datos recibidos, hemos cambiado el título de la página, aunque no es importante que lo haga así.*

script

```
1  <HTML>
2  <HEAD>
3  <TITLE>Form Results /Using Strings</TITLE>
4  <BODY>
5  <?php
6  /* Esta página recibe y manipula los datos
   generados por "form.html". */
7  $FirstName = trim($FirstName);
8  $LastName = trim($LastName);
9  $Email = trim($Email);
10 $Comments = trim($Comments);
11 print ("Your first name is
   $FirstName.<BR>\n");
12 print ("Your last name is
   $LastName.<BR>\n");
13 print ("Your E-mail address is
   $Email.<BR>\n");
14 print ("This is what you had to say:
   <BR>\n $Comments<BR>\n");
15 ?>
16 </BODY>
17 </HTML>
```

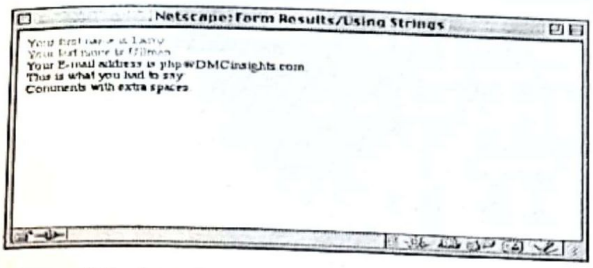


Figura 5.1. Los espacios superfluos son sutiles, pero indican una falta de previsión por parte del desarrollador web. La Figura 5.2, que muestra el código fuente de esta imagen, revela este punto.

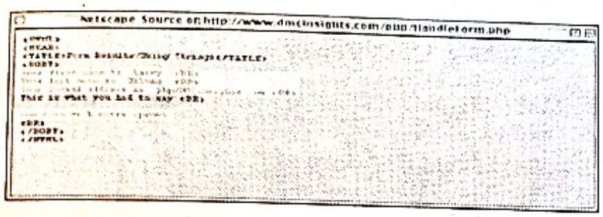


Figura 5.2. Visualizando el código fuente HTML está en disposición de ver los extraños espacios en blanco que se han pasado en el formulario. Aunque no son críticos en este caso, en ocasiones un número de líneas en blanco innecesarias pueden destrozar realmente el diseño HTML de una página.

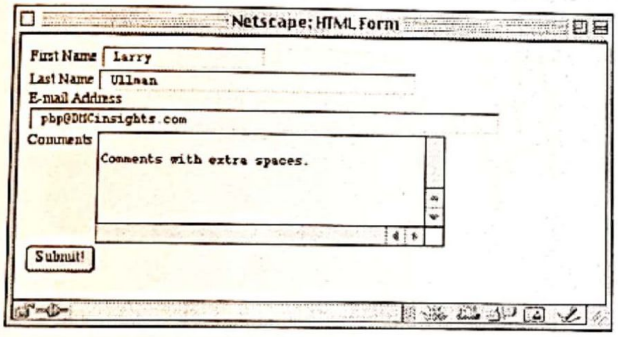


Figura 5.3. Por muchas razones, es muy fácil para el usuario medio añadir espacios en blanco cuando introduce los datos de un formulario. Prepararse para esta posibilidad denota una programación inteligente.

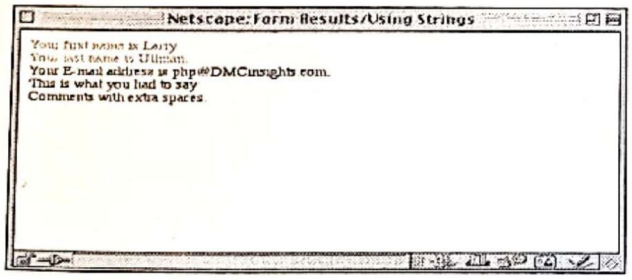


Figura 5.4. La buena programación frecuentemente se reduce a ajustar el número de errores que el usuario puede cometer, intencionadamente o no. Ahora, su página tiene un nivel ligeramente más alto de profesionalidad desde que recortó la entrada del usuario.

3. `$LastName = trim($LastName);`
   `$Email = trim($Email);`

   Recortar la dirección de correo electrónico sería particularmente útil, ya que los espacios añadidos podrían hacer que la dirección no fuera válida para enviar e-mails.

4. `$Comments = trim($Comments);`

   Independientemente de si la cadena se ha extraido de un fragmento de texto (como las variables $FirstName o $Email), deriva de un cuadro de texto HTML o de un párrafo escrito en un área de texto, trim() funcionará de la misma manera.

5. Guarde el *script* (como HandleForm.php), cárguelo en el servidor y pruébelo en el navegador web (véanse las Figuras 5.3, 5.4 y 5.5).

CONSEJO

- Si necesita recortar un exceso de espacios al principio o al final de la cadena, pero no en ambos a la vez, PHP ha dividido la función trim() en dos más específicas: rtrim() eliminará aquellos espacios encontrados al final de una cadena, y ltrim() eliminará los del comienzo. Ambas se utilizan como trim():

```
$String = rtrim($String);
$String = ltrim($String);
```

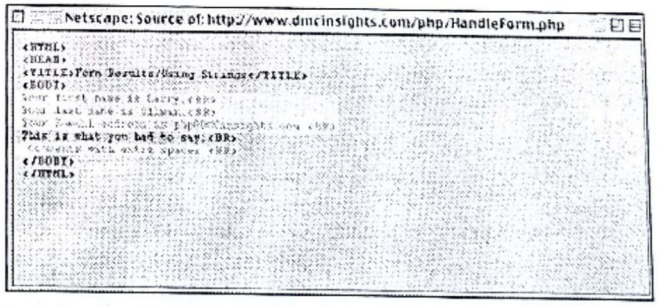


Figura 5.5. Compare el código HTML de la Figura 5.3, mostrado aquí, con el de la Figura 5.2 y verá fácilmente las diferencias que implica la función trim().

Conexión de cadenas (concatenación)

No es un término muy conocido, pero **concatenación** es un concepto útil. Se refiere al proceso de enlazar objetos. Especialmente en programación, se pueden concatenar cadenas. El punto (.) es el operador que realiza esta acción, y se utiliza así:

```
$NewString = $aString . $bString.
```

De esta forma puede enlazar tantas cadenas como desee. Puede incluso unir números a cadenas:

```
$NewString = $aString . $bString .
› $cNumber;
```

Esto funciona porque PHP está débilmente caracterizado, lo que significa que las variables no están bloqueadas en un formato particular. Aquí, la variable $cNumber se convertirá en una cadena y se añadirá al valor de la variable $NewString.

El *script* HandleForm.php contiene cadenas que se podrían concatenar de forma lógica. Es bastante frecuente e incluso se recomienda tener el nombre y el apellido del usuario como entradas separadas, como lo hizo con el formulario. Por otro lado, sería ventajoso poderse referir a los dos juntos como un nombre. Modificará el *script* teniendo esto en cuenta.

Cómo utilizar la concatenación en el *script*

1. Abra HandleForm.php en el editor de textos (véase el *Script* 5.2).

2. Cambie la línea 11 para que se lea:

   ```
   $Name = $FirstName . " " .
   › $LastName;
   ```
   (véase el *Script* 5.3).

   Ya que unirá los dos nombres en uno, no necesitará instrucciones de impresión separadas (líneas 11 y 12 del *Script* 5.2), de modo que sustituirá esta línea y después modificará la siguiente. También puede asegurarse de que la concatenación tenga lugar después de las instrucciones trim(), ya que no puede borrar los espacios extra dentro de una cadena utilizando

Script 5.3. La concatenación es uno de los trabajos más habituales de una variable cadena. Piense en ella como adición de cadenas.

script

```
1  <HTML>
2  <HEAD>
3  <TITLE>Form Results /Using Strings</TITLE
4  <BODY>
5  <?php
6  /* Esta página recibe y manipula
   los datos generados por "form.html". */
7  $FirstName = trim($FirstName);
8  $LastName = trim($LastName);
9  $Email = trim($Email);
10 $Comments = trim($Comments);
11 $Name = $FirstName . " " . $LastName;
12 print ("Your name is $Name.<BR>\n");
13 print ("Your E-mail address is
   $Email.<BR>\n");
14 print ("This is what you had to
   say:<BR>\n $Comments<BR>\n");
15 ?>
16 </BODY>
17 </HTML>
```

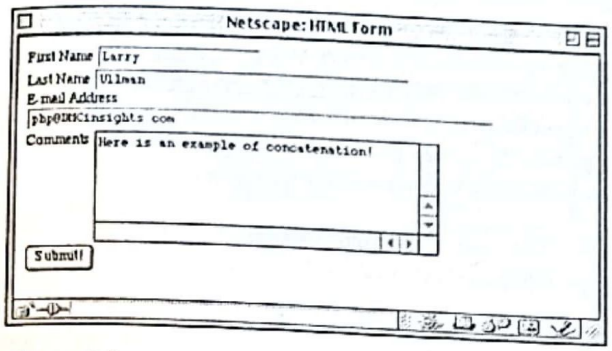


Figura 5.6. Aunque el formulario HTML en sí mismo no cambia, el script HandleForm.php será capaz de crear el nombre completo del usuario a partir de los datos individuales aquí introducidos (véase la Figura 5.7).

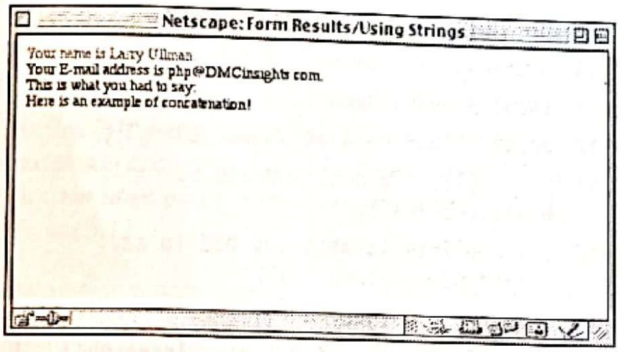


Figura 5.7. Mediante una simple concatenación de dos variables, puede derivar una variable más útil (el nombre completo del usuario).

trim() una vez que haya concatenado los dos nombres. Por último, ha insertado un espacio entre los dos nombres de manera que no se puedan ejecutar juntos.

3. Cambie la línea 12 a:

```
print ("Your name is $Name.<BR>\n");
```

4. Guarde el *script*, cárguelo en el servidor y pruébelo en el navegador web (véanse las Figuras 5.6 y 5.7).

CONSEJOS

- Debido a las características de PHP al tratar las variables, se podría obtener el mismo efecto utilizando $Name = "$FirstName $LastName";. Esto es porque las variables utilizadas dentro de comillas dobles se sustituyen por su valor cuando PHP las manipula. Sin embargo, el método formal de utilizar el punto para concatenar cadenas, como hizo aquí, se utiliza más habitualmente y, por tanto, recomendamos hacerlo de esa manera (lo que sucede en el código será más obvio).
- También podría haber escrito $FirstName = → $FirstName . " " . $LastName;, pero hay dos razones por las que no debería. Primera, hacerlo así habría sobrescrito el valor original de la variable $FirstName. Segunda, FirstName ya no sería una descripción apropiada del valor de la variable. Al programar debería intentar mantener siempre nombres de variables válidos.

Codificación y descodificación de cadenas

Al final del Capítulo 3, mostramos cómo utilizar el concepto que hay tras el método GET para enviar datos a una página añadiéndola al URL.

En aquel momento, sólo empleamos esta técnica para enviar una variable con un valor numérico o de una sola palabra. Utilizamos el mismo proceso en el Capítulo 4, de una forma muy parecida. Pero, ¿qué ocurre si quiere pasar varias palabras como valor de una variable?

Para estos casos dispone de la función urlencode(). Como su nombre indica, esta función toma una cadena y la codifica (cambia la forma) de modo que se puede pasar como parte de un URL. Sustituye los espacios por signos más (+) y convierte los caracteres especiales (por ejemplo, el apóstrofe) en una versión menos problemática. La sintaxis de esta función es:

```
$String = urlencode($String);
```

Para mostrar un uso de esta función, pasará la variable $Name que acaba de crear a una página que saludará al usuario con el nombre completo.

Cómo utilizar urlencode()

1. Abra HandleForm.php en el editor de textos (véase el *Script* 5.3).

2. Después de la línea 14, añada lo siguiente (véase el *Script* 5.4):

```
$Name = urlencode($Name);
print ("<P>Click <A HREF=\
→ "welcome.php?Name=$Name\">
→ here</A> to see your personalized
→ greeting!\n");
```

3. Cree un nuevo documento llamado welcome.php. Puede escribirlo utilizando la información que ha aprendido, o duplicar el código del *Script* 5.5. Esta nueva página recibirá el valor de $Name que le pase el *script* HandleForm.php. Lo usaremos para mostrar cómo puede pasar un valor de una página a otra y a otra (desde

Script 5.4. *Dos cosas a observar: primero, la etiqueta HREF es otro elemento HTML que necesita el uso de comillas dobles, las cuales deben expresarse como secuencias de escape dentro de la instrcción print(), y segundo, PHP sustituirá la variable $Name con su valor cuando se envíe al navegador (véase la Figura 5.10).*

script

```
1  <HTML>
2  <HEAD>
3  <TITLE>Form Results /Using Strings</TITLE>
4  <BODY>
5  <?php
6  /* Esta página recibe y manipula los
   datos generados por "form.html". */
7  $FirstName = trim($FirstName);
8  $LastName = trim($LastName);
9  $Email = trim($Email);
10 $Comments = trim($Comments);
11 $Name = $FirstName . " " . $LastName;
12 print ("Your name is $Name.<BR>\n");
13 print ("Your E-mail address is
   $Email.<BR>\n");
14 print ("This is what you had to say:
   <BR>\n $Comments<BR>\n");
15 $Name = urlencode($Name);
16 print ("<P>Click <A HREF=\"welcome.php?
   Name=$Name\"> here</A> to see your
   personalized greeting!\n");
17 ?>
18 </BODY>
19 </HTML>
```

Script 5.5. *Muy parecido al primer script "Hello, world!" del Capítulo 1, la página welcome.php crea un sencillo saludo ligeramente decorado en el navegador web (véase la Figura 5.11). Sin embargo, este saludo es personalizado.*

script

```
1  <HTML>
2  <HEAD>
3  <TITLE>Welcome!</TITLE>
4  <BODY>
5  <?php
6  print ("<B><CENTER>Hello, $Name.</CENTER>
   </B>\n");
7  ?>
8  </BODY>
9  </HTML>
```

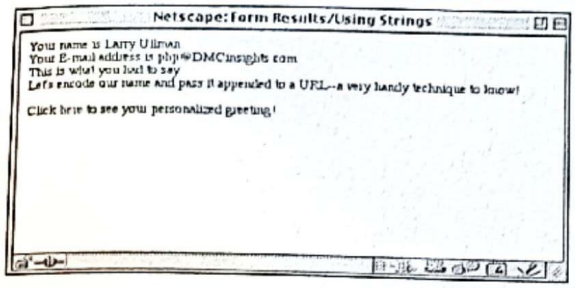


Figura 5.8. Si no crea la secuencia de escape de las comillas dobles en HREF, la línea añadida no aparecerá correctamente.

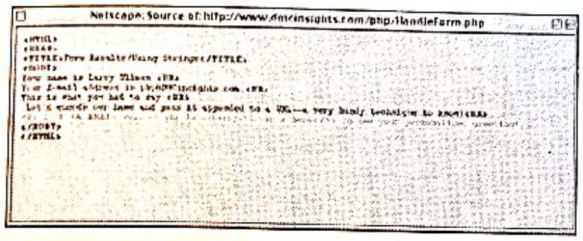


Figura 5.9. Éste es el código fuente HTML de la página PHP generada. Observe cómo el nombre codificado aparece añadido al enlace. Si no utiliza la función urlencode(), el resultado sería como el de la Figura 5.12.

Figura 5.10. La página PHP generada dinámicamente recibe una variable no desde un formulario, sino desde otro script PHP: una técnica común y muy útil.

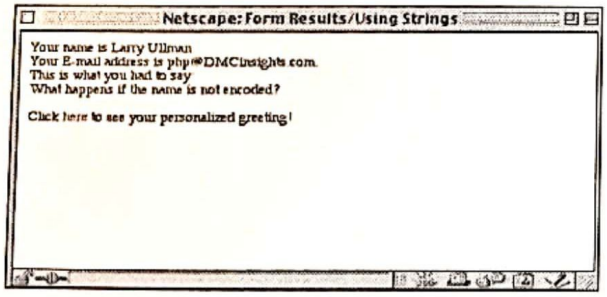


Figura 5.11. Sólo mirando la página, no puede saber si el valor del nombre se ha codificado correctamente o no. Sin embargo, el código fuente HTML (véase la Figura 5.12) y la página welcome.php (véase la Figura 5.13) revelarán cualquier problema.

form.html a HandleForm.php y a welcome.php).

4. Guarde ambos *scripts*, cárguelos en el servidor y pruébelos en el navegador web (véanse las Figuras 5.8, 5.9, 5.10, 5.11, 5.12, 5.13 y 5.14).

Figura 5.12. No será obvio hasta que llegue a la página weolcome.php (véase la Figura 5.13), pero el espacio entre "Larry" y "Ullman" provocará que el apellido salte de una página a otra, por lo que debe codificar las variables en el URL para pasarlas.

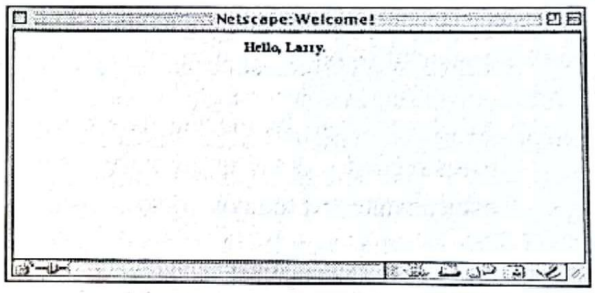


Figura 5.13. Si una página PHP recibida sólo imprime la primera palabra de una cadena, se debe a que la cadena no se ha pasado en un formato codificado.

Figura 5.14. Uno de los caracteres especiales que la función urlencode() dirige es el apóstrofe, traducido aquí a su equivalente ASCII, 0%27.

CONSEJOS

- La función urldecode() hace lo contrario de urlencode(), toma un URL codificado y lo devuelve en formato estándar.
- Recuerde que los valores enviados directamente desde un formulario son automáticamente codificados al URL antes de ser enviados y descodificados en la llegada al *script* receptor. La función urlencode() sólo es necesaria cuando debe codificar datos de forma manual (como en el ejemplo).
- En el Capítulo 11, utilizaremos funciones parecidas a urlencode(), addslashes() y stripslashes(). Lo anterior prepara los datos para la entrada en una base de datos creando como secuencias de escape los caracteres problemáticos (comillas simples, comillas dobles y la barra invertida). El último elimina las secuencias de escape de esos mismos caracteres. La sintaxis es exactamente como esperaría:

```
$Data = addslashes($Data);
$Data = stripslashes($Data);
```

Así como con la función urlencode(), los datos recibidos desde un formulario son automáticamente expresados como secuencias de escape para entradas de bases de datos apropiadas.

Encriptado y desencriptado de cadenas

Por razones de seguridad, el encriptado y desencriptado son un aspecto necesario en la mayoría de los sitios web, especialmente aqueéllos dedicados al comercio electrónico. Frecuentemente, para proteger los datos, los programadores los encriptan y alteran su estado, transformándolos a una forma que resulte más difícil, si no virtualmente imposible, de entender. Las claves de acceso son un ejemplo de una variable que se puede encriptar. Dependiendo del nivel de seguridad que quiera establecer, nombres de usuarios, direcciones electrónicas y números de teléfono son también candidatos al encriptado.

Observe, sin embargo, que sólo encriptar los datos no garantiza la seguridad de un sitio web. Primero, con suficiente esfuerzo, los datos encriptados se pueden descomponer. Segundo, ya que PHP es una tecnología del lado del servidor, sólo se puede introducir un nivel de seguridad una vez que el módulo PHP en el servidor haya recibido la información. Mientras está en tránsito desde la computadora del usuario al servidor, los datos se deben asegurar empleando otros métodos (como una conexión de *socket* seguro), si la seguridad es conveniente. Consulte el Apéndice B, para obtener más información y recursos sobre este tema.

Presentaremos tres funciones para encriptar y desencriptar las cadenas e incorporará una al *script* como demostración.

La primera función, crypt() se empleará para encriptar datos, pero observe que no hay opción disponible de desencriptado. De manera que puede encriptar una clave de acceso utilizando esta función y después puede almacenarla, pero no podrá determinar el valor desencriptado de la clave. Usando esto en una aplicación web, puede tener una clave de usuario encriptada en el registro y cuando el usuario se conecte, la clave introducida en ese momento también será encriptada, comparándose las dos versiones protegidas de la clave de acceso. La sintaxis para la utilización de crypt() es:

```
$Data = crypt($Data);
```

Una segunda función de encriptación es encrypt(), que puede desencriptarse utilizando la función llamada decrypt() (observe que crypt() y encrypt() son dos funciones distintas). Desafortunadamente, para poder utilizar estas dos funciones debe estar instalada la extensión crypt con el módulo PHP.

Deberá preguntar al ISP para ver si éste es el caso. No se sorprenda si estas dos funciones no están disponibles y por tanto está limitado al encriptado crypt() sin la posibilidad de desencriptado. (De hecho, debería acostumbrarse a la idea de que el ISP no soporte cualquier número de características, lo que se convertirá en un límite a lo que puede hacer. Un programador avanzado sabrá cómo salvar aquellos obstáculos inevitables.)

Cómo encriptar datos utilizando crypt()

1. Abra HandleForm.php en el editor de textos (véase el *Script* 5.4).
2. Sustituya las líneas 15 y 16 (donde utilizó urlencode() para pasar el nombre de usuario a la página welcome.php) con (véase el *Script* 5.6):

```
$CryptName = crypt($Name);
print ("<P>This is the crypt()
→ version of your name:
→ $CryptName\n");
```

Script 5.6. Hemos añadido las líneas crypt() solo para mostrar cómo funciona. Aunque esto en realidad no es una utilización productiva de la tecnología, revela rápidamente cómo funciona la codificación con cadenas.

script

```
1  <HTML>
2  <HEAD>
3  <TITLE>Form Results /Using Strings</TITLE>
4  <BODY>
5  <?php
6  /* Esta página recibe y manipula los
   datos generados por 'form.html'. */
7  $FirstName = trim($FirstName);
8  $LastName = trim($LastName);
9  $Email = trim($Email);
10 $Comments = trim($Comments);
11 $Name = $FirstName . ' ' . $LastName;
12 print ('Your name is $Name.<BR>\n');
13 print ('Your E-mail address is
   $Email.<BR>\n');
14 print ('This is what you had to say:
   <BR>\n $Comments<BR>\n');
15 $CryptName = crypt($Name);
16 print ("<P>This is the crypt() version
   of your name: $CryptName\n");
17 ?>
18 </BODY>
19 </HTML>
```

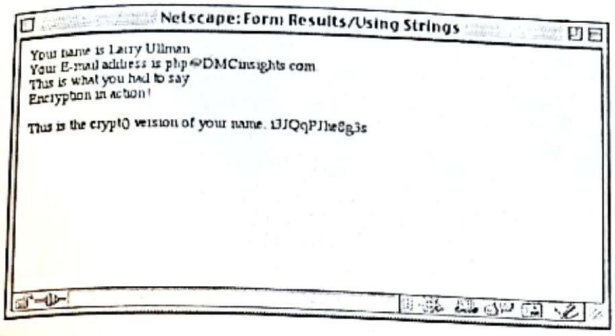


Figura 5.15. La función crypt() devuelve una versión larga de 12 caracteres única, de cualquier cosa que codifique.

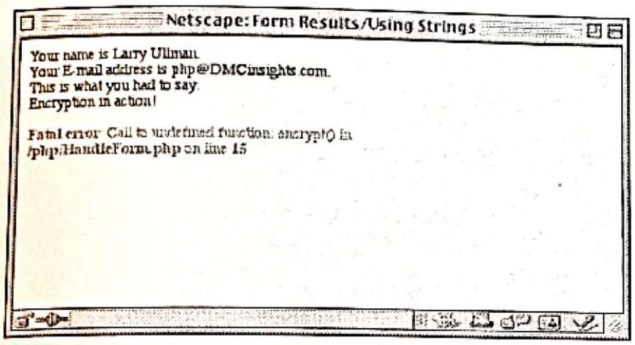


Figura 5.16. Hemos añadido una llamada a la función encrypt() en la página HandleForm.php, lo que provoca este mensaje de error, indicando que el servidor no soporta la función mencionada.

No hay razón para encriptar una versión codificada a un URL de su nombre, de manera que se ha sustituido esa parte del *script*.

3. Guarde el *script*, cárguelo en el servidor y pruébelo en el navegador (véase la Figura 5.15).

CONSEJOS

- Hay una función de encritptado parecida a crypt() llamada md5(), que se empleará más adelante en este capítulo. Para conocer más sobre su uso, así como de la sintaxis para encrypt() y decrypt(), consulte el Apéndice C.
- Una forma directa de saber si una función no está soportada por la instalación PHP en el servidor es que aparezca un mensaje de error como el de la Figura 5.16, derivado de la llamada de la función encrypt() en un servidor que no la soporta. Con precaución, antes de suponer que algo no tiene soporte, compruebe el deletreo y la sintaxis. (En la figura, si llamásemos a la función encrpyt() se generaría un mensaje de error similar.)
- Tenga cuidado de no confundir encriptar una cadena con codificarla. El encriptado se utiliza con fines de seguridad y cambia el texto completo. La codificación sólo sustituye ciertos caracteres con equivalentes que se pueden utilizar dentro de un URL.

Sustracción de partes de una cadena

Anteriormente en este capítulo aprendió a unir cadenas utilizando la concatenación. Además de hacer cadenas más largas a partir de cadenas más cortas, también puede extraer secciones de una cadena. Hay varias funciones internas que hacen esto de diferentes formas, y aquí trataremos dos. El truco de emplear un método para extraer una parte de una cadena es que debe saber algo acerca de la cadena en sí misma para hacerlo de forma eficaz.

La función strtok() crea una subcadena, denominada **muestra**, a partir de una cadena más larga basada en un separador predeterminado (quizá una coma o un espacio). Por ejemplo, si tiene usuarios que introducen el nombre completo en un campo (presumiblemente con el nombre y apellido separados por un espacio), puede averiguar el nombre con el código:

```
$FirstName = strtok($Name, " ");
```

$Name es el nombre de la variable recibida desde el formulario que contiene el nombre completo del usuario. Esta línea del programa le dice a PHP que saque todo desde $Name hasta que encuentre un espacio en blanco. Si el usuario introdujera el nombre completo en el formato LastName, FirstName, podría encontrar su apellido escribiendo:

```
$LastName = strtok($Name, ",");
```

Como ha guardado sabiamente los dos nombres en campos distintos en el formulario, no tendrá que emplear por el momento esta función.

Un segundo método para extraer partes de una cadena es refiriéndose a la posición indexada de los caracteres de la cadena. Cuando nos referimos al indexado de una cadena, queremos indicar la localización numérica de un carácter, contando desde el principio. Sin embargo, PHP, como la mayoría de los lenguajes de programación, comienza todos los índices con el número 0. De modo que para indexar la cadena "Larry", comenzaría con "L" en la posición 0, "a" en la primera, "r" en la 2, la segunda "r" en la tercera y la "y" en la cuarta. Incluso aunque la longitud de la cadena "Larry" sea 5, el indexado va de 0 a 4. Teniendo esto en cuenta, puede

utilizar la función substr() para crear una subcadena basada en la posición indexada de los caracteres de la subcadena, así:

```
$SubString = substr($String,0,10);
```

Primero, debe introducir la cadena maestra desde la que derivará la subcadena (aquí, $String). Segundo, indique doónde empieza la subcadena, así como la posición indexada (0, si quiere empezar con el primer carácter). Tercero, determine cuántos caracteres, desde el punto inicial, compondrán la subcadena (10). Si, en este caso, $String no tiene diez caracteres, la $SubString resultante terminará al final de $String.

Frecuentemente utilizará la función strlen() para determinar el punto final de la cadena. Esta función calcula la longitud de una cadena (es decir, el número de caracteres que contiene) y es muy fácil de usar:

```
$StringLength = strelen($String);
```

Utilizará la noción de substr(), con strlen() y md5(), para escribir un ingenioso programa de generación de claves de acceso.

Las claves de acceso que utilice serán más seguras cuanto más aleatorias sean. De hecho, las claves que no tienen ningún significado especial, no se basan en palabras comunes y utilizan una variedad de letras mayúsculas, minúsculas y números, son las más seguras. Este *script* creará una nueva clave aleatoria cada vez que cargue la página.

Cómo crear un generador de claves en PHP

1. Comience creando un nuevo documento PHP en el editor de textos con las líneas (véase el *Script 5.7*)

```
<HTML><HEAD><TITLE>Password
- Generator</TITLE><BODY><?php
$String = 'This is the text which
- will be encrypted so that we may
- create random and more secure
- passwords!';
```

Puede poner aquí cualquier cadena si lo desea, el texto específico no será una diferencia para nuestros fines. Este texto se encriptará para crear una cadena más aleatoria desde la que se tomará la clave.

```
$Length = 8;
```

Configurando la longitud de la clave como una variable, puede sencillamente cambiar este valor para obtener diferentes tamaños de clave. El límite para una clave es de treinta y dos caracteres, que es la longitud de una cadena encriptada utilizando

```
md5().$String = md5($String);
```

El uso de md5() es similar al de crypt(), pero generará una cadena de treinta y dos caracteres, en oposición a crypt(), que crea una cadena de doce caracteres. Consulte el manual de PHP o la página principal PHP para obtener más información sobre md5().

```
$StringLength = strlen($String);
```

Para extraer una subcadena, necesitará saber la longitud de la cadena encriptada, de modo que utilizaremos la función strlen(), que devuelve el número de caracteres que hay en la cadena dada. Aunque de hecho sepa que utilizando md5() la longitud de la cadena será siempre de treinta y dos caracteres, es mejor asegurarse, por lo que utilizará la función strlen(). De esta forma, podría cambiar sus métodos en el futuro (por ejemplo, utilizando encrypt() en lugar de md5()), pero esta línea seguiría funcionando correctamente.

Script 5.7. Le puede parecer que hizo un par de trucos para llegar a este punto, pero el script posterior es sólo un útil conglomerado de lo que ha aprendido a este punto. ¡Y todo en 16 líneas!

```
1  <HTML>
2  <HEAD>
3  <TITLE>Password Generator</TITLE>
4  <BODY>
5  <?php
6  $String = 'This is the text which will b
   encrypted so that we may create rando
   and secure passwords!';
7  $Length = 8; // Change this value to
   indicate how long your password should
   be. 32 character limit.
8  $String = md5($String);
9  $StringLength = strlen($String);
10 srand ((double) microtime() * 1000000);
11 $Begin = rand(0,($StringLength-$Length-
   // Seleccionar un punto de
   // inicio arbitrario.
12 $Password = substr($String, $Begin,
   $Length);
13 print ('Your recommended password
   is:<P><BIG>$Password</BIG>\n');
14 ?>
15 </BODY>
16 </HTML>
```

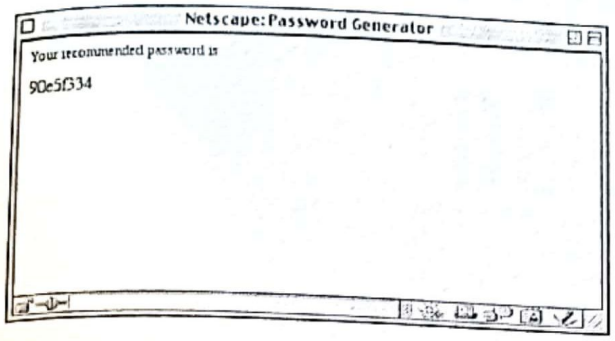


Figura 5.17. La página passwords.php realiza el trabajo de crear sus propias contraseñas seguras.

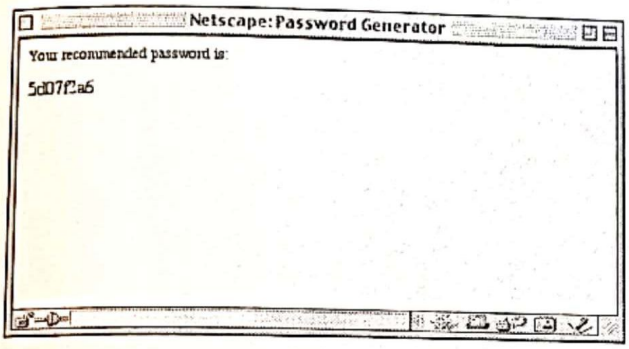


Figura 5.18. Cada vez que se vuelve a cargar la página, se distribuirá una nueva y distinta contraseña. (En la actualidad, debido al hecho de que md5() crea una cadena de 32 caracteres, existe un número finito de contraseñas únicas que este script puede generar, pero creará docenas de contraseñas diferentes.)

```
srand ((double) microtime() *
→ 1000000);
$Begin = rand(0,($StringLength -
→ $Length - 1));
```

Necesita determinar el punto inicial para la función substr(). La función rand() crea un número aleatorio entre (e incluido) un mínimo (aquí es 0) y un máximo. [Recuerde emplear primero la línea srand() o puede que no obtenga verdaderos resultados aleatorios a partir de rand()]. Ha configurado que el máximo sea la longitud de la cadena menos la longitud de la clave menos uno. Ésta es la razón: si la cadena encriptada es de treinta caracteres y necesita que las claves sean de ocho caracteres, entonces el último por el que puede empezar substr() y aún tener una clave de ocho caracteres es veintiuno (el cual se refiere al carácter en la posición veintidós de la cadena, debido al comportamiento de la indexación). Ponemos el cálculo entre paréntesis para que sea más claro, pero rand() no lo necesita.

```
$Password = substr($String, $Begin,
→ $Length);
```

El paso final es llamar a substr() con los valores determinados. Establecerá que la variable $Password es igual a una subcadena derivada de la variable $String, comenzando con la posición indexada $Begin y continuando con el número de caracteres indicado por $Length.

```
print ("Your recommended password is:
→ <P><BIG>$Password</BIG>\n");
```

Ahora imprimiremos los resultados (categóricamente).

2. Cierre el código PHP y HTML con:

```
?></BODY></HTML>
```

3. Guarde el *script* como passwords.php, cárguelo en el servidor y pruébelo en el navegador (véanse las Figuras 5.17 y 5.18).

CONSEJOS

- En lugar de configurar la variable $Length al principio de la página, podría utilizar el truco URL/GET para enviar a la página un valor de longitud añadiendo ?Length=8 al URL (si lo hace así, asegúrese de eliminar la línea 7).
- Las bases de datos tienen un formato de datos que almacena las fechas como YYYY-MM-DD (o algo parecido). Ya que sabrá el formato exacto de la cadena recuperada de la base de datos, las subcadenas año, mes y día se podrían calcular utilizando substr():

```
$Year = substr($Date,0,4);
$Month = substr($Date,5,2);
$Day = substr($Date,8,2);
```

- El empleo de strlen() es para una situación en la que no quiera escribir $Password = strlen ($Password);, en cuyo caso deberá sustituir el valor actual de la clave por un número que indique cuántos caracteres había, pero ya no existen, en la variable.

ESTRUCTURAS DE CONTROL 6

Ahora que hemos abordado las bases de los tipos de variables (a parte de las tablas, que trataremos en el Capítulo 7), podemos entrar en el tema de la programación (las estructuras de control, es decir, condiciones y bucles). En el proceso, terminaremos la explicación de varios usos de los operadores de PHP. (Ya hemos abordado la aritmética y la asignación en los capítulos dedicados a las variables.)

Las estructuras condicionales son necesarias en los lenguajes de programación. Permiten establecer un parámetro y realizar una acción basada en los resultados de ese parámetro. Esto le permite hacer los sitios web más dinámicos. Por ejemplo, si la hora del día es antes del mediodía, puede saludar al usuario con la expresión "¡Buenos días", si es después, puede imprimir "¡Buenas tardes!". En PHP hay dos condicionales generales, if y switch, que usará en este capítulo.

Al hablar de condicionales, presentaremos dos categorías de operadores, de comparación y lógicos. Habitualmente los utilizará en los condicionales junto a los conceptos PHP de TRUE y FALSE.

Finalmente, comenzará a programar con bucles, que le permitirán repetir una acción un número específico de veces. Los bucles pueden ahorrarle tiempo de programación y le ayudarán a obtener la máxima funcionalidad en las tablas, como verá en el siguiente capítulo. PHP soporta dos construcciones principales de bucle, while (y el equivalente do...while) y for.

La condición if

El estructura condicional básica en programación es el if estándar (que solía llamarse condición if-then; el then ahora está implícito).

La sintaxis de esta condición es muy sencilla:

```
if (condición) {
    instrucciones;
}
```

La condición debe ir entre paréntesis y el área de instrucciones debe comenzar después de un corchete. La sección de instrucciones condicionales está donde se sitúan los comandos ejecutables (por ejemplo, imprimir una cadena, añadir dos números juntos, etc.). Cada instrucción separada (o comando) debe tener un punto y coma que indique el final de la línea de comando; sin embargo no hay límite para el número de instrucciones que escriba como resultado de una condición. Cierre después la sección de instrucciones con otro corchete. Habitualmente los programadores sitúan estas instrucciones en sangría desde la línea inicial if para indicar que son el resultado de una condición, pero esto no es necesario sintácticamente. El fallo al usar un punto y coma después de cada instrucción, olvidar un paréntesis de apertura o cierre o un corchete, o utilizar un punto y coma después de los corchetes o los paréntesis, puede dar lugar a errores.

PHP utiliza los conceptos de TRUE y FALSE para determinar si se ejecutan o no las instrucciones. Si la condición es TRUE, se ejecutarán las instrucciones; si es FALSE, no se ejecutarán. La siguiente sección explica TRUE y FALSE con más detalle.

Para empezar a trabajar con la condición if, vuelva a escribir la página calculador del Capítulo 4, de modo que funcione sólo si se ha presentado la cantidad necesaria. Esto evitará que los cálculos se hagan sin todos los datos requeridos, lo que provocaría resultados ilógicos y erróneos en la página web.

Script 6.1. En el script numbers.php original, los cálculos se hacen y los resultados se imprimen basándose en la suposición de que los valores $Quantity y $Discount se reciben, lo que no es una buena práctica en programación.

script

```
1  <HTML>
2  <HEAD>
3  <TITLE>Using Numbers</TITLE>
4  </HEAD>
5  <BODY>
6  <?php
7  /* $Quantity se debe pasar a esta
   página desde un formulario o mediante
   un URL. $Discount es opcional. */
8  $Cost = 2000.00;
9  $Tax = 0.06;
10 $Quantity = abs($Quantity);
11 $Discount = abs($Discount);
12 $Tax++; // $Tax is now worth 1.06.
13 $TotalCost = (($Cost * $Quantity) -
   $Discount) * $Tax;
14 $Payments = round ($TotalCost, 2) / 12;
15 // Imprimir los resultados.
16 print ("You requested to purchase
   $Quantity widget(s) at \$$Cost
   each.\n<P>");
17 print ("The total with tax, minus your
   \$$Discount, comes to $");
18 printf ("%01.2f", $TotalCost);
19 print (".\n<P>You may purchase the
   widget(s)in 12 monthly installments of $");
20 printf ("%01.2f", $Payments);
21 print (" each.\n<P>");
22 ?>
23 </BODY>
24 </HTML>
```

Script 6.2. El uso frecuente de estructuras condicionales hace la programación más profesional configurando parámetros particulares antes de abordar los procesos correspondientes. Aquí asegurará que se recibió una $Quantity antes de proceder con los cálculos.

```
script
1  <HTML>
2  <HEAD>
3  <TITLE>Conditionals</TITLE>
4  </HEAD>
5  <BODY>
6  <?php
7  /* $Quantity se debe pasar a esta página
   desde un formulario o mediante un URL.
   $Discount es opcional. */
8  $Cost = 2000.00;
9  $Tax = 0.06;
10 if ($Quantity) {
11     $Quantity = abs($Quantity);
12     $Discount = abs($Discount);
13     $Tax++; // $Tax is now worth 1.06.
14     $TotalCost = (($Cost * $Quantity) -
       $Discount) * $Tax;
15     $Payments = round ($TotalCost, 2) / 12;
16     // Imprimir los resultados.
17     print ("You requested to purchase
       $Quantity widget(s) at \$$Cost each.
       \n<P>");
18     print ("The total with tax, minus
       your \$$Discount, comes to $");
19     printf ("%01.2f", $TotalCost);
20     print (".\n<P>You may purchase the
       widget(s)in 12 monthly installments
       of $");
21     printf ("%01.2f", $Payments);
22     print (" each.\n<P>");
23 }
24 ?>
25 </BODY>
26 </HTML>
```

Cómo crear una condición if

1. Abra numbers.php (véase el *Script* 6.1) en el editor de textos.
2. Cambie el título (línea 3) a

   ```
   <TITLE>Conditionals</TITLE>
   ```

   (véase el *Script* 6.2).
3. Cambie las líneas 10 a 23 para que sean el resultado de una condición if.

   ```
   if ($Quantity) {
       $Quantity = abs($Quantity);
       $Discount = abs($Discount);
       $Tax++; // $Tax is now worth
       → 1.06.
       $TotalCost = (($Cost *
       → $Quantity) - $Discount) * $Tax;
       $Payments = round ($TotalCost, 2)
       → / 12;
       // Imprimir los resultados.
       print ("You requested to purchase
       → $Quantity widget(s) at \$$Cost
       → each.\n<P>");
       print ("The total with tax,
       → minus your \$$Discount, comes
       → to $");
       printf ("%01.2f", $TotalCost);
       print (".\n<P>You may purchase
       → the widget(s) in 12 monthly
       → installments of $");
       printf ("%01.2f", $Payments);
       print (" each.\n<P>");
   }
   ```

   En PHP, utilizar sencillamente un nombre de variable como condición (aquí lo tiene con $Quantity) es lo mismo que decir: "si la variable $Quantity existe (es decir, tiene un valor distinto a cero)". Aquí está diciendo a PHP que ejecute las siguientes líneas sólo si $Quantity tiene un valor distinto de cero.

   En otras palabras, los cálculos sólo se realizarán si hay una cantidad que utilizar.

4. Guarde el *script*, cárguelo en el servidor y pruébelo en el navegador web con la información necesaria $Quantity y sin ella (véanse las Figuras 6.1y 6.2).

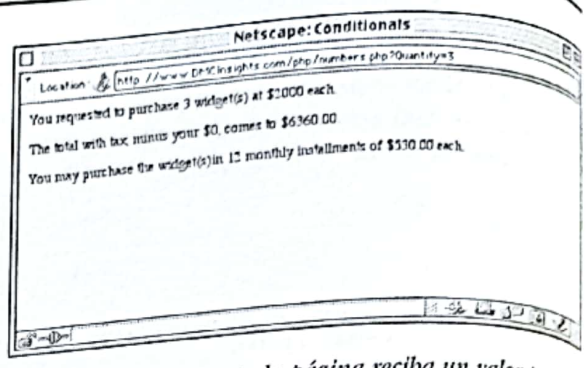


Figura 6.1. Tan pronto la página reciba un valor para la variable $Quantity, procederá como antes de añadir el condicional if.

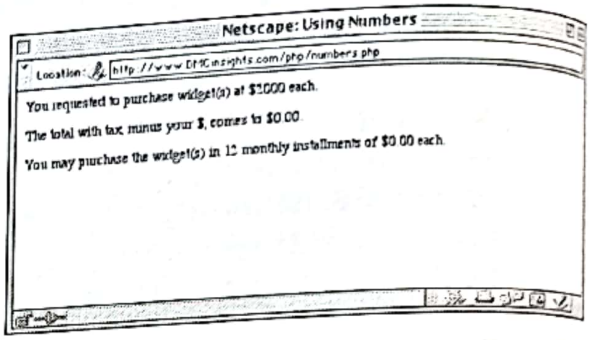


Figura 6.2. Gracias al condicional if, el sitio web nunca volverá a producir resultados indeseados como éstos.

CONSEJOS

- Si el área de instrucciones de la condición sólo tiene una línea, técnicamente no necesitará los corchetes. Si éste es el caso, puede sustituir la condición por una línea como ésta:

  ```
  If (condición) instrucción;
  ```

  Mencionamos este hecho aquí porque puede encontrar el código en este formato. Sin embargo, le sugerimos que utilice siempre el formato multilínea (como muestra la introducción de la sintaxis anterior) para mejorar la coherencia y minimizar los errores.

- En PHP puede anidar sin problema las condiciones if, situando un segundo if como resultado de una condición anterior, siempre que recuerde cerrar las condiciones adecuadamente.

- Posteriormente en este mismo capítulo, aprenderá a ejecutar comandos si no se cumple una condición. Después, si no se recibe el valor adecuado, puede programar el *script* para solicitarlo de forma específica con el fin de proceder.

- También puede utilizar la función isset() para determinar si existe una variable. A diferencia de referirse a una variable por el nombre, como ha hecho aquí, isset() devolverá TRUE si la variable es igual a cero. Por ejemplo:

  ```
  $Quantity = 0;
     if ($Quantity) { ... // FALSE
    if (isset($Quantity)) { ... // TRUE
  ```

Más operadores

Explicamos la mayoría de los operadores de PHP en capítulos anteriores junto con las variables que los utilizan. Estos operadores incluyen aritmética para los números: suma (+), resta (-), multiplicación (*) y división (/), junto con los accesos directos incremental (++) y decremental (--) para incrementar o disminuir el valor de un número en uno. También está el operador de asignación (=), que se usa para establecer el valor de la variable, sin importar el tipo. También introdujimos las concatenación (.), para añadir una cadena a otra.

Todos son útiles para establecer el valor de una variable, pero no resultan de mucha utilidad con las estructuras condicionales. Ahora exploraremos los operadores de comparación y lógicos para trabajar con condiciones.

Comparación

La primera vez que introdujimos el operador de asignación (=), en el Capítulo 2, explicamos cuidadosamente que su significado no es exactamente lo que convencionalmente pensaría. En la línea `$Variable = 5;` no establecerá que `$Variable` "es igual a" 5 si no que "se configura al valor de" 5.

Al escribir estructuras condicionales, a menudo deseará ver si una variable es igual a un valor específico (quizá, para hacer corresponder nombres de usuarios o claves de acceso), lo que no puede hacer con el signo igual solo (ya que es para asignar un valor, no para comparar valores). Para este fin tiene el operador igual (==), que se crea usando dos signos iguales juntos.

```
$Variable = 5;
$Variable == 5;
```

Emplear en primer lugar estas dos líneas de código juntas configura el valor de $Variable a 5, después de hacer una instrucción TRUE en la que se ve si $Variable es igual a 5. Esto muestra la significativa diferencia que un signo igual más produce en el código PHP y por qué deseará distinguir atentamente entre los operadores de comparación y asignación.

No igual a, en PHP, se representa por un signo de exclamación unido a un signo igual (!=). De hecho, la exclamación, en general, significa "no". De modo que $Variable significa "$Variable existe y tiene un valor (diferente de cero)"; por contra, !$Variable es la forma de decir que "$Variable no existe y no tiene valor (o un valor cero)".

Los restantes operadores de comparación son idénticos a sus equivalentes matemáticos: menor que (<), mayor que (>), menor o igual a (<=) y mayor o igual a (>=).

Para añadir funcionalidad al calculador de dispositivos, volverá a escribir el *script* numbers.php de manera que sólo aplique un descuento para ventas superiores a 50 dólares.

Cómo utilizar operadores de comparación

1. Abra numbers.php (véase el *Script* 6.2) en el editor de textos.
2. Cambie el coste del dispositivo a un número mucho menor, de modo que sea difícil alcanzar el límite de 50 dólares (véase el *Script* 6.3, línea 8).

   ```
   $Cost = 20.00;
   ```

3. Después de la línea $Tax++; (línea 13), cambie la ecuación $TotalCost para que diga: $TotalCost = ($Cost * $Quantity);

   Como el descuento sólo se aplicará si el total es superior a 50 dólares, primero debería calcular el total por separado.

Script 6.3. Los operadores comparativos como menor o igual a (<=) aquí le permitirán especificar mejores condiciones numéricas en el código.

script

```
1  <HTML>
2  <HEAD>
3  <TITLE>Conditionals</TITLE>
4  </HEAD>
5  <BODY>
6  <?php
7  /* $Quantity se deba pasar a esta página
   desde un formulario o mediante un URL.
   $Discount es opcional. */
8  $Cost = 20.00;
9  $Tax = 0.06;
10 if ($Quantity) {
11     $Quantity = abs($Quantity);
12     $Discount = abs($Discount);
13     $Tax++; // $Tax is now worth 1.06.
14     $TotalCost = ($Cost * $Quantity);
15         if ($TotalCost >= 50) {
16             $TotalCost = $TotalCost -
               $Discount;
17         }
18     $TotalCost = $TotalCost * $Tax;
19     $Payments = round ($TotalCost, 2) / 12;
20     // Imprimir los resultados.
21     print ("You requested to purchase
       $Quantity widget(s) at \$$Cost
       each.\n<P>");
22     print ("The total with tax, minus your
       \$$Discount, comes to $");
23     printf ("%01.2f", $TotalCost);
24     print (".\n<P>You may purchase the
       widget(s)in 12 monthly installments of
       $");
25     printf ("%01.2f", $Payments);
26     print (" each.\n<P>");
27 }
28 ?>
29 </BODY>
30 </HTML>
```

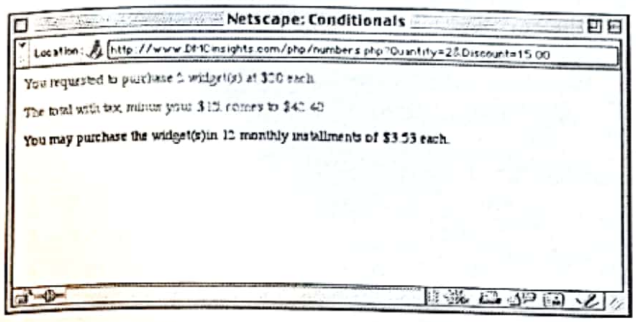


Figura 6.3. Si hace los cálculos pertinentes, verá que el descuento no se aplica porque el total es inferior a 50 dólares. Compare esto con los cálculos de la Figura 6.4.

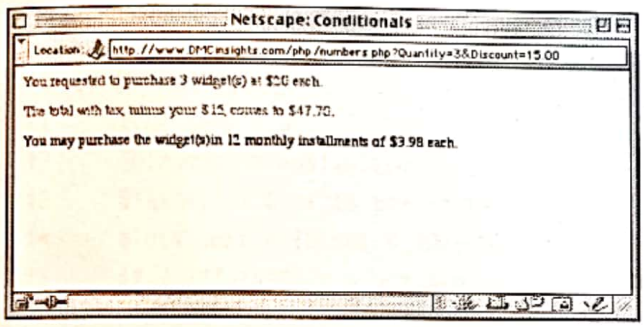


Figura 6.4. Ahora que el usuario ha comprado suficientes dispositivos como para que el total sea superior a 50 dólares, se ha aplicado el descuento.

4. Cree una comprobación condicional if para ver si el valor de la venta es superior a 50 dólares.

```
if ($TotalCost >= 50) {
    $TotalCost = $TotalCost -
    → $Discount;
}
```

Utilizando operadores comparativos dentro de la condición de if, puede especificar que el descuento sólo se aplicará si el valor de $TotalCost es mayor o igual a 50 dólares (no escriba el signo dólar en la condición). La resta de $Discount es la instrucción que se ejecuta si la condición es TRUE. Si la condición es FALSE, no se restará el descuento.

5. Ahora aplique la tasa al coste total.

```
$TotalCost = $TotalCost * $Tax;
```

6. El resto de este *script* está sin cambiar desde antes, incluyendo la determinación de los pagos mensuales y la impresión de los resultados.

7. Guarde el *script*, cárguelo en el servidor y pruébelo en el navegador web utilizando diferentes valores para $Quantity (véanse las Figuras 6.3 y 6.4).

CONSEJO

- En una condición if, si comete el error de escribir `$Variable = 5` en oposición a `$Variable == 5`, verá que las correspondientes instrucciones condicionales siempre se ejecutarán. Esto es porque, mientras la condición $Variable == 5 puede o no puede ser TRUE, la condición $Variable = 5 siempre será TRUE.

Lógicos

Los operadores lógicos le ayudarán a crear construcciones razonables, instrucciones que tienen un valor de TRUE o FALSE. En PHP, una condición es TRUE si es sencillamente un nombre de variable y esta variable tiene un valor distinto de cero (como ya ha visto), como:

```
$Variable = 5;
if ($Variable) { ...
```

Una condición también es TRUE si tiene sentido lógico, como:

```
if (5 >= 3) { ...
```

La condición será FALSE si se refiere a una variable y esta variable no tiene valor o si ha creado una construcción ilógica. La condición siguiente siempre será falsa:

```
if (5 <= 3) { ...
```

Para ir más allá de sencillas condiciones de una parte, PHP soporta seis tipos de operadores lógicos: dos versiones de and (AND o &&); dos versiones de or (OR o ||, un carácter llamado canalización puesto dos veces); not (NOT); y or not (XOR). Cuando hay dos opciones para un operador (como con and y or), sólo difieren en la precedencia. Consulte el Apéndice C, para ver la tabla de precedencia de operadores.

Utilizando paréntesis y operadores lógicos, puede crear condicionales más profundas para las condiciones if. Para una condicional AND, cada parte unida debe ser TRUE. Con OR, una subsección debe ser TRUE para que la condición completa sea TRUE. Estas condiciones son TRUE:

```
if ( (5 <= 3) OR (5 >= 3) ) { ...
if ( (5 > 3) AND (5 < 10) ) { ...
```

Estas condiciones son FALSE:

```
if ( (5 != 5) AND (5 > 3) ) { ...
if ( (5 != 5) OR (5 < 3) ) { ...
```

Script 6.4. En este script el operador lógico AND establece una condición específica bajo la que se imprimirá el mensaje. El AND necesitará que ambas subcondiciones sean TRUE para que la condición completa sea TRUE.

script

```
1  <HTML>
2  <HEAD>
3  <TITLE>Conditionals</TITLE>
4  </HEAD>
5  <BODY>
6  <?php
7  /* $Quantity se debe pasar a esta página
   desde un formulario o mediante el URL.
   $Discount es opcional. */
8  $Cost = 20.00;
9  $Tax = 0.06;
10 if ($Quantity) {
11     $Quantity = abs($Quantity);
12     $Discount = abs($Discount);
13     $Tax++; // $Tax is now worth 1.06.
14     $TotalCost = ($Cost * $Quantity);
15     if ( ($TotalCost < 50) AND
       ($Discount) ) {
16         print ("Your \$$Discount will not
           apply because the total value of the
           sale is under $50!\n<P>");
17     }
18     if ($TotalCost >= 50) {
19         $TotalCost = $TotalCost - $Discount;
20     }
21     $TotalCost = $TotalCost * $Tax;
22     $Payments = round ($TotalCost, 2) / 12;
23     // Imprimir los resultados.
24     print ("You requested to purchase
       $Quantity widget(s) at \$$Cost
       each.\n<P>");
25     print ("The total with tax, minus your
       \$$Discount, comes to $");
26     printf ("%01.2f", $TotalCost);
27     print (".\n<P>You may purchase the
       widget(s)in 12 monthly installments
       of $");
28     printf ("%01.2f", $Payments);
29     print (" each.\n<P>");
30 }
31 ?>
32 </BODY>
33 </HTML>
```

Cuando construya estructuras condicionales, recuerde dos cosas importantes: primera, para que las instrucciones resultantes de una condicional se ejecuten, la condicional debe tener un valor TRUE; segunda, utilizando paréntesis puede ignorar las reglas de precedencia y asegurar que los operadores están tratados en el orden de elección.

Para mostrar los operadores lógicos, debería añadir otra condicional a la página numbers.php permitiendo al usuario saber si el descuento no se aplica.

Cómo utilizar operadores lógicos

1. Abra numbers.php (véase el *Script* 6.3) en el editor de textos.
2. Después de que se calcule primero $TotalCost (véase el *Script* 6.3, línea 14) pero antes de la condición if ($TotalCost >= 50), añada lo siguiente (véase el *Script* 6.4):

```
if ( ($TotalCost < 50) AND
→ ($Discount) ) {
    print ("Your \$$Discount
    → discount will not apply
    → because the total value of
    → the sale is under
    → $50!\n<P>");
}
```

Esta estructura condicional preguntará dos cosas. Primero verá si $TotalCost es menor que 50 dólares. Segundo, determinará si existe un valor $Discount distinto de cero. Si ambos son TRUE, se imprimirá el mensaje.

Si al menos una de estas condiciones es FALSE, la condición completa también será FALSE (porque la condición principal se rige por el operador lógico AND) y, por tanto, no se imprimirá el mensaje.

3. Como éstos son los únicos cambios del *script*, ahora puede guardarlo, cargarlo en el servidor y probarlo en el navegador web (véanse las Figuras 6.5 y 6.6).

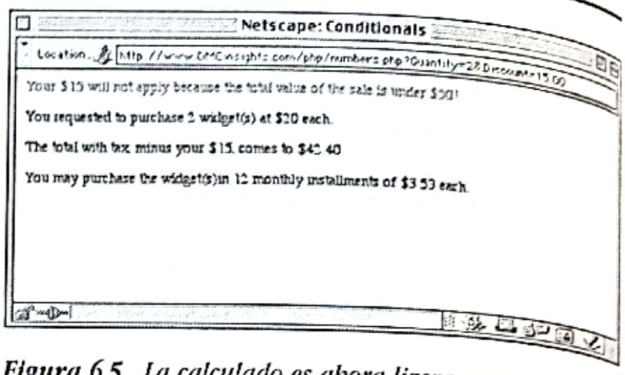


Figura 6.5. La calculado es ahora ligeramente más profesional al permitir al usuario saber que su descuento no se aplicará a la venta.

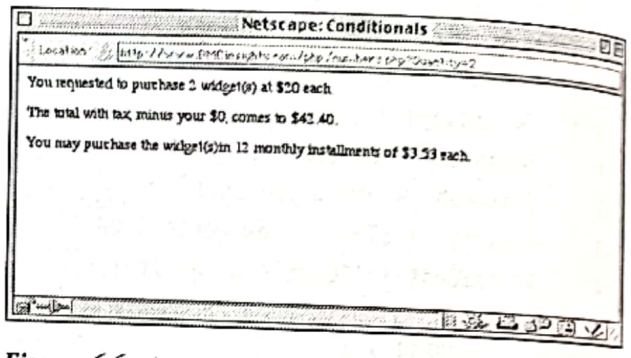


Figura 6.6. Aunque el valor de la venta total sea inferior a 50 dólares, no se imprime ningún mensaje al usuario porque $Discount no tiene un valor.

CONSEJOS

- Es una convención común en programación, que hemos mantenido en este libro, escribir los términos TRUE y FALSE en mayúsculas.
- En estructuras condicionales largas y complicadas es muy fácil olvidar abrir o cerrar paréntesis, lo que provocará mensajes de error o resultados inesperados. Encuentre algún sistema (como espaciar las estructuras condicionales que ya empleamos) para dar claridad al código.

Script 6.5. Normalmente verá que tiene sentido escribir estructuras condicionales if-else en lugar de sólo condicionales if estándar (como ha hecho aquí), ya que probablemente quiera realizar alguna acción si una condición que estaba esperando no se cumple.

script

```
1  <HTML>
2  <HEAD>
3  <TITLE> Conditionals </TITLE>
4  </HEAD>
5  <BODY>
6  <?php
7  /* $Quantity se debe pasar a esta página
   desde un formulario o mediante un URL.
   $Discount es opcional. */
8  $Cost = 20.00;
9  $Tax = 0.06;
10 if ($Quantity) {
11     $Quantity = abs($Quantity);
12     $Discount = abs($Discount);
13     $Tax++; // $Tax is now worth 1.06.
14     $TotalCost = ($Cost * $Quantity);
15     if ( ($TotalCost < 50) AND
       ($Discount) ) {
16         print ("Your \$$Discount will not
           apply because the total value of the
           sale is under $50!\n<P>");
17     }
18     if ($TotalCost >= 50) {
19         $TotalCost = $TotalCost - $Discount;
20     }
21     $TotalCost = $TotalCost * $Tax;
22     $Payments = round ($TotalCost, 2) / 12;
23     // Imprimir los resultados.
24     print ("You requested to purchase
       $Quantity widget(s) at \$$Cost each.
       \n<P>");
25     print ("The total with tax, minus your
       \$$Discount, comes to $");
26     printf ("%01.2f", $TotalCost);
27     print (".\n<P>You may purchase the wid-
       get(s)in 12 monthly installments of $");
28     printf ("%01.2f", $Payments);
29     print (" each.\n<P>");
30     } else {
31     print ("Please make sure that you
       have entered both a quantity and an
       applicable discount and then
       resubmit.\n"); }
32 ?>
33 </BODY>
34 </HTML>
```

Utilización de else

La formación lógica siguiente después de una condición if es la estructura condicional if-else (a veces llamado if-then-else). Éste le permitirá establecer una condición por la que se debe ejecutar una instrucción y después otra instrucción que se ejecutará si no se cumple la condición.

```
if (condición) {
    instrucciones;
} else {
    instrucciones2;
}
```

Es importante recordar al utilizar esta construcción que, a menos que la condición se cumpla explícitamente, la instrucción else se ejecutará. En otras palabras, las instrucciones después de else constituyen la acción por defecto, mientras que las instrucciones después de la condición if son la excepción de la regla. Ahora puede volver a escribir la página numbers.php incorporando else en la instrucción if.

Cómo utilizar else

1. Abra numbers.php en el editor de textos (véase el *Script* 6.4).
2. Inmediatamente después de cerrar el corchete de la condición if (línea 30), escriba (véase el *Script* 6.5):

```
else { print ("Please make sure that
→ you have entered both a quantity
→ and an applicable discount and
→ then resubmit.\n"); }
```

En lugar de mostrar una pantalla en blanco si la página no recibe una cantidad, será mejor imprimir un mensaje de error apropiado.

3. Guarde el *script*, cárguelo en el servidor y pruébelo en el navegador web (véase la Figura 6.7).

CONSEJO

- También puede añadir, en este momento, una condición if-else a numbers.php de manera que se imprima un mensaje si no se aplica el descuento. O, si lo desea, puede utilizar una condición if-else para imprimir "widget" si sólo se encarga un dispositivo y el plural "widgets" sin son más (en oposición a "widget(s)" como hemos estado escribiendo).

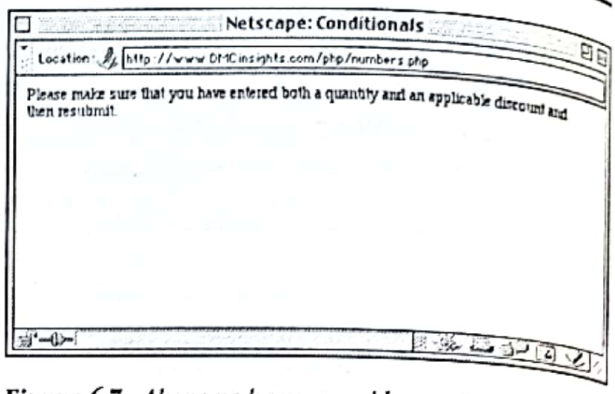


Figura 6.7. *Ahora ya ha conseguido un sitio web más profesional que reacciona de acuerdo con lo que el usuario o el sistema hacen: en este caso, se omite la cantidad. Los condicionales le ayudan a prepararse ante los supuestos que pueden o no ser realidad.*

Script 6.6. Ha anidado un if-elseif-else dentro de otra estructura condicional if-else, lo que es perfectamente aceptable mientras mantenga la sintaxis (continuar la sangría de las líneas siguiente ayudará).

```
script
1  <HTML>
2  <HEAD>
3  <TITLE> If-elseif Conditionals </TITLE>
4  <BODY>
5  <?php
6  if ($Username) {
7      print ("Good ");
8      if (date("A") == "AM") {
9          print ("morning, ");
10     } elseif ( ( date("H") >= 12 ) and
       ( date("H") < 18 ) ) {
11         print ("afternoon, ");
12     } else {
13         print ("evening, ");
14     } // Cerrar el if date.
15     print ("$Username");
16     print ("!\n");
17 } else {
18     print ("Please log in.\n");
19 } // Cerrar el if username.
20 ?>
21 </BODY>
22 </HTML>
```

Utilización de elseif

Similar a la estructura condicional if-else es if-elseif (o if-elseif-else). Actúa como una instrucción if de ejecución y puede ampliarse todo lo que requiera.

```
if (condición) {
    instrucciones;
} elseif (condición2) {
    instrucciones2;
}
```

Aquí hay otro ejemplo:

```
if (condición) {
    instrucciones;
} elseif (condición2) {
    instrucciones2;
} else {
    instrucciones3;
}
```

Creará una nueva página hello.php similar a la del Capítulo 3, que utiliza una estructura condicional if-elseif y la función date() para escribir un saludo personalizado para el usuario.

Cómo utilizar elseif

1. Cree un nuevo documento PHP en el editor de textos.
2. Escriba la cabecera HTML estándar y abra la sección PHP de la página (véase el *Script* 6.6).

   ```
   <HTML><HEAD><TITLE>If-elseif
   → Conditionals</TITLE><BODY>
   <?php
   ```

3. Cree la condición if principal.

   ```
   If ($Username) {
   ```

   Como en todo buen formulario, sólo imprimirá este saludo si tiene el nombre de usuario.

4. `print ("Good ");`

   Configurará el código para escribir la primera parte del saludo de forma separada, mejor que junto a cada uno de los tres saludos siguientes específicos. De esta manera, si después quiere cambiar el texto específico, sólo tendrá que hacer alteraciones en un sitio.

5. `if (date("A") == "AM") {`

   La función date() se utiliza para determinar cualquier información específica de fecha, día de la semana, mes, etc., basada en el parámetro del que depende. Aquí, date("A") devolverá "AM" o "PM". La función date() se tratará con más detalle en el Capítulo 13.

6. ```
   print ("morning, ");
   } elseif ( ( date("H") >= 12 ) and
   → ( date("H") < 18 ) ) {
   ```

   Date("H") devuelve la hora del día en formato militar. De modo que si es entre el mediodía y las 6 p.m., dirá "Good afternoon."

7. `print ("afternoon, ");`

   ```
   } else {
   print ("evening, ");
   ```

   Si no es la mañana ni la tarde, debe ser la noche, de manera que utilizará else como algo que sirva para todo.

8. `} // cerrar el if date.`

   ```
   print ("$Username");
   print ("!\n");
   } else {
   print ("Please log in.\n");
   ```

   Si no tiene un nombre de usuario, pedirá al usuario que se conecte antes de proceder.

9. `} // Cerrar el if username.`

   Los comentarios le ayudan a seguir las condiciones if complicadas y anidadas, ayudándole a asegurar que cierra los paréntesis correctamente.
   ```

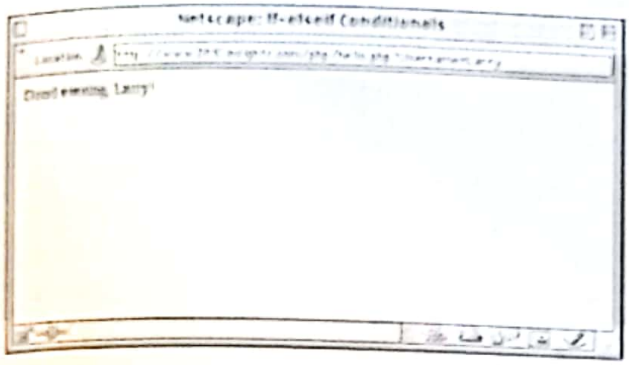


Figura 6.8. La página le saludará de esta forma si se le pasa un valor para $Username y la hora del servidor está comprendida entre las 6 PM y la medianoche.

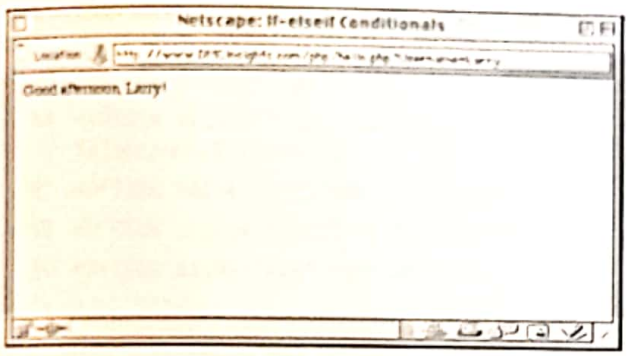


Figura 6.9. Ésta es la bienvenida que la página le dará asumiendo que recibe $Username y la hora está comprendida entre el mediodía y las 6 P.M.

10. Guarde el *script* como hello.php, cárguelo en el servidor y pruébelo en el navegador web (véanse las Figuras 6.8 y 6.9).

CONSEJOS

- Siempre debería poner else como la última parte de una condición, ya que se ejecutará a menos que una de las condiciones a las que apunta se haya cumplido.
- Puede utilizar elseif tantas veces como desee como parte de una condición.
- PHP le permite escribir elseif como dos palabras si prefiere hacerlo así:

```
if (condición) {
    instrucciones;
} else if (condición2) {
    instrucciones2;
}
```

- La función date() es muy útil, pero sólo refleja la hora en el servidor, no la hora donde está el usuario. (De hecho, si la hora en el servidor es incorrecta, la función date() reflejará la hora y fecha incorrectas.)

La condicional switch

Una vez llegue a un punto donde tenga muchas condicionales if-elseif-else, observará que ahorrará tiempo y la programación resultará más clara si utiliza en su lugar una condicional switch. La condicional switch sólo toma una condición posible: el valor de una variable.

```
switch ($Variable) {
    case "valor1":
        instrucciones1;
        break;
    case "valor2":
        instrucciones2;
        break;
    default:
        instrucciones3;
        break;
}
```

Es importante que comprenda cómo funciona una condicional switch. Comenzando desde el principio, una vez que PHP encuentra el argumento correspondiente al valor de la variable configurada, continuará ejecutando instrucciones hasta llegar al final de la condicional switch (el corchete de cierre) o hasta encontrar una instrucción break; en este punto saldrá de la construcción switch. Por tanto, es imperativo que cierre cada argumento (e incluso el argumento predeterminado, por coherencia) con un break.

La condicional switch es algo parecido a escribir esto:

```
if ($Variable == "valor1") {
    instrucciones1;
} elseif ($Variable=="valor2") {
    instrucciones2;
} else {
    instrucciones3;
}
```

Script 6.7. Este formulario HTML utiliza un menú desplegable para dar al usuario una lista de opciones (creado por el tipo de entrada SELECT).

```
script
1  <HTML>
2  <HEAD>
3  <TITLE>HTML Contact Form</TITLE>
4  </HEAD>
5  <BODY>
6  <FORM ACTION="HandleContact.php"
   METHOD=POST>
7  First Name <INPUT TYPE=TEXT
   NAME="FirstName" SIZE=20><BR>
8  Last Name <INPUT TYPE=TEXT NAME="LastName"
   SIZE=20><BR>
9  How would you prefer to be contacted:
   <SELECT NAME="ContactHow">
10 <OPTION VALUE="">Select One:</OPTION>
11 <OPTION VALUE="Telephone">
   Telephone</OPTION>
12 <OPTION VALUE="Mail">Mail</OPTION>
13 <OPTION VALUE="E-Mail">E-Mail</OPTION>
14 <OPTION VALUE="Fax">Fax</OPTION>
15 </SELECT><BR>
16 Comments <TEXTAREA NAME="Comments" ROWS=5
   COLS=40></TEXTAREA><BR>
17 <INPUT TYPE=SUBMIT NAME="SUBMIT"
   VALUE="Submit!">
18 </FORM>
19 </BODY>
20 </HTML>
```

Explicaremos por qué la condicional switch utiliza el valor de $Variable como condición. Primero comprobará si $Variable es igual a valor1 y, si lo es, ejecutará la instrucciones1. Si no, comprobará si $Variable es igual a valor2 y, si lo es, ejecutará instrucciones2. Si no se cumple ninguna condición, la acción predeterminada de la condición switch es ejecutar instrucciones3.

En la siguiente sección, "El bucle while", utilizará switch en combinación con un bucle para crear un formulario HTML que le diga cuántos días hay en un mes. Pero, de momento, para mostrar las capacidades de switch, escribirá un *script* sencillo que imprima un mensaje basado en lo que el usuario seleccione en un formulario HTML.

Cómo utilizar una condicional switch

1. Cree un nuevo documento HTML en el editor de textos.
2. Comience con la cabecera HTML estándar (véase el *Script* 6.7):

```
<HTML><HEAD><TITLE>HTML
→ Contact Form</TITLE></HEAD><BODY>
```

3. Haga un formulario que tome información de los usuarios y les dé la opción de cómo quieren conectarse.

```
<FORM ACTION="HandleContact.php"
→ METHOD=POST>
Nombre <INPUT TYPE=TEXT NAME=
→ "FirstName" SIZE=20><BR>
Apellido <INPUT TYPE=TEXT NAME=
→ "LastName" SIZE=20><BR>
How would you prefer to be contacted:
→ <SELECT NAME="ContactHow">
→ <OPTION VALUE="">Select One:
→ </OPTION>
<OPTION VALUE="Telephone">
→ Telephone</OPTION>
<OPTION VALUE="Mail">Mail</OPTION>
<OPTION VALUE="E-Mail">E-Mail
→ </OPTION>
<OPTION VALUE="Fax">Fax</OPTION>
</SELECT><BR>
```

Imaginamos esto como parte de un sistema de retroalimentación para una aplicación web mayor. Este documento particular y la página de tratamiento constituirán dos pasos en el proceso de retroalimentación. Aquí el usuario introducirá su nombre y después elegirá cómo desea que contacten con él.

4. Complete el formulario con un cuadro de comentarios, cierre el formulario y el documento HTML.

```
Comments <TEXTAREA NAME="Comments"
→ ROWS=5 COLS=40></TEXTAREA><BR>
<INPUT TYPE=SUBMIT NAME="SUBMIT"
→ VALUE="Submit!">
</FORM></BODY></HTML>
```

5. Guarde el formulario como contact.html y cárguelo en el servidor.

6. Ahora tendrá que crear la página que procesará una parte de la página contact.html.

7. Cree un nuevo documento PHP en el editor de textos.

8. Codifique la cabecera HTML estándar (véase el *Script* 6.8).

```
<HTML><HEAD><TITLE>Contact
→ Information Request</TITLE><BODY>
```

9. Cree un formulario HTML y abra la sección PHP.

```
<FORM ACTION="HandleContact2.php"
→ METHOD=POST><?php
```

Para recuperar y reunir más información del usuario, utilizará otro formulario.

10. Almacene los valores recuperados desde contact.html en elementos HIDDEN de formulario.

```
print ("<INPUT TYPE=HIDDEN
→ NAME=\"FirstName\"
→ VALUE=\"$FirstName\">\n");
print ("<INPUT TYPE=HIDDEN
→ NAME=\"LastName\"
→ VALUE=\"$LastName\">\n");
print ("<INPUT TYPE=HIDDEN
→ NAME=\"Comments\"
→ VALUE=\"$Comments\">\n");
```

Script 6.8. La condicional switch en este script utiliza el valor de $ContactHow para determinar qué solicitar del usuario: número de teléfono, número de fax, dirección electrónica o dirección postal. HIDDEN introduce tipos que también se utilizan para pasar junto con otros valores existentes.

script

```
1  <HTML>
2  <HEAD>
3  <TITLE>Contact Information Request</TITLE>
4  <BODY>
5  <FORM ACTION="HandleContact2.php"
   METHOD=POST>
6  <?php
7  // Transmite los valores recibidos
   utilizando tipos HIDDEN INPUT.
8  print ("<INPUT TYPE=HIDDEN
   NAME=\"FirstName\"
   VALUE=\"$FirstName\">\n");
9  print ("<INPUT TYPE=HIDDEN
   NAME=\"LastName\"
   VALUE=\"$LastName\">\n");
10 print ("<INPUT TYPE=HIDDEN
   NAME=\"Comments\"
   VALUE=\"$Comments\">\n");
11 print ("<INPUT TYPE=HIDDEN
   NAME=\"ContactHow\"
   VALUE=\"$ContactHow\">\n");
12
13 switch ($ContactHow) {
14     case "Telephone":
15         print("<B>Please enter a daytime
           phone number where you can be
           reached:</B><BR>\n");
16         print ("<INPUT TYPE=TEXT NAME=
           \"Telephone\" SIZE=10><BR>v");
17         print ("<INPUT TYPE=SUBMIT NAME=
           SUBMIT VALUE=\"Continue\">\n");
18         break;
19     case "Mail":
20         print("<B>Please enter your
           complete mailing address:
           </B><BR>\n");
21         print ("<TEXTAREA
           NAME=\"MailAddress\" ROWS=5
           COLS=40></TEXTAREA><BR>\n");
22         print ("<INPUT TYPE=SUBMIT NAME=
           SUBMIT VALUE=\"Continue\">\n");
23         break;
24     case "E-Mail":
25         print("<B>Please enter your
           E-Mail address:</B><BR>\n");
```

Script 6.8. Continuación

```
script
26          print ("<INPUT TYPE=TEXT NAME=
            "E-Mail\" SIZE=40><BR>\n");
27          print ("<INPUT TYPE=SUBMIT NAME=
            SUBMIT VALUE=\"Continue\">\n");
28          break;
29      case "Fax":
30          print("<B>Please enter your Fax
            number:</B><BR>\n");
31          print ("<INPUT TYPE=TEXT NAME=
            \"Fax\" SIZE=10><BR>\n");
32          print ("<INPUT TYPE=SUBMIT NAME=
            SUBMIT VALUE=\"Continue\">\n");
33          break;
34      default:
35          print("<B>Please go back and select
            how you would prefer to
            be contacted!</B><BR>\n");
36          break;
37  }
38  ?>
39  </FORM>
40  </BODY>
41  </HTML>
```

```
print ("<INPUT TYPE=HIDDEN
→ NAME=\"ContactHow\"
→ VALUE=\"$ContactHow\">\n");
```

Ya mencionamos en el Capítulo 3 que podría pasar información de una variable utilizando el tipo HIDDEN INPUT. Aquí el PHP situará todos los datos recogidos desde contact.html en elementos ocultos, de modo que los datos seguirán pasando a la siguiente página, HandleContact2.php.

11. Cree una condicional switch que reaccione de manera diferente basándose en la opción de contacto que seleccione el usuario en contact.html.

```
switch ($ContactHow) {
case "Telephone":
print("<B>Please enter a daytime
→ phone number where you can be
→ reached:</B><BR>\n");
print ("<INPUT TYPE=TEXT NAME=
→ \"Telephone\" SIZE=10><BR>v");
print ("<INPUT TYPE=SUBMIT NAME=
→ SUBMIT VALUE=\"Continue\">\n");
break;
case "Mail":
print("<B>Please enter your complete
→ mailing address:</B><BR>\n");
print ("<TEXTAREA NAME=
→ \"MailAddress\" ROWS=5 COLS=40>
→ </TEXTAREA><BR>\n");
print ("<INPUT TYPE=SUBMIT NAME=
→ SUBMIT VALUE=\"Continue\">\n");
break;
case "E-Mail":
print("<B>Please enter your E-Mail
→ address:</B><BR>\n");
print ("<INPUT TYPE=TEXT NAME=
→ \"E-Mail\" SIZE=40><BR>\n");
print ("<INPUT TYPE=SUBMIT NAME=
→ SUBMIT VALUE=\"Continue\">\n");
break;
case "Fax":
print("<B>Please enter your Fax
→ number:</B><BR>\n");
print ("<INPUT TYPE=TEXT NAME=
→ \"Fax\" SIZE=10><BR>\n");
```

```
print ("<INPUT TYPE=SUBMIT NAME=
 ‣ SUBMIT VALUE=\"Continue\">\n");
break;
default:
print("<B>Please go back and select
 ‣ how you would prefer to be
 ‣ contacted!</B><BR>\n");
break;
}
```

Esta condicional switch se configura para imprimir cosas distintas dependiendo del valor de $ContactHow. Si quieren ser telefoneados, la página pedirá un número de teléfono, si prefieren el correo electrónico, se les solicitará una dirección de correo electrónico, etc. Si $ContactHow no tiene valor, se ejecutará el argumento predeterminado y aparecerá un mensaje pidiéndoles que vuelvan a conectar a la página contact.html para seleccionar un método de contacto.

12. Cierre la sección PHP, después el formulario y la página HTML.

```
?></FORM></BODY></HTML>
```

13. Guarde el *script* como HandleContact.php, cárguelo en el servidor (en el mismo directorio que contact.html) y pruebe ambas páginas en el navegador web (véanse las Figuras 6.10, 6.11, 6.12, 6.13 y 6.14).

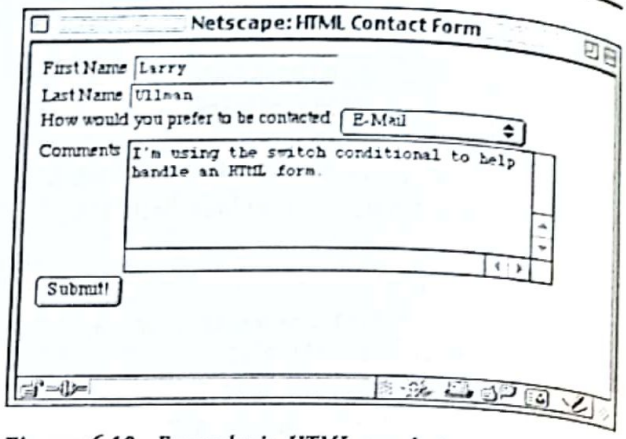


Figura 6.10. Formulario HTML creado por contact.html. Constituye el primer paso de un sistema de retroalimentación.

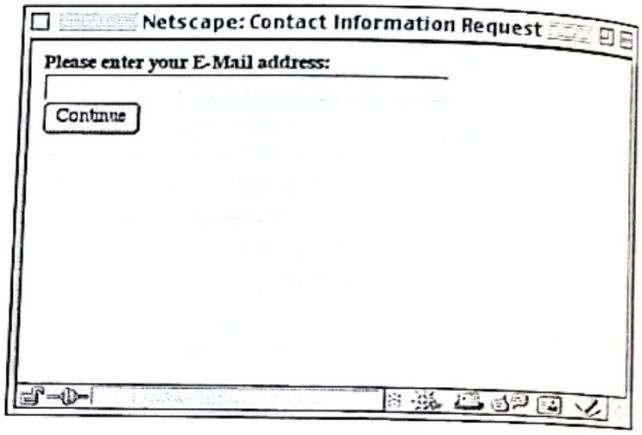


Figura 6.11. Puesto que solicitamos que contactaran con nosotros a través de correo electrónico (véase la Figura 6.10), HandleContact.php solicitará una dirección de correo electrónico y mostrará un tipo de entrada TEXT donde poder introducir esta información.

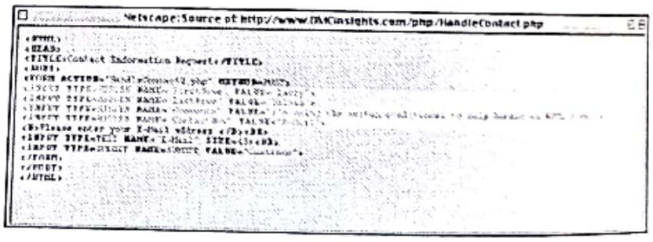


Figura 6.12. Código fuente de lo que se ve en la Figura 6.11. Observe que los campos HIDDEN ya están almacenando la información, de modo que será pasada en HandleContact2.php.

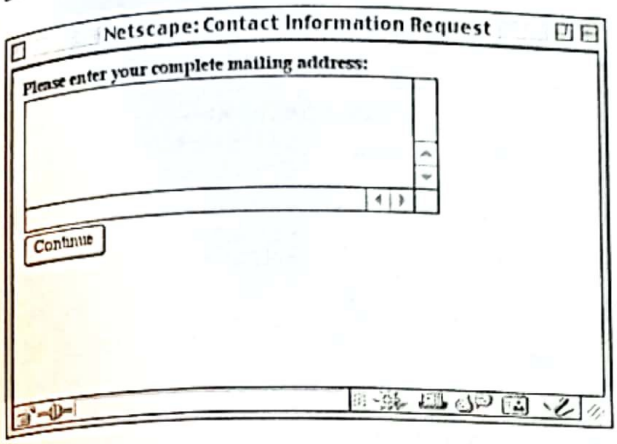


Figura 6.13. Si decidimos que queremos ser avisados mediante correo electrónico, HandleContact.php puede solicitar una dirección de correo electrónico y, en esta ocasión, proporcionar un campo TEXTAREA donde poder introducir esta información.

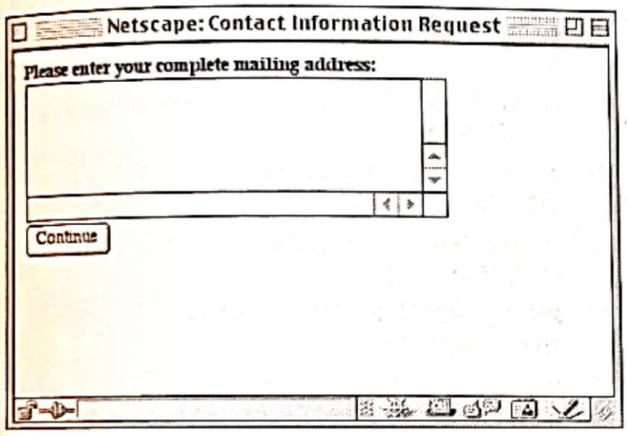


Figura 6.14. Si el usuario no selecciona una preferencia sobre cómo quiere ser contactado, verá este mensaje y no podrá seguir adelante.

CONSEJOS

- No hemos creado una página HandleContact2.php que trabaje con los resultados de esta página HandleContact.php (como dijimos, sólo íbamos a desarrollar dos pasos en un proceso más largo). Si quiere asegurarse de que HandleContact.php funciona de forma adecuada, escriba un *script* PHP sencillo que imprima los valores de las tablas.
- No necesitará un caso predeterminado en la condicional switch (puede configurarlo de modo que si el valor no se cumple explícitamente en uno de los casos, no ocurra nada), pero si se utiliza, debe nombrarse como el último caso.
- Si va a usar una cadena en la condicional switch, recuerde que distingue entre mayúsculas y minúsculas, lo que significa que "Value" no se corresponderá con una cadena "value".

El bucle while

Como sugerimos anteriormente en este capítulo, los bucles se utilizan para ejecutar una parte de código repetidamente. Puede desear crear un menú desplegable que consista en los días del mes (imprima los números del 1 al 31) e imprimir cada valor en una tabla. Para estos casos, y muchos más, utilizará un bucle (lo mostraremos en un ejemplo del siguiente capítulo).

El primero de los dos tipos de bucles que existen en PHP, el bucle while, está diseñado para seguir trabajando mientras la condición establecida sea TRUE. Este bucle comprobará el valor de la condición antes de cada repetición. Una vez que la condición sea FALSE, el bucle while finalizará.

```
while (condición) {
    instrucciones;
}
```

Para mostrar el bucle while, creará un *script* que generará dinámicamente un menú desplegable de fechas (mes, día, año) para un formulario HTML. Mientras que el formulario por sí mismo no hará nada así, podrá ver cómo puede utilizar PHP para mejorar y facilitar la creación de un elemento de formulario HTML estándar.

Cómo utilizar while

1. Cree un nuevo documento en el editor de textos. Escriba (véase el *Script* 6.9):

```
<HTML><HEAD><TITLE>Select Menu
</TITLE><BODY><?php
```

2. Configure el año actual utilizando la función date().

```
$Year = date ("Y");
```

Argumentando la función date() con el valor "Y", devolverá el año actual. Utilizará el valor para imprimir este año, además de los diez siguientes, en el menú seleccionado. Así, no necesitará cambiar el formulario año tras año.

Script 6.9. Los dos bucles while imprimirán rápidamente los requisitos HTML para generar dos menús desplegables (véase la Figura 6.16). Utilizando la función date() para basar el bucle del año en el año en curso, este script nunca estará anticuado.

script

```
1  <HTML>
2  <HEAD>
3  <TITLE>Select Menu</TITLE>
4  <BODY>
5  <?php
6  $Year = date ("Y");
7  // Crea el formulario.
8  print ("<FORM ACTION=\"$PHP_SELF\"
   METHOD=POST>\n");
9  // Crea el menú desplegable de los meses.
10 print ("Select a month:<BR>\n");
11 print ("<SELECT NAME=Month><OPTION>
   Choose One</OPTION>\n");
12 print ("<OPTION VALUE=January>January
   </OPTION>\n");
13 print ("<OPTION VALUE=February>February
   </OPTION>\n");
14 print ("<OPTION VALUE=March>March
   </OPTION>\n");
15 print ("<OPTION VALUE=April>April
   </OPTION>\n");
16 print ("<OPTION VALUE=May>May
   </OPTION>\n");
17 print ("<OPTION VALUE=June>June
   </OPTION>\n");
18 print ("<OPTION VALUE=July>July
   </OPTION>\n");
19 print ("<OPTION VALUE=August>August
   </OPTION>\n");
20 print ("<OPTION VALUE=September>September
   </OPTION>\n");
21 print ("<OPTION VALUE=October>October
   </OPTION>\n");
22 print ("<OPTION VALUE=November>November
   </OPTION>\n");
23 print ("<OPTION VALUE=December>December
   </OPTION>\n");
24 print ("</SELECT>\n");
25 // Crea el menú desplegable de los días.
26 print ("<P>Select a day:<BR>\n");
27 print ("<SELECT NAME=Day><OPTION>Choose
   One</OPTION>\n");
28 $Day = 1;
29 while ($Day <= 31) {
```

Script 6.9. Continuación

```
script
30      print ("<OPTION VALUE=$Day>$Day
        </OPTION>\n");
31      $Day++;
32  }
33  print ("</SELECT>\n");
34  // Crea el menú desplegable de los años.
35  print ("<P>Select a year:<BR>\n");
36  print ("<SELECT NAME=Year><OPTION>
    Choose One</OPTION>\n");
37  $EndYear = $Year + 10;
38  while ($Year <= $EndYear ) {
39      print ("<OPTION VALUE=$Year>$Year
        </OPTION>\n");
40      $Year++;
41  }
42  print ("</SELECT>\n");
43  print ("<P><INPUT TYPE=SUBMIT NAME=
    SUBMIT VALUE=\"Go!\"></FORM>\n");
44  ?>
45  </BODY>
46  </HTML>
```

3. Cree un formulario HTML que incluya un menú desplegable para cada mes.

```
print ("<FORM ACTION=\"$PHP_SELF\"
→ METHOD=POST>\n");
print ("Select a month:<BR>\n");
print ("<SELECT NAME=Month>
→ <OPTION>Choose One</OPTION>\n");
print ("<OPTION VALUE=January>
→ January</OPTION>\n");
print ("<OPTION VALUE=February>
→ February</OPTION>\n");
print ("<OPTION VALUE=March>March
→ </OPTION>\n");
print ("<OPTION VALUE=April>April
→ </OPTION>\n");
print ("<OPTION VALUE=May>May
→ </OPTION>\n");
print ("<OPTION VALUE=June>June
→ </OPTION>\n");
print ("<OPTION VALUE=July>July
→ </OPTION>\n");
print ("<OPTION VALUE=August>August
→ </OPTION>\n");
print ("<OPTION VALUE=September>
→ September</OPTION>\n");
print ("<OPTION VALUE=October>
→ October</OPTION>\n");
print ("<OPTION VALUE=November>
→ November</OPTION>\n");
print ("<OPTION VALUE=December>
→ December</OPTION>\n");
print ("</SELECT>\n");
```

4. Utilice un bucle while para crear el menú desplegable para los días.

```
print ("<P>Select a day:<BR>\n");
print ("<SELECT NAME=Day><OPTION>
→ Choose One</OPTION>\n");
$Day = 1;
while ($Day <= 31) {
    print ("<OPTION VALUE=$Day>$Day
    → </OPTION>\n");
    $Day++;
}
print ("</SELECT>\n");
```

El primer paso será configurar el valor de $Day a 1. Se debe hacer antes de llamar al bucle. El bucle entonces verá si $Day es menor o igual a 31. Si lo es, imprimirá el valor de $Day como una opción en el menú seleccionado y después incrementará el valor de $Day en 1. Este proceso continuará hasta que $Day sea igual a 32; en este punto, PHP detendrá el bucle y continuará el *script*. Utilizar un bucle que genere el requisito HTML es mucho más fácil que imprimir treinta y una líneas de código a la vez.

5. Utilice otro bucle while para crear el menú desplegable de los años.

```
print ("<P>Select a year:<BR>\n");
print ("<SELECT NAME=Year><OPTION>
→ Choose One</OPTION>\n");
$EndYear = $Year + 10;
while ($Year <= $EndYear ) {
    print    ("<OPTION    VALUE=$Year>
    → $Year</OPTION>\n");
    $Year++;
}
print ("</SELECT>\n");
```

Los años enumerados en el menú desplegable se generarán basándose en el año actual, más los diez siguientes. El año actual se asignó anteriormente a la variable $Year. El último año entonces, $EndYear, es igual a $Year más 10 (este valor se podría cambiar fácilmente para imprimir 5 o 15 años). El bucle establece que mientras el valor de $Year sea menor o igual al valor de $EndYear, se debería imprimir el valor de $Year y después incrementar el valor en uno.

6. Cree un botón de envío, cierre el formulario, PHP y HTML.

```
print ("<P><INPUT TYPE=SUBMIT NAME=
→ SUBMIT VALUE=\"Go!\"></FORM>\n");
?></BODY></HTML>
```

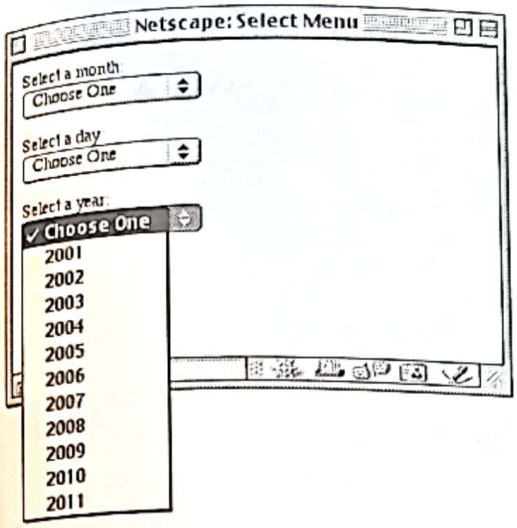


Figura 6.15. La página select.php genera tres menús desplegables que permiten al usuario seleccionar el día, mes y año. Puede pegar fácilmente este código en un formulario HTML más grande donde necesite estos menús.

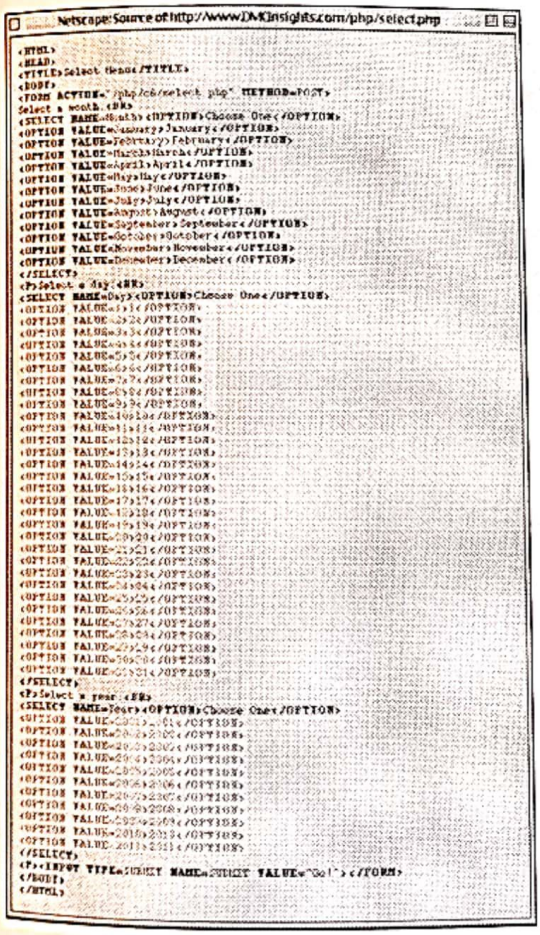


Figura 6.16. Código fuente del formulario mostrado en la Figura 6.15. Observe cómo una docena de líneas del Script 6.9 (que contienen dos bucles while) generan de 40 a 50 líneas de código HTML (los menús de selección del día y el año).

Aunque esta página está diseñada para ser parte de un formulario HTML mayor y, por tanto, no hace todo por sí misma, para mantener la coherencia hemos añadido el botón de envío.

7. Guarde la página como select.php, cárguela en el servidor y pruébela en el navegador web (véanse las Figuras 6.15 y 6.16).

CONSEJOS

- También tiene la opción de utilizar el bucle do...while que garantizará que las instrucciones se ejecuten al menos una vez, lo que no es necesariamente cierto en el bucle while.

```
do {
   instrucciones;
} while (condición);
```

- Asegúrese de los elementos que necesita incluir en un bucle y de los que se pueden excluir. Un fallo al incrementar el valor de $Day o $Year dentro de un bucle crearía un bucle sin fin (como, por ejemplo, $Day sería siempre menor o igual a 31). De forma similar, la inclusión de etiquetas <SELECT> dentro del bucle crearía múltiples menús desplegables.

- Así como la estructura condicional if se puede escribir en una línea si sólo tiene una instrucción, lo mismo se puede hacer con el bucle while, aunque, de nuevo, no se lo recomendamos.

El bucle for

El bucle for está diseñado para realizar las instrucciones específicas para un determinado número de repeticiones (a diferencia de while, que se ejecuta hasta que la condición es FALSE). Normalmente puede utilizar una variable ficticia en el bucle con este fin. La sintaxis para el bucle for es más complicada que la del bucle while y aunque los usos de estos bucles pueden coincidir fácilmente, encontrará uno más adecuado que el otro para algunas tareas.

```
for (expresión inicial; condición;
→ expresión de cierre) {
    instrucciones;
}
```

La expresión inicial se ejecutará una vez, la primera vez que se llame al bucle. Entonces la condición se usa para determinar si se ejecutan o no las instrucciones. Finalmente, la expresión de cierre se ejecutará después de cada vez que la condición sea TRUE, pero sólo después de que se ejecuten las instrucciones. Por tanto, para imprimir cada valor en una tabla, codificaría:

```
for ($n = 0; $n < count($Array); $n++) {
    print ("$Array[$n]<BR>\n");
}
```

Puede ayudar a la comprensión de la sintaxis del bucle for rescribir el bucle while $Day desde el *Script* 6.9 como un bucle for.

El código original era:

```
$Day = 1;
while ($Day <= 31) {
    print ("<OPTION VALUE=$Day>$Day
    → </OPTION>\n");
    $Day++;
}
```

Script 6.10. Este breve script es un uso sencillo del bucle for, que repite un proceso mil veces.

```
script
1  <HTML>
2  <HEAD>
3  <TITLE>Prime Numbers</TITLE>
4  <BODY>
5  <?php
6  // Cambiar este valor si quiere imprimir
   más números primos.
7  for ($n = 1; $n <= 1000; $n++) {
8      if ( ($n == 1) OR ($n == 2) OR
       ($n == 3) OR ($n == 5)) {
9          print("$n<BR>\n");
10     } elseif (($n % 2 != 0) AND
       ($n % 3 != 0) AND ($n % 5 != 0)) {
11         print("$n<BR>\n");
12     } // Cierra el IF.
13 } // Cierra el FOR.
14 ?>
15 </BODY>
16 </HTML>
```

Primero se configuró el valor de $Day, después el bucle while estableció la condición ($Day <= 31). Finalmente, si era TRUE, se ejecutó la instrucción print() y $Day se incrementó. Para un bucle for, el mismo código sería así:

```
for ($Day = 1; $Day <= 31; $Day++) {
    print ("<OPTION
VALUE=$Day>$Day</OPTION>\n");
}
```

Como ejemplo común de programación, utilizará el bucle for para imprimir todos los números primos entre 1 y 1.000.

Cómo escribir un bucle for

1. Cree un nuevo documento PHP en el editor de textos. Escriba (véase el *Script* 6.10):

   ```
   <HTML><HEAD><TITLE>Prime Numbers
   → </TITLE><BODY><?php
   ```

2. Comience el bucle for.

   ```
   for ($n = 1; $n <= 1000; $n++) {
   ```

   Utilizando la sintaxis del bucle for, $n se configura como una variable ficticia. Se configura inmediatamente a 1; después el bucle comprobará si $n es menor o igual a 1.000. En otras palabras, este bucle se ejecutará 1.000 veces. Finalmente, si la condición del bucle ($n <= 1000) es TRUE, la instrucción(s) se ejecutará, $n se incrementará, y el proceso comenzará de nuevo.

3. Escriba los contenidos (es decir, la sección de instrucción(s) del bucle).

   ```
   if ( ($n == 1) OR ($n == 2) OR
   → ($n == 3) OR ($n == 5)) {
       print("$n<BR>\n");
   } elseif (($n % 2 != 0) AND ($n
   → % 3 != 0) AND ($n % 5 != 0)) {
       print("$n<BR>\n");
   }
   ```

Un número es primo si no es divisible por otro número que no sea él mismo y 1. En otras palabras, si divide el número por otro número que no sea él mismo o 1, siempre obtendrá un resto, entonces el número es primo. Por ejemplo, 4 se puede dividir entre 2, de modo, que no es primo mientras que 7 no se puede dividir entre 2, 3 ó 5, lo que es la base para establecer los números primos. De manera informal, sabe que 1, 2, 3 y 5 son primos, de manera que si $n es igual a cualquiera de ellos, se imprimirá automáticamente. Observe que utilizamos el operador lógico OR porque si una de estas condiciones es TRUE, deseará imprimir $n.

Si $n no es igual a 1, 2, 3 ó 5, deberá comprobar si es primo. Para hacerlo, compruebe si en la división de $n entre 2, 3 ó 5 se obtiene un resto, calculado utilizando el operador de módulo (%). El módulo devuelve el resto de una división y aunque no sea el operador más útil, aquí es inestimable. De modo que 4 módulo 2 es 0 y 7 módulo 5 es 0,4. Si se devuelve un valor distinto de cero al dividir $n entre 2, 3 y 5, el número es primo y se debería imprimir. Observe que usamos el operador lógico AND porque si alguna de estas condiciones es FALSE, el número no es primo.

4. Cierre el bucle for, el PHP y el HTML.

```
}
?>
</BODY>
</HTML>
```

5. Guarde el *script* como primes.php, cárguelo en el servidor y pruébelo en el navegador web (véase la Figura 6.17).

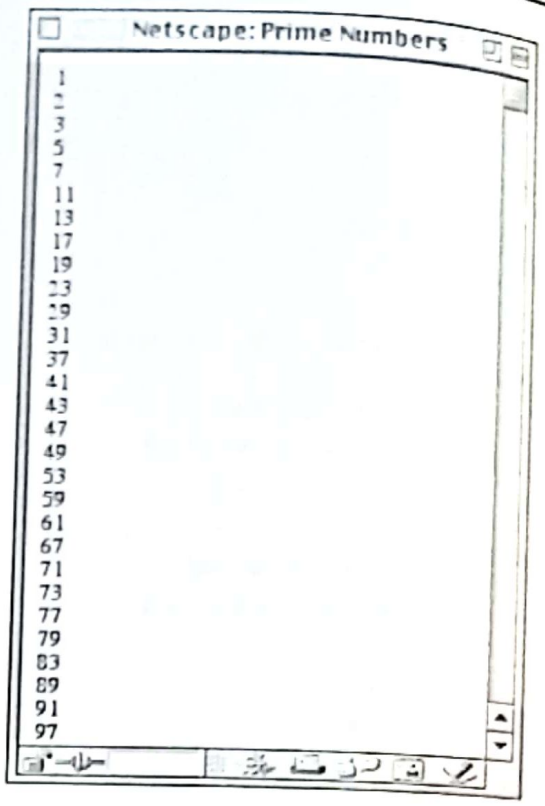


Figura 6.17. *Listado de los números primos hasta el 100, calculados automáticamente y mostrados utilizando un bucle for. Las 16 líneas del Script 6.10 generan aproximadamente 300 líneas en la ventana HTML.*

CONSEJO

- Aunque hay una amplia cantidad de coincidencias respecto en cuanto al uso de las dos construcciones de bucle principales, while y for, al programar descubrirá que a veces uno es más lógico que el otro. El bucle while se utiliza frecuentemente en la recuperación de datos desde una base de datos (consulte el Capítulo 11) y el bucle for es un estándar para trabajar con arrays (como descubrirá en el Capítulo 7).

USO DE LOS ARRAYS 7

El último tipo de variable que trataremos en este libro es el de los arrays. (Los objetos están fuera del enfoque de este libro, pero una vez que se habitúe a usar PHP, estamos seguros de que deseará aprender a utilizarlos.)

Los arrays constituyen una idea complicada pero a la vez muy útil: son una colección de valores ensamblados en una variable anulada. Un array puede consistir en números y/o cadenas (y/u otros arrays), que permiten a dicha variable mantener exponencialmente más información que una cadena simple o un número. Por ejemplo, si quiere crear una lista de comida empleando cadenas, el código podría ser el siguiente:

```
$Item1 = 'manzanas';
$Item2 = 'bananas';
$Item3 = 'naranjas';
```

Necesitrá crear una nueva cadena por cada elemento que añada. Esto es muy engorroso y hace difícil volver a referirse a la lista completa o a cualquier valor específico del código más tarde. Puede simplificar enormemente los problemas colocando la lista en un array (por ejemplo, $Items), que contenga todo lo que necesite incluir en la lista. Al igual que un array, la lista se podrá aumentar, ordenar, buscar, etc. Recordando este contexto, vamos a adentrarnos en la sintaxis de los arrays.

Los otros tipos de variables con los que ha trabajado (números y cadenas), tienen un nombre de variable y un valor correspondiente (por ejemplo, $FirstName podría equivaler a "Larry"). Los arrays también utilizan un nombre, derivado de los mismos convenios (el signo de dólar seguido por una letra o subrayado, o por cualquier combinación de letras, números y subrayados), pero los arrays se diferencian en que pueden contener lo que llamaríamos elementos múltiples. Cada elemento tiene un índice o clave (se pueden usar los dos indistintamente), que se emplea para acceder al valor del elemento. Los términos índice o clave hacen referencia a un número o palabra empleado como punto de referencia de los valores. Un array puede tener un número o una cadena clave, dependiendo de cómo se estructure.

Piense que un array es como una tabla compuesta por dos columnas. La primera columna sería el índice (que actúa como un número de fila o un nombre) y la segunda el valor de dicha fila. Al usar el índice, determina el valor almacenado en un punto concreto de la tabla (véanse las Figuras 7.1 y 7.2).

Generalmente, el formato de un array parece el mismo que el de cualquier otra variable, excepto que incluye una clave entre corchetes ([]) cuando hace referencia a elementos concretos.

Por eso, $Array hace referencia al array como un todo, pero $Array[0] señalará el primer elemento del array. (Los caracteres en una cadena se indexan comenzando desde cero, como se explicó en el Capítulo 5. Los elementos de un array se indexan de la misma manera.)

En este capítulo trataremos los fundamentos del manejo de los arrays. Presentaremos varios conceptos clave y aprenderá a incorporarlos a los *scripts*.

Índice o clave	Valor
1	Cómo empezar con PHP
2	Variables
3	Formularios HTML y PHP
4	Utilización de números
5	Utilización de cadenas
6	Estructuras de control
7	Uso de los arrays

Figura 7.1. *Una tabla de hoja de cálculo es una buena metáfora para saber qué es un array. Este array, que almacena los títulos de los capítulos, utiliza números a modo de índice (o clave).*

Índice o clave	Valor
Chapter_1	Cómo empezar con PHP
Chapter_2	Variables
Chapter_3	Formularios HTML y PHP
Chapter_4	Utilización de números
Chapter_5	Utilización de cadenas
Chapter_6	Estructuras de control
Chapter_7	Uso de los arrays

Figura 7.2. *En esta versión en forma de tabla del array, el índice (o clave) son palabras, no números.*

Script 7.1. Siempre resulta más adecuado incluir muchos comentarios que pocos, porque lo que parece obvio a la hora de iniciar la programación, puede no serlo tanto cuando tenga que volver a ella al cabo del tiempo.

script

```
1  <HTML>
2  <HEAD>
3  <TITLE>Using Arrays</TITLE>
4  <BODY>
5  <?php
6  $Soups = array(
7  "Monday"=>"Clam Chowder", "Tuesday"=>
   "White Chicken Chili",
8  "Wednesday"=>"Vegetarian"
9  );
10 print ("$Soups<P>\n");
11 ?>
12 </BODY>
13 </HTML>
```

Creación de un array

Para la creación de arrays debe utilizar array(), cuya sintaxis es:

```
$List = array ("manzanas", "bananas",
"naranjas");
```

En este ejemplo (donde no se ha especificado un índice para los elementos), el primer artículo, manzanas, ha sido indexado automáticamente como 0, el segundo como 1, y el tercero como 2. Cuando use un array(), puede asignar el índice de la siguiente manera:

```
$List = array (1=>"manzanas",
→ 2=>"bananas", 3=>"naranjas");
```

El valor del índice que especifique no tiene por qué ser un número, también puede emplear palabras. Esta técnica de indexado es muy práctica para la creación de listas con más significados. Como en el ejemplo, puede crear una lista donde registre la sopa del día durante todos los días de la semana.

Cómo crear un array

1. Cree un nuevo documento PHP en el editor de textos.

2. Escriba un título HTML estándar (véase el *Script* 7.1).

   ```
   <HTML><HEAD><TITLE>Using Arrays
   → </TITLE><BODY>
   ```

3. Comience la sección PHP del *script* y use la función array() para crear un array.

   ```
   <?php
   $Soups = array(
   "Monday"=>"Clam Chowder",
   "Tuesday"=>"White Chicken Chili",
   "Wednesday"=>"Vegetarian"
   );
   ```

   Éste es el formato adecuado para iniciar (crear y asignar un valor a) un array en PHP, empleando cadenas como índices.

4. Imprima el array al navegador web.

```
print ("$soups<P>\n");
```

5. Cierre los documentos PHP y HTML.

```
?></BODY></HTML>
```

6. Guarde el documento como soups.php, cárguelo en el servidor y pruébelo en el navegador web (véase la Figura 7.3).

CONSEJO

- La práctica de comenzar cada índice en cero es estándar en PHP y en la mayoría de lenguajes de programación. Por muy poco natural que parezca, ha prevalecido, por lo que sólo le quedan dos posibilidades. La primera, comenzar manualmente los arrays indexando desde la posición 1. Segunda, olvidar toda una vida contando desde 1. Decida la manera que le resulte más sencilla, pero los programadores ya se han habituado a este método tan extraño de construcción.

Figura 7.3. Puesto que un array está estructurado de forma diferente a otros tipos de variables, una solicitud de imprimir un array dará como resultado la impresión de la palabra "array". Aunque esta impresión no es tan útil como imprimir los valores individuales del array (como aprenderá), sí le confirma que el array se ha creado satisfactoriamente.

Adición de elementos a un array

En PHP, una vez haya creado el array, puede añadir elementos extra asignándolos directamente con el operador de asignación (el signo igual), similar al modo de asignar un valor a una cadena o a un número. Cuando lo haga, puede especificar la clave del elemento añadido o no, pero en cualquier caso, debe hacer referencia al array con los corchetes. Para añadir dos elementos a la lista $List, escriba:

```
$List[] = "pears";
$List[] = "tomatoes";
```

Si no especifica la clave, cada elemento será añadido al array existente, siendo indexado con el siguiente número lógico. Suponiendo que fuese el array de la sección anterior, donde estaban indexados como 1, 2 y 3, pears ocuparían ahora la cuarta posición y tomatoes la quinta.

Si especifica el índice, el valor se asignará a la ubicación y cada valor existente que esté indexado en ese punto se eliminará, como en:

```
$List[3] = "pears";
$List[4] = "tomatoes";
```

Ahora, el valor del elemento en la cuarta posición del array es tomatoes, y ningún elemento de $List es igual a naranjas. Recordando esto, le recomendamos precaución, ya que es mejor que no incluya una clave específica cuando añada los valores a los arrays, a no ser que quiera eliminar algún dato existente. (Sin embargo, si los arrays emplean cadenas de índices, debería utilizar claves específicas para que los valores no se pierdan.)

Para probar este proceso, volverá a escribir soups.php, para añadir más elementos al array. Para ver la diferencia que se produce al añadir más elementos, imprima los elementos que hay en el array antes y después de las nuevas entradas.

Al igual que puede saber la longitud de un cadena (cuántos caracteres contiene), usando strlen(), también puede determinar el número de elementos de un array empleando count().

```
$HowMany = count($Array);
```

Cómo añadir elementos a un array

1. Abra soups.php en el editor de texto.
2. Tras iniciar el array usando array(), añada (véase el *Script* 7.2):

```
$HowMany = count($Soups);
print ("The array contains
→ $HowMany.<P>\n");
```

La función count() determinará cuántos elementos hay en $Soups. Asignando el valor a la variable, podrá imprimirlo fácilmente.

3. Añada tres elementos más al array.

```
$Soups["Thursday"] = "Chicken
→ Noodle";
$Soups["Friday"] = "Tomato";
$Soups["Saturday"] = "Cream of
→ Broccoli";
```

4. Recuente cuántos elementos hay en el array e imprima su valor.

```
$HowManyNow = count($Soups);
print ("The array now contains
→ $HowManyNow.<P>\n");
```

5. Guarde el *script*, cárguelo en el servidor y pruébelo en el navegador web (véase la Figura 7.4).

La función que permite añadir un array dentro de otro es nueva en PHP 4.0. Piense que es una concatenación de arrays. La función array_merge() trabaja de la siguiente manera:

```
$NewArray = array_merge ($OneArray,
→ $TwoArray);
```

Puede volver a escribir la página soups.php usando esta nueva función si trabaja con un servidor que tenga PHP 4.0.

***Script* 7.2.** *Puede añadir directamente elementos a un array, a la vez que les asigna un valor con el operador de asignación. La función count() le ayuda a vigilar los elementos que contienen los arrays.*

script

```
1  <HTML>
2  <HEAD>
3  <TITLE>Using Arrays</TITLE>
4  <BODY>
5  <?php
6  $Soups = array(
7  "Monday"=>"Clam Chowder", "Tuesday"=>
   "White Chicken Chili",
8  "Wednesday"=>"Vegetarian"
9  );
10 $HowMany = count($Soups);
11 print ("The array contains $HowMany
   elements.<P>\n");
12 $Soups["Thursday"] = "Chicken Noodle";
13 $Soups["Friday"] = "Tomato";
14 $Soups["Saturday"] = "Cream of Broccoli";
15 $HowManyNow = count($Soups);
16 print ("The array now contains $HowManyNow
   elements.<P>\n");
17 ?>
18 </BODY>
19 </HTML>
```

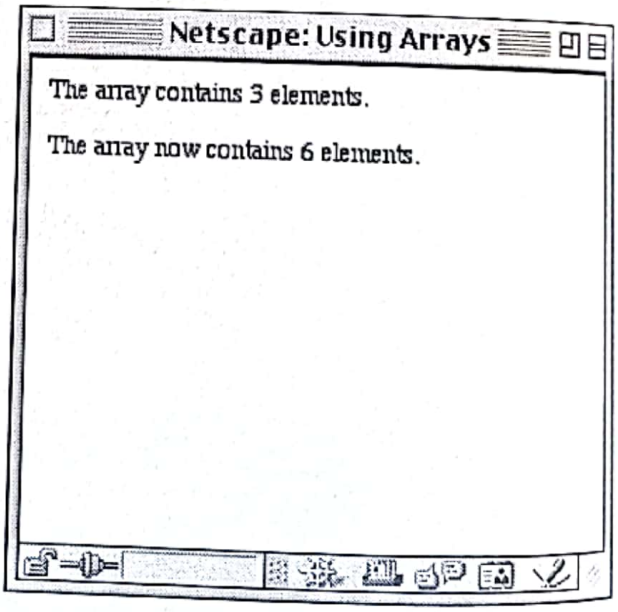


***Figura* 7.4.** *Una forma directa de asegurarse de que los elementos nuevos se han añadido satisfactoriamente al array consiste en contar el número de elementos antes y después de efectuar las adiciones.*

Script 7.3. La función array_merge() es nueva en PHP 4. Es una de las nuevas funciones de arrays diseñada para ahorrarle tiempo al programador.

script

```
1  <HTML>
2  <HEAD>
3  <TITLE>Using Arrays</TITLE>
4  <BODY>
5  <?php
6  $Soups = array(
7  "Monday"=>"Clam Chowder", "Tuesday"=>"White
   Chicken Chili",
8  "Wednesday"=>"Vegetarian"
9  );
10 $HowMany = count($Soups);
11 print ("The \$Soups array contains $HowMany
   elements.<P>\n");
12 $Soups2 = array(
13 "Thursday"=>"Chicken Noodle",
14 "Friday"=>"Tomato",
15 "Saturday"=>"Cream of Broccoli"
16 );
17 $HowMany2 = count($Soups2);
18 print ("The \$Soups2 array contains $How-
   Many2 elements.<P>\n");
19 $TheSoups = array_merge ($Soups, $Soups2);
20 $HowMany3 = count($TheSoups);
21 print ("The \$TheSoups array contains
   $HowMany3 elements.<P>\n");
22 ?>
23 </BODY>
24 </HTML>
```

Cómo unir dos arrays

1. Abra soups.php en el editor de textos.

2. Tras iniciar el array $Soups, cuente e imprima el número de arrays que contiene (véase el *Script* 7.3):

```
$HowMany = count($Soups);
print ("The \$Soups array contains
→ $HowMany elements.<P>\n");
```

3. Cree un segundo array, y cuente e imprima el número de elementos que contiene.

```
$Soups2 = array(
→ "Thursday"=>"Chicken Noodle",
→ "Friday"=>"Tomato",
→ "Saturday"=>"Cream of Broccoli"
);
$HowMany2 = count($Soups2);
print ("The \$Soups2 array contains
→ $HowMany2 elements.<P>\n");
```

4. Una los dos arrays en un nuevo array.

```
$TheSoups = array_merge ($Soups,
→ $Soups2);
```

Asegúrese de que coloca los arrays en este orden ($Soups, y después, $Soups2), para que los elementos que van de Thursday a Saturday, se añadan a los elementos que van de Monday a Wednesday, y no al revés.

5. Cuente e imprima el número de elementos del nuevo array que ha creado.

```
$HowMany3 = count($TheSoups);
print ("The \$TheSoups array contains
→ $HowMany3 elements.<P>\n");
```

6. Cierre los documentos PHP y HTML.

7. Guarde el archivo, cárguelo en el servidor y pruébelo en el navegador web (véase la Figura 7.5).

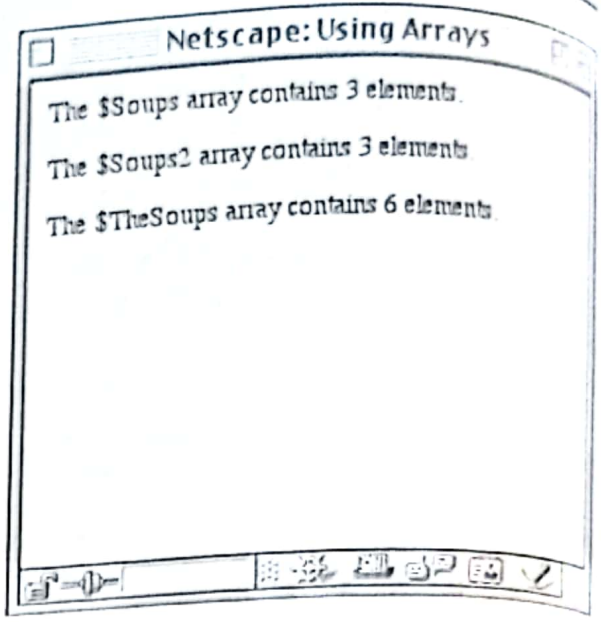


Figura 7.5. La función count() se utiliza aquí para asegurarnos de que todos los elementos de los dos arrays están presentes en el array de nueva creación.

CONSEJOS

- Sea prudente cuando añada directamente elementos a un array. Existe una manera correcta de hacerlo ($Array[] = "Add This", o $Array[1] = → "Add This";) y una forma incorrecta ($Array = → "Add This"). Olvidar el uso de los paréntesis motiva que el valor añadido reemplace al array existente y tan sólo se obtenga una cadena o un número.
- Algunas de las nuevas funciones de PHP 4 para manejar arrays no han podido tratarse aquí. El manual de PHP, disponible en la página principal de PHP, los trata a fondo. No use una función específica de PHP 4 si el servidor está funcionando con PHP 3.

Acceso a los elementos del array

Sólo existe una manera de recobrar un elemento específico (o valor) de un array, sin importar la forma en la que lo establezca: hacer referencia al índice. Una opción para manejar los arrays es, usando el operador de asignación, asignar un valor de elemento específico a una variable separada:

```
$Total = $Array[1];
```

Al hacer esto puede conservar el valor original en el array y manipularlo separadamente como una variable.

Sin embargo, también puede hacer una referencia directa a un valor de elemento específico y emplearlo casi siempre como si fuese una cadena:

```
print ("The total of your order comes
→ to $Array[Total]");
```

Cuando imprima un array, debe omitir las comillas dobles que utilizaría normalmente en el índice cuando sean antagónicas con la instrucción print(). Esta línea le dará problemas:

```
print ("The total of your order comes
→ to $Array["Total"]");
```

Éste es un ejemplo donde la utilización de comillas no le dará ningún problema:

```
$Array["Name"] = trim($Array["Name"]);
```

Irónicamente, la función que hace que los arrays sean tan útiles (pudiendo almacenar varios valores en una variable), también pone una limitación que no tienen las variables de otros tipos, o sea, que debe saber las claves del array para poder acceder a los elementos. Si la clave se configura usando cadenas, como las del array $Soups, que se refieren a $Soups[1], no tendremos que señalar nada. Para este asunto, como las variables permiten la utilización de mayúsculas y minúsculas, $Soups["monday"] no tendría valor cuando indexó Clam Chowder en Soups["Monday"].

Afortunadamente, la manera más rápida y fácil de acceder a los valores de un array es usar un bucle en conjunción con la función each(). Esta función crea un array del índice y de los valores del array. Dentro de un bucle, se puede repetir each() tantas veces como elementos tenga el array, hasta que devuelva cada valor.

```
for ($n = 0; $n < count($Array); $n++) {
    $Line = each ($Array);
    print ("La clave es igual a
$Line[key].<BR>El valor es igual a
→ $Line[value].");
}
```

En este ejemplo, la función each() creará un array llamado $Line, con la clave y el valor para $Array y para la posición actual. Piense que un array es como tener un señalizador interno. Al comenzar a usar la función each(), el puntero está en el primer elemento de $Array. La función each() recuperará esos valores y los asignará a $Line[key] (y también a $Line[0]) y a $Line[value] (también a $Line[1]), y después desplazará el puntero hacia el siguiente elemento. La segunda vez que llame a la función each(), recuperará el segundo conjunto de valores y se repetirá el proceso hasta que no haya más elementos en el array y el bucle se haya extinguido.

Ahora puede volver a escribir el *script* soups.php para aplicar lo aprendido. Además de poder imprimir simplemente los elementos que haya en el array (como ha hecho hasta este punto), ahora también puede acceder a los valores actuales.

Cómo imprimir los valores de cualquier array

1. Cree un nuevo documento PHP en el editor de textos.
2. Escriba el título HTML estándar (véase el *Script* 7.4).

```
<HTML><HEAD><TITLE>Using Arrays
→ </TITLE><BODY>
```

Script 7.4. *Un bucle es la manera más normal de acceder a cada elemento de un array. En este script, la función each() se emplea para determinar las claves y valores del array que se imprimirán en el navegador web.*

script

```
1  <HTML>
2  <HEAD>
3  <TITLE>Using Arrays</TITLE>
4  <BODY>
5  <?php
6  $Soups = array(
7      "Monday"=>"Clam Chowder",
8      "Tuesday"=>"White Chicken Chili",
9      "Wednesday"=>"Vegetarian",
10     "Thursday"=>"Chicken Noodle",
11     "Friday"=>"Tomato",
12     "Saturday"=>"Cream of Broccoli"
13 );
14 for ($n = 0; $n < count($Soups); $n++) {
15     $Line = each ($Soups);
16     print ("$Line[key]'s soup is $Line[va-
       lue].<P>\n");
17 }
18 ?>
19 </BODY>
20 </HTML>
```

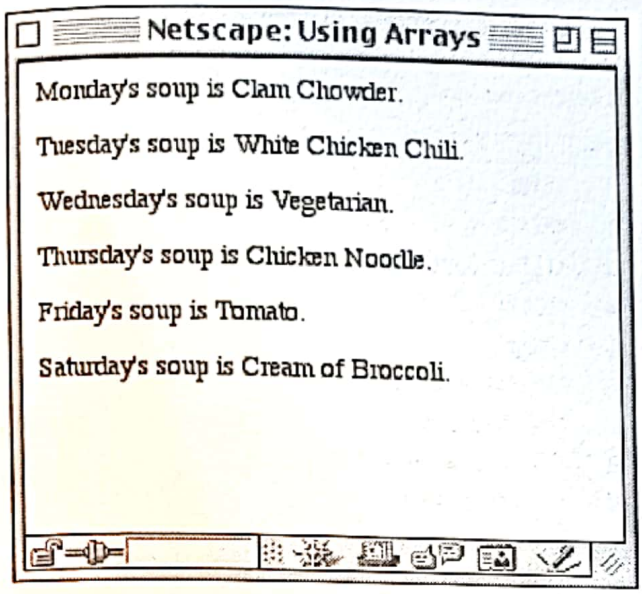


Figura 7.6. La ejecución del bucle para cada elemento de este array genera esta página. La función each() permite a PHP acceder a cada clave y valor sin conocimiento previo de lo que eran.

3. Comience la sección PHP de la página e inicie el array $Soups.

```
<?php
$Soups = array(
"Monday"=>"Clam Chowder",
"Tuesday"=>"White Chicken Chili",
"Wednesday"=>"Vegetarian",
"Thursday"=>"Chicken Noodle",
"Friday"=>"Tomato",
"Saturday"=>"Cream of Broccoli"
);
```

4. Inicie un bucle para acceder a cada elemento del array.

```
for ($n = 0; $n < count($Soups);
→ $n++) {
```

Este bucle asigna 0 a la variable $n. Después comprobará si esta variable es menor que el número de elementos del array. Si lo es, ejecutará el bucle para incrementar dicha variable.

5. Utilice la función each() para recuperar las claves y los valores; después imprímalos.

```
$Line = each ($Soups);
print ("$Line[key]'s soup is
→ $Line[value].<P>\n");
```

El bucle asignará las claves y los valores del array a $Line mediante la función each(). Después imprimirá la clave y el valor.

6. Cierre el bucle y las páginas PHP y HTML.

7. `}?></BODY></HTML>`

8. Guarde la página como soups.php, cárguela en el servidor y pruébela en el navegador web (véase la Figura 7.6).

CONSEJO

- Una alternativa es ajustar una variable al valor de count($Array), y emplearla en el bucle. De este modo, el PHP no tiene que recontar el array en cada interación.

Clasificación de los arrays

PHP soporta varios modos de clasificar un array (cuando decimos clasificar, nos referimos a una clasificación alfabética si es una cadena, y a una clasificación numérica si es un número). Cuando clasifique un array, debe recordar que está formado por varios pares de claves y valores. Así, un array puede clasificarse basándose en los valores de las claves. También puede clasificar los valores y mantener las claves correspondientes emparejadas, o clasificar los valores y hacer que reciban claves nuevas.

Para clasificar los valores, sin dar importancia a las claves, use sort(). Para clasificar estos valores (de nuevo sin dar importancia a las claves), pero en orden inverso, emplee rsort(). La sintaxis para cada función de clasificación es la siguiente:

```
function($Array);
```

Por lo que sort() y rsort() quedarán simplemente como:

```
sort($Array);
rsort($Array);
```

Para clasificar los valores, utilice asort() mientras se mantenga la correlación entre el valor y la clave. Para clasificarlos en orden inverso, emplee arsort() mientras se mantenga la correlación con la clave.

Para clasificar las claves, use ksort() mientras se mantenga la correlación entre la clave y el valor. Para clasificarlas de forma inversa, utilice krsort().

Por último, shuffle() reorganiza aleatoriamente el orden del array.

Como ejemplo de clasificación de arrays, creará una lista de estudiantes y la puntuación que han obtenido en un examen. Después clasificará esta lista, primero por la puntuación y después por el nombre.

Cómo clasificar un array

1. Abra el editor de textos, cree y titule un nuevo documento PHP.
2. Comience a codificar los documentos HTML y PHP estándares (véase el *Script* 7.5):

```
<HTML><HEAD><TITLE>Sorting Arrays
→ </TITLE><BODY><?php
```

Script 7.5. Existen varias funciones diferentes en PHP para clasificar arrays, incluyendo arsort() y ksort(), que ya hemos empleado.

script

```
1  <HTML>
2  <HEAD>
3  <TITLE>Sorting Arrays</TITLE>
4  <BODY>
5  <?php
6  $Grades = array(
7      "Richard"=>"95",
8      "Sherwood"=>"82",
9      "Toni"=>"98",
10     "Franz"=>"87",
11     "Melissa"=>"75",
12     "Roddy"=>"85"
13 );
14 print ("Originally, the array looks like
   this:<BR>");
15 for ($n = 0; $n < count($Grades); $n++) {
16     $Line = each ($Grades);
17     print ("$Line[key]'s grade is
       $Line[value].<BR>\n");
18 }
19 arsort($Grades);
20 reset($Grades);
21 print ("<P>After sorting the array by value
   using arsort(), the array looks like
   this:<BR>");
22 for ($n = 0; $n < count($Grades); $n++) {
23     $Line = each ($Grades);
24     print ("$Line[key]'s grade is
       $Line[value].<BR>\n");
25 }
26 ksort($Grades);
27 reset($Grades);
28 print ("<P>Tras clasificar el array por
   claves usando ksort(), este es el
   resultado:<BR>");
29 for ($n = 0; $n < count($Grades); $n++) {
30     $Line = each ($Grades);
31     print ("$Line[key]'s grade is
       $Line[value].<BR>\n");
32 }
33 ?>
34 </BODY>
35 </HTML>
```

3. Cree el array:

```
$Grades = array(
"Richard"=>"95",
"Sherwood"=>"82",
"Toni"=>"98",
"Franz"=>"87",
"Melissa"=>"75",
"Roddy"=>"85"
);
```

4. Imprima un subtítulo, y después imprima cada elemento del array empleando un bucle.

```
print ("Originally, the array looks
→ like this:<BR>");
for ($n = 0; $n < count($Grades);
→ $n++) {
    $Line = each ($Grades);
    print ("$Line[key]'s grade is
    → $Line[value].<BR>\n");
}
```

5. Clasifique el array en orden inverso a los valores, para determinar quién ha obtenido la mejor nota.

```
arsort($Grades);
```

Como ya ha determinado quién ha obtenido la mejor nota, necesitará usar arsort() en lugar asort(). La última clasifica el array en orden numérico, por lo que el resultado sería 75, 82, 85, etc., y no el deseado 98, 95, 87, etc.

Puede emplear también arsort() y no rsort() para mantener la relación entre el valor y la clave (con rsort() no podría).

6. Reajuste el puntero del array utilizando la función reset().

```
reset($Grades);
```

Para asegurarse de que el bucle comienza con el primer elemento del array $Grade, utilice la función reset(), que devuelve el puntero al primer elemento del array. Se debe hacer esto porque el bucle anterior ha trasladado el puntero hasta el final del array.

7. Imprima de nuevo el array (con precaución), usando otro bucle.

```
print ("<P>After sorting the array
→ by value using arsort(), the
→ array looks like this:<BR>");
for ($n = 0; $n < count($Grades);
→ $n++) {
    $Line = each ($Grades);
    print ("$Line[key]'s grade is
    → $Line[value].<BR>\n");
}
```

8. Ahora clasifique el array por claves, poniendo en orden alfabético los nombres de los estudiantes, y reajuste de nuevo el array.

```
ksort($Grades);
reset($Grades);
```

La función ksort() organizará el array por claves (en este caso, alfabéticamente) mientras se mantiene la correlación entre clave y valor.

9. Imprima un subtítulo y el array una vez más.

```
print ("<P>After sorting the array
→ by key using ksort(), the array
→ looks like this:<BR>");
for ($n = 0; $n < count($Grades);
→ $n++) {
    $Line = each ($Grades);
    print ("$Line[key]'s grade is
    → $Line[value].<BR>\n");
}
```

10. Cierre el *script* con las etiquetas estándar de PHP y HTML:

```
?></BODY></HTML>
```

11. Guarde el *script* como sort.php, cárguelo en el servidor, y pruébelo en el navegador web (véase la Figura 7.7).

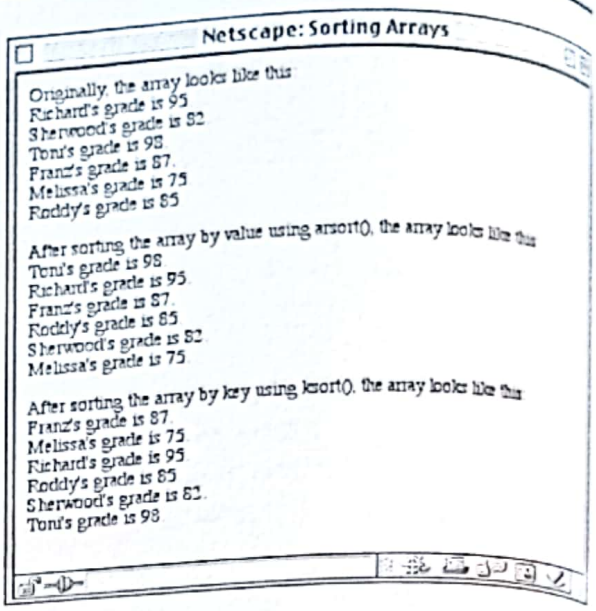


Figura 7.7. Puede ordenar un array de varias formas con distintos resultados. Preste mucha atención a si quiere o no mantener la asociación clave-valor al seleccionar una función de ordenación.

Transformaciones entre cadenas y arrays

Ahora que entiende las cadenas y los arrays, introduciremos dos funciones para cambiar entre ambos formatos. La primera, implode(), transforma un array en un cadena. La segunda, explode(), hace lo contrario. Éstas son algunas buenas razones para emplear estas funciones:

- Puede que quiera transformar un array en una cadena para pasar el valor añadido a un URL (lo que no sería fácil con un array).
- Puede que quiera transformar un array en una cadena para almacenar información en una base de datos.
- Puede que quiera transformar una cadena en un array, para convertir un campo de texto limitado por comas en sus partes separadas.

La sintaxis para el uso de explode() es:

```
$Array = explode ($Separator, $String);
```

El separador hace referencia a cualquier carácter o caracteres que se definen donde finaliza un elemento del array y comienza otro. Normalmente sería una coma o un espacio en blanco. El código sería:

```
$Array = explode(",", $String);
```

O bien:

```
$Array = explode(" ", $String);
```

Para pasar de un array a una cadena tiene que definir cuál va ser el separador y PHP hará el resto.

```
$String = implode($Glue, $Array);
$String = implode(",", $Array);
```

O bien:

```
$String = implode(" ", $Array);
```

Para mostrarle el uso de explode() e implode(), creará un formulario HTML que tenga una cadena de nombres de usuario delimitada por comas. PHP transformará la cadena en un array para que se pueda clasificar la lista. Finalmente, el código creará y devolverá la cadena ordenada alfabéticamente.

Cómo transformar cadenas y arrays

1. Cree un nuevo documento HTML en el editor de textos.

2. Escriba el título HTML estándar (véase el *Script* 7.6):

```
<HTML><HEAD><TITLE>HTML Form
→ </TITLE></HEAD><BODY>
```

3. Cree un formulario HTML con una salida TEXT.

```
<FORM ACTION="HandleList.php"
→ METHOD=POST>
Enter the words you want alphabetized
→ with each individual word separated
→ by a space:<BR>
<INPUT TYPE=TEXT NAME="List"
→ SIZE=80><BR>
```

Es muy importante instruir al usuario en casos como éste. Por ejemplo, si introduce una lista delimitada por comas, no podrá trabajar adecuadamente con la cadena (tras completar ambos *scripts*, intente usar comas en lugar de los espacios y vea el resultado).

4. Cree un botón de envío, y después cierre la página HTML y el formulario.

```
<INPUT TYPE=SUBMIT NAME="SUBMIT"
→ VALUE="Submit!">
</FORM></BODY></HTML>
```

5. Guarde el *script* como list.html y cárguelo en el servidor.

Ahora escriba la página HandleList.php para procesar los datos generados por list.html.

Script 7.6. *Éste es un sencillo formulario HTML donde el usuario puede enviar una lista de palabras. Incluir las instrucciones detalladas para el usuario es una norma de diseño web muy prudente.*

script

```
1  <HTML>
2  <HEAD>
3  <TITLE>HTML Form</TITLE>
4  </HEAD>
5  <BODY>
6  <FORM ACTION="HandleList.php" METHOD=POST>
7  Enter the words you want alphabetized with
   each individual word separated by a
   space:<BR>
8  <INPUT TYPE=TEXT NAME="List" SIZE=80><BR>
9  <INPUT TYPE=SUBMIT NAME="SUBMIT"
   VALUE="Submit!">
10 </FORM>
11 </BODY>
12 </HTML>
```

Script 7.7. Ya que las funciones explode() e implode() son simples y poderosas, puede enviar rápida y fácilmente una lista de palabras (prácticamente de cualquier longitud) en sólo dos líneas.

```
script
1  <HTML>
2  <HEAD>
3  <TITLE>Alphabetizing Example</TITLE>
4  <BODY>
5  <?php
6  /* Esta página manipula y recibe los datos
   generados por "list.html". */
7  $Array = explode (" ", $List);
8  sort ($Array);
9  $NewList = implode ("<BR>", $Array);
10 print ("An alphabetized version of your
   list is:<BR>$NewList");
11 ?>
12 </BODY>
13 </HTML>
```

6. Cree un nuevo documento HTML en el editor de textos.

7. Escriba el título HTML estándar y abra la sección PHP de la página (véase el *Script* 7.7):

```
<HTML><HEAD><TITLE>Alphabetizing
→ Example</TITLE><BODY><?php
```

8. Transforme la cadena entrante, $List, en un array.

```
$Array = explode (" ", $List);
```

Esta línea de código creará un array nuevo, $Array, fuera de la cadena $List. Cada espacio entre palabras de $List indica que ajusta la nueva palabra como un nuevo elemento de array. Comenzando en la primera palabra, $Array[0], se incluye un espacio en $List, después llega la segunda palabra, $Array[1], y así sucesivamente hasta el final de $List.

9. Clasifique el array alfabéticamente.

```
sort ($Array);
```

Como no necesita mantener las asociaciones entre clave y valor en $Array, puede emplear sort() en lugar de asort(), que ha empleado anteriormente.

10. Cree una nueva cadena de arrays clasificados.

```
$NewList = implode ("<BR>", $Array);
```

Imprima la nueva lista, y como los arrays no se imprimen tan fácilmente como las cadenas, transforme primero $Array en una cadena a la que llamará $NewList. La cadena resultante comenzará con el valor $Array[0], seguido de la etiqueta HTML <BR>, y seguido también por el valor $Array[1], etc. El uso de <BR> en lugar de un espacio o de una coma, da a la lista un formato más legible cuando se imprime en un navegador.

11. Imprima la nueva lista al navegador.

```
print ("An alphabetized version of
› your list is:<BR>$NewList");
```

12. Cierre la sección de PHP y la página HTML.

```
?></BODY></HTML>
```

13. Guarde la página como HandleList.php, cárguela en el servidor (en el mismo directorio que list.html) y pruebe los dos *scripts* en el navegador web (véanse las Figuras 7.8 y 7.9).

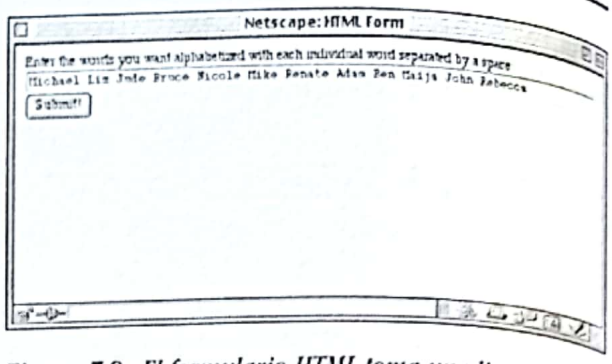


Figura 7.8. El formulario HTML toma una lista de palabras que después ordena alfabéticamente el script HandleList.php (véase la Figura 7.9).

CONSEJO

- También puede atravesar rápidamente un código escrito utilizando la función join(), que es idéntica a implode().

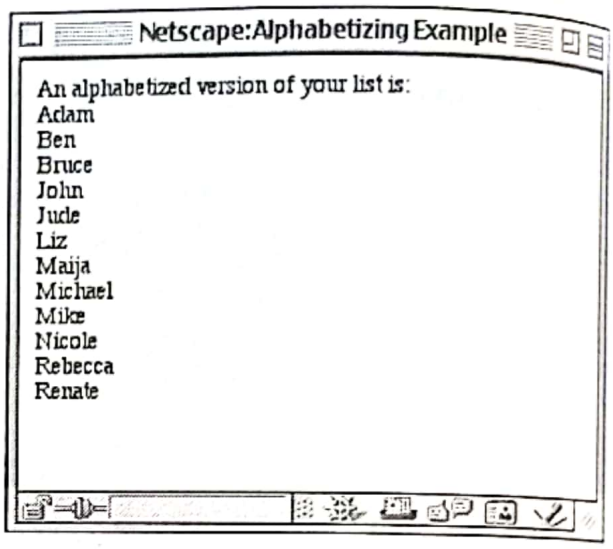


Figura 7.9. La misma lista, ordenada alfabéticamente por el usuario. Es un proceso rápido y sencillo de codificar, pero imposible sin la existencia de los arrays.

Script 7.8. Éste es el formulario HTML con un array para los nombres INPUT. No tiene que especificar las claves de cada array, porque puede etiquetar cada campo Array[], en cuyo caso HandleForm.php mantendría los datos de First Name en Array[0], de Last Name en Array[1], E-mail Address en Array[2], y Comments en Array[3].

```
script
1  <HTML>
2  <HEAD>
3  <TITLE>HTML Form</TITLE>
4  </HEAD>
5  <BODY>
6  <FORM ACTION="HandleForm.php" METHOD=POST>
7  First Name <INPUT TYPE=TEXT NAME=
   "Array[FirstName]" SIZE=20><BR>
8  Last Name <INPUT TYPE=TEXT NAME=
   "Array[LastName]" SIZE=40><BR>
9  E-mail Address <INPUT TYPE=TEXT NAME=
   "Array[Email]" SIZE=60><BR>
10 Comments <TEXTAREA NAME="Array[Comments]"
   ROWS=5 COLS=40></TEXTAREA><BR>
11 <INPUT TYPE=SUBMIT NAME="SUBMIT"
   VALUE="Submit!">
12 </FORM>
13 </BODY>
14 </HTML>
```

Creación de un array desde un formulario

En este capítulo hemos establecido arrays desde una página PHP. Sin embargo, también puede enviar un array de datos a un *script* PHP a través de un formulario HTML. La codificación es tan sólo un poco más complicada que la que ha hecho hasta ahora. Puede rescribir la página de retroalimentación del Capítulo 3, para que genere un array en lugar de variables individuales.

Cómo crear un array con un formulario HTML

1. Cree un nuevo documento HTML en el editor de textos.

2. Cree el título estándar de HTML (véase el *Script* 7.8).

```
<HTML><HEAD><TITLE>HTML Form
→ </TITLE></HEAD><BODY>
```

3. Comience el formulario HTML.

```
<FORM ACTION="HandleForm.php"
→ METHOD=POST>
```

4. Cree tres cuadros de entrada TEXT con arrays para los atributos NAME.

```
First Name <INPUT TYPE=TEXT NAME=
→ "Array[FirstName]" SIZE=20><BR>
Last Name <INPUT TYPE=TEXT NAME=
→ "Array[LastName]" SIZE=40><BR>
E-mail Address <INPUT TYPE=TEXT NAME=
→ "Array[Email]" SIZE=60><BR>
```

Antes, cuando creó el formulario, el *script* HandleForm.php recibió variables como $FirstName y $LastName.

Ahora, recibirá $Array[FirstName], $Array[LastName], etc., que crearán automáticamente el array indexado. Hemos omitido las comillas alrededor de la clave del array (por ejemplo, Array["LastName"]), lo que es perfectamente aceptable, y ayuda a evitar errores sintácticos en tal situación.

5. Cree un TEXTAREA que también sea parte del array.

```
Comments <TEXTAREA NAME=
→ "Array[Comments]" ROWS=5
→ COLS=40></TEXTAREA><BR>
```

6. Guarde el *script* como form.html y cárguelo en el servidor.

   Necesitará escribir una nueva página HandleForm.php para reflejar los cambios para form.html.

7. Cree un nuevo documento PHP en el editor de textos.

8. Comience con el título estándar de HTML, seguido de la etiqueta inicial PHP (véase el *Script* 7.9).

```
<HTML><HEAD><TITLE>Form Results/
→ Using Arrays</TITLE><BODY><?php
```

9. Seleccione los elementos FistName y LastName del array y únalos para formar un nuevo elemento del array.

```
$Array["Name"] =
→ $Array["FirstName"] . " " .
→ $Array["LastName"];
```

   Ha añadido otra pieza de información al array, pero puede seguir gestionando todos los datos con la única variable.

10. Imprima el nombre completo enviado al navegador, con el fin de confirmar que recibió con éxito los datos del formulario.

```
print ("Your full name is
→ $Array[Name].<BR>\n");
```

11. Guarde la página como HandleForm.php, cárguela en el servidor (en el mismo directorio que form.html) y revise ambas páginas en el navegador (véanse las Figuras 7.10, 7.11 y 7.12).

CONSEJOS

- También puede usar arrays en formularios HTML cuando utilice varios cuadros de verificación o un menú contextual en el que se pueden seleccionar varias opciones.

Script 7.9. No hay ningún problema con PHP si recibe un número de cadenas de un formulario HTML, o sólo una cadena, como sucede en este caso. Sin embargo, el uso de un array le dejaría menos variables con las que trabajar.

script

```
1  <HTML>
2  <HEAD>
3  <TITLE>Form Results/Using Arrays</TITLE>
4  <BODY>
5  <?php
6  /* Esta página recibe y maneja los datos
   generados por "form.html". */
7  $Array["Name"] = $Array["FirstName"] . "
   " . $Array["LastName"];
8  print ("Your full name is
   $Array[Name].<BR>\n");
9  ?>
10 </BODY>
11 </HTML>
```

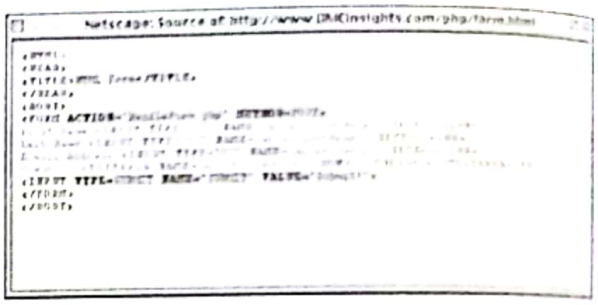


Figura 7.10. Código fuente de la página form.html, ahora que utiliza un array para la entrada de nombres.

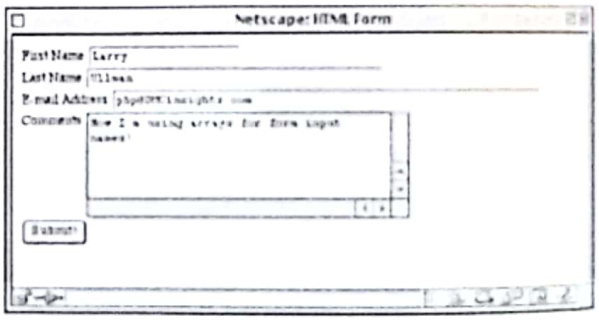


Figura 7.11. Al usuario no le importan los nombres que utilice en el formulario, pues normalmente sólo tiene esta vista. Con esto en mente, seleccione nombres que faciliten su trabajo de programación.

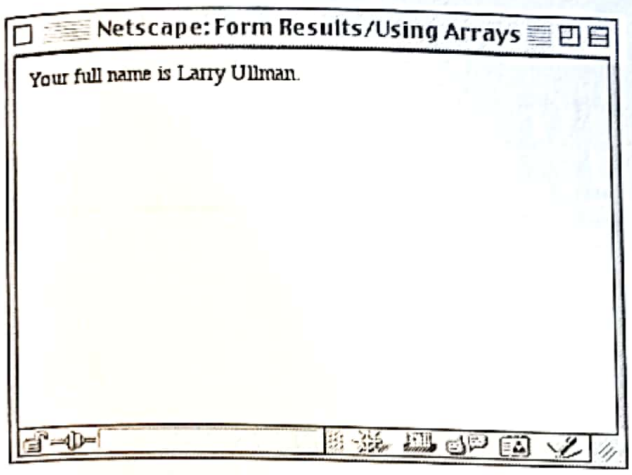


Figura 7.12. Para confirmar que los datos del formulario (véase la Figura 7.11) se han construido satisfactoriamente para la página HandleForm.php, se imprime este sencillo mensaje.

Creación de arrays multidimensionales

Los arrays multidimensionales (el equivalente PHP a un *hash* de Perl) son al mismo tiempo sencillos y complejos. Éste es otro tema en el que posiblemente quiera ahondar, una vez se haya familiarizado con todos los conceptos.

La teoría que se esconde trasun array multidimensional es que puede crear un array con más información incluso que en el caso de un array estándar, incorporando otros arrays para valores, en lugar de sólo cadenas y números. Por ejemplo:

```
$Array1 = array("manzanas", "bananas",
→ "naranjas");
$Array2 = array("filetes",
→ "hamburguesas", "chuletas de cerdo");
$List = array("frutas"=>$Array1,
→ "carnes"=>$Array2,
"otros"=>"cacahuetes",
"efectivo" => 30.00);
```

Ahora, este array, $List, consta de una cadena (cacahuetes), un número de punto flotante (30.00), y dos arrays (frutas y carnes).

Es complicado señalar un elemento dentro de un array que se encuentra en otro array. La clave es continuar añadiendo índices tanto como sea necesario. Así, en el ejemplo, las bananas están en $List["frutas"][1].

En primer lugar, señale el elemento (en este caso, un array) en el array $List, usando ["frutas"]. Después, señale el elemento dentro de ese array basándose en su posición (el segundo elemento, así que use el índice [1]).

CONSEJOS

- Puede crear un array multidimensional en una instrucción usando una serie de llamadas a array() anidadas (en lugar de hacerlo a través de varios pasos, como en el caso de este ejemplo), pero no se lo recomendamos porque es muy fácil cometer errores sintácticos, dado que una instrucción se vuelve más y más anidada. Cuando necesite utilizar arrays multidimensionales, comience asignando cada array individual tal y como le mostramos aquí, colocándolo después en otro array, mejor que intentar programarlo con instrucciones array() anidadas.
- También puede resultar complicado intentar imprimir un valor de un array multidimensional. Las versiones de PHP anteriores a la 4 no le permiten imprimir este elemento directamente. En PHP 4, puede imprimir desde arrays multidimensionales dentro de una cadena: print ("The value I want to print is {$Array[index1] → [index2]}.");. Pero print("The value I want to → print is $Array[index1][index2]."); no funcionará en cualquier versión de PHP.

EXPRESIONES REGULARES

8

Si sólo hubiera dos cosas que debiera entender acerca de las expresiones regulares, serían, primero, que las expresiones regulares son sumamente útiles en programación avanzada, y, dos que son muy sencillas de comprender pero más complejas de escribir. Sin embargo, una vez haya entendido las reglas que rigen la escritura de expresiones regulares, este conocimiento adicional dará su fruto en forma de expresiones regulares que mejorarán la calidad de su programación y le ahorrarán tiempo a la larga.

En este capítulo identificaremos las expresiones regulares, discutiremos detalladamente su construcción y proporcionaremos algunos ejemplos útiles que demostrarán sus capacidades.

¿Qué son las expresiones regulares?

Un capítulo sobre las expresiones regulares sólo puede cubrir la punta del iceberg, por lo que tendrá que aprender mucho por su cuenta. Si le interesan estas expresiones y quiere aprender más sobre ellas, el Apéndice C le será de gran ayuda.

Piense en las expresiones regulares como en un sistema elaborado de patrones de coincidencia. Primero escribe el patrón, y después utiliza una de las funciones integradas de PHP para aplicar el patrón a una cadena de texto (las expresiones regulares son específicas para su uso con cadenas). En esencia, PHP cuenta con dos funciones para su uso con expresiones regulares y patrones de coincidencia (una sensible al uso de mayúsculas y minúsculas, y otra que no lo es) y otras dos para patrones de coincidencia y sustitución del texto coincidente por otro texto (de nuevo, una sensible al uso de mayúsculas y minúsculas, y otra que no lo es).

Empezaremos con la definición y comparación de un patrón sencillo, para después adentrarnos en modelos más complejos y terminar con la comparación y sustitución de patrones.

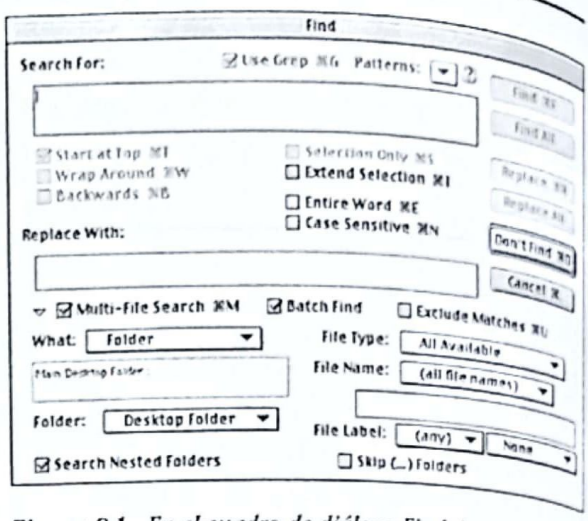


Figura 8.1. En el cuadro de diálogo Find de BBEdit, tiene la posibilidad de utilizar expresiones regulares (incluso a través de varios archivos o carpetas) activando la casilla de verificación Use Grep. BBEdit también puede almacenar patrones de expresiones regulares y, además, cuenta con algunas ya integradas.

CONSEJOS

- Algunos editores de textos, como BBEdit para Macintosh, TextPad para Windows y emacs para Unix, permiten el uso de expresiones regulares para comparar y sustituir patrones dentro y a través de varios documentos (véase la Figura 8.1). Ésta puede ser otra buena razón para aprender expresiones regulares y, quizá, puede llegar a ser algo a tener en consideración a la hora de elegir el editor de textos.
- El manual de PHP examina las diferencias a la hora de tratar las expresiones regulares entre PHP y Perl (consulte el apartado "Perl-compatible Regular Expression Functions" de la sección cuarta de la referencia de funciones). Los principiantes en PHP puede que quieran leer primero las páginas del manual.

Definición de un patrón sencillo

Antes de utilizar alguna de las funciones de expresiones regulares integradas de PHP, debe ser capaz de definir un patrón que luego utilizará la función para sus propósitos de comparación. PHP dispone de un conjunto de reglas para la creación de patrones, muchas de las cuales son similares a las utilizadas en Perl, C y Java. Puede utilizar estas reglas por separado o combinándolas, lo que hará que el patrón sea sencillo o muy complejo.

Para explicar cómo se crean los patrones, empezaremos con una introducción a los símbolos que se utilizan, después explicaremos cómo agrupar los caracteres y terminaremos con las clases. La combinación de símbolos, grupos y clases define un patrón. Como norma de formateo, definiremos los patrones entre comillas dobles ("") e indicaremos aquello con lo que debe coincidir el patrón en cursiva.

Literales

El primer tipo de carácter que utilizará para la definición de patrones será un literal. Un literal es un valor que se escribe exactamente igual a como se interpreta. Por ejemplo, el patrón "a" sólo coincidirá con la letra *a*, "ab" sólo coincidirá con *ab*, etc.

Los literales le permiten realizar comparaciones exactas, pero las expresiones regulares resultarían inútiles si sólo permitieran la comparación de literales (en cuyo caso sólo necesitaría algunas funciones de cadena, explicadas en el Capítulo 5). También puede utilizar símbolos específicos, que tienen su propio significado, para crear patrones menos rígidos. Los explicaremos más adelante.

Metacaracteres

Un paso más allá de los literales en términos de complejidad están los metacaracteres. Son símbolos especiales que tienen un significado más allá de su valor literal. Mientras una "a" simplemente significa *a*, el primer metacarácter, el punto (.), coincide con cualquier carácter (".." coincide con *a*, *b*, *c*, etc.). Esto es muy directo, aunque debe tener en cuenta que si quiere referirse literalmente a un metacarácter, debe expresarlo como secuencia de escape, lo mismo que se hace con unas comillas para poder imprimirlas. Así pues, "\." coincide con el punto.

A continuación tiene tres metacaracteres que le permiten múltiples ocurrencias: "a*" coincide con ninguna o más *a* (*a*, *aa*, *aaa*, etc.); "a+" coincide con una o más *a* (*a*, *aa*, *aaa*, etc., aunque aquí debe existir al menos una *a*); y "a?" coincidirá con una o ninguna *a*.

Para coincidir con una cierta cantidad de una letra, indique dicha cantidad entre llaves ({}), declarando un número específico, un mínimo, o un mínimo y un máximo. Así, "a{3}" sólo coincidirá con *aaa*; "a{3,}" coincidirá con *aaa*, *aaaa*, etc. (tres o más *a*); y "a{3,5}" coincidirá sólo con *aaa*, *aaaa* y *aaaaa* (entre tres y cinco).

Por último, tenemos el acento circunflejo (^), que coincide con cualquier cadena que empiece por el carácter que sigue al acento. También tenemos el signo de dólar ($), que coincide con todo lo que termine con el carácter precedente. En consecuencia, "^a." coincidirá con cualquier cadena de dos caracteres que empiece por *a*, seguida por cualquier otro carácter, mientras que ".a$" coincidirá con cualquier cadena de dos caracteres que termine con una *a*. Por consiguiente, "^a$" sólo coincidirá con *a*, lo que hace que este patrón sea idéntico a "a".

Tabla 8.1. Ésta es una lista bastante completa de los caracteres especiales utilizados para definir expresiones regulares (incluyendo los metacaracteres, pero no los literales: a, b, c, etc.).

Caracteres especiales para expresiones regulares

Carácter	Coincide con...
.	cualquier carácter
^a	empieza por "a"
a$	termina en "a"
a+	al menos tiene una "a"
a?	ninguna o una "a"
\n	línea nueva
\t	tabulador
\	secuencia de escape
(ab)	"ab" agrupadas
a|b	"a" o "b"
a{2}	"aa"
a{1,}	"a", "aa", "aaa", etc.
a{1,3}	"a", "aa", "aaa"
[a-z]	cualquier letra en minúscula
[A-Z]	cualquier letra en mayúscula
[0-9]	cualquier dígito

Las expresiones regulares también hacen uso del carácter de canalización (|) como equivalente de la operación lógica OR. Por tanto, "a|b" coincidirá con las cadenas *a* o *b* (el uso de la canalización dentro de los patrones se conoce con el nombre de **alternancia**).

Por supuesto, todo esto es una pequeña muestra de las repeticiones de coincidencia de una letra en una cadena, pero estos ejemplos son una buena forma de demostrar el funcionamiento de los símbolos. Debe empezar por comprender el significado de los distintos símbolos y cómo se utilizan.

- Cuando utilice las llaves para indicar un número de caracteres, siempre debe incluir el número mínimo, mientras que el máximo es opcional ("a{3}" y "a{3,}" son aceptables, pero "a{,3}" no lo es).
- Para incluir caracteres especiales (^.[]$()|*?{}\) en un patrón, debe expresarlos como secuencias de escape (con una barra invertida por delante de ellos). Esto es así para los metacaracteres y los símbolos de agrupación (paréntesis y corchetes). También puede utilizar la barra invertida para comparar con los caracteres de línea nueva ("\n") y tabulación ("\t"), esencialmente para crear un metacarácter fuera de su significado literal.
- La Tabla 8.1 presenta una lista con todos los símbolos y combinaciones a modo de referencia (junto con aquellos que describiremos de ahora en adelante).

Comparación de patrones

Existen dos funciones integradas en PHP expresamente con el propósito de comparar un patrón con una cadena: ereg() y eregi(). La única diferencia entre las dos es que ereg() es sensible a las mayúsculas y minúsculas de los patrones, mientras que eregi() es insensible al uso de mayúsculas y minúsculas, lo que la hace más particular. La última es generalmente la más utilizada, a menos que deba ser más explícito (quizá por propósitos de seguridad, como el control de contraseñas). Ambas funciones se evaluarán como verdaderas (TRUE) si el patrón coincide, y como falsas (FALSE) si no coincide. A continuación presentamos dos formas de utilizar estas funciones:

```
ereg("patrón", "cadena");
```

O:

```
$Pattern = "patrón";
$String = "cadena";
eregi($Pattern, $String);
```

Durante el resto del capítulo, asignaremos el patrón a una variable, como en el segundo ejemplo de los anteriores, para prestar más atención al propio patrón (el corazón de cualquier expresión regular).

Para demostrar la comparación de un patrón, creará HandleForm.php, que funcionará en conjunción con el archivo form.html, creado en el Capítulo 7 (véase el *Script* 8.1). El *script* HandleForm.php validará la dirección de correo electrónico enviada.

Script 8.1. *El form.html original fue diseñado para aceptar una dirección de correo electrónico y unos comentarios por parte del usuario, así como su nombre y apellidos.*

script

```
1  <HTML>
2  <HEAD>
3  <TITLE>HTML Form</TITLE>
4  </HEAD>
5  <BODY>
6  <FORM ACTION="HandleForm.php" METHOD=POST>
7  First Name <INPUT TYPE=TEXT NAME=
   "Array[FirstName]" SIZE=20><BR>
8  Last Name <INPUT TYPE=TEXT NAME=
   "Array[LastName]" SIZE=40><BR>
9  E-mail Address <INPUT TYPE=TEXT NAME=
   "Array[Email]" SIZE=60><BR>
10 Comments <TEXTAREA NAME="Array[Comments]"
   ROWS=5 COLS=40></TEXTAREA><BR>
11 <INPUT TYPE=SUBMIT NAME="SUBMIT"
   VALUE="Submit!">
12 </FORM>
13 </BODY>
14 </HTML>
```

Script 8.2. Este script hace un uso muy práctico de las expresiones regulares para validar que el usuario no sólo introdujo una dirección de correo electrónico en el formulario HTML, sino que realmente envió una dirección correcta.

script

```
1. <HTML>
2. <HEAD>
3. <TITLE>Using Regular Expressions</TITLE>
4. <BODY>
5. <?php
6. /* Esta página recibe y manipula los datos
   generados por "form.html". */
7. if (($Array["FirstName"]) AND
   ($Array["LastName"])) {
8.     $Array["Name"] = $Array["FirstName"] .
       " " . $Array["LastName"];
9. } else {
10.    print ("Please enter your first and
       last names.<BR>\n");
11. }
12. $Pattern = ".+@.+\..+";
13. if (eregi($Pattern, $Array["Email"])) {
14.    print ("Your information has been re-
       ceived!<BR>\n");
15. } else {
16.    print ("Please enter a valid email ad-
       dress!\n");
17. }
18. ¿>
19. </BODY>
20. </HTML>
```

Cómo comparar un patrón utilizando eregi

1. Cree un nuevo *script* HandleForm.php en el editor de textos.

2. Comience con la cabecera estándar HTML y PHP (véase el *Script* 8.2):

```
<HTML>
<HEAD>
<TITLE>Using Regular Expressions
→ </TITLE>
<BODY>
<?php
/* Esta página recibe y manipula los
→ datos generados por "form.html". */
```

3. Cree las estructuras condicionales necesarias.

```
if (($Array["FirstName"]) AND
→ '($Array["LastName"])) {
    $Array["Name"] =
    → '$Array["FirstName"] . " " .
    → '$Array["LastName"];
} else {
    print ("Please enter your first
    → and last names.<BR>\n");
}
```

Podría utilizar una expresión regular para asegurarse de que se ha introducido un nombre correcto, constituido únicamente por caracteres alfabéticos, pero es suficiente con asegurarse de que la variable tiene un valor. Después, si el usuario introduce un valor tanto para el nombre como para los apellidos, todos estos nombres son combinados en un solo elemento y asignados al array. Pero, si alguno es omitido, se imprime una solicitud.

4. A continuación, establezca el patrón para la comparación de la dirección de correo electrónico:

```
$Pattern = ".+@.+\..+";
```

Se trata del patrón para comparar una dirección de correo electrónico correcta. Aunque es bastante simple, funciona bien (posteriormente en este capítulo desarrollaremos un patrón más elaborado, aunque sólo sea para compararlos).

El primer paso del patrón indica que la dirección de correo electrónico debe empezar con al menos un carácter (.+). En segundo lugar, encontramos el símbolo @, que es necesario en toda dirección de correo electrónico. En tercer lugar, el patrón insiste en que a continuación debe aparecer al menos un carácter. En cuarto lugar, una dirección de correo electrónico debe incluir un punto. Por último, debe haber al menos un carácter concluyendo la cadena (no puede terminar con un punto).

5. Utilice el patrón para verificar una coincidencia con la dirección de correo electrónico enviada.

```
if (eregi($Pattern,
$Array["Email"])) {
print ("Your information has been
 . received!<BR>\n");
} else {
    print ("Please enter a valid
     . email address!\n");
}
```

Para utilizar la función de comparación eregi(), la alimentará con el patrón definido anteriormente y con la variable $Array["Email"] procedente de form.html. Si PHP encuentra el patrón dentro de la variable, la condición será verdadera y se imprimirá el mensaje apropiado. Si no encuentra el patrón dentro de la variable, se visualizará una solicitud instando a la introducción de una dirección de correo electrónico correcta. Como este tipo de direcciones hacen distinción entre mayúsculas y minúsculas, utilice eregi() en lugar de ereg().

6. Guarde el *script*, cárguelo en el servidor y pruébelo en el navegador (véanse las Figuras 8.2, 8.3 y 8.4).

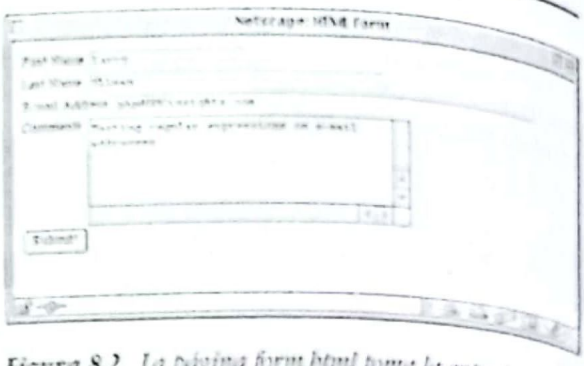


Figura 8.2. La página form.html toma la entrada del usuario y la envía para su comprobación mediante expresiones regulares a la página HandleForm.php (véase la Figura 8.3).

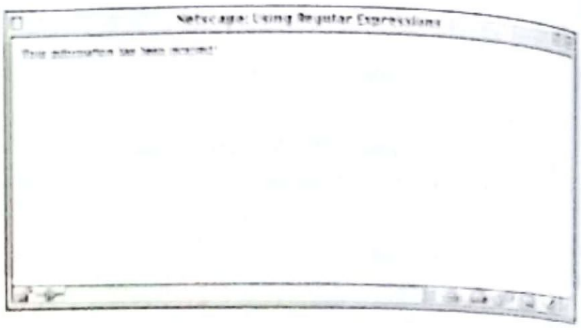


Figura 8.3. Utilizando eregi(), si el usuario introduce una dirección de correo electrónico que coincide con el patrón definido, el script HandleForm.php creará este mensaje. Compare esto con la Figura 8.4.

Figura 8.4. Aquí hemos proporcionado pbp@DMCinsights, que no es una dirección de correo electrónico correcta, obteniendo como resultado un mensaje de error.

CONSEJOS

- Aunque el aprendizaje de cómo escribir y ejecutar sus propias expresiones regulares demuestra su dedicación por la programación, numerosos ejemplos (incluyendo variantes del patrón de comparación de direcciones de correo electrónico) están disponibles en distintos sitios web dedicados a PHP. En el Apéndice C encontrará una referencia de los mismos.
- Para propósitos de demostración, hemos asignado un patrón a una variable a la que luego hemos hecho referencia en la función eregi(). Sin embargo, puede colocar el patrón directamente dentro de la función, si así lo desea:

```
if (eregi(".+@.+\..+", $Array
→ ["Email"])) {...
```

- La validación de un formulario, similar a lo codificado anteriormente, también puede lograrse como un proceso del lado del cliente utilizando JavaScript y expresiones regulares. Dependiendo de las circunstancias, puede utilizar las expresiones regulares en uno u otro lenguaje, o en ambos.

Definición de patrones más complejos

Una vez ha comprendido cómo utilizar literales y metacaracteres para crear un patrón, es hora de pasar a las agrupaciones y las clases, dos conceptos que le permitirán crear patrones más complejos.

Agrupaciones

Utilizando los símbolos básicos establecidos hasta ahora, puede empezar a incorporar paréntesis para agrupar caracteres en patrones más complicados. Las agrupaciones funcionan como esperaba: "(abc)" sólo coincidirá con *abc*, "(trucha)" sólo coincidirá con *trucha*. Estos ejemplos son discutibles, puesto que "abc" también sólo coincide con *abc*. Es cuando utilice metacaracteres con paréntesis cuando verá realmente en qué afectan las agrupaciones a los patrones.

En esencia, piense en los paréntesis como si fuera a establecer un literal nuevo. Lógicamente, mientras "a" es un literal que sólo coincide con *a*, "(abc)" es un literal que sólo coincide con *abc*. A partir de esta noción, los cuantificadores, en lugar de aplicarse únicamente al literal inmediatamente anterior, se aplicarán al grupo completo. Así pues, "a{3}" coincide con *aaa*, pero "(abc){3}" coincide con *abcabcabc*. Un ejemplo mejor es "bon+", que sólo coincidirá con una cadena que empiece por *bon* seguida por una o más letras *n* (por ejemplo, *bonnet*); pero "(bon)+" coincidirá con una cadena que empiece por *bon*, seguida por ninguna o más cadenas *bon* (*bonbon*). Los paréntesis restringen y controlan el patrón. De forma similar, mientras "yes|no" coincide con *yeso* o con *yeno* (*ye* más la letra *s* o *n* más la letra *o*), "(Yes)|(No)" acepta una de las dos palabras en su totalidad, que realmente es lo más correcto.

Tabla 8.2. Aquí tiene las clases predefinidas más comunes integradas en PHP, que le ahorrarán un tiempo precioso al no tener que crear sus propias clases.

Clases predefinidas para las expresiones regulares

Clase	Coincide con...
[[:alpha:]]	cualquier letra.
[[:digit:]]	cualquier dígito.
[[:alnum:]]	cualquier letra o dígito.
[[:space:]]	cualquier espacio en blanco.
[[:upper:]]	cualquier letra mayúscula.
[[:lower:]]	cualquier letra minúscula.
[[:punct:]]	cualquier marca de puntuación.

Clases

Independientemente de cómo combine letras en varios grupos, esto sólo le será útil para buscar correspondencias con palabras completas. Pero, ¿qué pasa si busca una coincidencia con cualquier palabra de cuatro letras minúsculas o cualquier secuencia de números? Para ello, defina y utilice clases (conocidas formalmente como clases de caracteres).

Las clases se crean colocando caracteres dentro de corchetes ([]). Por ejemplo, puede buscar la coincidencia de cualquier vocal con "[aeiou]". O puede utilizar el guión para indicar un rango de caracteres: "[a-z]" representa cualquier letra minúscula y "[A-Z]" cualquier letra mayúscula; "[A-Za-z]" es cualquier letra en general; y "[0-9]" coincide con cualquier dígito. Habrá observado que todos estos patrones sólo coinciden con un único carácter, pero "[A-Z]{3}" debe coincidir con *abc*, *def*, etc.

Dentro de los corchetes, el símbolo del acento circunflejo, que normalmente se utiliza para indicar el comienzo de una cadena, se utiliza para excluir un carácter. Así, "[^a]" coincidirá con cualquier carácter que no sea una *a*.

PHP ya incluye algunas clases que le serán muy útiles en su trabajo de programación. Se trata de las clases "[[:alpha:]]", que representa cualquier letra (equivalente a "[A-Za-z]"), "[[:digit:]]", que representa cualquier número (o "[0-9]"), y "[[:alnum:]]", que representa cualquier letra o número (otro modo de escribir "[A-Za-z0-9]").

Definiendo sus propias clases y utilizando las clases integradas en PHP (véase la Tabla 8.2), puede construir los mejores patrones para las expresiones regulares.

Ejemplos de patrones

Utilizando la información introducida anteriormente, puede crear algunos patrones muy útiles.

```
"^([0-9]{5})(-[0-9]{4})?$"
```

El patrón anterior es un patrón que sirve para verificar un código postal, que empieza con cinco números, posiblemente seguidos por un guión y cuatro dígitos más. En el primer grupo de paréntesis estamos indicando precisamente que necesitamos cinco dígitos (la clase seguida por las llaves); el acento circunflejo lo utilizamos para indicar que esto debe ser el principio de la cadena. Después, el segundo grupo de paréntesis empieza con un guión, seguido por cuatro dígitos. El carácter de interrogación hace referencia a que esta sección es opcional (es decir, puede haber ninguno o uno de éstos), pero utilizamos el carácter de dólar para especificar que, de existir, debe ser al final de la cadena.

Aquí tiene un patrón más complejo para verificar una dirección de correo electrónico: "^([0-9a-z]+)([0-9a-z\.-_]+)@([0-9a-z\.-_]+)\.([0-9a-z]+)".

Si utiliza eregi() para la verificación, no debe preocuparse por la caja de los caracteres, sobre si la clase sólo incluye "a-z" y no "A-Z".

El primer paso del patrón indica que la dirección de correo electrónico debe empezar con al menos una letra o un número, seguido por cualquier cantidad de letras, números, caracteres de subrayado, puntos y guiones.

En segundo lugar encontramos el carácter @, que es necesario en cualquier dirección de correo electrónico.

En tercer lugar, el patrón insiste en al menos una letra, número, guión, punto o carácter de subrayado.

En cuarto lugar, una dirección de correo electrónico debe incluir un punto.

Script 8.3. El patrón complejo de validación de una dirección de correo electrónico se ha insertado en el script HandleForm.php para proporcionar un nivel más específico de validación.

```
script
1  <HTML>
2  <HEAD>
3  <TITLE>Using Regular Expressions</TITLE>
4  <BODY>
5  <?php
6  /* Esta página recibe y manipula los datos
   generados por "form.html". */
7  if (($Array["FirstName"]) AND
   ($Array["LastName"])) {
8      $Array["Name"] = $Array["FirstName"] .
       " " . $Array["LastName"];
9  } else {
10     print ("Please enter your first and
       last names.<BR>\n");
11 }
12
13 $Pattern = "^([0-9a-z]+)([ 0-9a-z\.-
   _]+)@([0-9a-z\.-_]+)\.([0-9a-z]+) ";
14 if (eregi($Pattern, $Array["Email"])) {
15     print ("Your information has been
       received!<BR>\n");
16 } else {
17     print ("Please enter a valid email
       address!\n");
18 }
19 ?>
20 </BODY>
21 </HTML>
```

Por último, debe aparecer al menos una letra o un número que concluya la cadena (ésta no puede acabar con un punto).

El espacio, un carácter literal, marcará el final de la cadena (para que, si la dirección de correo electrónico se encuentra dentro de otro texto, las palabras que siguen a la dirección no se incluyan en el patrón). El *Script* 8.3 es una modificación de HandleForm.php utilizando este patrón más específico.

Referencia regresiva

Para terminar, existe un concepto más que debemos explicar en relación a la definición de patrones: la referencia regresiva. Aprenderá más sobre este proceso posteriormente en este capítulo, cuando expliquemos la búsqueda y sustitución de texto.

En el patrón de verificación del código postal, "^([0-9]{5})(-[0-9]{4})?$", observe que existen dos grupos encerrados entre paréntesis: "^([0-9]{5})" y "(-[0-9]{4})". En el patrón de una expresión regular, PHP enumera automáticamente los grupos entre paréntesis empezando por 1. La referencia regresiva le permite referirse a cada sección individual utilizando una doble barra invertida (\\) por delante del número correspondiente. Por ejemplo, si quiere comparar el código postal 94710-0001 con este patrón, la referencia regresiva a \\1 le proporcionará el valor 94710. Verá un ejemplo de esto en acción en la siguiente sección.

Patrones de búsqueda y sustitución

Aunque las funciones ereg() y eregi() funcionan muy bien para la validación de una cadena, puede llevar su programación un paso más allá buscando una coincidencia con un patrón y luego sustituyéndola por un patrón ligeramente distinto o por un texto específico. La sintaxis para el uso de las dos funciones es la siguiente:

```
eregi_replace("patrón", "sustitución",
→ "cadena");
```

O:

```
$Pattern = "patrón";
$Replace = "sustitución";
$String = "cadena";
eregi_replace($Pattern, $Replace,
→ '$String);
```

Una de las razones por las que querría utilizar esto sería para convertir la dirección de un sitio web introducida por el usuario (un URL) en un enlace HTML en el que puede hacerse clic, encapsulándolo en las etiquetas <A HREF="URL"></A>. Modificaremos los *scripts* form.html (véase el *Script* 8.1) y HandleForm.php (véase el *Script* 8.3) para hacer exactamente esto.

Cómo utilizar eregi_replace para buscar y reemplazar un patrón

1. Abra form.html en un editor de textos. Modificará ligeramente la página form.html para aceptar un URL y una descripción.

2. Cambie la línea 9 (véase el *Script* 8.4), que toma una dirección de correo electrónico, por esta otra:

```
URL <INPUT TYPE=TEXT NAME=
→ "Array[URL]" SIZE=60><BR>
```

3. Cambie la línea 10, los comentarios, por esta otra:

```
Description <TEXTAREA NAME=
→ "Array[Description]" ROWS=5
→ COLS=40></TEXTAREA><BR>
```

Script 8.4. Hemos modificado ligeramente la página form.html para que acepte un URL de envío y descripciones, en lugar de una dirección de correo electrónico y unos comentarios.

script

```
1  <HTML>
2  <HEAD>
3  <TITLE>HTML Form</TITLE>
4  </HEAD>
5  <BODY>
6  <FORM ACTION="HandleForm.php" METHOD=POST>
7  First Name <INPUT TYPE=TEXT
   NAME="Array[FirstName]" SIZE=20><BR>
8  Last Name <INPUT TYPE=TEXT
   NAME="Array[LastName]" SIZE=40><BR>
9  URL <INPUT TYPE=TEXT NAME="Array[URL]"
   SIZE=60><BR>
10 Description <TEXTAREA
   NAME="Array[Description]" ROWS=5
   COLS=40></TEXTAREA><BR>
11 <INPUT TYPE=SUBMIT NAME="SUBMIT"
   VALUE="Submit!">
12 </FORM>
13 </BODY>
```

Script 8.5. La función eregi_replace() convertirá automáticamente el URL enviado por el usuario en un enlace en el que puede hacerse clic. Esto es posible principalmente gracias a la referencia regresiva.

script

```
1  <HTML>
2  <HEAD>
3  <TITLE>Using Regular Expressions</TITLE>
4  <BODY>
5  <?php
6  /* Esta página recibe y manipula los datos
   generados por "form.html". */
7  if (($Array["FirstName"]) AND
   ($Array["LastName"])) {
8      $Array["Name"] = $Array["FirstName"] .
       " " . $Array["LastName"];
9  } else {
10     print ("Please enter your first and
       last names.<BR>\n");
11 }
12 $Pattern = "(http://)?([^[:space:]]+)
   ([[:alnum:]\.,-_?/&=])";
13 $Replace = "<a href=\"http://\\2\\3\"
   target=\"_new\">\\2\\3</a>";
14 $Array["URL"] = eregi_replace($Pattern,
   $Replace, $Array["URL"]);
15 print ("Your submission--$Array[URL]--
   has been received!<BR>\n");
16 ?>
17 </BODY>
18 </HTML>
```

4. Guarde el *script* y cárguelo en el servidor.

   Después, actualice la página HandleForm.php.

5. Abra HandleForm.php en el editor de textos (véase el *Script* 8.3).

6. Sustituya la línea 12 por el siguiente código (véase el *Script* 8.5):

7. `$Pattern = "(http://)?([^[:space:]]+)
   ([[:alnum:]\.,-_?/&=])";`

   Aquí tiene un patrón libre configurado para reconocer un URL. De hecho, este patrón es mucho mejor para el propósito de búsqueda y sustitución que la validación de los datos introducidos, a la vez que no es demasiado restrictivo.

   El patrón contiene tres grupos: la cadena http://, el grueso del URL y la parte posterior del mismo. Un URL puede empezar con la cadena http:// o no. Para comprobar esto, empezamos con http seguida de los dos puntos y las dos barras, y utilizamos el signo de interrogación para indicar que la sección completa es opcional.

   Después de esto, un URL contiene caracteres que no pueden ser espacios (la segunda, y mucho más liberal, agupación): letras, números, guiones, caracteres de subrayado, puntos, etc. Por último, permite muchos más caracteres alfanuméricos, así como la barra, el signo de interrogación, el *ampersand* y el signo de igualdad. Este último paréntesis coincide con la parte final de un URL, como, por ejemplo, .com/php/.

8. `$Replace = "<a href=\"http://`
   `→ \\2\\3\" target=\"_new\">`
   `→ \\2\\3</a>";`

   Aquí es donde se define el texto de sustitución. Utilizando la técnica de la referencia regresiva, puede tomar una subsección de una cadena y colocarla dentro de otra. Puesto que la sección http// es opcional, la mejor forma de asegurar la coherencia es ignorándola en la sustitución (observe que no hacemos referencia a \\1) e insertándola manualmente, de modo que todos los URL se visualicen con el mismo formato. Después de realizar este ajuste y de añadir la etiqueta <a href=, insertamos el resto del URL, seguido por la etiqueta HTML de cierre (</a>).

9. `$Array["URL"] = eregi_replace`
   `→ ($Pattern, $Replace,`
   `→ '$Array["URL"]);`

   Restablecemos el valor del URL a la cadena de sustitución nueva. Después, cada vez que el URL se envíe al navegador (como en la siguiente línea), aparecerá como un enlace en el que puede hacerse clic. Esta forma alterada también se almacenará en una base de datos o archivo para un uso futuro.

10. `print ("Your submission—$Array[URL]—`
    `→ has been received!<BR>\n");`

    Por último, finalizamos la instrucción if.

Guarde el *script*, cárguelo en el servidor y pruébelo en el navegador (véanse las Figuras 8.5, 8.6 y 8.7). Si el *script* funciona como esperamos, tomará el URL enviado por un usuario, lo validará y, después, lo convertirá en un enlace en el que puede hacerse clic. Se trata de un proceso que probablemente utilizará en muchas aplicaciones web.

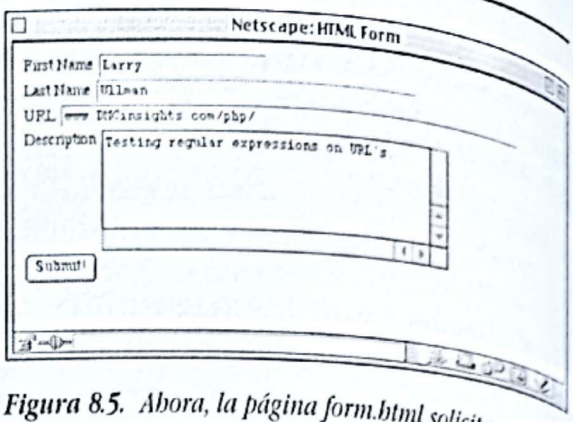


Figura 8.5. Ahora, la página form.html solicita un URL y una descripción; el primero se validará utilizando expresiones regulares (véase la Figura 8.6).

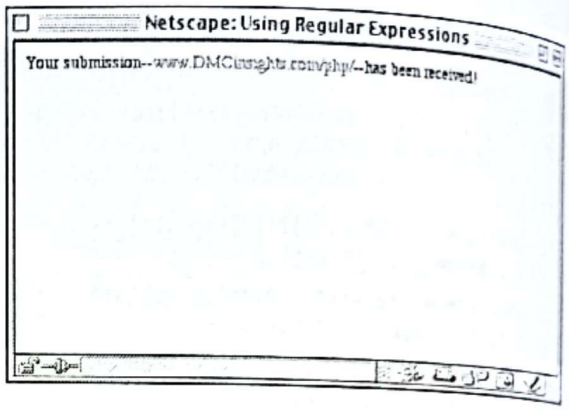


Figure 8.6. Con la función eregi_replace(), puede convertir el URL enviado, con o sin el prefijo http://, en un enlace en el que puede hacerse clic.

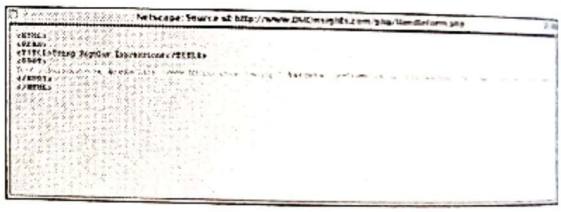


Figura 8.7. El código fuente de la página revela las etiquetas <A HREF> que se han añadido al URL introducido por el usuario.

CREACIÓN DE FUNCIONES 9

En el mismo nivel de utilidad que las expresiones regulares, pero mucho más fáciles de entender e implementar, se encuentran las funciones. Las funciones a las que nos vamos a referir en este capítulo no son exactamente las mismas que se ha encontrado hasta ahora, como phpinfo(), count() y eregi_replace(). Mientras que dichas funciones ya han sido definidas por PHP, ahora usted tendrá que crear las suyas propias. Sin embargo, una vez las haya creado, las podrá manipular de una manera muy similar a las funciones integradas en PHP.

Crear funciones puede ahorrarle mucho tiempo durante su programación. De hecho, constituyen un paso muy grande hacia la creación de aplicaciones web, y construyen una librería sólida de código PHP que puede emplear en proyectos futuros.

En este capítulo verá cómo escribir sus propias funciones para que realicen tareas específicas. Y, después, aprenderá a pasar información a la función, a que su función devuelva un valor, y entenderá cómo las funciones y las variables trabajan al unísono.

Creación y uso de funciones simples

Mientras programe, descubrirá que existen ciertas secciones de código que usa más habitualmente, ya sea en un solo *script* o en varios. Colocar esas rutinas en una función autodefinida le ahorrará tiempo y facilitará la programación, especialmente cuando los sitios web se hacen más grandes. Cuando haya creado una función, las acciones de dicha función se ejecutarán cada vez que la llame, al igual que print() envía texto al navegador cada vez que la utiliza.

La sintaxis para la creación de una función definida por el usuario es la siguiente:

```
function NombreFuncion () {
    instrucciones;
}
```

Puede emplear las mismas convenciones de nombres para el nombre de la función que las que utilizó para las variables, pero sin el signo de dólar inicial. La norma más importante es que sean coherentes. Le sugerimos también que cree nombres de funciones que tengan un significado, del mismo que debe escribir nombres de variables representativos (CrearCabecera será un nombre mejor para una función que Funcion1). Recuerde que no se usan espacios, ya que parecería que hay dos palabras separadas en el nombre de la función, y provocaría la emisión de mensajes de error (el subrayado es un sustituto lógico para el espacio; por ejemplo, Crear_Cabecera es un nombre de función válido).

Cualquier código correcto de PHP puede ir dentro del área de instrucciones de la función, incluyendo las llamadas a otras funciones. No existe ningún límite en el número de instrucciones que tiene una función, pero asegúrese de que cada una de ellas finaliza con un punto y coma.

El formato de una función no es importante mientras tenga los elementos que necesite. Dichos elementos incluyen la palabra function, el nombre de la función, los paréntesis de apertura y cierre, las llaves de apertura y cierre, y las instrucciones. Es conveniente sangrar las instrucciones de la función, como hemos hecho en el ejemplo de la sintaxis, por motivos de legibilidad. En cualquier caso, seleccione un estilo de formato

Script 9.1. Éste es el script de contraseñas original que no utiliza funciones. Va a crear su primera función empleando el código que se encuentra en la mitad de esta página.

```
script

1  <HTML>
2  <HEAD>
3  <TITLE>Password Generator</TITLE>
4  <BODY>
5  <?php
6  $String = "This is the text which will be
   encrypted so that we may create random
   and secure passwords!";
7  $Length = 8; // Cambie este valor para
   indicar la longitud de la contraseña.
   Límite 32 caracteres.
8  $String = md5($String);
9  $StringLength = strlen($String);
10 srand ((double) microtime() * 1000000);
11 $Begin = rand(0,($StringLength-$Length-
   1)); // Seleccionar punto inicial arbitrario.
12 $Password = substr($String, $Begin,
   $Length);
13 print ("Your recommended password is:
   <P><BIG>$Password</BIG>\n");
14 ?>
15 </BODY>
16 </HTML>
```

que le guste (con una sintaxis correcta y un sentido lógico) y cree sus propias funciones.

Llame a la función haciendo referencia a ella al igual que haría con cualquier función integrada. La línea de código NombreFuncion(); hará que las instrucciones que forman parte de la función definida anteriormente, se ejecuten.

Vamos a comenzar por rescribir el *script* generador de contraseñas del Capítulo 5, como función definida por el usuario.

Cómo crear y llamar a una función básica

1. Abra passwords.php en el editor de textos (véase el *Script* 9.1).

2. Cree la primera línea de la página con <?php, al contrario que el comienzo estándar de la cabecera HTML (véase el *Script* 9.2).

   Para asegurarnos de que la función se ha creado antes de que se llame (se exige en PHP 3), escribiremos la función en la primera parte del *script*, incluso antes de cualquier código HTML.

3. `function CreatePassword () {`

   El nombre de la función será CreatePassword, que es un nombre representativo de lo que hace la función y, por tanto, será fácil de recordar.

4. Ahora, coloque el código PHP (líneas 6 a 13) de passwords.php en la función.

   Como se indica en el *Script* 9.2, recomendamos sangrar las líneas respecto al nombre de la función, para indicar que pertenecen a dicha función.

```
$String = "This is the text which will
→ be encrypted so that we may create
→ random and secure passwords!";
$Length = 8; $String = md5($String);
$StringLength = strlen($String);
srand ((double) microtime() *
→ 1000000);
$Begin = rand(0,($StringLength-
→ $Length-1)); $Password = substr
→ ($String, $Begin, $Length);
print ("Your recommended password is:
→ <P><BIG>$Password</BIG><P>\n");
```

5. Cierre la función de la línea siguiente con una llave: }.

   La omisión de las llaves de apertura o cierre es una causa de errores muy común, por lo que debe asegurarse de que escribe la sintaxis correctamente.

6. Cierre el código PHP con ?>.

   Como va a escribir parte de la página en HTML, necesita cerrar la sección PHP. Puede mantener la sección PHP abierta y utilizar print() para enviar el código HTML al navegador, en cuyo caso debería saltar este paso.

7. Escriba el código de cabecera HTML estándar,

```
<HTML><HEAD><TITLE>Password Generator,
→ within a Function</TITLE><BODY>
```

8. Comience una nueva sección PHP de la página con <?php.

   Actualmente puede cerrar y abrir varias secciones de código PHP en un documento HTML. Es muy normal que los programadores lo hagan.

Script 9.2. Colocar la función al principio del script es una buena manera de separarla. Asimismo, sangrar las instrucciones de la función ayuda a indicar su relación con la misma. Después, en el cuerpo principal del script, una línea de código ejecuta las instrucciones de la función.

script

```
1  <?php
2  function CreatePassword () {
3      $String = "This is the text which
       will be encrypted so that we may
       create random and secure passwords!";
4      $Length = 8; // Cambie este valor para
       indicar la longitud de la contraseña.
       Límite 32 caracteres.
5      $String = md5($String);
6      $StringLength = strlen($String);
7      srand ((double) microtime() * 1000000);
8      $Begin = rand(0,($StringLength-
       $Length-1)); // Seleccionar punto
       inicial arbitrario.
9      $Password = substr($String, $Begin,
       $Length);
10     print ("Your recommended password is:
       <P><BIG>$Password</BIG><P> \n");
11 } // Fin de la función CreatePassword.
12 ?>
13 <HTML>
14 <HEAD>
15 <TITLE>Password Generator within a
   Function</TITLE>
16 <BODY>
17 <?php
18 CreatePassword(); // Llamar a la función.
19 ?>
20 </BODY>
21 </HTML>
```

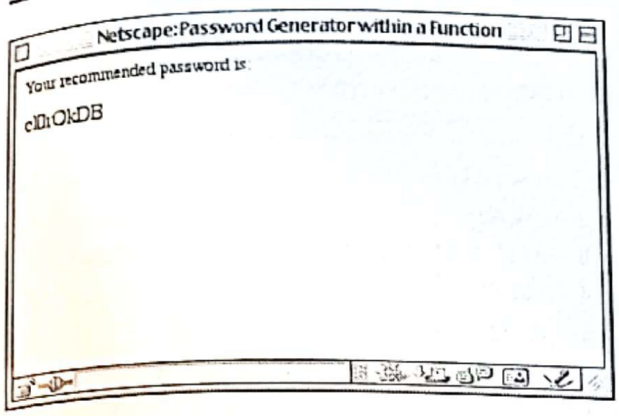


Figura 9.1. Colocando varias líneas PHP de password.php en una función, empezará a modularizar el código. Es el primer paso hacia la creación de sitios web dinámicos, aun cuando el resultado visual parece idéntico al anterior a crear la función.

9. `CreatePassword();`

   Una vez que ha creado su función, simplemente tiene que llamarla por su nombre (use el nombre correcto) para ejecutarla. Asegúrese de incluir los paréntesis.

10. `?></BODY></HTML>`

    Cierre la segunda parte de PHP y finalice el código HTML.

11. Guarde el *script*, cárguelo en el servidor y pruébelo en el navegador (véase la Figura 9.1).

CONSEJOS

- Si el servidor con que trabaja funciona con PHP 3, debe definir la función antes de llamarla. Aunque esto no sucede en PHP 4, le recomendamos que defina sus funciones al comienzo del *script*, asegurándose así de que siempre estarán creadas antes de llamarlas.
- Los nombres de las funciones para las funciones que ha creado no hacen distinción entre mayúsculas y minúsculas, pero las funciones que ya existen en PHP sí la hacen. De todos modos, no tendrá ningún problema en el código si emplea, por ejemplo, crearclave en lugar de un más adecuado CrearClave.

Creación y llamada a funciones que admiten argumentos

Aunque la posibilidad de crear una función sencilla es muy útil, escribir una que tome una entrada y que después haga algo con esa entrada, es aún mejor. La entrada que toma una función se llama argumento, y no existen límites para los argumentos que pueda admitir una función. Las funciones que toman argumentos es un concepto que ya ha visto antes: la función print() toma una cadena como argumento que después envía al navegador.

La sintaxis para escribir funciones que toman argumentos es la siguiente:

```
function NombreFunción ($Argumento1,
→ $Argumento2, etc.) {
    instrucciones;
}
```

Dichos argumentos estarán en forma de variables a las que se asigna el valor de la función cuando se la llama. Se llama mucho más a las funciones que admiten entradas que a las que no, por lo que sólo tiene que recordar pasar los valores necesarios. Puede hacer esto cuando vaya a pasar una variable:

```
NombreFunción ($Variable1, $Variable2,
→ etc.);
```

O colocar los valores entre comillas, como en la siguiente instrucción:

```
NombreFunción ("Valor1", "Valor2", etc.);
```

O una combinación de ambas:

```
NombreFunción ($Variable1, "Valor2",
→ etc.);
```

Observe que los argumentos se basan casi literalmente en que el primer valor de la función será igual al primer valor de la línea de llamada, el segundo valor de la función coincide con el segundo valor de llamada, etc. Las funciones no son lo suficientemente inteligentes para comprender intuitivamente cómo quiere el usuario que se asocien los valores. Esto también sucede si comete un error al pasar un valor, en cuyo caso la función asumirá que el valor es nulo (nulo no es matemáticamente un 0, que es un valor, pero está muy cer-

Script 9.3. La página hello.php crea un saludo personalizado en varias líneas de código dentro del resto de la página.

script

```
1  <HTML>
2  <HEAD>
3  <TITLE>If-elseif Conditionals</TITLE>
4  <BODY>
5  <?php
6  if ($Username) {
7      print ("Good ");
8      if (date("A") == "AM") {
9          print ("morning, ");
10     } elseif ( ( date("H") >= 12 ) and
       ( date("H") < 18 ) ) {
11         print ("afternoon, ");
12     } else {
13         print ("evening, ");
14     } // Cierra if date.
15     print ("$Username");
16     print ("!\n");
17 } else {
18     print ("Please log in.\n");
19 } // Cierra if username.
20 ?>
21 </BODY>
22 </HTML>
```

Script 9.4. Aquí tiene una función que toma un argumento que debe pasarle en la llamada. Para asegurarse de que la función se ejecuta correctamente, sólo la llamará si $Username tiene un valor.

script

```
1  <?php
2  function GreetUser ($TheUser) {
3      print ("Good ");
4      if (date("A") == "AM") {
5          print ("morning, ");
6      } elseif ((date("H")>12) and
       (date("H")<18)) {
7          print ("morning, ");
8      } else {
9          print ("evening, ");
10     } // Cierra if date.
11     print ("$TheUser");
12     print ("!\n");
13 } // Final de la función GreetUser.
14 ?>
15 <HTML>
16 <HEAD>
17 <TITLE>The GreetUser Function</TITLE>
18 <BODY>
19 <?php
20 if ($Username) {
21     GreetUser ($Username); // Llama a la
       función.
22 } else {
23     print ("Please log in.\n");
24 } // Cierra username if.
25 ?>
26 </BODY>
27 </HTML>
```

cano a la idea de "nada"). Lo mismo ocurre si una función toma cuatro argumentos y el usuario pasa tres (el cuarto será nulo).

Para demostrar las funciones que toman argumentos, va a rescribir la página hello.php del Capítulo 6. Colocará el código de saludo en una función simple que tome un argumento (el nombre del usuario).

Cómo crear y llamar a una función que tome un argumento

1. Abra hello.php en el editor de textos (véase el *Script* 9.3).
2. Comience por la etiqueta PHP de apertura, <?php. Después, tome las líneas 7 a 16 del *script* original (véase el *Script* 9.3), y colóquelas en la función que está en el principio de la página (véase el *Script* 9.4).

```
function GreetUser ($TheUser) {
print ("Good ");
   if (date("A") == "AM") {
      print ("morning, ");
   } elseif ((date("H")>12) and
   → (date("H")<18)) {
print ("morning, ");
   } else {
      print ("evening, ");
   } // Cierra if date.
   print ("$TheUser");
   print ("!\n");
} // Final de la función GreetUser.
```

Después de dar a la función un nombre adecuado, ésta continuará teniendo las mismas líneas que se han usado antes.

Cuando llame a la función, la variable $TheUser recibirá el valor de $Username (consulte la línea 21 del *Script* 9.4).

3. Cierre la sección inicial de PHP y cree la cabecera HTML.

```
?><HTML><HEAD><TITLE>The GreetUser
 ‣ Function</TITLE><BODY>
```

4. Vuelva a abrir la sección PHP y coloque la estructura condicional if-else, incluyendo una llamada a la nueva función.

```
<?php
if ($Username) {
   GreetUser ($Username); // Llama a
    ‣ la función.
} else {
   print ("Please log in.\n");
} // Cierra username if.
```

Debe mantener la condición if ($Username), para que sólo se llame a la función si existe $Username.

5. Cierre el PHP y el HTML.

```
?></BODY></HTML>
```

6. Guarde el *script*, cárguelo en el servidor y pruébelo en el navegador (véase la Figura 9.2). Recuerde enviar al *script* un valor Username (añadiéndolo al URL, o a través de un formulario HTML), u obtendrá un resultado como el de la Figura 9.3.

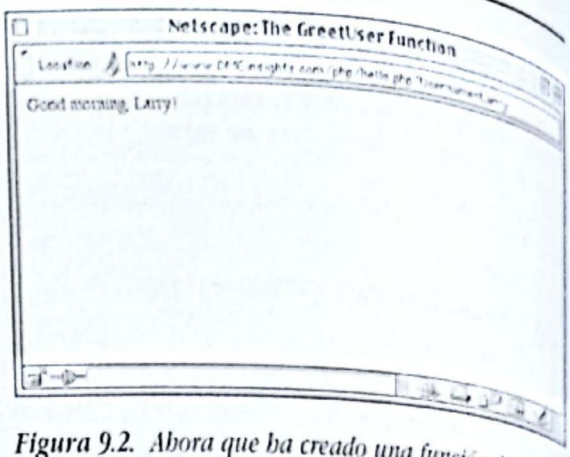


Figura 9.2. Ahora que ha creado una función de bienvenida, aunque puede parecer similar a la página hello.php original, este saludo puede generarse múltiples veces sin rescribir el código.

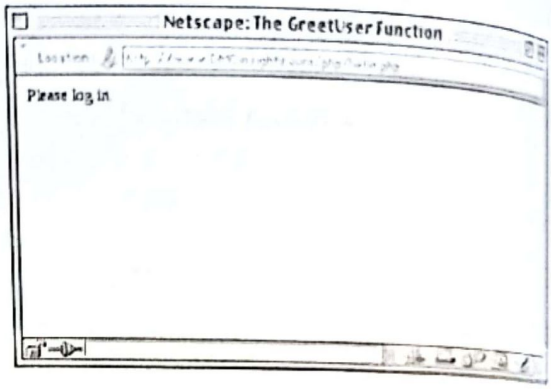


Figura 9.3. A medida que aprenda cosas más avanzadas acerca de la programación en PHP, no olvide los hechos simples. La omisión de enviar a hello.php un valor Username (en comparación con el URL de la Figura 9.2) genera este mensaje, como prudentemente lo programó.

CONSEJOS

- En el Capítulo 13 verá cómo se coloca cierta información en archivos externos que se pueden usar en varias páginas. Un archivo externo es el mejor lugar para que el usuario ponga sus propias funciones, como GreetUser, por lo que podrán ser accesibles en todos los sitios web.
- Técnicamente, si pasa un número como argumento a una función, no necesita estar entre cadenas, pero, en cualquier caso, no pasa nada si las emplea para que sea coherente con el número de argumentos pasados.

Script 9.5. Ésta es la versión más reciente de numbers.php, modificado por última vez en el Capítulo 6. Va a modularizar el proceso en este script creando dos funciones.

```
script
1  <HTML>
2  <HEAD>
3  <TITLE>Conditionals</TITLE>
4  </HEAD>
5  <BODY>
6  <?php
7  /* $Quantity debe pasarse a esta página
   desde un formulario o a través de un URL.
   $Discount es opcional. */
8  $Cost = 20.00;
9  $Tax = 0.06;
10 if ($Quantity) {
11     $Quantity = abs($Quantity);
12     $Discount = abs($Discount);
13     $Tax++; // $Tax tiene ahora un valor
       de 1.06.
14     $TotalCost = ($Cost * $Quantity);
15     if ( ($TotalCost < 50) AND ($Discount) ) {
16         print ("Your \$$Discount will not
           apply because the total value of the
           sale is under $50!\n<P>");
17     }
18     if ($TotalCost >= 50) {
19         $TotalCost = $TotalCost - $Discount;
20     }
21     $TotalCost = $TotalCost * $Tax;
22     $Payments = round ($TotalCost, 2) / 12;
23     // Imprimir los resultados.
24     print ("You requested to purchase
       $Quantity widget(s) at \$$Cost
       each.\n<P>");
25     print ("The total with tax, minus your
       \$$Discount, comes to $");
26     printf ("%01.2f", $TotalCost);
27     print (".\n<P>You may purchase the
       widget(s)in 12 monthly installments of
       $");
28     printf ("%01.2f", $Payments);
29     print (" each.\n<P>");
30 } else {
31     print ("Please make sure that you have
       entered both a quantity and
       an applicable discount and then
       resubmit.\n"); }
32 ?>
33 </BODY>
34 </HTML>
```

Creación y uso de funciones que devuelven un valor

Una vez que ya ha comenzado a escribir funciones que toman argumentos, el siguiente paso va a ser la creación de una función de retorno y una salida para un valor. Para hacerlo se necesitan dos pasos más. El primero es usar la instrucción return en la función, y el segundo, asignar la salida de algún modo cuando llame a la función. Normalmente, asignaría el valor de retorno a la variable, pero también podría, por ejemplo, imprimir la salida directamente. Éste es el formato básico para una función que toma un argumento y devuelve un valor:

```
function NombreFunción ($Argumento) {
    instrucciones;
    return $Valor;
}
```

Normalmente, esta función se usaría con una línea de código como la siguiente:

```
$Valor = NombreFunción($Variable);
```

Observe que hemos asignado el valor de retorno de la función a una variable. Para demostrar mejor este concepto y sus varias utilidades, va a crear dos funciones que se basan en la página numbers.php, que comenzamos en el Capítulo 4 y modificamos después en el Capítulo 6.

Cómo crear y utilizar una función que devuelve un valor

1. Abra numbers.php en el editor de textos (véase el *Script* 9.5).

2. Abra la sección inicial de PHP y dé comienzo a la primera función, CalculateTotal(), que contiene las líneas 14 a 21 de la página original (véase el *Script* 9.6):

```
<?php
function CalculateTotal ($HowMany,
→ $Price, $TaxRate, $Savings) {
    $TaxRate++; // $TaxRate tiene
    → ahora un valor de 1.06.
    $TheCost = ($Price * $HowMany);
    if ( ($TheCost < 50) AND
    → ($Savings) ) {
        print ("Your \$$Savings will
        → not apply because the
        → total value of the sale
        → is under $50!\n<P>");
    }
    if ($TheCost >= 50) {
        $TheCost = $TheCost -
        → $Savings;
    }
    $TheCost = $TheCost * $TaxRate;
    return $TheCost;
}
```

Esta función toma cuatro argumentos: el número de dispositivos adquiridos, el precio de cada uno, el impuesto aplicable y el descuento que puede tener el cliente. Después se efectúan los cálculos necesarios para determinar el coste total que se devuelve.

3. Cree la segunda función:

```
function CalculatePayments ($Amount,
→ $NumberPayments) {
    $Payments = round($Amount, 2) /
    → $NumberPayments;
    $Payments = sprintf ("%01.2f",
    → $Payments);
    return $Payments;
}
```

La función CalculatePayments() tendrá también dos argumentos (la cantidad total y el número de pagos) y efectuará un cálculo sencillo. Primero, la función determina la cantidad del pago empleando la misma fórmula que utilizó originalmente numbers.php. Después, el valor $Payments se formatea usando la función

Script 9.6. Las dos funciones de este script devuelven un valor. Cuando se llama a la función CalculateTotal, el valor de retorno se asigna a la variable. Cuando se llama a la función CalculatePayments, se imprime el valor de retorno en el navegador.

script

```
1  <?php
2  function CalculateTotal ($HowMany,
   $Price, $TaxRate, $Savings) {
3      $TaxRate++; // $TaxRate tiene ahora el
       valor 1.06.
4      $TheCost = ($Price * $HowMany);
5      if ( ($TheCost < 50) AND ($Savings) ) {
6          print ("Your \$$Savings will not
           apply because the total value of the
           sale is under $50!\n<P>");
7      }
8      if ($TheCost >= 50) {
9          $TheCost = $TheCost - $Savings;
10     }
11     $TheCost = $TheCost * $TaxRate;
12     return $TheCost;
13 } // Final de la función CalculateTotal.
14 function CalculatePayments ($Amount,
   $NumberPayments) {
15     $Payments = round($Amount, 2) /
       $NumberPayments;
16     $Payments = sprintf ("%01.2f",
       $Payments);
17     return $Payments;
18 } // Final de la función CalculatePayments.
19 ?>
20 <HTML>
21 <HEAD>
22 <TITLE>Calculation Functions</TITLE>
23 </HEAD>
24 <BODY>
25 <?php
26 $Cost = 20.00;
27 $Tax = 0.06;
28 if ($Quantity) {
29     $Quantity = abs($Quantity);
30     $Discount = abs($Discount);
31     $TotalCost = CalculateTotal ($Quantity,
       $Cost, $Tax, $Discount);
32     // Imprimir los resultados.
33     print ("You requested to purchase
       $Quantity widget(s) at \$$Cost
       each.\n<P>");
```

Script 9.6. Continuación

```
script
34      print ("The total with tax, minus your
        \$$Discount, comes to $");
35      printf ("%01.2f", $TotalCost);
36      print (".\n<P>You may purchase the
        widget(s) in 12 monthly installments
        of $");
37      print (CalculatePayments($TotalCost,
        "12"));
38      print (" each.\n<P>");
39  } else {
40      print ("Please make sure that you
        have entered both a quantity and an
        applicable discount and then
        resubmit.\n");
41  }
42  ?>
43  </BODY>
44  </HTML>
```

sprintf(), la cual, como hemos dicho en el Capítulo 4, funciona exactamente igual que printf(), salvo que no imprime nada en el navegador, tan sólo altera la variable. Finalmente, se devuelve el valor de $Payments.

Las ventajas de colocar los cálculos en una función son dobles: primero, será más fácil de encontrar y modificar, ya que está situada al principio del *script*, en lugar de estar ubicada en medio del código, y segundo, si quiere repetir la acción en el *script*, puede hacerlo sin tener que repetir el código.

4. Cierre la sección inicial de PHP y cree después la cabecera HTML básica.

```
?><HTML><HEAD><TITLE>Calculation
 › Functions</TITLE></HEAD><BODY>
```

5. Abra una nueva sección PHP y ajuste sus variables:

```
<?php
$Cost = 20.00;
$Tax = 0.06;
```

No hemos cambiado el valor de la variable $Cost, pero puede ajustarlo también como $Tax, con el valor que prefiera.

6. Escriba ahora el motor del *script* PHP.

```
if ($Quantity) {
    $Quantity = abs($Quantity);
    $Discount = abs($Discount);
    $TotalCost = CalculateTotal
     › ($Quantity, $Cost, $Tax,
     › $Discount);
    // Imprimir los resultados.
    print ("You requested to purchase
     › $Quantity widget(s) at \$$Cost
     › each.\n<P>");
    print ("The total with tax,
     › minus 'your \$$Discount, comes
     › to $");
    printf ("%01.2f", $TotalCost);
    print (".\n<P>You may purchase
     › the widget(s)in 12 monthly
     › installments of $");
    print (CalculatePayments
     › ($TotalCost, "12"));
```

```
    print (" each.\n<P>");
} else {
    print ("Please make sure that
    → you have entered both a
    → quantity and an applicable
    → discount and then
    → resubmit.\n");
}
```

Excepto por las llamadas a las funciones, esta sección es exacta a la que escribió en el Capítulo 6, pero ha aumentado la eficiencia de la página al separar los cálculos en funciones propias. Esta sección de código también demuestra los dos modos de usar las funciones que devuelven un valor. En la línea 31 se asigna a una variable el valor de retorno de la función CalculateTotal(), y en la línea 37 se imprime rápidamente el resultado de la llamada a la función CalculatePayment().

7. Cierre la última sección PHP y finalice el código HTML.

   ?></BODY></HTML>

8. Guarde el *script*, cárguelo en el servidor y pruébelo en el navegador (véase la Figura 9.4). No se olvide de enviar al *script* un valor Quantity (y un valor Discount, si quiere), añadiéndolo al URL, o a través de un formulario HTML. Si el *script* no recibe un valor Quantity, verá el mensaje de error "Please make sure …", generado en la línea 40.

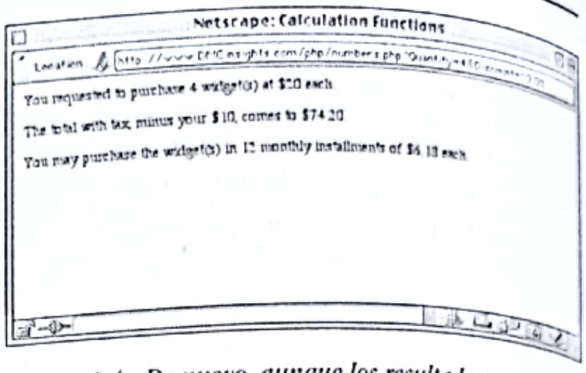


Figura 9.4. De nuevo, aunque los resultados actuales son iguales a los obtenidos cuando no tenía funciones, estas funciones se pueden aplicar universalmente a lo largo de todo un sitio web. La creación de funciones como estas dos acelera los proyectos de programación futuros al crear una robusta biblioteca de código reutilizable.

CONSEJOS

- Podría cambiar la línea sprintf() por la línea printf() en la función CalculatePayments(), imprimir el resultado y eliminar la instrucción return. Aunque en este ejemplo no lo hemos hecho, es perfectamente razonable que su función imprima directamente cualquier resultado.
- Sólo podrá ejecutar una instrucción return por función, pero cada función puede tener varias instrucciones return. Por ejemplo, si quiere escribir una función que compruebe una condición, y que devuelva si la condición se ha cumplido. En este caso, debería escribir en su función:

```
if (condición) {
    return TRUE;
} else {
    return FALSE;
}
```

  El resultado que devuelve la función es TRUE o FALSE, indicando si se ha cumplido la condición.
- Una instrucción return sólo puede devolver un valor. Para devolver varios valores, necesita emplear arrays. Consulte el Apéndice C para obtener más información sobre el retorno de varios valores.

Variables y funciones

No hemos presentado el concepto de variable en el Capítulo 2, porque sin comprender las funciones, su estudio no tendría sentido. Ahora que ya se familiarizado razonablemente con las funciones, revisaremos el tema de las variables y discutiremos con más detalle cómo trabajan juntas variables y funciones.

Ámbito de las variables y la instrucción global

Como ha visto en la segunda sección de este capítulo, "Creación y llamada a funciones que admiten argumentos", puede enviar variables a una función pasándolas como argumentos. No obstante, también puede llamar a una variable de una función utilizando la instrucción global. Esto es debido al ámbito de la variable. Dicho ámbito es esencialmente el dominio en el que existe. Por defecto, las variables que escribe en un *script* existen durante toda la vida del mismo. Por el contrario, las variables de entorno (como $PHP_SELF) existen a través de un servidor.

Las funciones crean un nuevo nivel de ámbito. Las variables de función (los argumentos de una función, así como cualquier variable definida en dicha función), sólo existen en esa función y no son accesibles fuera de ella (por ejemplo, existen variables locales con un ámbito local). Asimismo, a una variable que está fuera de una función sólo se le podrá hacer referencia pasándola a dicha función como un argumento o utilizando la instrucción global. La instrucción global significa: "quiero que esta variable de la función sea la misma que la que está fuera de la función". En otras palabras, la instrucción global convierte una variable local con un ámbito local en una variable global con un ámbito global. Cualquier cambio que se haga en la variable de la función se pasa también a la variable cuando esté fuera de la función (suponiendo que se llame a la función, claro está) sin usar el comando return. La sintaxis de una instrucción global es la siguiente:

```
function NombreFunción ($Argumento) {
    global $Variable;
instrucciones;
}
```

Mientras exista $Variable fuera de la función, tendrá el mismo valor que cuando estuvo dentro. Esto nos conduce a un tema paralelo respecto a funciones y variables: debido al ámbito de la variable, la que está en una función es una entidad diferente (quizá con otro valor) de la que está fuera, aunque las dos tengan el mismo nombre (y asumiendo que el usuario no emplee la instrucción global en la función). Vamos a tratar este tema más profundamente.

Normalmente, se emplea una línea de llamada a la función, como NombreFunción ($Valor1);, en cuyo caso la función (aquí, NombreFunción ($Argumento1)) iguala después el valor de $Argumento1 con el de $Valor1, con lo que sus valores son los mismos, pero sus nombres son diferentes. Sin embargo, si el nombre del argumento de la función fuese también $Valor1 (con lo que en la línea de creación de la funmción se leería NombreFunción ($Valor1)), la variable $Valor1 de la función asume el mismo valor que la original $Valor1 fuera de la función, pero son variables separadas. Una tiene un ámbito dentro de la función y otra fuera. Por este motivo, debemos tener cuidado cuando escribamos las funciones, y usar nombres de variables diferentes en la línea que defina a la función de los que hemos empleado en la línea de llamada a la función para evitar confusiones. Hemos desarrollado este tema porque aún no ha tratado con nombres diferentes. Podría utilizar exactamente el mismo nombre en la función y en la línea de llamada por motivos de conveniencia (es más fácil de recordar que los argumentos se han pasado de esa manera), pero recuerde que no son las mismas variables. Lo que le suceda al valor de una variable en una función, se queda dentro de esa función, a no ser que emplee una instruccción global, lo que haría que las dos variables fuesen la misma.

Para comprender mejor este concepto, va a volver a trabajar en el *script* numbers.php usando la instrucción global.

Cómo usar la instrucción global

1. Abra numbers.php en el editor de textos (véase el *Script* 9.6).
2. Elimine los argumentos $Price y $TaxRate de la función CalculateTotal (véase el *Script* 9.7), para que la línea 2 quede como:

   ```
   function CalculateTotal ($HowMany,
   → $Savings) {
   ```

   Las variables $Price y $TaxRate se traerán a la función como $Cost y $Tax usando la instrucción global para que no sea necesario tener que pasarlas como argumentos nunca más.
3. Añada dos instrucciones globales:

   ```
   global $Cost;
   global $Tax;
   ```

   Estas dos instruciones le dicen a la función que incorpore las mismas variables $Cost y $Tax que existen fuera de la función.

Script 9.7. Como los valores $Cost y $Tax han sido solicitados por la función CalculateTotal, se pueden traer fácilmente utilizando la instrucción global. Cuando efectúe el cambio, recuerde que no tiene que pasarlos como argumentos, ya que podría confundir al programador y crear unos errores muy raros.

script

```
1  <?php
2  function CalculateTotal ($HowMany,
   $Savings) {
3      global $Cost;
4      global $Tax;
5      $Tax++; // $Tax tiene ahora el valor
       1.06.
6      $TotalCost = ($Cost * $HowMany);
7      if ( ($TotalCost < 50) AND
       ($Savings) ) {
8          print ("Your \$$Savings will not
           apply because the total value of the
           sale is under $50!\n<P>");
9      }
10     if ($TotalCost >= 50) {
11         $TotalCost = $TotalCost - $Savings;
12     }
13     $TotalCost = $TotalCost * $Tax;
14     return $TotalCost;
15 } // Final de la función CalculateTotal.
16 function CalculatePayments ($Amount, $Num-
   berPayments) {
17     $Payments = round($Amount, 2) /
       $NumberPayments;
18     $Payments = sprintf ("%01.2f",
       $Payments);
19     return $Payments;
20 } // Final de la función CalculatePayments.
21 ?>
22 <HTML>
23 <HEAD>
24 <TITLE>Calculation Functions</TITLE>
25 </HEAD>
26 <BODY>
27 <?php
28 $Cost = 20.00;
29 $Tax = 0.06;
30 print ("The tax value is currently
   \$$Tax .\n<P>");
31 if ($Quantity) {
32     $Quantity = abs($Quantity);
33     $Discount = abs($Discount);
```

Script 9.7. Continuación

```
script
34      $TotalCost = CalculateTotal
        ($Quantity, $Discount);
35      print ("After calling the function, the
        tax value is now \$$Tax .\n<P>");
36      // Imprimir los resultados.
37      print ("You requested to purchase
        $Quantity widget(s) at \$$Cost
        each.\n<P>");
38      print ("The total with tax, minus your
        \$$Discount, comes to $");
39      printf ("%01.2f", $TotalCost);
40      print (".\n<P>You may purchase the
        widget(s) in 12 monthly installments
        of $");
41      print (CalculatePayments($TotalCost,
        "12"));
42      print (" each.\n<P>");
43  } else {
44      print ("Please make sure that you
        have entered both a quantity and an
        applicable discount and then
        resubmit.\n");
45  }
46  ?>
47  </BODY>
48  </HTML>
```

4. Edite el resto de la función empleando $Cost en lugar de $Price, $Tax en lugar de $TaxRate, y $TotalCost en lugar de $TheCost:

```
$Tax++; // $Tax tiene ahora el
→ valor 1.06.
$TotalCost = ($Cost * $HowMany);
if ( ($TotalCost < 50) AND
→ ($Savings) ) {
    print ("Your \$$Savings will
    → not apply because the
    → total value of the sale
    → is under $50!\n<P>");
}
if ($TotalCost >= 50) {
    $TotalCost = $TotalCost -
    → $Savings;
}
$TotalCost = $TotalCost * $Tax;
return $TotalCost;
```

Como la función usa ahora nombres de variables diferentes, debe cambiar los cálculos. Para obtener un ámbito de la variable comprensible, también hemos cambiado la variable $TheCost para que se llame $TotalCost. Recuerde que la variable $TotalCost de la función es una variable diferente a la que se encuentra fuera, incluso si al final ambas tienen el mismo valor.

5. Siguiendo hacia abajo en el *script*, después de la línea $Tax = 0.06; (línea 29), imprima el valor actual de la variable para monitorizar cómo cambia durante el transcurso de la página.

```
print ("The tax value is currently
→ '\$$Tax .\n<P>");
```

Para demostrar que la variable global lleva a la variable $Tax a la función, y que cada cambio que se haga en dicha función se aplica globalmente, imprima el valor de $Tax antes y después de llamar a la función.

6. Cambie la llamada a la función CalculateTotal(), para que $Tax y $Cost no se pasen más como argumentos.

```
$TotalCost = CalculateTotal '($Quan-
tity, $Discount);
```

Como la función ahora sólo toma dos argumentos, si pasa cuatro argumentos, como solía hacer en el *script*, provocaría un error.

7. Imprima de nuevo el valor de $Tax.

```
print ("After calling the function,
→ the tax value is now \$$Tax .\
→ n<P>");
```

Sin la instrucción global, el valor que se ha impreso aquí y el valor que se ha impreso más arriba serían el mismo. Como la variable global $Tax se modificó en la función, verá impresos valores diferentes.

8. Guarde el *script*, cárguelo en el servidor y pruébelo en el navegador (véase la Figura 9.5). No se olvide de enviar al *script* un valor Quantity (y un valor Discount, si lo desea), añadiéndolos a un URL o a través de un formulario HTML.

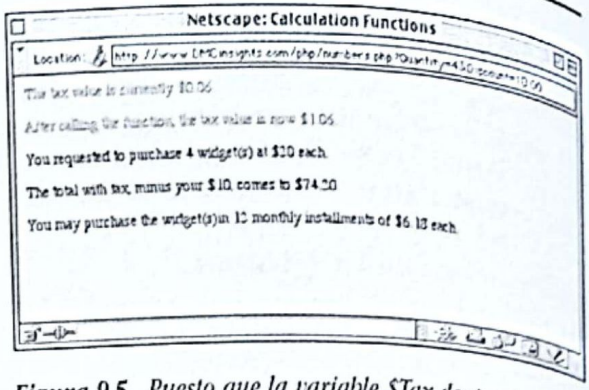


Figura 9.5. Puesto que la variable $Tax dentro de la función es la misma variable que la variable $Tax externa a la función (a causa de la instrucción global), cada cambio que realice en su valor dentro de la función se aplicará universalmente.

Configuración de valores de argumento predeterminados

Mientras escribe las funciones, PHP le permite predefinir un valor para sus argumentos. La función utilizará este valor predeterminado a no ser que reciba un argumento que pueda sobrescribir dicho valor. En otras palabras, al configurar un valor para un argumento, representa ese argumento opcional cuando se llama a la función.

Al usar la función CalculatePayments() como en el ejemplo (véase el *Script* 9.7), puede predefinir 12 como el número de pagos mensuales si crea una función como ésta:

```
function CalculatePayments ($Amount,
→ $NumberPayments = "12") {
```

Si llama a la función con el código CalculatePayments ($Amounts); podría trabajar sin problemas, pero si codifica CalculatePayments ($Amounts, "24");, establecería la variable $NumberPayments a 24, no al valor predeterminado 12.

Puede establecer el valor predeterminado de un argumento si quiere asumir algún valor, pero permitiendo alguna otra posibilidad. Sin embargo, recuerde que los argumentos predeterminados se deben escribir siempre después de los argumentos estándar (los no predeterminados). Esto se debe a que PHP asigna directamente valores a los argumentos en el orden en el que los recibe de la línea de llamada. Así, no es posible omitir un valor para el primer argumento, y sí incluir uno para el segundo (lo que significa que se envía un valor que se adjudicará automaticamente al primer argumento, no al segundo). Por ejemplo, si una función se escribe del siguiente modo:

```
function CalculateTotal ($HowMany,
→ $Price = "20.00", $TaxRate = "0.06") {
```

y llama a la función con esta línea:

```
CalculateTotal (3, "0.07");
```

con la intención de ajustar $HowMany a 3, dejar $Price en 20.00, y cambiar $TaxRate a 0.07, tendrá algunos problemas. El resultado final sería que $HowMany se ajusta a 3, $Price se ajusta a 0.07, y $TaxRate permanece en 0.06, lo cual no es el resultado esperado. La manera adecuada de lograrlo sería escribiendo el código:

```
CalculateTotal (3, "20.00", "0.07");
```

Vamos a volver a trabajar con la página numbers.php (véase el *Script* 9.7) para incorporar el concepto de configuración de valores de argumento predeterminados.

ómo escribir una función que utiliza alores predeterminados

1. Abra numbers.php (véase el *Script* 9.7) en el editor de textos.

2. Añada un valor predeterminado a la variable $Savings en la función CalculateTotal() (véase el *Script* 9.8):

```
function CalculateTotal ($HowMany,
→ $Savings = '0') {
```

3. Ha establecido el valor de $Savings como predeterminado para que sea 0. Si no se pasa ningún argumento a la función, se asumirá que no se aplica ningún descuento. Edite la segunda función y ajuste el argumento $NumberPayments al valor predetrerminado de 12.

```
function CalculatePayments ($Amount,
→ $NumberPayments = '12') {
```

Si se envían los dos argumentos a la función CalculatePayments(), $NumberPayments se establecerá al segundo valor en lugar de al predeterminado.

4. Elimine las dos líneas del cuerpo principal del *script* que han impreso el valor $Tax, ya que no son relevantes.

5. Después, llame a la función CalculatePayments para que ya no envíe un argumento para el número de pagos.

```
print (CalculatePayments
→ ($TotalCost));
```

La llamada a CalculatePayments() ahora sólo pasa el valor de $TotalCost como un argumento, ya que $NumberPayments tiene un valor predeterminado. Puede pasar un valor $NumberPayments hasta aquí si desea establecer un número diferente a 12 como número de pagos mensuales.

Script 9.8. La página numbers.php se escribe ahora como se escribe una función normalmente. Establecer el valor predeterminado en la función crea un argumento opcional.

script

```
1  <?php
2  function CalculateTotal ($HowMany,
   $Savings = '0') {
3      global $Cost;
4      global $Tax;
5      $Tax++; // $Tax tiene ahora el valor 1.06.
6      $TotalCost = ($Cost * $HowMany);
7      if ( ($TotalCost < 50) AND
       ($Savings) ) {
8          print ('Your \$$Savings will not
           apply because the total value of
           the sale is under $50!\n<P>');
9      }
10     if ($TotalCost >= 50) {
11         $TotalCost = $TotalCost - $Savings;
12     }
13     $TotalCost = $TotalCost * $Tax;
14     return $TotalCost;
15 } // Final de la función CalculateTotal.
16 function CalculatePayments ($Amount,
   $NumberPayments = '12') {
17     $Payments = round($Amount, 2) /
       $NumberPayments;
18     $Payments = sprintf ('%01.2f',
       $Payments);
19     return $Payments;
20 } // Final de la función CalculatePayments.
21 ?>
22 <HTML>
23 <HEAD>
24 <TITLE>Calculation Functions</TITLE>
25 </HEAD>
26 <BODY>
27 <?php
28 $Cost = 20.00;
29 $Tax = 0.06;
30 if ($Quantity) {
31     $Quantity = abs($Quantity);
32     $Discount = abs($Discount);
33     $TotalCost = CalculateTotal ($Quantity,
       $Discount);
34     // Imprimir los resultados.
35     print ('You requested to purchase
       $Quantity widget(s) at \$$Cost
       each.\n<P>');
```

Script 9.8. Continuación

```
script
36      print ("The total with tax, minus your
        \$$Discount, comes to $");
37      printf ("%01.2f", $TotalCost);
38      print (".\n<P>You may purchase the
        widget(s) in 12 monthly installments
        of $");
39      print (CalculatePayments($TotalCost));
40      print (" each.\n<P>");
41  } else {
42      print ("Please make sure that you have
        entered both a quantity and an
        applicable discount and then
        resubmit.\n");
43  }
44  ?>
45  </BODY>
46  </HTML>
```

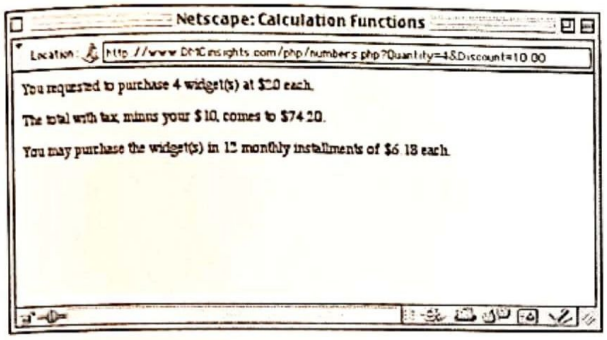


Figura 9.6. Aunque el usuario final, visualizando las páginas en su navegador, nunca sería capaz de indicar la diferencia entre el script que utiliza funciones y el que no las usa (compare esto con la Figura 9.4), usted ha incrementado la posibilidad de reutilizar su código.

6. Guarde el *script*, cárguelo en el servidor y pruébelo en el navegador (véase la Figura 9.6). No se olvide de enviar al *script* un valor Quantity (y un valor Discount, si quiere), agregándolo al URL o mediante un formulario HTML.

CONSEJO

- En cada ejemplo de este capítulo hemos pasado argumentos a funciones según su valor, lo que significa que una copia del valor de una variable se envía a la función, no la propia variable. También puede pasar argumentos según sus referencias, lo que le permite modificar una variable en una función, pero es un tema demasiado complejo para este libro. Consulte el Apéndice C para obtener más información al respecto.
- Existen diferentes corrientes de pensamiento para la convención de nombres que se emplea en los argumentos de las funciones. Por un lado, utilizar el mismo nombre de la función como una variable exterior, facilita el recuerdo de cómo coinciden los valores. Por otro lado, usar nombres de argumentos genéricos en la función la hace menos específica del *script*. Puede emplear un método híbrido añadiendo una "f" minúscula al comienzo de los nombres de la funciones, como $fHowMany y $fTaxRate. Algunos programadores identifican las variables globales como $gVariable. De nuevo, la clave es que su convención de nombres sea coherente.

ARCHIVOS Y DIRECTORIOS 10

Para llevar realmente las aplicaciones web al siguiente nivel, necesitará un método para almacenar y recuperar los datos. Los sitios de comercio electrónico, portales de Internet y proveedores de contenido dirigido por plantillas hacen uso de todo tipo de escritura y lectura del servidor.

Hay dos formas principales de almacenar datos con PHP: utilizar archivos (y directorios) o bases de datos. En este capítulo trataremos la primero, y en el siguiente capítulo presentaremos la segunda forma. Merece la pena comprender los dos métodos.

Al menos, hay dos ventajas en usar archivos y directorios para almacenar datos en lugar de bases de datos: 1) no requieren conocimiento de bases de datos (un tema complicado), y 2) la mayoría de servidores web cargan extras en su factura por el acceso a las bases de datos, mientras que el acceso a los archivos es gratis. Mientras que una base de datos será inevitablemente más potente que un sistema basado en archivos, ¡se sorprenderá de lo que puede hacer sólo enviando y recuperando información de simples documentos de texto en el servidor!

En este capítulo conocerá los permisos de archivo, y entonces aprenderá a: escribir a y leer desde archivos; crear directorios; manipular cargas de archivos desde un formulario HTML; y realizar otras tareas comunes, tanto con archivos como con directorios (renombrar, eliminar, etc.).

Permisos de archivo

Antes de pasar a los ejemplos de escribir a, y leer desde, un archivo, deberíamos mencionar brevemente los permisos de archivo. Éste es otro tema (como la seguridad y las expresiones regulares) lo suficientemente amplio que puede que quiere aprender mucho más de lo aquí expuesto: en tal caso, consulte el Apéndice C. Sin embargo, abordaremos lo suficiente para comenzar.

Los permisos de archivo y directorio identifican quién puede hacer qué con el archivo o directorio. Las opciones son: **leer**, **escribir** y **ejecutar** (en realidad, los archivos pueden ser designados como ejecutables, mientras que los directorios son lugares donde buscar). Por encima de ello, estas opciones se pueden configurar de forma diferente para tres únicos tipos de usuarios: el **propietario** del archivo, es decir, la persona que lo pone en el servidor; los miembros de un grupo particular, configurado por un administrador del servidor; y **otros**, es decir, aquellos que no están en las dos categorías anteriores. También existe el nivel de cualquier implicado que incluye a todos los usuarios anteriores.

Habitualmente, al propietario de un archivo se le dan por defecto los permisos de lectura y escritura, mientras que los "otros" sólo pueden leer un archivo (los grupos realmente no entran en juego aquí). La ejecución de un archivo es una cuestión de seguridad básica y afortunadamente no afecta a los *scripts* PHP (Perl puede escribir un archivo ejecutable, pero, por lo general, PHP no puede, ya que es un lenguaje de *scripting*, no un lenguaje de programación). Poder escribir en un archivo también puede ser una cuestión de seguridad y sólo se debería designar esta opción cuando fuera absolutamente necesario.

En este capítulo trabajará con un archivo de texto en el servidor llamado data.txt. Dependiendo de la configuración de su servidor (por ejemplo, si está trabajando en el servidor de un ISP o en su propia computadora personal), puede que quiera configurar los permisos en este archivo con el tiempo. Si el archivo no tiene los permisos correctos para hacer lo que su *script* PHP va a pedirle, verá un mensaje de error similar al de la Figura 10.1. Antes de proceder con este capítulo, debería crear data.txt en el servidor.

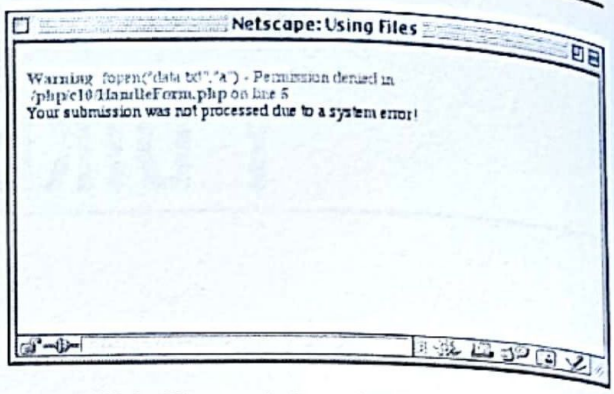


Figura 10.1. El mensaje "permission denied..." es el resultado de intentar hacer algo con un archivo que no está permitido por el servidor. Aquí, el servidor ha denegado la función fopen(), que intentaba abrir data.txt con intención de escribir en él.

```
$ chmod 0666 data.txt
$ ls -l
-rw-rw-rw-  ullman  375  data.txt
```

Figura 10.2. En el indicador de comandos del shell, en el directorio donde está ubicado data.txt, la línea chmod 0666 data.txt cambia los permisos de este archivo. Puede comprobar los permisos escribiendo ls -l. Este comando revela lo siguiente: data.txt tiene permisos de lectura y escritura para las tres categorías de usuarios; el propietario del archivo es ullman; y el tamaño del archivo es de 375 bytes.

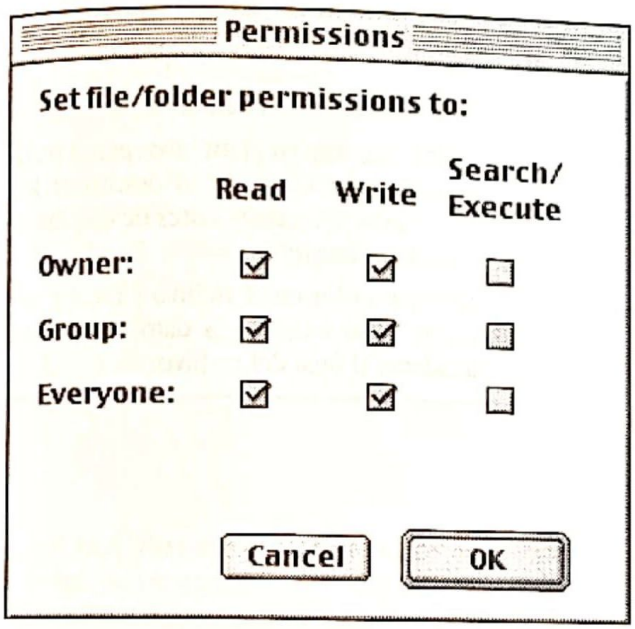


Figura 10.3. La aplicación Fetch FTP para Macintosh utiliza esta ventana desplegable para permitirle configurar los permisos de un archivo.

CONSEJO

- Si ya le son familiares telnet y chmod, probablemente también entienda lo que significa el número 0666 (véase la Figura 10.2), aunque lo explicaremos para aquellos a los que no les sea familiar. El 0 es un prefijo necesario, y cada 6 equivale a los permisos escribir (4) más leer (2). Primero se asigna 6 al propietario, después al grupo y después a los otros. Comparativamente, 0777 permite escribir (4) más leer (2) más ejecutar (1) a los tres tipos de usuarios. Esto sólo es aplicable a los sistemas operativos derivados de Unix (por ejemplo, Linux y Solaris).

Cómo crear data.txt

1. Abra el editor de textos y cree un documento nuevo.
2. Sin escribir nada más en el archivo, grábelo como data.txt.
3. Cargue el archivo en el servidor.

Esto puede parecer un proceso extraño; sin embargo, para configurar los permisos de un archivo, éste, primero, debe existir. De momento, el archivo está en blanco, aunque usará PHP para escribir datos en él más tarde.

Dependiendo de su situación, hay un número de formas en las que puede configurar los permisos de un archivo. El resultado final deseado para este ejemplo es dar a algunos de los otros o a todos el permiso para leer y escribir (pero no ejecutar) data.txt. Necesitará comprobar con el ISP cómo establecer los permisos de archivo, pero lo más probable es que esté implicado en una de las siguientes opciones:

Cómo configurar los permisos de un archivo

- Algunos ISP ofrecen a los usuarios un panel de control basado en la Web donde pueden configurar los permisos, así como configurar otros parámetros del hospedaje.
- Si tiene acceso telnet al servidor, navegue al directorio donde está localizado data.txt y utilice el comando chmod (en un servidor Unix) para modificar los permisos (véase la Figura 10.2). Para más información sobre la configuración de permisos utilizando chmod en servidores Unix, consulte el consejo que hay más adelante.
- Puede cambiar los permisos de un archivo utilizando el cliente FTP (véase la Figura 10.3).
- Si va a trabajar en su propio servidor Windows NT, puede modificar los permisos de un archivo haciendo clic con el botón derecho en el archivo, seleccionando Propiedades y haciendo clic en la ficha Seguridad.

Escritura en archivos

Ya que necesita escribir en un archivo para leerlo después, abordaremos primero la escritura. Escribir en un archivo en el servidor es un proceso que consta de tres pasos: primero, abra el archivo; segundo, escriba los datos en él; y tercero, cierre el archivo. Afortunadamente, las funciones internas de PHP para hacer estos pasos son bastante obvias:

```
$FileName = "data.txt";
$FilePointer = fopen ($FileName, "mode");
fwrite ($FilePointer, "data to be
 › written");
fclose ($FilePointer);
```

Al manipular archivos, lo primero que hacemos es asignar el nombre de archivo y la ruta (dónde está en el servidor) a una variable, por conveniencia. Utilice convenciones comunes de rutas cuando asigne el valor (por ejemplo, si el archivo estuviera en el directorio files, que es un subgrupo del directorio actual, el nombre del archivo sería files/data.txt).

Una vez que haya identificado el archivo, debe crear un puntero de archivo cuando lo abra. Este puntero de archivo, asignado a la variable nombrada apropiadamente $FilePointer, es utilizado por PHP para referirse al archivo abierto.

La consideración más importante al abrir el archivo es el modo que utilizará. Dependiendo de lo que intente hacer con el archivo, el modo le dicta cómo abrirlo. El modo más permisivo es a+, que permite leer o escribir en un archivo. Creará el archivo si no existe y añadirá los nuevos datos al final del archivo automáticamente. Por el contrario, r sólo permitirá leer desde un archivo. La Tabla 10.1 enumera todos los modos posibles.

La función fwrite() escribirá los datos nuevos, enviados como segundo argumento en la llamada a la función, en el archivo de acuerdo con el modo seleccionado. Finalmente, cierre el archivo refiriéndose una vez más al puntero del mismo.

Vamos a crear un formulario que almacene los URL enviados por el usuario en un archivo separado.

Tabla 10.1. El modo a utilizar cuando se abre un archivo es más que patente. Las cosas más importantes a recordar son: si va a leer, escribir o ambos, y qué hacer con los contenidos existentes (añadir o sobrescribir).

Modos de archivo	
Modo	Permite
r	Sólo leer desde el archivo.
w	Sólo escribir en el archivo, pero creará el archivo si no existe o descartará los contenidos existentes antes de escribir si el archivo existe.
a	Añadir nuevos datos al final del archivo y crear el archivo si no existe.
r+	Leer y escribir en el archivo.
w+	Leer y escribir en el archivo, pero creará el archivo si no existe o descartará los contenidos existentes antes de escribir si el archivo existe.
a+	Leer y escribir en el archivo y crear el archivo si no existe. Los datos nuevos se añadirán al final del archivo.

Script 10.1. Ésta es la versión de form.html existente. Si quiere, puede eliminar las líneas de los apellidos, ya que no serán necesarias aquí (consulte el Script 10.2).

```
1  <HTML>
2  <HEAD>
3  <TITLE>HTML Form</TITLE>
4  </HEAD>
5  <BODY>
6  <FORM ACTION="HandleForm.php" METHOD=POST>
7  First Name <INPUT TYPE=TEXT
   NAME="Array[FirstName]" SIZE=20><BR>
8  Last Name <INPUT TYPE=TEXT
   NAME="Array[LastName]" SIZE=40><BR>
9  URL <INPUT TYPE=TEXT NAME="Array[URL]"
   SIZE=60><BR>
10 Description <TEXTAREA NAME=
   "Array[Description]" ROWS=5
   COLS=40></TEXTAREA><BR>
11 <INPUT TYPE=SUBMIT NAME="SUBMIT"
   VALUE="Submit!">
12 </FORM>
13 </BODY>
```

Cómo escribir en un archivo externo

1. Si lo desea, abra el archivo form.html del Capítulo 8 (véase el *Script* 10.1) en el editor de textos.

   Haremos algunos pequeños cambios en el formulario, pero puede saltarse este paso, si quiere, sin que esto afecte al resultado final.

2. Elimine las líneas primera y última (véase el *Script* 10.2).

3. Grabe la página y cárguela en el servidor.

   Ahora escribirá un nuevo *script* HandleForm.php que procese los datos generados por form.html.

4. Cree un nuevo documento PHP en el editor de textos.

Script 10.2. Para mantener las cosas de forma ordenada, hemos eliminado las líneas primera y última de form.html. Sin embargo, puesto que no cambiamos el nombre de las dos entradas importantes, Array[URL] y Array[Description], mantuvimos la página como estaba, lo que no fue un problema.

```
1  <HTML>
2  <HEAD>
3  <TITLE>HTML Form</TITLE>
4  </HEAD>
5  <BODY>
6  <FORM ACTION="HandleForm.php" METHOD=POST>
7  URL <INPUT TYPE=TEXT NAME="Array[URL]" SIZE=60><BR>
8  Description <TEXTAREA NAME="Array[Description]" ROWS=5 COLS=40></TEXTAREA><BR>
9  <INPUT TYPE=SUBMIT NAME="SUBMIT" VALUE="Submit!">
10 </FORM>
11 </BODY>
12 </HTML>
```

5. Comience definiendo la función PHP (véase el *Script* 10.3):

```
<?
function WriteToFile ($URL,
→ $Description) {
```

La función WriteToFile() toma dos argumentos, el URL y la descripción del URL que el usuario introdujo en form.html.

6. Identifique y abra el archivo.

```
$TheFile = "data.txt";
$Open = fopen ($TheFile, "a");
```

Estas dos líneas identifican el archivo y lo abren sólo para escritura (creando, si es necesario, y escribiendo los datos nuevos al final del archivo si el archivo existe). El archivo data.txt referido aquí debería ser el mismo que creó y al que asignó los permisos adecuados en la sección anterior de este capítulo.

7. Ahora, creará una condición basada en el éxito de la función fopen().

```
if ($Open) {
    fwrite ($Open, "$URL\
    → t$Description\n");fclose
    → ($Open);
    $Worked = TRUE;
} else {
    $Worked = FALSE;
}
```

Si el archivo se abre correctamente, la función añadirá los datos nuevos a los anteriores. El formato para los datos es el URL seguido de un carácter de tabulación (creado usando una secuencia de escape con la letra *t*), y después la descripción seguida de un carácter de nueva línea (creado mediante una secuencia de escape con la letra *n*). Entonces se cierra el archivo.

8. Devuelva el valor $Worked y cierre la función.

```
    return $Worked;
} // Fin de la función WriteToFile.
```

Si el archivo se abrió correctamente, la variable $Worked se establece a TRUE y se devuelve para indicar el éxito de la función.

Script 10.3. Este script muestra algunas de las técnicas aprendidas hasta este momento, las cuales debe considerar utilizar al programar. Primero, la función se escribe como el primer elemento del script. Segundo, se comprueba la exactitud de los datos enviados por el usuario utilizando expresiones regulares. Tercero, se envían mensajes al navegador en caso de que haya problemas.

script

```
1  <?
2  function WriteToFile ($URL, $Description) {
3  /* La función WriteToFile toma dos
   argumentos--URL y Description--que
   se escribirán en un archivo externo. */
4      $TheFile = "data.txt";
5      $Open = fopen ($TheFile, "a");
6      if ($Open) {
7          fwrite ($Open,
           "$URL\t$Description\n");
8          fclose ($Open);
9          $Worked = TRUE;
10     } else {
11         $Worked = FALSE;
12     }
13     return $Worked;
14 } // Fin de la función WriteToFile.
15 ?>
16 <HTML>
17 <HEAD>
18 <TITLE>Using Files</TITLE>
19 <BODY>
20 <?php
21 /* Esta página recibe y procesa los
   datos generados por "form.html". */
22 $Pattern = "(http://)?([^[:space:]]+)
   ([[:alnum:]\.,-_?/&=])";
23 if (eregi($Pattern, $Array["URL"])) {
24     $Replace = "<a href=\"http://\\2\\3\"
       target=\"_new\">\\2\\3</a>";
25     $Array["URL"] = eregi_replace($Pattern,
       $Replace, $Array["URL"]);
26     $CallFunction = WriteToFile
       ($Array["URL"], $Array["Description"]);
27     if ($CallFunction) {
28         print ("Your submission--
           $Array[URL]--has been
           received!<BR>\n");
29     } else {
30         print ("Your submission was
           not processed due to a system
           error!<BR>\n");
```

Script 10.3. Continuación

script

```
31      }
32  } else {
33      print ("Please enter a valid Web
        address!\n");
34  }
35  ?>
36  </BODY>
37  </HTML>
```

9. Cierre el código PHP inicial y cree la cabecera HTML estándar.

```
?>
<HTML><HEAD><TITLE>Using Files
→ </TITLE><BODY>
```

10. Abra la sección PHP principal.

```
<?php
```

11. Ahora, defina el patrón de la expresión regular:

```
$Pattern = "(http://)?([^[:space:]]+)
→ ([[:alnum:]\.,-_?/&=])";
```

Una vez más, las expresiones regulares se utilizarán para confirmar que el URL es correcto y para convertirlo en un enlace en el que hacer clic.

12. Cree una condición basada en la ejecución satisfactoria de la expresión regular.

```
if (eregi($Pattern, $Array["URL"]))
{
    $Replace = "<a href=\"http://
    → \\2\\3\" target=\"_new\">
    → \\2\\3</a>";
    $Array["URL"] = eregi_replace
    → ($Pattern, $Replace,
    → $Array["URL"]);
    $CallFunction = WriteToFile
    → ($Array["URL"],
    → $Array["Description"]);
```

Si el URL enviado por el usuario coincide con la expresión regular, se cambiará entonces usando eregi_replace() y el concepto de referencia regresiva. Después, se llama a la función WriteToFile().

13. Ahora, PHP editará un mensaje basado en los resultados de la llamada a la función.

```
if ($CallFunction) {
    print ("Your submission--
    → $Array[URL]--has been
    → received!<BR>\n");
} else {
    print ("Your submission was not
    → processed due to a system
    → error!<BR>\n");
}
```

La variable $CallFunction será igual al valor devuelto en $Worked desde la función. Si PHP pudiera abrir el archivo, $Worked y, por tanto, $CallFunction serían TRUE, y else sería FALSE. De acuerdo a esto, se mostrará un mensaje al usuario indicando el resultado.

14. Cierre la sentencia if, PHP y HTML.

```
} else {
print ("Please enter a valid Web
→ address!\n");
}
?>
</BODY></HTML>
```

15. Guarde el *script* como HandleForm.php, cárguelo en el servidor (en la misma ubicación que form.html y data.txt) y pruébelos en el navegador web (véanse las Figuras 10.4, 10.5 y 10.6).

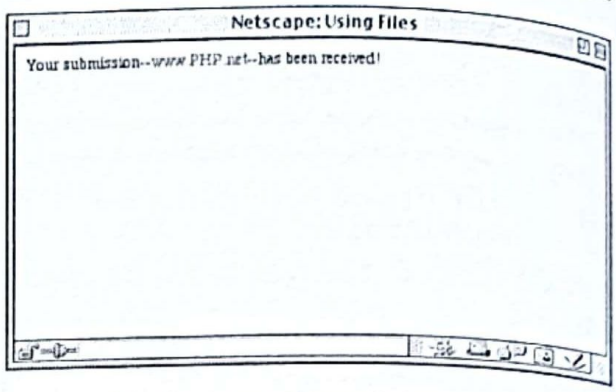


Figura 10.4. Tan pronto el usuario introduce una dirección de correo electrónico correcta y el script puede escribir los datos en el archivo, todos los usuarios deberían ver esto.

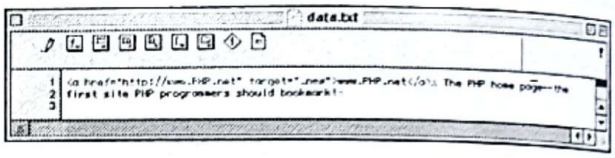


Figura 10.5. Después de ejecutar las páginas form.html y HandleForm.php al menos una vez, puede descargar data.txt en su computadora y ver su contenido en un editor de textos. Observe que la etiqueta HTML A HREF, generada con la función eregi_replace(), también se escribe en el archivo.

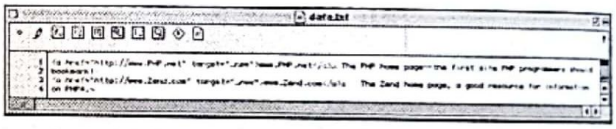


Figura 10.6. Para cada envío adicional de form.html, se añadirá otra línea de información a data.txt. Este editor de textos en particular (BBEdit) también indica las tabulaciones (con el triángulo) y las líneas nuevas (el último carácter en las líneas 2 y 4).

- Como un paso extra de comprobación de seguridad, puede utilizar la función is_writeable() para determinar si el servidor le permitirá escribir datos en el archivo antes de intentar abrirlo. A continuación puede ver cómo debería empezar a incorporarlo (esto es sólo el comienzo del *script*):

```
$TheFile = "data.txt";
if (is_writeable ($TheFile)) {
$Open = fopen ($TheFile, "a");
```

- En los servidores Windows, asegúrese de crear una secuencia de escape con las barras invertidas en una ruta de acceso o utilice barras inclinadas en su lugar. Por ejemplo, podría indicar c:\\php\\data.txt o c:/php/data.txt, pero no c:\php\data.txt, ya que la barra invertida es el carácter de escape. Los servidores Unix utilizan la barra inclinada como el estándar, por lo que no es un problema en esos servidores.

Script 10.4. Este script es más fácil de entender mediante la creación de cuatro funciones: una para leer del archivo, otra para escribir en él, otra para crear un formulario y otra para manipular el formulario. De esta manera, el cuerpo principal de la página es una sola condición que llama a las funciones.

script

```
1  <HTML>
2  <HEAD>
3  <TITLE>Storing URLs in an External
   File</TITLE>
4  </HEAD>
5  <BODY>
6  <?php
7  function WriteToFile ($URL, $Description) {
8  /* La función WriteToFile toma dos
   argumentos--URL y Description--que se
   escribirán en un archivo externo. */
9      $TheFile = "data.txt";
10     $Open = fopen ($TheFile, "a");
11     if ($Open) {
12         fwrite ($Open, "$URL\
           t$Description\n");
13         fclose ($Open);
14         $Worked = TRUE;
15     } else {
16         $Worked = FALSE;
17 }
18     return $Worked;
19 } // Fin de la función WriteToFile.
20
21 function ReadFromFile () {
22 /* La función ReadFromFile muestra toda la
   información almacenada en un archivo
   externo. */
23     $TheFile = "data.txt";
24     $Open = fopen ($TheFile, "r");
25     if ($Open) {
26         print ("URLs currently listed in
           the data file:<P>\n");
27         $Data = file ($TheFile);
28         for ($n = 0; $n < count($Data);
           $n++) {
29             $GetLine = explode("\t",
               $Data[$n]);
30             print ("$GetLine[0]<BR>
               \n$GetLine[1]<P>\n");
31         }
32         fclose ($Open);
```

Lectura desde archivos

Ahora que ha codificado un *script* que escribe datos en un archivo, es el momento de crear uno que pueda leer esa información. Leer datos de un archivo es casi tan fácil como escribirlos. Hay, de nuevo, tres pasos. Primero se abre el archivo, después se lee del archivo y, finalmente, se cierra el archivo.

```
$FileName = "data.txt";
$FilePointer = fopen ($FileName,
→ "mode");
$Array = file ($FileName);
fclose ($FilePointer);
```

La función file() es una valiosa herramienta interna de PHP. Lee todo de un archivo y sitúa esa información en un array: cada elemento del array queda definido por un retorno de carro o un salto de línea. Si data.txt contiene dos líneas de información, cada una de las cuales termina con un salto de línea (véase la Figura 10.6, pero no confunda el hecho de que cada línea parezca desbordarse en una segunda línea), el array correspondiente contendrá dos elementos. El primer elemento será igual a la primera línea de data.txt y el segundo elemento será igual a la segunda línea. Una vez que los datos están almacenados en un array, se pueden manipular o imprimir fácilmente.

Ahora utilizará este conocimiento para crear un *script* que reúna los URL enviados por un usuario, además de mostrar en pantalla los URL enviados anteriormente, ¡todo en una página!

Cómo leer desde un archivo

1. Cree un nuevo documento en el editor de textos.
2. Comience con la cabecera HTML estándar (véase el *Script* 10.4):

```
<HTML><HEAD>
<TITLE>Storing URLs in an External
→ File</TITLE>
</HEAD><BODY>
```

Ya que esta página actuará como el archivo en sí mismo y procesará los resultados del formulario, la escribiremos de manera un poco diferente a lo que hemos visto; por ello, no

empezaremos la página con las funciones como hacemos habitualmente.

3. Abra una sección de código PHP y duplique la función WriteToFile() de HandleForm.php (véase el *Script* 10.3).

```
<?php
function WriteToFile ($URL,
→ $Description) {
$TheFile = "data.txt";
$Open = fopen ($TheFile, "a");
if ($Open) {
fwrite ($Open,
→ "$URL\t$Description\n");
fclose ($Open);
$Worked = TRUE;
} else {
$Worked = FALSE;
}
return $Worked;
} // Fin de la función WriteToFile.
```

No debe haber cambiado ninguna de las líneas de las aquí mostradas, ¡una de las ventajas de programar con funciones!

4. Cree una nueva función para recuperar datos del archivo.

```
function ReadFromFile () {
$TheFile = "data.txt";
$Open = fopen ($TheFile, "r");
if ($Open) {
print ("URLs currently listed in the
→ data file:<P>\n");
$Data = file ($TheFile);
for ($n = 0; $n < count($Data);
→ $n++) {
$GetLine = explode("\t", $Data[$n]);
print ("$GetLine[0]<BR>
→ \n$GetLine[1]<P>\n");
}
fclose ($Open);
print ("<HR><P>\n");
} else {
print ("Unable to read from
→ data.txt!<BR>\n");
}
} // Fin de la función ReadFromFile.
```

Esta función empieza como WriteToFile() con la identificación del archivo, el intento de abrirlo

Script 10.4. Continuación

script

```
33          print ("<HR><P>\n");
34      } else {
35          print ("Unable to read from
            data.txt!<BR>\n");
36      }
37  } // Fin de la función ReadFromFile.
38
39  function CreateForm () {
40  /* La función CreateForm visualizará el
    formulario HTML. */
41      print ("Add a URL to the data file:\n");
42      print ("<FORM ACTION=\"urls.php\"
        METHOD=POST>\n");
43      print ("URL <INPUT TYPE=TEXT NAME=
        \"Array[URL]\" SIZE=60><BR>\n");
44      print ("Description <TEXTAREA
        NAME=\"Array[Description]\" ROWS=5
        COLS=40></TEXTAREA><BR>\n");
45      print ("<INPUT TYPE=HIDDEN
        NAME=\"BeenSubmitted\"
        VALUE=\"TRUE\">\n");
46      print ("<INPUT TYPE=SUBMIT
        NAME=\"SUBMIT\" VALUE=\"Submit!\">
        </FORM>\n");
47  } // Fin de la función CreateForm.
48
49  function HandleForm () {
50      global $Array;
51      $Pattern = "(http://)?([^[:space:]]+)
        ([[:alnum:]\.,-_?/&=])";
52      if (eregi($Pattern, $Array["URL"])) {
53          $Replace = "<a href=
            \"http://\\2\\3\"
            target=\"_new\">\\2\\3</a>";
54          $Array["URL"] = eregi_replace
            ($Pattern, $Replace, $Array["URL"]);
55          $CallFunction = WriteToFile
            ($Array["URL"],
            $Array["Description"]);
56          if ($CallFunction) {
57              print ("Your submission--
                $Array[URL]--has been
                received!<P><HR><P>\n");
58          } else {
59              print ("Your submission
                was not processed due to a
                system error!<BR>\n");
60          }
```

Script 10.4. Continuación

```
script
61      } else {
62          print ("Please enter a valid Web
            address!\n");
63      }
64  } // Fin de la función HandleForm.
65
66  /* La siguiente condición determina
    si se procesa el formulario, dependiendo
    de si $BeenSubmitted es TRUE o no. */
67  if ($BeenSubmitted) {
68      HandleForm();
69  }
70  ReadFromFile();
71  CreateForm();
72  ?>
73  </BODY>
74  </HTML>
```

(usando r como el modo para que PHP pueda leer los datos) y después continúa, suponiendo que abrió correctamente el archivo.

La función imprimirá un título y leerá los datos del archivo en un array llamado $Data. Cada elemento de $Data es una cadena (en el formato de un URL, seguido de un tabulador, su descripción y un salto de línea). Convertirá cada cadena en un nuevo array, llamado $GetLine, efectuando un bucle para cada elemento $Data. En el bucle, la función explode() creará $GetLine a partir de la cadena separando los elementos basándose en la localización del tabulador. Utilizando esta nueva variable, puede editar las partes de la cadena original individualmente.

Finalmente, el archivo se cierra, se envía algo de formato HTML al navegador y, entonces, se completa la condición, generando un mensaje de error si PHP no puede abrir el archivo.

5. Ahora, escriba una tercera función, la cual creará el formulario de envío HTML con cada llamada.

```
function CreateForm () {
    print ("Add a URL to the data
    → file:\n");
    print ("<FORM ACTION=\"urls.php\"
    → METHOD=POST>\n");
    print ("URL <INPUT TYPE=TEXT NAME=
    →\"Array[URL]\" SIZE=60><BR>\n");
    print ("Description <TEXTAREA
    → NAME=\"Array[Description]\"
    → ROWS=5 COLS=40></TEXTAREA>
    → <BR>\n");
    print ("<INPUT TYPE=HIDDEN NAME=
    → \"BeenSubmitted\" VALUE=
    → \"TRUE\">\n");
    print ("<INPUT TYPE=SUBMIT NAME=
    → \"SUBMIT\" VALUE=\"Submit!\">
    → </FORM>\n");
} // Fin de la función CreateForm.
```

Esta función es casi lo mismo que la página form.html con dos cambios principales. Primero, la etiqueta ACTION del formulario ahora envía los datos del formulario a urls.php, que será

el nombre de este *script*. Segundo, hemos añadido un campo oculto que establece el valor de $BeenSubmitted a TRUE. Verá por qué hacemos esto en breve.

6. Escribimos una cuarta función con el fin de manipular el formulario.

```
function HandleForm () {
   global $Array;
```

Esta función se basa en el *script* anterior, HandleForm.php. Lo primero que haremos será traer la variable $Array, utilizando la sentencia global, que contiene los datos del formulario.

7. El corazón de esta función está tomado del cuerpo principal de la página anterior, HandleForm.php (véase el *Script* 10.4).

```
   $Pattern = "(http://)?
   → ([^[:space:]]+)([[:alnum:]
   → \.,-_?/&=])";
   if (eregi($Pattern, $Array
   → ["URL"])) {
       $Replace = "<a href=\"http://
       → \\2\\3\" target=\"_new\">
       → \\2\\3</a>";
       $Array["URL"] =
       → eregi_replace($Pattern,
       → $Replace, $Array["URL"]);
       $CallFunction = WriteToFile
       → ($Array["URL"],
       → $Array["Description"]);
       if ($CallFunction) {
           print ("Your submission--
           → $Array[URL]--has been
           → received!<P><HR><P>
           → \n");
       } else {
           print ("Your submission
           → was not processed due
           → to a system error!
           → <BR>\n");
       }
   } else {
       print ("Please enter a
       → valid Web address!\n");
   }
} // Fin de la función HandleForm.
```

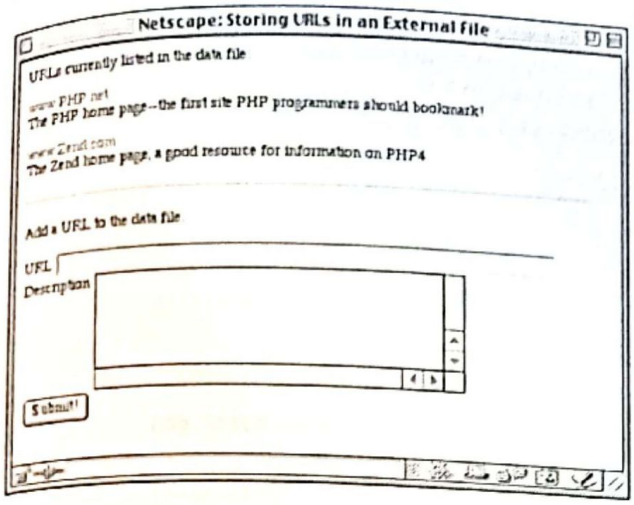


Figura 10.7. La primera vez que el usuario llega a la página, $BeenSubmitted no es igual a TRUE, puesto que nunca se ha llamado a HandleForm() (véase el Script 10.4). Una vez que el formulario se ha enviado, el código PHP manipulará los resultados e imprimirá una respuesta como la de la Figura 10.8.

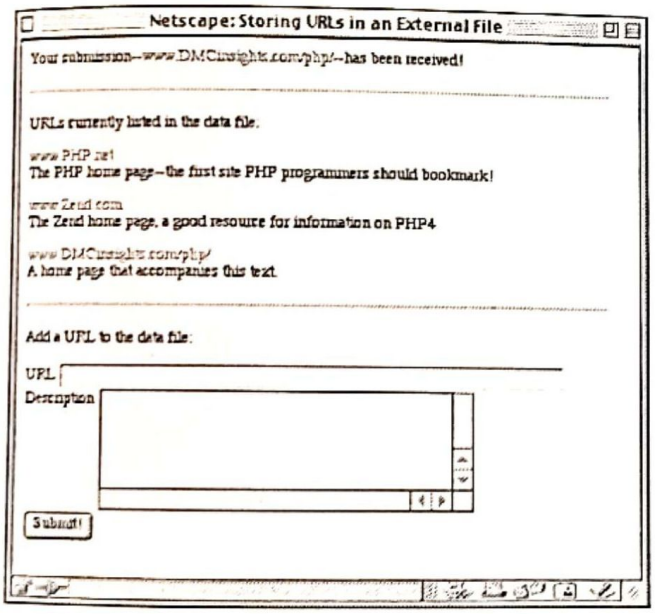


Figura 10.8. Después de que el usuario envíe el formulario, urls.php primero manipulará los datos y después visualizará la información del archivo y creará un formulario nuevo.

La función recorre los procesos de la expresión regular e imprimirá un mensaje de error si se introduce un URL incorrecto. Suponiendo que se haya enviado un URL correcto, la línea $CallFunction y su condición correspondiente vienen después.

8. Finalmente, cree la condición que decida si se manipula o no el formulario.

```
if ($BeenSubmitted){
   HandleForm();
}
ReadFromFile();
CreateForm();
```

Si el valor de $BeenSubmitted es TRUE (lo cual significa que se ha enviado el formulario), los datos se procesarán.

Si el valor de $BeenSubmitted es FALSE, lo que significa que el usuario no ha enviado nada, no se llama a la función HandleForm(). En cualquier caso, los datos existentes se muestran desde el archivo y entonces se crea un formulario para que el usuario pueda introducir otro URL.

9. Cierre el PHP y el HTML.

```
?></BODY></HTML>
```

10. Guarde la página como urls.php, cárguela en el servidor y pruébela en el navegador web (véanse las Figuras 10.7, 10.8 y 10.9).

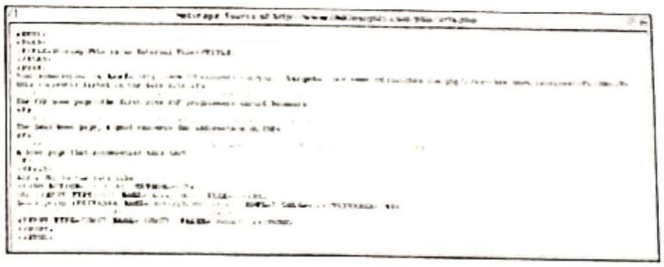


Figura 10.9. Código fuente de urls.php, después de haberse enviado el formulario. Observe que en todos los URL se puede hacer clic y que el formulario utiliza un valor oculto, del que el usuario no es consciente, que añade funcionalidad al diseño.

Directorios

Entender cómo leer y escribir archivos en el servidor es solamente una parte del proceso de almacenamiento de datos. Es probable que también quiera emplear directorios al mismo tiempo. Puede pensar en un directorio de igual forma que en una carpeta (es una subsección del disco duro que puede almacenar otros archivos y directorios). Por defecto, todas sus páginas web están en un directorio específico (con frecuencia llamado www o html), dentro del cual puede crear más directorios que almacenen imágenes, datos, etc.

El comando para crear un directorio en PHP es:

```
mkdir ("ruta", "permisos");
```

Ruta constituirá el nombre y la localización del directorio, en tanto que los permisos (en forma de un cero seguido de tres dígitos, como 0666) asignarán derechos de acceso al directorio.

Vamos a crear un *script* que, al conectarse nuevos usuarios, cree un nuevo directorio para ellos.

Cómo crear un directorio nuevo

1. Primero creará una sencilla página de registro que tome un nombre de usuario y una clave de acceso. Abra el editor de textos e inicie un nuevo documento HTML (véase el *Script* 10.5).

```
<HTML><HEAD><TITLE>Registration
→ Form</TITLE></HEAD><BODY>
```

2. Cree un formulario cuyo atributo ACTION sea HandleNewUser.php y cuyo METHOD sea POST.

```
<FORM ACTION="HandleNewUser.php"
→ METHOD=POST>
```

Como el formulario tomará una clave de acceso, asegúrese de utilizar POST y no GET, ya que este último es menos seguro.

Script 10.5. Éste es un formulario de registro muy sencillo que sólo pide dos campos. El formulario de sus aplicaciones será probablemente más complicado, pero el proceso será el mismo.

script

```
1  <HTML>
2  <HEAD>
3  <TITLE>Registration Form</TITLE>
4  </HEAD>
5  <BODY>
6  <FORM ACTION="HandleNewUser.php"
   METHOD=POST>
7  Username <INPUT TYPE=TEXT NAME=
   "Array[Username]" SIZE=15><BR>
8  Password <INPUT TYPE=PASSWORD
   NAME="Array[Password]" SIZE=15><BR>
9  <INPUT TYPE=SUBMIT NAME="SUBMIT"
   VALUE="Submit!">
10 </FORM>
11 </BODY>
12 </HTML>
```

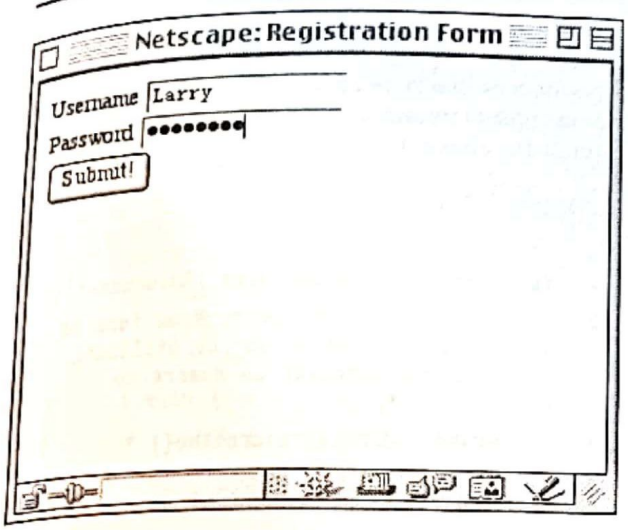


Figura 10.10. Éste es el formulario de registro básico. El atributo PASSWORD de INPUT TYPE oculta al usuario la información aquí introducida.

3. Cree dos entradas, una TEXT para el nombre de usuario y una PASSWORD para la clave de acceso.

```
Username <INPUT TYPE=TEXT NAME=
→ "Array[Username]" SIZE=15><BR>
Password <INPUT TYPE=PASSWORD NAME=
→ "Array[Password]" SIZE=15><BR>
```

Asegúrese de que TYPE sea igual a PASSWORD para el campo Password, de modo que el texto introducido no sea visible (véase la Figura 10.10).

4. Cree un botón de envío, cierre el formulario y el documento HTML.

```
<INPUT TYPE=SUBMIT NAME="SUBMIT"
→ VALUE="Submit!">
</FORM></BODY></HTML>
```

5. Guarde la página como NewUser.html y cárguela en el servidor.

Ahora necesitará construir un archivo de datos en blanco llamado users.txt y configurar los permisos adecuados, tal y como hizo con data.txt.

6. Cree un documento en blanco nuevo en el editor de textos.

7. Guarde el documento como users.txt y cárguelo en el servidor, en el mismo directorio que NewUser.html.

8. Configure los permisos de users.txt para que todo el mundo pueda leerlo y escribir en él.

El tercer paso es hacer un directorio en el que PHP creará los demás directorios.

9. Utilizando FTP o la aplicación telnet (o el panel de control basado en la Web proporcionado por su ISP), cree un nuevo directorio llamado users, en el mismo directorio que NewUser.html y users.txt.

10. Configure los permisos en el directorio users de manera que todos puedan leerlo, y escribir y buscar en él.

    Para finalizar, escribirá el *script* HandleNewUser.php, que procesará la información del formulario y creará el nuevo directorio.

11. Abra el editor de textos y cree un documento PHP nuevo.

12. Comience abriendo PHP y escriba la primera función (véase el *Script* 10.6).

```
<?
function MakeDirectoryName
→ ($Username) {
    srand ((double) microtime() *
    → 1000000);srand(mktime());
    $Name = rand() . $Username;
    return $Name;
} // Fin de la función
→ MakeDirectoryName.
```

    El directorio name se creará utilizando Username y un número aleatorio. El nombre se genera argumentando la función srand() con una fecha y hora y tomando de forma aleatoria un número basado en ellas. La concatenación de este número a $Username asegurará un solo directorio name cada vez, el cual es devuelto por la función.

13. Copie la función WriteToFile() del *Script* 10.4 y modifíquela ligeramente:

```
function WriteToFile ($Username,
→ $Password) {
    $TheFile = "users.txt";
    $Open = fopen ($TheFile, "a");
    if ($Open) {
        $Password = md5 ($Password);
```

    Una de las cosa que deberá cambiar aquí es el nombre del archivo de texto (ahora users.txt). También añadirá una línea que encriptará la clave antes de almacenarla (normalmente querrá almacenar una versión encriptada por razones de seguridad).

Script 10.6. Aunque aquí lo escribimos como dos funciones, podría crear una tercera función explícitamente con el fin de crear el directorio. Para hacer esto necesitaría separar la condición mkdir() de la función WriteToFile().

script

```
1  <?
2  function MakeDirectoryName ($Username) {
3  /* La función MakeDirectoryName toma un
   argumento--usersname--que se utilizará
   para crear y devolver un nombre de
   directorio. */
4      srand ((double) microtime() *
       1000000);
5      $Name = rand() . $Username;
6  return $Name;
7  } // Fin de la función MakeDirectoryName.
8
9  function WriteToFile ($Username,
   $Password) {
10 /* La función WriteToFile toma dos
   argumentos--URL y Description--que se
   escribirán en un archivo
   externo. */
11 $TheFile = "users.txt";
12 $Open = fopen ($TheFile, "a");
13 if ($Open) {
14 $Password = md5 ($Password);
15 $Directory = "users/" .
   MakeDirectoryName ($Username);
16 fwrite ($Open, "$Username\t$Password\
   t$Directory\n");
17 fclose ($Open);
18 if (!(mkdir ($Directory, "0777"))) {
19 $Directory = FALSE;
20 }
21 } else {
22 $Directory = FALSE;
23 }
24 return $Directory;
25 } // Fin de la función WriteToFile.
26 ?>
27 <HTML>
28 <HEAD>
29 <TITLE>Using Directories</TITLE>
30 <BODY>
31 <?php
32 /* Esta página recibe y procesa los datos
   generados por "NewUser.html". */
```

Script 10.6. Continuación

script

```
33
34 if (($Array[Username]) &&
   ($Array[Password])) {
35     $Check = WriteToFile
       ($Array[Username], $Array[Password]);
36     if ($Check) {
37         print ("Your request was
           successfully processed!<BR>\n");
38     } else {
39         print ("Your request was not
           processed due to a system
           error!<BR>\n");
40     }
41 } else {
42     print ("Please enter a Username and
       Password!\n");
43 }
44 ?>
45 </BODY>
46 </HTML>
```

14. Cree el directorio name.

```
$Directory = 'users/' .
 ‣ MakeDirectoryName ($Username);
```

El nuevo nombre de directorio se creará llamando a la función MakeDirectoryName y estableciendo que el directorio será un subdirectorio de users.

15. Añada los nuevos datos al archivo.

```
fwrite ($Open, "$Username\t$Password\
 ‣ t$Directory\n");
fclose ($Open);
```

16. Intente crear un directorio y escriba una condición que manipule el éxito de la acción.

```
if (!(mkdir ($Directory, "0777"))) {
$Directory = FALSE;
}
} else {
$Directory = FALSE;
}
return $Directory;
} // Fin de la función WriteToFile.
```

A la variable $Directory se le asigna como valor el nuevo nombre de directorio sólo si el archivo se ha abierto correctamente. Si no se puede abrir el archivo o no se puede crear el directorio, $Directory es igual a FALSE, indicando que la función falló al realizar las tareas solicitadas, de modo que la variable será devuelta.

17. Cierre el área de código PHP y cree la cabecera HTML.

```
?><HTML><HEAD><TITLE>Using
 ‣ Directories</TITLE><BODY>
```

18. Abra una segunda sección PHP y después cree la condición principal.

```
<?php
if (($Array[Username]) &&
 ‣ ($Array[Password])) {
     $Check = WriteToFile
      ‣ ($Array[Username],
      ‣ $Array[Password]);
```

```
if ($Check) {
        print ("Your request was
        successfully
        processed!<BR>\n");
    } else {
        print ("Your request was
        not processed due to a
        system error!<BR>\n");
    }
} else {
    print ("Please enter a Username
    and Password!\n");
}
```

Esta parte le debería parecer fácil por ahora. Hay una pregunta para ver si ambos campos se han rellenado y, si es así, se procederá a escribir los datos en el archivo y crear el nuevo directorio. Se envía un mensaje apropiado basado en el valor devuelto por la función.

19. Cierre esta última sección PHP y la página HTML.

    ?></BODY></HTML>

20. Guarde la página como HandleNewUser.php, cárguela en el servidor (en la misma ubicación que el directorio users, el archivo users.txt y la página NewUser.html) y pruébela en el navegador web (véanse las Figuras 10.11 y 10.12).

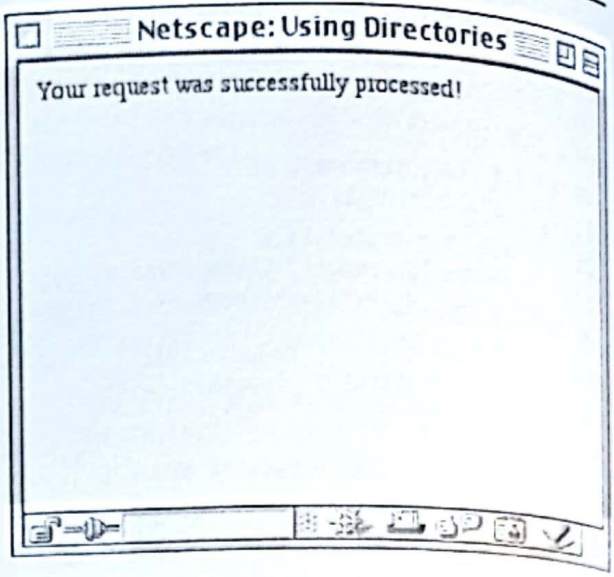


Figura 10.11. Si todo funciona según el diseño, éste es el único mensaje que ve el usuario.

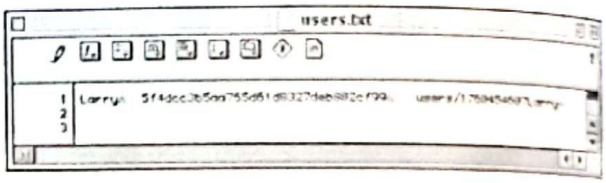


Figura 10.12. El archivo users.txt muestra tres campos de información separados por tabuladores: el nombre del usuario, una versión encriptada de la contraseña y el nombre del directorio del usuario.

CONSEJOS

- También puede asegurarse de que la página funciona como debería mirando comprobando si en el directorio users del servidor están los nuevos subdirectorios (mediante telnet o FTP).
- En algún momento, aunque no para estos fines, puede que quiera crear un sistema que garantice nombres de usuarios únicos. El proceso para hacer esto es bastante sencillo: antes de intentar crear el directorio, utilice PHP para comprobar si en la lista de nombres de usuarios ya existe alguno con el mismo nombre que se acaba de registrar. Si no se encuentra ninguna coincidencia, el nombre nuevo es aceptable. Si el nombre de usuario ya está en uso, PHP crea un mensaje de error pidiendo un nuevo nombre de usuario.

Script 10.7. La simplicidad de este script también debería ser indicio de la naturalidad con que PHP manipula los formularios HTML. Hay tres pasos para cargar archivos: cambiar el formulario HTML en consecuencia, utilizar copy() para mover el archivo a la ubicación adecuada y después eliminar el archivo utilizando unlink().

script

```
1  <HTML>
2  <HEAD>
3  <TITLE>Handling File Uploads</TITLE>
4  </HEAD>
5  <BODY>
6  <?php
7  /* La siguiente condición determina si el
   formulario se manipula o no, dependiendo
   de si $File existe o no. */
8  if ($File) {
9      print ("File name: $File_name<P>\n");
10     print ("File size: $File_size<P>\n");
11     if (copy ($File, "users/$File_name")) {
12         print ("Your file was successfully
           uploaded!<P>\n");
13     } else {
14         print ("Your file could not be
           copied.<P>\n");
15     }
16     unlink ($File);
17 }
18
19 print ("Upload a file to the server:\n");
20 print ("<FORM ACTION=\"FileUpload.php\"
   METHOD=POST ENCTYPE=\"multipart/form-
   data\">\n");
21 print ("File <INPUT TYPE=FILE
   NAME=\"File\" SIZE=20><BR>\n");
22 print ("<INPUT TYPE=SUBMIT NAME=
   \"SUBMIT\" VALUE=\"Submit!\"></FORM>\n");
23 ?>
24 </BODY>
25 </HTML>
```

Manipulación de cargas de archivos

La manipulación de formularios HTML utilizando PHP es una hazaña extraordinariamente sencilla. Sin tener en cuenta los datos enviados, PHP puede procesarlos fácil y directamente. Esto también es así cuando el usuario carga un archivo mediante un formulario HTML.

Para dar al usuario la opción de cargar un archivo, se deben hacer dos cambios en el formulario HTML estándar. Primero, la línea inicial FORM debe incluir el código ENCTYPE = "multipart/form-data", que permite a HTML saber que espera un archivo, además de otros datos. Segundo, el elemento <INPUT TYPE=FILE NAME=NAME> se utiliza para crear el campo necesario.

En los formularios, el elemento INPUT TYPE=FILE permite que el usuario especifique un archivo almacenado en la computadora del usuario que se cargará en el servidor. Una vez que ha ocurrido esto, puede utilizar PHP para tratar el archivo.

Cuando se carga un archivo, el servidor lo coloca en un directorio temporal. Su primera responsabilidad es guardar el archivo en una ubicación permanente. La función copy() se emplea para duplicar el archivo en una nueva ubicación.

copy ("NombreOrigen", "NombreDestino");

Entonces, debería eliminar el archivo temporal, como una cuestión formal, utilizando unlink(). Escribirá un *script* muy básico que cargará un archivo y lo almacenará en el directorio users. Como el *script* urls.php, este ejemplo también creará y procesará el formulario HTML, todo en una página.

Cómo utilizar PHP para cargar archivos

1. Cree un nuevo documento PHP en el editor de textos.

2. Comience con la cabecera HTML estándar (véase el *Script* 10.7):

```
<HTML><HEAD><TITLE>Handling File
→ Uploads</TITLE></HEAD><BODY>
```

3. Abra la sección PHP y cree una condición que pregunte si manipular o no la carga del archivo.

```
<?php
if ($File) {
```

La información del archivo cargado se almacenará en una variable apropiadamente llamada $File. Si esta variable existe (es decir, tiene un valor), la procesará.

4. Imprima el nombre y el tamaño del archivo, a modo de confirmación.

```
print ("File name: $File_name<P>\n");
print ("File size: $File_size<P>\n");
```

Cuando se carga un archivo, se configuran algunas variables. Tomando el nombre de la variable principal (en este caso, $File) y añadiendo el nombre y el tamaño, también puede recuperar esa información para el archivo.

5. Intente copiar el archivo en el directorio users e imprima un mensaje basándose en el éxito de la condición.

```
if (copy ($File, "users/
→ $File_name")) {
print ("Your file was successfully
→ uploaded!<P>\n");
} else {
print ("Your file could not be
→ copied.<P>\n");
}
```

Al copiar el archivo, el origen se almacena en la variable $File y el destino es o bien absoluto (por ejemplo, c:/php/data.txt) o relativo a la localización de este archivo (por ejemplo, "php/data.txt"). Aquí hemos utilizado una referencia relativa, que indica poner el archivo en el directorio users que está localizado en el mismo directorio que esta página.

6. Ahora, elimine el archivo y cierre la condición.

```
    unlink ($File);
}
```

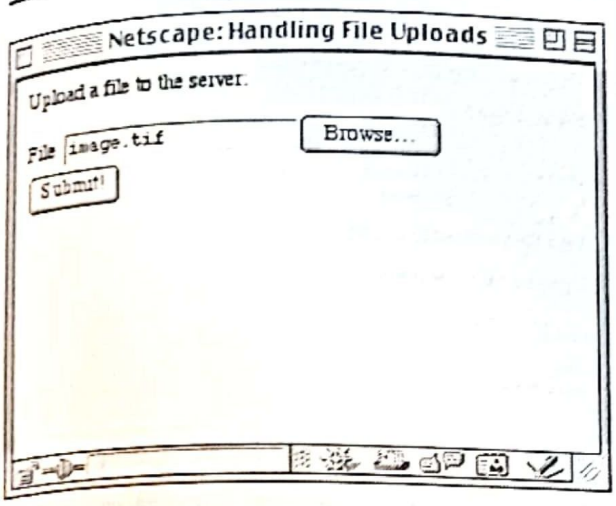


Figura 10.13. El tipo de entrada FILE crea un botón que, una vez que se ha hecho clic en él, permite al usuario navegar por su computadora hasta localizar el archivo deseado. Seleccionado el archivo, su nombre se mostrará automáticamente en la ventana.

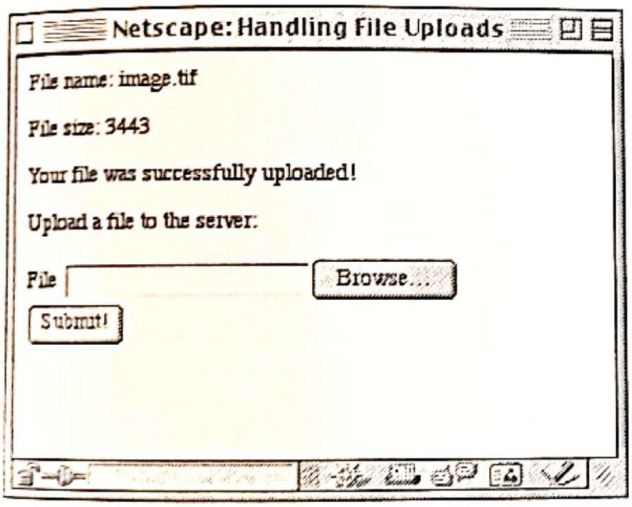


Figura 10.14. La manipulación satisfactoria del archivo se refleja con el mensaje correspondiente. También puede acceder al mismo tiempo al nombre y tamaño en bytes del archivo.

Lo más probable es que el archivo sea automáticamente eliminado por el servidor en algún momento posterior, pero es mejor tener las cosas ordenadas mientras siga, por lo que elimine el archivo inmediatamente una vez que haya terminado con él.

7. Edite el formulario HTML para cargar el archivo.

```
print ("Upload a file to the
→ server:\n");
print ("<FORM ACTION=
→ \"FileUpload.php\" METHOD=POST
→ ENCTYPE=\"multipart/form-
→ data\">\n");
print ("File <INPUT TYPE=FILE
→ NAME=\"File\"><BR>\n");
print ("<INPUT TYPE=SUBMIT NAME=
→ \"SUBMIT\" VALUE=\"Submit!\">
→ </FORM>\n");
```

Asegúrese de añadir el código ENCTYPE a la etiqueta de apertura FORM. La línea INPUT TYPE=FILE es manifiesta.

8. Cierre PHP y HTML.

```
?></BODY></HTML>
```

9. Guarde el archivo como FileUpload.php, cárguelo en el servidor (en el mismo directorio que el directorio users) y pruébelo en el navegador web (véanse las Figuras 10.13 y 10.14).

CONSEJOS

- Hemos decidido utilizar el directorio users creado anteriormente porque sabemos que sus permisos hacen que todos puedan escribir en él. Si intenta copiar un archivo en un directorio que no tenga el conveniente permiso de escritura, los resultados serían como los que se muestran en la Figura 10.15.

- El tamaño máximo de un archivo que se cargue está determinado por varios factores. Primero, el servidor puede tener un límite configurado. Segundo, PHP por sí mismo puede imponer un límite (en el archivo de configuración php.ini). Tercero, puede determinar un tamaño de archivo máximo codificando <INPUT TYPE=HIDDEN NAME="MAX_FILE_SIZE" VALUE="2048"> en el formulario antes de la entrada FILE.

  El campo value se expresa en bytes.

- Un tipo de archivo, que se asigna a la variable $File_type cuando se carga $File, es lo mismo que el tipo MIME (*Multipurpose Internet Mail Extensions*), que utilizan las aplicaciones de correo electrónico y los navegadores web para saber cómo tratar un archivo. Un tipo MIME podría ser image/jpeg o text/html.

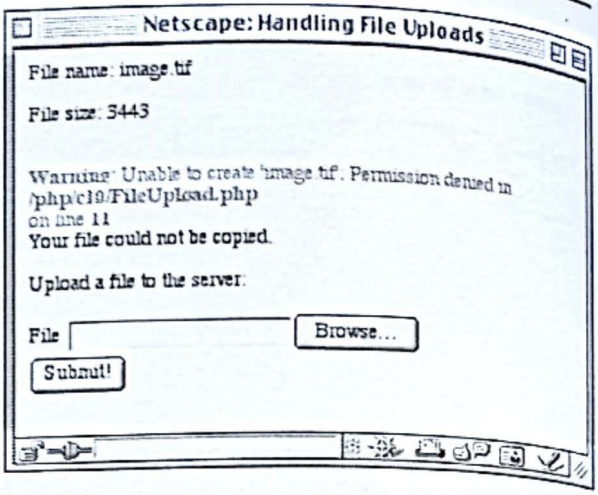


Figura 10.15. *El intento de copiar el archivo en una ubicación que no tiene los permisos adecuados genera este mensaje de error. Sabrá que el archivo se ha actualizado satisfactoriamente cuando se visualicen el nombre y el tamaño del archivo. El código PHP apenas copia el archivo al destino sugerido.*

Renombrar y borrar archivos y directorios

Hay unas cuantas funciones útiles relacionadas con los archivos y los directorios integradas en PHP que trataremos brevemente. Incluyen renombrar y borrar archivos, así como enumerar archivos localizados en un directorio. Primero revisaremos la sintaxis de algunas de ellas y después mostraremos su funcionamiento dentro del contexto de un *script* PHP muy útil.

Primero, está la función rename(), que funciona como esperaría que lo hiciera:

```
rename ("nombre_antiguo", "nombre_nuevo");
```

Se puede utilizar tanto con archivos como con directorios.

Otra función que utilizaremos en esta sección es filesize(), que determinará el tamaño de un archivo en bits. Este valor se puede asignar a una variable o imprimir.

```
$Number = filesize ("NombreArchivo");
```

Un concepto importante que mostraremos en el siguiente *script* implica el listado de todos los archivos que se encuentran en un directorio. Tal *script* le permite ver directorios sin acceso FTP y se puede utilizar para crear un archivo de documentos *online*. Leer un directorio es bastante parecido a leer desde un archivo. Primero abra el directorio, después vea los archivos uno a uno y, por último, cierre el directorio. En lugar de utilizar un puntero de archivo, como se hacía al leer archivos, se utiliza un manejador de directorio, aunque el concepto es el mismo.

```
$Handle = opendir ("ruta");
readdir ($Handle);
closedir ($Handle);
```

Como la función readdir() identifica un archivo cada vez, necesitará situarla en un bucle, así:

```
while (readdir ($Handle)) {
    instrucciones;
}
```

Pondrá todas estas capacidades juntas en una página que constituirá un panel de control basado en la Web para visualizar y administrar directorios.

Cómo crear el panel de control de directorios

1. Abra el editor de textos e inicie un nuevo documento PHP.

2. Cree la cabecera HTML estándar (véase el *Script* 10.8).

```
<HTML><HEAD><TITLE>Viewing Files in
→ a Directory</TITLE></HEAD><BODY>
```

3. Inicie una tabla y después comience la sección PHP.

```
<TABLE BORDER=0 WIDTH="60%"
→ CELLSPACING=2 CELLPADDING=2
→ ALIGN=CENTER>
<?php
```

En interés de crear una página más atractiva y ordenada, los datos de guardarán en una tabla.

4. Escriba una serie de condiciones que pregunten si se realizarán o no ciertas acciones basándose en la existencia de ciertas variables: $Upload (cargar archivos), $Rename (renombrar archivos) y $Delete (borrar archivos).

```
if ($Upload) {
print ("<TR><TD COLSPAN=4
→ ALIGN=CENTER>Uploaded file name:
→ $File_name</TD></TR>\n");
print ("<TR><TD COLSPAN=4
→ ALIGN=CENTER>Uploaded file size:
→ $File_size</TD></TR>\n");
if (copy ($File,
→ "users/$File_name")) {
```

Script 10.8. En lugar de emplear funciones, esta página utiliza condiciones, pero los resultados son comparables. Para probar sus conocimientos, quizá debería reescribir estas funciones de uso, especialmente la sección que muestra los archivos de un directorio.

script

```
1  <HTML>
2  <HEAD>
3  <TITLE>Viewing Files in a Directory</TITLE>
4  </HEAD>
5  <BODY>
6  <TABLE BORDER=0 WIDTH="60%"
   CELLSPACING=2 CELLPADDING=2
   ALIGN=CENTER>
7  <?php
8  /* Este archivo lista toda la información
   para archivos en un directorio y permite
   al usuario eliminar, cargar y renombrar
   archivos. */
9
10 if ($Upload) { // Manipula la carga
   de archivos.
11     print ("<TR><TD COLSPAN=4
       ALIGN=CENTER>Uploaded file name:
       $File_name</TD></TR>\n");
12     print ("<TR><TD COLSPAN=4
       ALIGN=CENTER>Uploaded file size:
       $File_size</TD></TR>\n");
13     if (copy ($File, "users/$File_name')) {
14         print ("<TR><TD COLSPAN=4
           ALIGN=CENTER>Your file,
           $File_name, was successfully
           uploaded!</TD></TR>\n");
15     } else {
16         print ("<TR><TD COLSPAN=4
           ALIGN=CENTER>Your file,
           $File_name, could not be
           copied.</TD></TR>\n");
17     }
18     unlink ($File);
19     print ("<TR><TD COLSPAN=4
       ALIGN=CENTER> </TD></TR>\n");
20 }
21
22 if ($Delete) { // Manipula la eliminación
   de archivos.
23     for ($i = 0; $i < count ($Delete);
       $i++) {
24         if ( unlink ("users/$Delete
           [$i]") ) {
```

Script 10.8. Continuación

```
script
25              print ("<TR><TD COLSPAN=4
                ALIGN=CENTER>Your file,
                $Delete[$i], was successfully
                deleted!</TD></TR>\n");
26          } else {
27              print ("<TR><TD COLSPAN=4
                ALIGN=CENTER>Your file,
                $Delete[$i], could not be
                deleted.</TD></TR>\n");
28          }
29      }
30      print ("<TR><TD COLSPAN=4
        ALIGN=CENTER> </TD></TR>\n");
31  }
32
33  if ($Rename) { // Manipula el renombrado
    de archivos.
34      for ($n = 0; $n < count ($Rename);
        $n++) {
35          $OldFilename = $Rename[$n];
36          $Old = "users/$OldFilename";
37          $New = "users/$NewName
            [$OldFilename]";
38          if ( rename ($Old, $New) ) {
39              print ("<TR><TD COLSPAN=4
                ALIGN=CENTER>Your file,
                $Rename[$n], was successfully
                renamed!</TD></TR>\n");
40          } else {
41  print ("<TR><TD COLSPAN=4 ALIGN=CENTER>
    Your file, $Rename[$n], could not be
    renamed.</TD></TR>\n");
42          }
43      }
44      print ("<TR><TD COLSPAN=4
        ALIGN=CENTER> </TD></TR>\n");
45  }
46
47  // Inicio del formulario.
48  print ("<FORM ACTION=\"files.php\"
    METHOD=POST ENCTYPE=\"multipart/form-
    data\">\n");
49  print ("<TR><TD><B>File Name</B></TD>
    <TD><B>File Size</B></TD>
    <TD><B>Delete</B></TD>
    <TD><B>Rename</B> (Enter the New Name
    in the Box)</TD></TR>\n");
50
```

```
print ("<TR><TD COLSPAN=4
→ ALIGN=CENTER>Your file, $File_name,
→ was successfully uploaded!
→ </TD></TR>\n");
} else {
print ("<TR><TD COLSPAN=4
→ ALIGN=CENTER>Your file, $File_name,
→ could not be copied.
→ </TD></TR>\n");
}
unlink ($File);
print ("<TR><TD COLSPAN=4
→ ALIGN=CENTER> </TD></TR>\n");
}
```

La variable $Upload se configurará si el usuario cargó un archivo. De aquí en adelante, si $Upload existe, el archivo cargado se manipulará como se ha visto anteriormente.

La última sentencia print, que creará una fila de tabla en blanco, está ahí sólo con fines estéticos, un tema que también continúa con las dos condiciones siguientes. El espacio adicional que crea esa línea del código hará la página web más atractiva.

5.
```
if ($Delete) {
for ($i = 0; $i < count ($Delete);
→ $i++) {
if ( unlink ("users/$Delete[$i]") ) {
print ("<TR><TD COLSPAN=4
→ ALIGN=CENTER>Your file, $Delete[$i],
→ was successfully deleted!</TD>
→ </TR>\n");
} else {
print ("<TR><TD COLSPAN=4
→ ALIGN=CENTER>Your file, $Delete[$i],
→ could not be deleted.</TD>
→ </TR>\n");
}
}
print ("<TR><TD COLSPAN=4
→ ALIGN=CENTER> </TD></TR>\n");
}
```

Utilizará una variable que identifique si los archivos necesitan o no borrarse, como hizo con los archivos cargados. Como se pueden elimi-

nar múltiples archivos, se ha creado un array y el bucle procesará cada elemento del mismo eliminándolos por turno.

6.
```
if ($Rename) {
for ($n = 0; $n < count ($Rename);
→ $n++) {
$OldFilename = $Rename[$n];
$Old = "users/$OldFilename";
$New = "users/$NewName
→ [$OldFilename]";
if ( rename ($Old, $New) ) {
print ("<TR><TD COLSPAN=4
→ ALIGN=CENTER>Your file, $Rename[$n],
→ was successfully renamed!</TD>
→ </TR>\n");
} else {
print ("<TR><TD COLSPAN=4
→ ALIGN=CENTER>Your file, $Rename[$n],
→ could not be renamed.</TD>
→ </TR>\n");
}
}
print ("<TR><TD COLSPAN=4
→ ALIGN=CENTER> </TD></TR>\n");
}
```

Lo mismo que hemos explicado para renombrar archivos se puede aplicar a la eliminación de archivos, respecto a recorrer los elementos del array. Una vez que ha asignado al archivo los nombres nuevo y anterior, una llamada a rename() manipulará las alteraciones.

7. Cree el formulario HTML, asegurándose de incluir el código ENCTYPE que permite que se cargue el archivo.

```
print ("<FORM ACTION=\"files.php\"
→ METHOD=POST ENCTYPE=\"multipart/
→ form-data\">\n");
```

8. Imprima las cabeceras de la tabla.

```
print ("<TR><TD><B>File Name</B>
→ </TD><TD><B>File Size</B>
→ </TD><TD><B>Delete</B></TD><TD>
→ <B>Rename</B> (Enter the New Name
→ in the Box)</TD></TR>\n");
```

Script 10.8. Continuación

script

```
51 // Lee los archivos del directorio.
52 $Open = opendir ("users");
53 while ($Files = readdir ($Open)) {
54     $Filename = "users/" . $Files;
55     if (is_file ($Filename)) {
56         $Size = filesize ("users/$Files");
57         print ("<TR><TD>$Files</TD>
           <TD>$Size</TD><TD><INPUT
           TYPE=CHECKBOX NAME=\"Delete[]\"
           VALUE=\"$Files\"></TD><TD><INPUT
           TYPE=CHECKBOX NAME=\"Rename[]\"
           VALUE=\"$Files\"><INPUT TYPE=TEXT
           NAME=\"NewName[$Files]\"></TD>
           </TR>\n");
58     }
59 }
60 closedir ($Open);
61
62 // Proporciona la opción de carga.
63 print ("<TR><TD COLSPAN=4
   ALIGN=CENTER> </TD></TR>\n");
64 print ("<TR><TD COLSPAN=4
   ALIGN=CENTER><INPUT TYPE=CHECKBOX
   NAME=\"Upload\" VALUE=\"Yes\">Upload a
   file to the server:<INPUT TYPE=FILE
   NAME=\"File\" SIZE=20></TD></TR>\n");
65 print ("<TR><TD COLSPAN=4
   ALIGN=CENTER><INPUT TYPE=SUBMIT
   NAME=\"SUBMIT\" VALUE=\"Submit!\">
   </FORM></TD></TR>\n");
66 ?>
67 </TABLE>
68 </BODY>
69 </HTML>
```

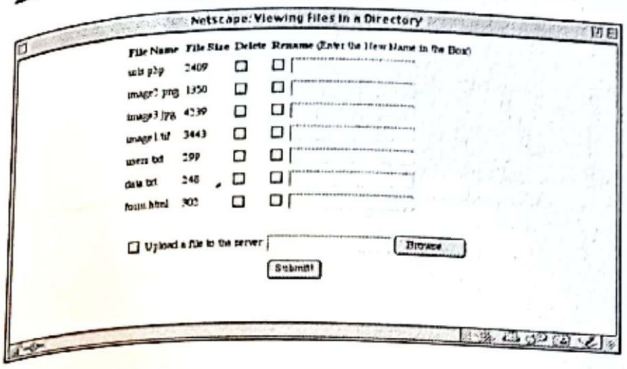
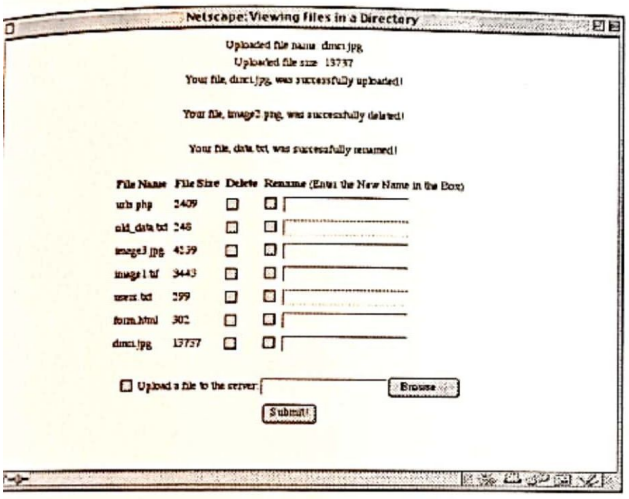


Figura 10.16. Al llegar por primera vez a la página, esto es lo que el usuario ve. Se visualizan unas casillas de verificación que permiten al usuario borrar, renombrar y cargar archivos.



gura 10.17. Aquí hemos cargado un archivo nuevo, rrado otro y renombrado un tercero. Se informa de da operación satisfactoria y se visualiza una lista tualizada con el contenido del directorio.

9. Escriba el código para leer desde el directorio.

```
$Open = opendir ("users");
while ($Files = readdir ($Open)) {
$Filename = "users/" . $Files;
if (is_file ($Filename)) {
$Size = filesize ("users/$Files");
print ("<TR><TD>$Files</TD>
→ <TD>$Size</TD><TD><INPUT
→ TYPE=CHECKBOX NAME=\"Delete[]\"
→ VALUE=\"$Files\"></TD><TD><INPUT
→ TYPE=CHECKBOX NAME=\"Rename[]\"
→ VALUE=\"$Files\"><INPUT TYPE=TEXT
→ NAME=\"NewName[$Files]\"></TD>
→ </TR>\n");
}
}
closedir ($Open);
```

Aquí tiene cómo hemos efectuado el bucle por cada archivo (y directorio) localizado en el directorio users. Para nuestros propósitos, sólo queremos tratar con archivos, no con directorios, de forma que empleamos la función is_file() para asegurarnos de que sólo se listarán los archivos, evitando los directorios.

10. Cree la opción de carga en el formulario.

```
print ("<TR><TD COLSPAN=4
→ ALIGN=CENTER> </TD></TR>\n");
print ("<TR><TD COLSPAN=4
→ ALIGN=CENTER><INPUT TYPE=CHECKBOX
→ NAME=\"Upload\" VALUE=\"Yes\">
→ Upload a file to the server:
→ <INPUT TYPE=FILE NAME=\"File\"
→ SIZE=20></TD></TR>\n");
print ("<TR><TD COLSPAN=4
→ ALIGN=CENTER><INPUT TYPE=SUBMIT
→ NAME=\"SUBMIT\"
VALUE=\"Submit!\">
→ </FORM></TD></TR>\n");
```

11. Cierre PHP y HTML.

```
?></TABLE></BODY></HTML>
```

12. Guarde el *script* como files.php, cárguelo en el servidor (en la misma localización que el directorio users) y pruébelo en el navegador web (véanse las Figuras 10.16 y 10.17).

CONSEJOS

- Si quisiera, utilizando las páginas NewUser.html y HandleNewUser.php, podría copiar files.php (con algunas modificaciones) en cada nuevo directorio de usuario. De esta forma, cuando los usuarios se conectaran, podrían ser enviados a sus paneles de control donde podrían manipular los archivos almacenados.
- Para informarse sobre otras funciones de archivos y directorios, consulte en el manual de PHP la sección "Directories and Filesystem".

BASES DE DATOS 11

Puede parecer extraño decir que Internet no sería lo que es actualmente si no fuera por la existencia de las bases de datos. De hecho, el propio PHP probablemente no sería tan popular o útil si no fuera por su soporte integrado de numerosos tipos de bases de datos. (Como regla general, el soporte de bases de datos por parte de PHP y su facilidad de uso son cosas que no encontrará utilizando *scripts* CGI.)

Una base de datos es una colección de tablas (una tabla está a su vez compuesta por filas y columnas) que almacenan información. Las bases de datos se utilizan por todo Internet. Los sitios de comercio electrónico utilizan bases de datos para almacenar las especificaciones de los productos (como su precio y color, por ejemplo), así como los datos de los clientes, mientras que los sitios de contenido colocan artículos y noticias en las bases de datos.

Existen actualmente muchos servidores de bases de datos o Sistemas de administración de bases de datos (DBMS, *Database Management Systems*), que funcionan en distintas plataformas. (Técnicamente, un DBMS es el software que interactúa con la base de datos apropiada. Sin embargo, cada vez más y más personas utilizan los términos de **base de datos** y **DBMS** como sinónimos.) En cualquier sistema operativo, Oracle está considerado como el mejor DBMS, pero su precio lo coloca fuera del alcance de la mayoría y queda para las aplicaciones más grandes y con buena financiación. Para Windows y Windows NT se suele recurrir a Access o SQL Server, ambos sistemas muy útiles pero incompatibles con la plataforma cruzada.

En este capítulo utilizaremos MySQL (véase la Figura 11.1) como ejemplo de DBMS. Aunque MySQL, que está disponible para la mayoría de las plataformas, no es tan potente como otros servidores de bases de datos, tiene la suficiente velocidad y funcionalidad para la mayoría de los propósitos y su precio (gratuito para los servidores Unix) lo convierte en una de las opciones más comunes para el desarrollo web. Si está utilizando un servidor proporcionado por un ISP o un *host* web, verifique qué DBMS tiene disponible para su uso (seguramente con un coste adicional). Si está ejecutando su propio servidor, debería considerar la instalación de MySQL (www.MySQL.com u otro servidor de bases de datos).

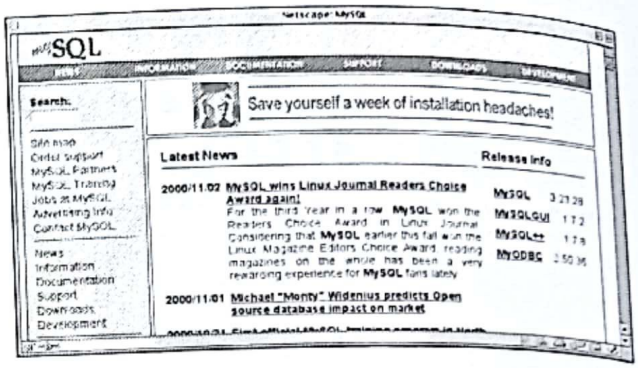


Figura 11.1. Puede obtener MySQL en la página principal de este DBMS (www.MySQL.com). En esta dirección también encontrará documentación dedicada a la instalación y uso de MySQL, así como el importe de las licencias aplicables.

Las bases de datos se crean, actualizan y leen utilizando SQL (*Structured Query Language*, Lenguaje de consultas estructurado). Sorprendentemente hay pocos comandos en SQL, y algunos son una bendición y otros una maldición. SQL fue diseñado para que su escritura se pareciese al inglés, lo que lo hace muy amigable para los que conocen este idioma, pero es necesaria alguna reflexión para crear instrucciones SQL más elaboradas con sólo los comandos disponibles. Debe ser consciente de que al incorporar una base de datos a las aplicaciones web incrementa el número potencial de errores, así que compruebe y vuelva a comprobar que la base de datos funciona como espera.

Le conduciremos por el proceso de desarrollo de una base de datos sencilla que registra la retroalimentación del usuario. Aunque en este capítulo aprenderá lo suficiente para empezar, puede que quiera consultar el Apéndice C, una vez finalizado este capítulo, donde encontrará algunas referencias que le servirán para aprender más sobre este tema.

Puesto que utilizaremos MySQL, todas las funciones que utilicemos en este capítulo serán específicas de MySQL. Por ejemplo, para conectar a una base de datos en MySQL, la función adecuada es mysql_connect(), pero si utiliza PostgreSQL, debería utilizar pg_connect(). Si no está utilizando el DBMS MySQL, deberá utilizar el manual de PHP (disponible en www.PHP.net) para encontrar los nombres de las funciones adecuadas.

Conexión y creación de una base de datos

Cuando estuvo trabajando con archivos de texto en el Capítulo 10, vio que primero tenía que crear un puntero al archivo mientras lo abría. Este puntero actuaba entonces como punto de referencia al archivo abierto. Con las bases de datos se recurre a un proceso similar. En primer lugar, tiene que establecer una conexión con el servidor de bases de datos (en este caso, MySQL). Esta conexión se utilizará entonces como punto de acceso para los comandos futuros. La sintaxis de MySQL para la conexión a una base de datos es la siguiente:

```
$Link = mysql_connect ("host",
→ "usuario", "contraseña");
```

El enlace se establece con tres argumentos: el *host*, que casi siempre será localhost, el nombre del usuario, y la contraseña para este nombre de usuario. Estos dos últimos parámetros dictarán los permisos que tiene sobre la base de datos.

Estos permisos son un poco más complicados que los de los archivos, pero es necesario que los comprenda: a distintos tipos de usuarios se les asignan diferentes capacidades de bases de datos. Por ejemplo, un administrador de DBMS puede crear bases de datos nuevas y borrar las antiguas (dentro de su DBMS puede haber docenas de bases de datos), pero un administrador de nivel inferior sólo tiene la posibilidad de crear y modificar tablas dentro de una sola base de datos. El usuario más básico sólo puede leer, pero no modificar, las tablas.

Su ISP le otorgará seguramente el segundo tipo de acceso (control sobre una sola base de datos, pero no sobre el propio DBMS) y establecerá su base de datos inicial. Si está trabajando en su propio servidor o tiene acceso de administrador, tiene la capacidad de crear bases de datos nuevas.

El código para crear una base de datos nueva es:

```
mysql_create_db ("nombre_base_datos",
→ $Link);
```

Observe cómo el valor $Link establecido al conectar con la base de datos se utiliza para continuar trabajando con ella, al igual que nos referimos a un puntero de archivo para leer o escribir en él.

Después de ejecutar cualquier comando contra la base de datos, se considera buena idea cerrar el enlace, del mismo modo que cierra un archivo abierto:

```
mysql_close ($Link);
```

En el primer ejemplo de base de datos de este capítulo, creará una base de datos nueva, para lo que tiene que disponer de acceso como administrador. Si su ISP le restringe el acceso, debería pedirle a su ISP que cree una base de datos inicial, con lo que podría saltar a la siguiente sección, "Creación de una tabla".

Cómo conectar a MySQL y crear una base de datos nueva

1. Cree un documento PHP nuevo en el editor de textos.
2. Comience con la cabecera HTML estándar (véase el *Script* 11.1).

   ```
   <HTML><HEAD><TITLE>Creating a
   → Database</TITLE><BODY>
   ```

3. Cree la sección PHP del *script* y establezca las variables de bases de datos.

   ```
   <?php
   $Host = "localhost";
   $User = "user";
   $Password = "password";
   $DBName = "NewDatabase";
   ```

   Asignando estos valores a variables, puede alterar más fácilmente el *script* para que funcione con otras bases de datos en el futuro. Si está trabajando en el servidor del *host* web o del ISP, deberían proporcionarle el nombre de usuario y la contraseña.

   Cuando cree el nombre de una base de datos, no utilice espacios en blanco, por la misma razón que no los utiliza en los nombres de las funciones y las variables.

Script 11.1. *La creación de una base de datos consta de tres pasos: enlace con la base de datos, uso de la función mysql_create_db() y cierre del enlace. Le instamos a que adquiera el hábito de asignar todas las dependencias de la base de datos (host, nombre de usuario, contraseña y nombre de la base de datos) a variables, para que en el futuro le resulten más sencillas las modificaciones.*

script

```
1  <HTML>
2  <HEAD>
3  <TITLE>Creating a Database</TITLE>
4  <BODY>
5  <?php
6  // Establece variables para el acceso a la
   base de datos:
7  $Host = "localhost";
8  $User = "user";
9  $Password = "password";
10 $DBName = "NewDatabase";
11
12 $Link = mysql_connect ($Host, $User,
   $Password);
13 if (mysql_create_db ($DBName, $Link)) {
14     print ("The database, $DBName, was
       successfully created!<BR>\n");
15 } else {
16     print ("The database, $DBName, could
       not be created!<BR>\n");
17 }
18 mysql_close ($Link);
19 ?>
20 </BODY>
21 </HTML>
```

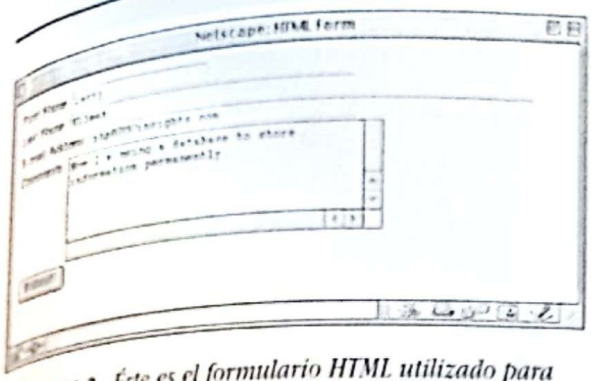


Figura 11.2. Éste es el formulario HTML utilizado para introducir datos en la base de datos. La entrada que aquí se se almacenará en la tabla Feedback de la base de datos NewDatabase.

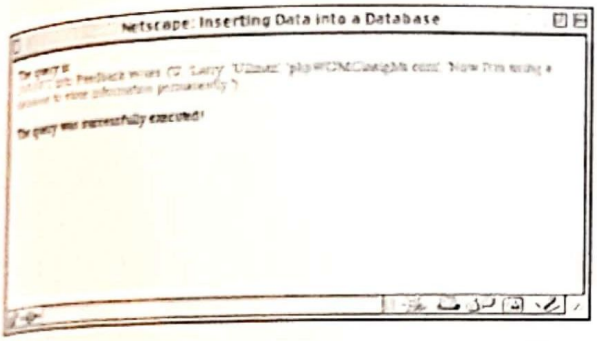


Figura 11.3. Observe cómo PHP genera con el apóstrofe una secuencia de escape en el comentario para que no provoque problemas cuando se envíe la consulta a la base de datos. Sin esta secuencia de escape, SQL pensaría que la última columna consistiría únicamente en la cadena 'Now I', lo que provocaría un error.

4. Conecte con la base de datos.

```
$Link = mysql_connect ($Host, $User,
 ‣ $Password);
```

Este *script* intentará conectar con el DBMS MySQL en el servidor, utilizando el nombre de *host*, nombre de usuario y contraseña. Si el nombre de usuario y la contraseña que introduce no coinciden con los privilegios de base de datos preestablecidos (por ejemplo, si está utilizando un nombre de usuario o contraseña erróneos), obtendrá un mensaje de error en este punto de la ejecución del *script*.

5. Intente crear la base de datos nueva e imprima un mensaje indicando el resultado del intento.

```
if (mysql_create_db ($DBName, $Link)) {
    print ("The database, $DBName,
     ‣ was successfully
     ‣ created!<BR>\n");
} else {
    print ("The database, $DBName,
     ‣ could not be created!<BR>\n");
}
```

Si el *script* puede crear con éxito la base de datos, verá un mensaje indicándoselo (véase la Figura 11.2). Si, por alguna razón, es imposible crear la base de datos, verá varios errores de MySQL, así como el error "could not be created!" generado por esta condición (véase la Figura 11.3).

6. Cierre el enlace MySQL, después cierre la sección PHP y finalmente HTML.

```
mysql_close ($Link);
?></BODY></HTML>
```

No es estrictamente necesario cerrar el enlace MySQL, pues lo hará automáticamente el servidor una vez que el *script* PHP deje de ejecutarse, pero le recomendamos que siempre incluya este paso, como cuestión de forma y eficiencia.

7. Guarde el *script* como CreateDB.php, cárguelo en el servidor y pruébelo en el navegador.

CONSEJOS

- PHP incluye soporte para la mayoría de bases de datos: dBase, FilePro, mSQL, MySQL, Oracle, PostgreSQL y Sybase. Si utiliza un tipo de base de datos para la que no tiene soporte directo (por ejemplo, Access o SQL Server), deberá utilizar las funciones ODBC (*Open Database Connectivity*, Conectividad de base de datos abierta) de PHP, junto con los controladores ODBC de la base de datos para interactuar con dicha base de datos. Consulte la sección "Recursos de bases de datos" del Apéndice C si desea más información.

- La combinación del uso de PHP y MySQL es actualmente tan común que hay dos términos que debe conocer para identificar los servidores configurados con PHP y MySQL: LAMP (*Linux operating system, Apache web server, MySQL DBMS, PHP*) y WAMP (*Windows operating system, Apache Web server, MySQL DBMS, PHP*).

Creación de una tabla

Una vez creada la base de datos inicial, puede empezar a crear tablas individuales dentro de ella. Recuerde que una base de datos está constituida por múltiples tablas, cada una de las cuales tiene columnas y filas. Para este sencillo ejemplo creará una tabla en la que podrá almacenar datos.

Para crear una tabla dentro de una base de datos, utilizará SQL (el lenguaje que entienden las bases de datos). El proceso de consultar una base de datos involucra la escritura de una consulta SQL y después hacer una consulta a la base de datos utilizando:

```
mysql_db_query ():
$Query = "el texto de consulta va aquí
→ en formato SQL";
mysql_db_query ("NombreBaseDatos",
→ $Query, $Link);
```

Puesto que SQL es muy parecido al inglés, la consulta para crear una tabla nueva se lee como sigue:

```
$Query = "CREATE table NOMBRETABLA
→ (columna1, columna2, etc.)";
```

Para cada columna, separadas entre sí por comas, debe indicar su nombre y su tipo. Los tipos más comunes son TEXT (texto) e INT (entero). Puesto que es extremadamente recomendable crear una primera columna que actúe como clave primaria (una columna utilizada para referirse a cada fila), una simple consulta podría ser ésta:

```
$Query = "CREATE table TablaNueva (id
→ INT PRIMARY KEY, information TEXT)";
```

La clave primaria de una tabla es una columna especial de valores únicos que se utiliza para referirnos a las filas de la tabla. La base de datos crea un índice ordenado de esta columna para navegar más rápidamente a través de la tabla. Una tabla sólo puede tener una clave primaria que normalmente se configura como una columna de enteros autoincrementable. La primera fila tendrá la clave 1, la segunda la clave 2, y así sucesivamente. Al referirse a la clave podrá recuperar los valores de esa fila.

Puede visitar el sitio web de MySQL (véase la Figura 11.1) si desea más información sobre SQL. De todos modos, siguiendo nuestras indicaciones debería ser capaz de acometer algunas de las tareas básicas sobre una base de datos.

Para este ejemplo creará una tabla para almacenar la información enviada mediante un formulario HTML. En la siguiente sección del capítulo escribirá el *script* que almacenará los datos enviados en la tabla que va a crear en un momento.

Cómo crear una tabla nueva

1. Cree un documento PHP nuevo en el editor de textos.

2. Codifique la cabecera HTML estándar (véase el *Script* 11.2):

```
<HTML><HEAD><TITLE>Creating a
› Table</TITLE><BODY>
```

3. Abra la sección PHP del *script* y configure las variables de la base de datos.

```
<?php
$Host = "localhost";
$User = "user";
$Password = "password";
$DBName = "NewDatabase";
$TableName = "Feedback";
```

Observe que hemos añadido una variable más a la lista existente en el *Script* 11.1, $TableName. Nos ayudará a crear nombres de tabla relevantes, por lo que esta tabla, pensada para almacenar la retroalimentación del usuario, se llama Feedback. En MySQL, los nombres de las tablas, así como los de las columnas, hacen distinción entre minúsculas y mayúsculas.

4. Establezca un enlace con el servidor MySQL.

```
$Link = mysql_connect ($Host, $User,
› '$Password);
```

Script 11.2. *La creación de una tabla, y la mayoría de las otras tareas con las bases de datos, se manipulan mediante la escritura de la consulta adecuada y utilizando después la función mysql_db_query().*

script

```
1  <HTML>
2  <HEAD>
3  <TITLE>Creating a Table</TITLE>
4  <BODY>
5  <?php
6  // Establece variables para el acceso a la
   base de datos:
7  $Host = 'localhost';
8  $User = 'user';
9  $Password = 'password';
10 $DBName = 'NewDatabase';
11 $TableName = 'Feedback';
12
13 $Link = mysql_connect ($Host, $User,
   $Password);
14 $Query = "CREATE table $TableName (id INT
   UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY
   KEY, FirstName TEXT, LastName TEXT,
   EmailAddress TEXT, Comments TEXT)";
15 if (mysql_db_query ($DBName, $Query,
   $Link)) {
16     print ("The query was successfully
       executed!<BR>\n");
17 } else {
18     print ("The query could not be
       executed!<BR>\n");
19 }
20 mysql_close ($Link);
21 ?>
22 </BODY>
23 </HTML>
```

5. Escriba la consulta.

```
$Query = "CREATE table $TableName (id
→ INT UNSIGNED NOT NULL
→ AUTO_INCREMENT PRIMARY KEY,
→ FirstName TEXT, LastName TEXT,
→ EmailAddress TEXT, Comments TEXT)";
```

Hemos dividido la consulta en partes más reconocibles. En primer lugar, para crear una tabla nueva, escriba `CREATE table $TableName` (la variable $TableName será reemplazada por el valor de $TableName durante la ejecución. Después, dentro de los paréntesis, escriba una lista con las columnas que desea, separándolas entre sí mediante comas.

La primera columna de la tabla se llama id y será un entero sin signo (INT UNSIGNED, lo que significa que sólo podrá almacenar enteros positivos). Al incluir las palabras NOT NULL, está indicando que esta columna debe tener un valor para cada fila. Los valores se incrementarán automáticamente en una unidad para cada fila (AUTO_INCREMENT); esta columna será la clave primaria.

Las siguientes cuatro columnas serán de tipo TEXT (texto), la primera llamada FirstName, la segunda LastName, la tercera EmailAddress y la cuarta Comments.

6. Consulte la base de datos e imprima un mensaje indicando el éxito de la consulta.

```
if (mysql_db_query ($DBName, $Query,
→ '$Link)) {
    print ("The query was
    → successfully executed!<BR>\n");
} else {
    print ("The query could not be
    → executed!<BR>\n");
}
```

Si la consulta se ha creado satisfactoriamente (en cuyo caso la condición es TRUE), verá únicamente el mensaje "successfully executed" (véase la Figura 11.4). Si la tabla no se ha creado debido a un error SQL, el resultado se parecerá al que muestra la Figura 11.5. Sin embargo, si la tabla no puede crearse debido a un problema con los permisos de la base de datos, verá un error parecido al de la Figura 11.3.

7. Cierre el enlace MySQL, después cierre la sección PHP y, por último, cierre HTML.

```
mysql_close ($Link);
?></BODY></HTML>
```

8. Guarde el *script* como CreateTable.php, cárguelo en el servidor y pruébelo en el navegador.

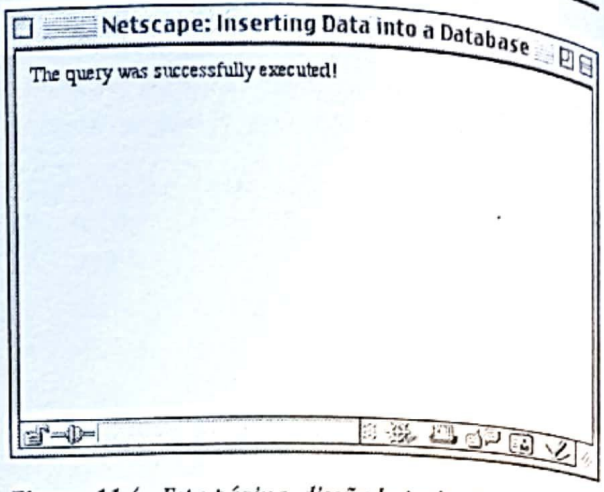


Figura 11.4. *Esta página, diseñada teniendo en mente al usuario, no le ayudará en el desarrollo de su sitio. Sin embargo, si consulta el código fuente de la página podrá ver cuál es la consulta SQL exacta (véase la Figura 11.5).*

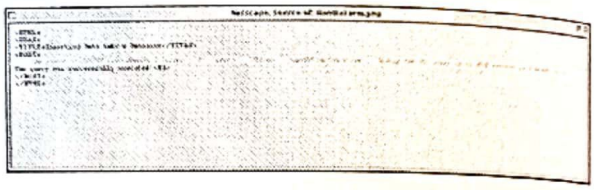


Figura 11.5. *Durante el desarrollo de un sitio web puede imprimir consultas SQL como un comentario HTML para comprobar doblemente la exactitud de lo que el código hace.*

CONSEJO

- No es necesario que escriba las consultas SQL parcialmente en mayúsculas como en los ejemplos que aquí está viendo, ya que tan sólo lo hacemos para que distinga mejor SQL de los nombres de la tabla y de las columnas. Los nombres de la tabla y de las columnas hacen distinción entre mayúsculas y minúsculas, mientras que SQL generalmente no la hace.

Script 11.3. En capítulos precedentes ya ha visto variantes de este formulario. Como con todos los formularios HTML, lo más importante es recordar los nombres del campo INPUT para que luego pueda referirse a ellos desde PHP.

script

```
1  <HTML>
2  <HEAD>
3  <TITLE>HTML Form</TITLE>
4  </HEAD>
5  <BODY>
6  <FORM ACTION="HandleForm.php" METHOD=POST>
7  First Name <INPUT TYPE=TEXT
   NAME="Array[FirstName]" SIZE=20><BR>
8  Last Name <INPUT TYPE=TEXT
   NAME="Array[LastName]" SIZE=40><BR>
9  E-mail Address <INPUT TYPE=TEXT
   NAME="Array[Email]" SIZE=60><BR>
10 Comments <TEXTAREA NAME="Array[Comments]"
   ROWS=5 COLS=40></TEXTAREA><BR>
11 <INPUT TYPE=SUBMIT NAME="SUBMIT"
   VALUE="Submit!">
12 </FORM>
13 </BODY>
14 </HTML>
```

Envío de datos a una base de datos

Como ya hemos mencionado, esta base de datos almacenará la retroalimentación del usuario en una tabla para su visualización en el futuro. Es un uso muy simple pero relevante de una base de datos. En la última sección creó la tabla necesaria, formada por cinco columnas: id, FirstName, LastName, EmailAddress y Comments. El proceso de añadir información a una tabla es similar a crear la tabla, en relación a las funciones utilizadas, pero la consulta SQL será diferente.

```
$Query = "INSERT into $TableName values
→ ('valor1', 'valor2', 'valor3', etc.)";
mysql_db_query ("nombre_base_datos",
→ $Query, '$Link);
```

La consulta empieza con INSERT into $TableName values. Después, dentro de los paréntesis, el valor de cada columna debe especificarse entre comillas simples, separando los distintos valores mediante comas. ¡Debe indicar tantos valores como columnas tiene la tabla, o la consulta no funcionará! Después, la consulta es enviada a MySQL utilizando mysql_db_query().

Para demostrar esto, utilizará un formulario HTML que toma el nombre, los apellidos, la dirección de correo electrónico y los comentarios del usuario. El *script* PHP manipula el formulario colocando la información enviada en la base de datos.

Cómo introducir datos en una base de datos desde un formulario HTML

1. Cree un documento HTML nuevo en el editor de textos para crear el formulario HTML.
2. Codifique la cabecera HTML estándar (véase el *Script* 11.3):

```
<HTML><HEAD><TITLE>HTML Form
→ </TITLE><BODY>
```

3. Cree un formulario.

```
<FORM ACTION="HandleForm.php"
→ METHOD=POST>
```

4. Codifique las cuatro entradas de texto.

```
First Name <INPUT TYPE=TEXT
→ NAME="Array[FirstName]"
→ SIZE=20><BR>
→ Last Name <INPUT TYPE=TEXT
→ NAME="Array[LastName]" SIZE=40><BR>
→ E-mail Address <INPUT TYPE=TEXT
→ NAME="Array[Email]" SIZE=60><BR>
→ Comments <TEXTAREA
→ NAME="Array[Comments]" ROWS=5
→ COLS=40></TEXTAREA><BR>
```

Puede hacer más atractivo el formulario, pero asegúrese de tomar nota de los nombres de las variables de entrada, pues los necesitará en la página HandleForm.php.

5. Añada un botón de envío, cierre el formulario y cierre la página HTML.

```
<INPUT TYPE=SUBMIT NAME="SUBMIT"
→ 'VALUE="Submit!">
</FORM>
</BODY>
</HTML>
```

6. Guarde la página como form.html y cárguela en el servidor.

Ahora escribirá la página HandleForm.php, que tomará los datos generados por el formulario y los almacenará en la base de datos.

7. Cree un documento PHP nuevo en el editor de textos.

8. Comience con la cabecera HTML estándar (véase el *Script* 11.4).

```
<HTML><HEAD><TITLE>Inserting Data
→ into a Database</TITLE><BODY>
```

Script 11.4. *La instrucción de consulta para añadir información a la base de datos es muy directa, pero asegúrese de que el número de valores entre paréntesis coincide con el número de columnas en la tabla de la base de datos.*

script

```
1  <HTML>
2  <HEAD>
3  <TITLE>Inserting Data into a Database
   </TITLE>
4  <BODY>
5  <?php
6  /* Esta página recibe y manipula los datos
   generados por "form.html". */
7  // Corta los datos entrantes.
8  $Array["FirstName"] = trim
   ($Array["FirstName"]);
9  $Array["LastName"] = trim
   ($Array["LastName"]);
10 $Array["Email"] = trim ($Array['Email']);
11 $Array["Comments"] = trim
   ($Array["Comments"]);
12
13 // Establece variables para el acceso a la
   base de datos:
14 $Host = "localhost";
15 $User = "user";
16 $Password = "password";
17 $DBName = "NewDatabase";
18 $TableName = "Feedback";
19
20 $Link = mysql_connect ($Host, $User,
   $Password);
21 $Query = "INSERT into $TableName values
   ('0', '$Array[FirstName]',
   '$Array[LastName]', '$Array[Email]',
   '$Array[Comments]')";
22 print ("The query is:<BR>$Query<P>\n");
23 if (mysql_db_query ($DBName, $Query,
   $Link)) {
24     print ("The query was successfully
       executed!<BR>\n");
25 } else {
26     print ("The query could not be
       executed!<BR>\n");
27 }
28 mysql_close ($Link);
29 ?>
30 </BODY>
31 </HTML>
```

9. Comience la sección PHP de la página y recorte todos los datos entrantes para eliminar los espacios sobrantes.

```
<?php
$Array["FirstName"] = trim
→ ($Array["FirstName"]);
$Array["LastName"] = trim
→ ($Array["LastName"]);
$Array["Email"] = trim
→ ($Array["Email"]);
$Array["Comments"] = trim
→ ($Array["Comments"]);
```

10. Establezca las variables para el acceso a la base de datos.

```
$Host = "localhost";
$User = "user";
$Password = "password";
$DBName = "NewDatabase";
$TableName = "Feedback";
```

11. Conecte con MySQL y después escriba la consulta.

```
$Link = mysql_connect ($Host, $User,
→ $Password);
$Query = "INSERT into $TableName
→ values ('0', '$Array[FirstName]',
→ '$Array[LastName]', '$Array[Email]',
→ '$Array[Comments]')";
```

La consulta empieza con el código INSERT into $TableName values, seguido de cinco valores (uno por cada columna y en el mismo orden) encerrados entre comillas simples y separados mediante comas. Puesto que la columna id se ha definido como AUTO_INCREMENT, puede utilizar el 0 como valor y MySQL lo convertirá automáticamente en el siguiente valor lógico de id.

12. Sólo para comprobar si hay errores, imprima la consulta en el navegador.

```
print ("The query is:<BR>$Query<P>\n");
```

Siempre que tenga dificultades con la interacción con la base de datos, compruebe primero que la consulta es lógica y correcta. Aunque no querrá que el usuario vea la consulta, puede utilizar esta técnica mientras practica y depura (consulte el consejo que viene a continuación).

13. Cree una estructura condicional basándose en el resultado de la consulta a la base de datos.

```
if (mysql_db_query ($DBName, $Query,
→ '$Link)) {
     print ("The query was
     → successfully executed!<BR>\n");
} else {
     print ("The query could not be
     → executed!<BR>\n");
}
```

14. Cierre el enlace MySQL, después cierre la sección PHP y, finalmente, HTML.

```
mysql_close ($Link);
?></BODY></HTML>
```

15. Guarde el *script* como HandleForm.php, cárguelo en el servidor (en el mismo directorio que form.html) y después pruebe ambas páginas en el navegador (véanse las Figuras 11.2 y 11.3).

CONSEJO

- Si quiere probar las instrucciones SQL, sin hacerlas visibles al usuario, añada esta línea de código al *script*: `print ("<!--The query is $Query-->\n");`. Ahora, el código PHP escribirá la consulta como un comentario HTML, haciéndola visible si se accede al código fuente de la página (véanse las Figuras 11.4 y 11.5).

Recuperación de datos de una base de datos

El último proceso que mostraremos será la recuperación de datos de una base de datos establecida. Aunque todavía seguirá utilizando la función mysql_db_query(), la recuperación de datos es ligeramente distinta de la inserción de datos, en el sentido de que tiene que asignar la información recuperada a una variable para poder usarla. Pasaremos por este paso en un momento.

La consulta más sencilla para leer de una tabla es la siguiente:

```
$Query = "SELECT * from $TableName";
```

El asterisco es equivalente a decir todo (por tanto, esta consulta quiere decir: "selecciona todo lo que contiene $TableName"). Esta instrucción de cuatro palabras será casi siempre suficiente para la recuperación de datos.

Sin embargo, puede, en nuestro ejemplo, limitar la consulta escribiendo `SELECT FirstName, Comments from $TableName`. Esta consulta solicita únicamente la información de dos columnas (FirstName y Comments).

Otra forma de restringir la consulta consiste en escribir algo parecido a esto: `SELECT * from $Tablename where (FirstName='Larry')`. En este caso deseamos la información de cualquier columna de la tabla, pero sólo de aquellas filas donde la columna FirstName sea igual a Larry. Estos son buenos ejemplos de cómo SQL utiliza con efectividad y flexibilidad unos pocos términos.

La principal diferencia entre la recuperación y la inserción de datos en una base de datos está en que tiene que manipular de forma distinta la consulta. Es preferible asignar los resultados de la consulta a una variable:

```
$Result = mysql_db_query ($DBName,
→ '$Query, $Link);
```

En lenguaje vulgar, esta variable conoce ahora el resultado de la consulta. Para acceder a las múltiples filas de información recuperada, debe ejecutar la variable $Result mediante un bucle.

```
while ($Row = mysql_fetch_array
→ ($Result)) {
    instrucciones;
}
```

Con cada iteración del bucle, la siguiente fila de información de la consulta (almacenada en $Result) se convertirá en un *array* llamado $Row. Este proceso continuará hasta que no queden más filas de información. La mejor forma de comprender este sistema es probándolo, por lo que ahora escribiremos un *script* que lee la información almacenada en la tabla Feedback (puede que quiera pasar un par de veces más por form.html para construir la tabla).

Cómo recuperar datos de una tabla

1. Cree un documento PHP nuevo en el editor de textos.
2. Comience con la cabecera HTML estándar (véase el *Script* 11.5).

```
<HTML><HEAD><TITLE>Retrieving Data
→ from a Database</TITLE><BODY>
```

3. Comience la sección PHP de la página y establezca las variables para la base de datos.

```
<?php
$Host = "localhost";
$User = "user";
$Password = "password";
$DBName = "NewDatabase";
$TableName = "Feedback";
```

Script 11.5. La consulta SQL para la recuperación de todos los datos de la tabla es muy sencilla, pero para que PHP acceda a cada elemento, tiene que efectuar un bucle para recorrer los resultados fila a fila.

script

```
1  <HTML>
2  <HEAD>
3  <TITLE>Retrieving Data from a
   Database</TITLE>
4  <BODY>
5  <?php
6  // Establece variables para el acceso a la
   base de datos:
7  $Host = "localhost";
8  $User = "user";
9  $Password = "password";
10 $DBName = "NewDatabase";
11 $TableName = "Feedback";
12
13 $Link = mysql_connect ($Host, $User,
   $Password);
14
15 $Query = "SELECT * from $TableName";
16 $Result = mysql_db_query ($DBName, $Query,
   $Link);
17
18 // Crea una tabla.
19 print ("<TABLE BORDER=1 WIDTH=\"75%\"
   CELLSPACING=2 CELLPADDING=2
   ALIGN=CENTER>\n");
20 print ("<TR ALIGN=CENTER VALIGN=TOP>\n");
21 print ("<TD ALIGN=CENTER
   VALIGN=TOP>Name</TD>\n");
22 print ("<TD ALIGN=CENTER VALIGN=TOP>Email
   Address</TD>\n");
23 print ("<TD ALIGN=CENTER
   VALIGN=TOP>Comments</TD>\n");
24 print ("</TR>\n");
25
26 // Busca los resultados de la base de
   datos.
27 while ($Row = mysql_fetch_array
   ($Result)) {
28     print ("<TR ALIGN=CENTER
       VALIGN=TOP>\n");
29     print ("<TD ALIGN=CENTER
       VALIGN=TOP>$Row[FirstName]
       $Row[LastName]</TD>\n");
30     print ("<TD ALIGN=CENTER
       VALIGN=TOP>$Row[EmailAddress]</TD>\n");
```

Script 11.5. Continuación

```
31      print ("<TD ALIGN=CENTER
        VALIGN=TOP>$Row[Comments]</TD>\n");
32      print ("</TR>\n");
33  }
34  mysql_close ($Link);
35  print ("</TABLE>\n");
36  ?>
37  </BODY>
38  </HTML>
```

4. Conecte con la base de datos, y escriba y ejecute la consulta.

```
$Link = mysql_connect ($Host, $User,
→ $Password);
$Query = "SELECT * from $TableName";
$Result = mysql_db_query ($Query,
→ $Link);
```

La consulta es directa y muy útil. Después, como ya indicamos anteriormente, los resultados de la consulta se almacenarán en una variable, que se recorrerá posteriormente mediante un bucle.

5. Inicie una tabla HTML para visualizar los resultados de la consulta.

```
print ("<TABLE BORDER=1 WIDTH=\"75%\"
→ CELLSPACING=2 CELLPADDING=2
→ ALIGN=CENTER>\n");
print ("<TR ALIGN=CENTER
→ VALIGN=TOP>\n");
print ("<TD ALIGN=CENTER
→ VALIGN=TOP>Name</TD>\n");
print ("<TD ALIGN=CENTER
→ VALIGN=TOP>Email Address</TD>\n");
print ("<TD ALIGN=CENTER
→ VALIGN=TOP>Comments</TD>\n");
print ("</TR>\n");
```

Puesto que está recuperando información de una tabla y visualizándola en un formulario HTML, tiene sentido crear una tabla HTML para mejorar la apariencia de los datos.

6. Cree un bucle que recupere cada fila de la variable hasta que no queden más filas.

```
while ($Row = mysql_fetch_array
→ ($Result)) {
```

Lo que este bucle está haciendo es configurando la variable $Row a un *array* compuesto por la primera fila de la tabla en $Result. El bucle ejecutará entonces los siguientes comandos (paso 7). Una vez que el bucle regresa al comienzo, asignará la siguiente fila, si existe. El bucle continuará con este proceso hasta que no queden más filas de información que obtener.

7. Imprima la información de la base de datos en la tabla HTML.

```
print ("<TR ALIGN=CENTER
→ VALIGN=TOP>\n");
print ("<TD ALIGN=CENTER
→ VALIGN=TOP>$Row[FirstName]
→ $Row[LastName]</TD>\n");
print ("<TD ALIGN=CENTER
→ VALIGN=TOP>$Row[EmailAddress]
→ </TD>\n");
print ("<TD ALIGN=CENTER
→ VALIGN=TOP>$Row[Comments]
→ </TD>\n");
```

Puesto que utiliza la función mysql_fetch_array(), puede referirse a cada columna individualmente en la fila del mismo modo que en un *array*. Las claves del *array* son los nombres de las columnas de la tabla: FirstName, LastName, EmailAddress y Comments (no hay ninguna necesidad de imprimir id).

8. Cierre la fila HTML y cierre el bucle while.

```
    print ("</TR>\n");
}
```

Simplemente para asegurarnos de que queda claro, diremos que este bucle asignará una fila de los datos de la tabla al array $Row, y después los imprimirá como una fila de una tabla HTML.

9. Cierre el enlace MySQL, cierre la tabla HTML, después cierre la sección PHP y, por último, la propia página HTML.

```
mysql_close ($Link);
print ("</TABLE>\n");
?></BODY></HTML>
```

10. Guarde el *script* como DisplayDB.php, cárguelo en el servidor y pruébelo en el navegador (véase la Figura 11.6).

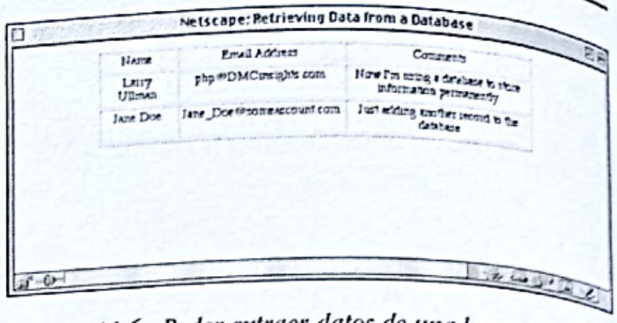


Figura 11.6. Poder extraer datos de una base de datos para luego crear una página web dinámica es un gran uso de PHP y algo que nunca podrá lograr sólo con HTML.

COOKIES 12

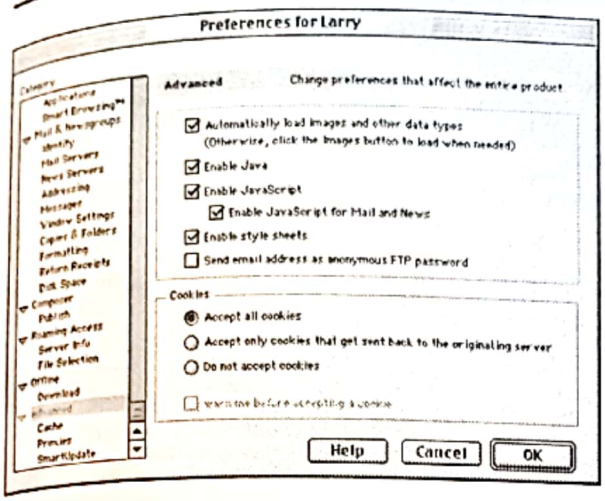


Figura 12.1. En Netscape Navigator puede establecer los permisos relacionados con las cookies seleccionando Edición, Preferencias, Avanzadas. En Internet Explorer 5.0 (para Macintosh) debe seleccionar Edición, Preferencias, Cookies, aunque las distintas versiones de los navegadores están organizadas de forma diferente, especialmente al comparar Windows con Macintosh.

Las *cookies* son una de las herramientas menos entendida y más difamada de la actual World Wide Web. Mientras muchos usuarios tienen interés en las cuestiones de seguridad que, según se rumorea, crean las *cookies*, poca gente se da cuenta de los beneficios que ofrecen al usuario.

Antes de la existencia de las *cookies*, atravesar un sitio web era un viaje sin historia. Aunque el navegador sigue la pista de las páginas que visita, permitiéndole usar el botón Atrás para volver a las páginas anteriormente visitadas e indicando los enlaces visitados con un color diferente, el servidor no guarda el registro de quién ha visto qué página. Esto es así tanto para los sitios que no emplean *cookies*, como para los usuarios que las han desactivado en los navegadores web (véase la Figura 12.1).

¿Por qué es un problema? Sin que el servidor sea capaz de seguir la pista de un usuario, puede no haber carros de compra para que haga compras *online*. Si las *cookies* no existieran o estuvieran desactivadas en el navegador web, los usuarios no podrían utilizar Hotmail, ya que necesitan un registro de usuario.

Las *cookies* son una forma de que un servidor almacene la información sobre el usuario (en la computadora de éste) de modo que el servidor pueda recordar al usuario el curso de la visita. Piense en una *cookie* como en una etiqueta nombre: usted dice el nombre al servidor y éste le proporciona una etiqueta nombre. Entonces el servidor sabrá a quién se está refiriendo gracias a esta etiqueta.

Esto nos lleva a otro punto sobre las cuestiones de seguridad relacionadas con las *cookies*. Las *cookies* han tenido una mala aceptación, ya que los usuarios creen que permiten que el servidor tenga demasiada información sobre ellos. Sin embargo, una *cookie* sólo puede utilizarse para almacenar la información que le dé, de modo que será tan segura como lo desee.

PHP dispone de buen soporte para *cookies*.

En este capítulo aprenderá a configurar, recuperar la información y borrar una *cookie*. También verá algunos de los parámetros opcionales que puede usar para establecer límites a la existencia de una *cookie*.

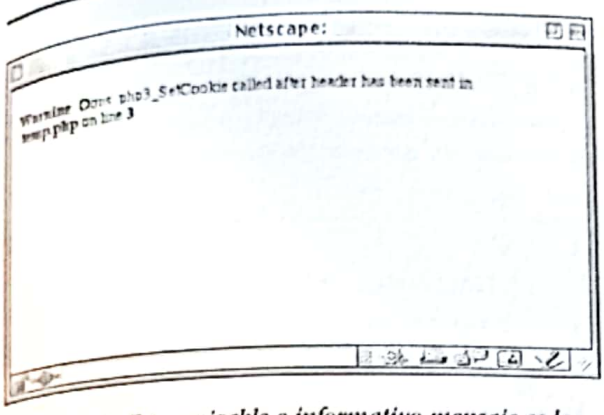


Figura 12.2. Este amigable e informativo mensaje es lo que se ve cuando se llama a la función setcookie() después de que algo, incluso una línea en blanco, se haya enviado al navegador.

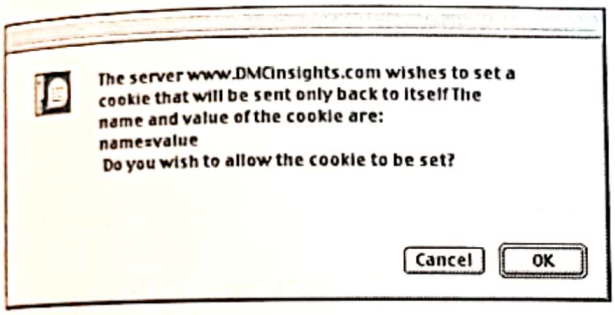


Figura 12.3. Si el usuario tiene activada la opción "Advertir antes de aceptar un cookie" en sus Preferencias (véase la Figura 12.1), aparecerá un aviso parecido a éste por cada cookie enviada.

Creación y lectura de *cookies*

Lo más importante a entender con respecto a las *cookies* es que se deben enviar desde el servidor al cliente antes que ninguna otra información. Si el servidor intentara enviar una *cookie* una vez que el navegador web hubiera recibido código HTML, incluso un extraño espacio en blanco, resultaría un mensaje erróneo y la *cookie* no se enviaría (véase la Figura 12.2). Éste es el error más común relacionado con las *cookies*.

Las *cookies* se envían utilizando la función setcookie():

```
setcookie ("nombre", "valor");
```

Esta línea de código enviará una *cookie* al navegador con el nombre y el valor especificados (véase la Figura 12.3).

Puede seguir enviando *cookies* al navegador utilizando la función setcookie(), aunque está limitado por los protocolos web a enviar como máximo veinte *cookies* de un servidor a un usuario:

```
setcookie ("nombre2", "valor2");
setcookie ("nombre3", "valor3");
```

Para recuperar un valor de una *cookie*, sólo necesitará hacer referencia al nombre de la *cookie* como una variable (el nombre precedido de un signo de dólar), igual que si se refiriera a un elemento de formulario HTML como una variable en la página. Por ejemplo, para recuperar el valor de la *cookie* configurada con la línea setcookie ("UserName", "Larry");, utilizará la variable $UserName.

Como ejemplo de configuración de *cookies*, crearemos una página que permita al usuario especificar el texto y los colores de fondo de una página.

Cómo enviar y recuperar *cookies* con PHP

1. Cree un nuevo documento PHP en el editor de textos comenzando con la etiqueta PHP (véase el *Script* 12.1).

   ```
   <?php
   ```

2. Escriba una estructura condicional que enviará la *cookie* si el formulario es enviado.

   ```
   if ($BeenSubmitted) {
   ```

   Igual que ha hecho antes, la variable $BeenSubmitted se utilizará para determinar si el formulario se ha enviado o no. Si es TRUE, entonces PHP procesará el formulario.

3. Configure las *cookies* y los colores de la página.

   ```
   setcookie("BGColor",
   → "$NewBGColor");
   setcookie("TextColor",
   → "$NewTextColor");
   $BGColor = $NewBGColor;
   $TextColor = $NewTextColor;
   ```

   Si el formulario ha sido enviado, PHP enviará dos *cookies* con los valores para los colores del fondo y del texto. El *script* establecerá los valores actuales ($BGColor y $TextColor) a los valores elegidos ($NewBGColor y $NewTextColor) para que se refleje el cambio inmediatamente.

4. Termine la estructura condicional.

   ```
   } else {
       if (!$BGColor) {
           $BGColor = "WHITE";
       }
       if (!$TextColor) {
           $TextColor = "BLACK";
       }
   }
   ```

   Si el formulario no se ha enviado, PHP preguntará para asegurarse de que los colores del fondo y del texto no se han asignado. Si no han sido asignados, se les asignarán los valores predeterminados.

Script 12.1. Se utilizarán dos cookies para almacenar las elecciones del usuario en relación a los colores del fondo y del texto. Los valores enviados del formulario también se asignarán a la página de manera que el cambio solicitado se aplique inmediatamente.

script

```
1  <?php
2  if ($BeenSubmitted) {
3      setcookie("BGColor", "$NewBGColor");
4      setcookie("TextColor", "$NewTextColor");
5      $BGColor = $NewBGColor;
6      $TextColor = $NewTextColor;
7  } else {
8      if (!$BGColor) {
9          $BGColor = "WHITE";
10     }
11     if (!$TextColor) {
12         $TextColor = "BLACK";
13     }
14 }
15 ?>
16 <HEAD>
17 <TITLE>User Customization</TITLE>
18 </HEAD>
19 <?
20 print ("<BODY BGCOLOR=$BGColor
   TEXT=$TextColor>\n");
21 ?>
22 Currently your page looks like this!
23 <FORM ACTION="cookies.php" METHOD=POST>
24 Select a new background color:
25 <SELECT NAME="NewBGColor">
26 <OPTION VALUE=WHITE>WHITE</OPTION>
27 <OPTION VALUE=BLACK>BLACK</OPTION>
28 <OPTION VALUE=BLUE>BLUE</OPTION>
29 <OPTION VALUE=RED>RED</OPTION>
30 <OPTION VALUE=GREEN>GREEN</OPTION>
31 </SELECT>
32 Select a new text color:
33 <SELECT NAME="NewTextColor">
34 <OPTION VALUE=WHITE>WHITE</OPTION>
35 <OPTION VALUE=BLACK>BLACK</OPTION>
36 <OPTION VALUE=BLUE>BLUE</OPTION>
37 <OPTION VALUE=RED>RED</OPTION>
38 <OPTION VALUE=GREEN>GREEN</OPTION>
39 </SELECT>
```

Script 12.1. Continuación

```
script
40 <INPUT TYPE=HIDDEN NAME=BeenSubmitted
   VALUE=TRUE>
41 <INPUT TYPE=SUBMIT NAME="SUBMIT"
   VALUE="Submit!">
42 </FORM>
43 </BODY>
44 </HTML>
```

5. Cierre la sección inicial PHP y cree la cabecera HTML.

```
?>
<HEAD>
<TITLE>User Customization</TITLE>
</HEAD>
```

6. Construya otra sección PHP para imprimir la etiqueta <BODY> con los valores apropiados para los colores de fondo y texto.

```
<?
print ("<BODY BGCOLOR=$BGColor
→ TEXT=$TextColor>\n");
?>
```

7. Imprima una sentencia sencilla que revelará el color del texto.

```
Currently your page looks like this!
```

8. Cree un formulario HTML que se enviará a sí mismo.

```
<FORM ACTION="cookies.php"
→ METHOD=POST>
```

También podría hacer que PHP imprimiera el valor de $PHP_SELF, que siempre se refiere a la página.

9. Construya dos menús desplegables para que el usuario seleccione los colores del fondo y del texto.

```
Select a new background color:
<SELECT NAME="NewBGColor">
<OPTION VALUE=WHITE>WHITE</OPTION>
<OPTION VALUE=BLACK>BLACK</OPTION>
<OPTION VALUE=BLUE>BLUE</OPTION>
<OPTION VALUE=RED>RED</OPTION>
<OPTION VALUE=GREEN>GREEN</OPTION>
</SELECT>
Select a new text color:
<SELECT NAME="NewTextColor">
<OPTION VALUE=WHITE>WHITE</OPTION>
<OPTION VALUE=BLACK>BLACK</OPTION>
<OPTION VALUE=BLUE>BLUE</OPTION>
<OPTION VALUE=RED>RED</OPTION>
<OPTION VALUE=GREEN>GREEN</OPTION>
</SELECT>
```

Cada menú desplegable sólo tiene cinco opciones. Puede añadir más opciones a estos menús, si quiere, tantas como los colores HTML válidos a los que se limite.

10. Codifique una variable oculta que indicará si el formulario se ha enviado o no.

```
<INPUT TYPE=HIDDEN
 NAME=BeenSubmitted VALUE=TRUE>
```

Ésta es la variable que indicará al *script* que el formulario se ha enviado.

11. Añada el botón de envío, y cierre el formulario y la página HTML.

```
<INPUT TYPE=SUBMIT NAME='SUBMIT'
 VALUE='Submit!'>
</FORM>
</BODY>
</HTML>
```

12. Guarde la página como cookies.php, cárguela en el servidor y pruébela en el navegador web (véanse las Figuras 12.4, 12.5, 12.6, 12.7 y 12.8).

- El valor de una *cookie* se codifica en un URL automáticamente cuando se envía y se descodifica el URL para ser recibida por la página PHP.

  Como sabe, esto también es cierto para los valores enviados por formularios HTML.

- Es importante recordar que el valor de una *cookie* siempre tendrá preferencia sobre los valores pasados mediante un formulario. En cookies.php, por ejemplo, si las entradas del formulario se llamaran BGColor y TextColor tendría problemas, ya que las *cookies* BGColor y TextColor sobrescribirían los valores del formulario.

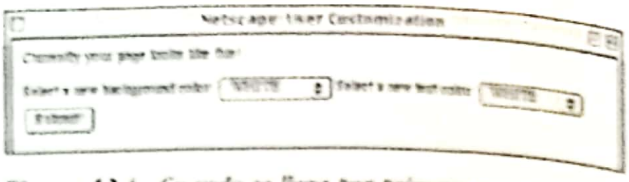


Figura 12.4. Cuando se llega por primera vez a cookies.php, la página se parece a ésta, con los colores predeterminados: fondo blanco y texto negro.

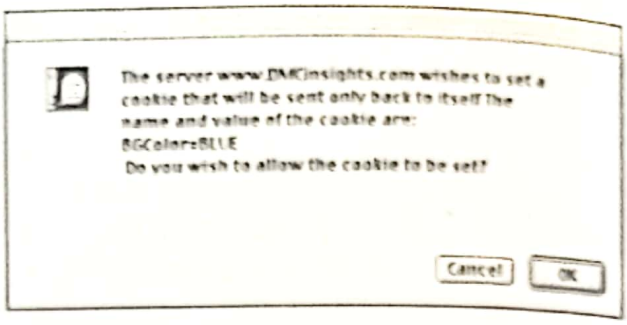


Figura 12.5. Éste es el mensaje que verá el usuario cuando se haga la primera llamada a setcookie() si ha optado por ser advertido antes de aceptar una cookie. Esta cookie almacena el valor BLUE en una cookie llamada BGColor.

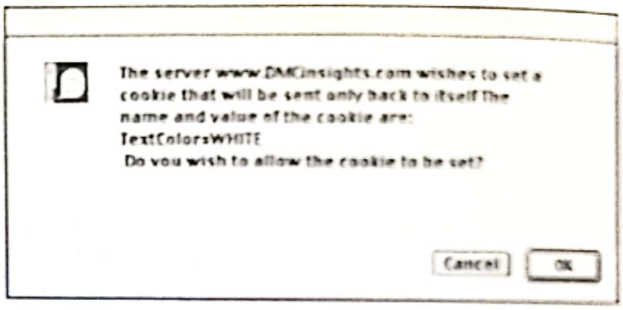


Figura 12.6. La segunda cookie que se envía se llama TextColor y tiene el valor WHITE. Si el usuario no ha activado en sus preferencias la opción "Advertir antes de aceptar un cookie", no verá estos mensajes.

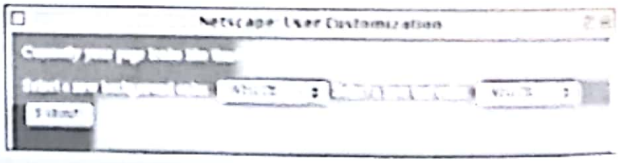


Figura 12.7. Al recibir los valores del formulario HTML, la página cookies.php enviará las dos cookies para almacenar la información y después reflejar los cambios en la página.

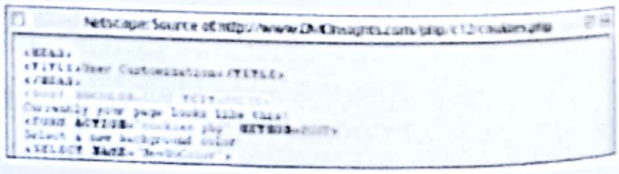


Figura 12.8. Estudiando el código fuente de la página, también puede saber cómo cambian los valores del color.

Adición de parámetros a una *cookie*

Aunque pasar sólo los argumentos nombre y valor a la función setcookie() será suficiente para la mayoría de los usos de las *cookies*, debe prestar atención a los demás argumentos disponibles. La función puede tomar hasta cuatro argumentos más, cada uno de los cuales limitará la operación de la *cookie*.

```
setcookie ("nombre", "valor", "expiración", "ruta",
→ "dominio", "seguridad");
```

El argumento expiración se utiliza para establecer el tiempo específico de existencia de una *cookie*.

Si no se especifica, la *cookie* seguirá siendo funcional hasta que el usuario cierre el navegador. Normalmente, el tiempo de expiración se configura añadiendo un número particular de minutos u horas al tiempo actual. Esta línea de código establecerá que el tiempo de expiración de la *cookie* sea una hora (60 segundos por 60 minutos) a partir del momento actual:

```
setcookie ("nombre", "valor", time()_+ "3600");
```

Debido a que el tiempo de expiración se calculará como el valor de time() más 3600, este argumento particular no se entrecomilla (ya que no quiere pasar literalmente time() + 3600 como expiración, sino el resultado de ese cálculo).

Los argumentos ruta y dominio se utilizan para limitar una *cookie* a una carpeta específica dentro de un sitio web (la ruta) o a un dominio específico. Por ejemplo, podría limitar a una *cookie* a que existiera sólo mientras un usuario esté dentro de la carpeta del dominio:

```
setcookie ("nombre", "valor", time()_+ 3600, "/user/");
```

Eliminación de una *cookie*

Lo último que hay que entender al usar *cookies* es cómo borrarlas. Aunque una *cookie* expirará automáticamente cuando el navegador del usuario se cierre o cuando se cumpla la fecha/hora de expiración, a veces deberá borrarla manualmente. Por ejemplo, los sitios web que tienen registrados usuarios y capacidades de conexión probablemente querrán borrar algunas *cookies* cuando el usuario se desconecte.

Aunque la función setcookie() puede tomar hasta seis argumentos, sólo uno es realmente necesario: el nombre. Si envía una *cookie* que consiste en un nombre sin un valor, tendrá el mismo efecto que borrar la *cookie* que ya existe con el mismo nombre. Por ejemplo, para crear la *cookie* UserName, utilice esta línea:

```
setcookie("UserName", "Larry");
```

Para borrar la *cookie* UserName, codificaría:

```
setcookie("UserName", "");
```

Como precaución añadida, también puede configurar una fecha de expiración en el pasado.

```
setcookie("UserName", "", time() - 60);
```

Para mostrar este elemento, añadiremos un botón reset a la página cookies.php que destruirá las *cookies* enviadas y mostrará en pantalla los colores predeterminados una vez más.

Script 12.3. Para volver a iniciar todos los valores, se envían cookies en blanco con los mismos nombres que las cookies existentes. También se ha añadido un cuadro de activación de reinicio al formulario HTML.

script

```
1  <?php
2  if ($BeenSubmitted) {
3      if ($Reset) {
4          setcookie("BGColor", "", time()-
           "100");
5          setcookie ("TextColor", "" ,
           time()-"100");
6          $BGColor = "WHITE";
7          $TextColor = "BLACK";
8      } else {
9          setcookie("BGColor",
           "$NewBGColor", time()+
           "1000000");
10         setcookie ("TextColor",
           "$NewTextColor", time()+
           "1000000");
11         $BGColor = $NewBGColor;
12         $TextColor = $NewTextColor;
13     }
14 } else {
15     if (!$BGColor) {
16         $BGColor = "WHITE";
17     }
18     if (!$TextColor) {
19         $TextColor = "BLACK";
20     }
21 }
22 ?>
23 <HEAD>
24 <TITLE>User Customization</TITLE>
25 </HEAD>
26 <?
27 print ("<BODY BGCOLOR=$BGColor
   TEXT=$TextColor>\n");
28 ?>
29 Currently your page looks like this!
30 <FORM ACTION="cookies.php" METHOD=POST>
31 Select a new background color:
32 <SELECT NAME="NewBGColor">
33 <OPTION VALUE=WHITE>WHITE</OPTION>
34 <OPTION VALUE=BLACK>BLACK</OPTION>
35 <OPTION VALUE=BLUE>BLUE</OPTION>
36 <OPTION VALUE=RED>RED</OPTION>
```

Script 12.3. Continuación

```
37 <OPTION VALUE=GREEN>GREEN</OPTION>
38 </SELECT>
39 Select a new text color:
40 <SELECT NAME="NewTextColor">
41 <OPTION VALUE=WHITE>WHITE</OPTION>
42 <OPTION VALUE=BLACK>BLACK</OPTION>
43 <OPTION VALUE=BLUE>BLUE</OPTION>
44 <OPTION VALUE=RED>RED</OPTION>
45 <OPTION VALUE=GREEN>GREEN</OPTION>
46 </SELECT>
47 <P><INPUT TYPE=Checkbox NAME=Reset
   VALUE=TRUE> Check this box to reset the
   colors.<P>
48 <INPUT TYPE=HIDDEN NAME=BeenSubmitted
   VALUE=TRUE>
49 <INPUT TYPE=SUBMIT NAME="SUBMIT"
   VALUE="Submit!">
50 </FORM>
51 </BODY>
52 </HTML>
```

Cómo eliminar una *cookie*

1. Abra cookies.php en el editor de textos (véase el *Script* 12.2).

2. Añada otra estructura condicional a la condicional if ($BeenSubmitted) existente (véase el *Script* 12.3, líneas 3 a 8).

```
if ($Reset) {
     setcookie("BGColor", "",
     → time()- "100");
     setcookie ("TextColor", "" ,
     → time()-"100");
     $BGColor = "WHITE";
     $TextColor = "BLACK";
} else {
```

Si el formulario se ha enviado, PHP primero comprobará si $Reset es TRUE.

Si lo es, el *script* borrará las *cookies* existentes enviando *cookies* sin valor con el mismo nombre. La página también restablecerá los valores de color para la página existente.

3. Complete la estructura condicional if ($Reset).

```
setcookie("BGColor", "$NewBGColor",
→ time()+ "1000000");
     setcookie ("TextColor",
     → "$NewTextColor",
     → time()+"1000000");
     $BGColor = $NewBGColor;
     $TextColor = $NewTextColor;
}
```

Si el formulario se ha enviado pero $Reset no es TRUE, el formulario se manipulará como en el *Script* 12.2.

4. Dentro del formulario HTML, añada una casilla de verificación para que el usuario vuelva a restablecer los colores (véase el *Script* 12.3, línea 47).

```
<P><INPUT TYPE=Checkbox NAME=Reset
→ VALUE=TRUE>Check this box to reset
· the colors.<P>
```

Esta casilla de verificación indicará a PHP si se restablecerán o no los valores. Hemos añadido dos etiquetas de párrafo para que el formulario HTML parezca más elegante.

5. Guarde el *script*, cárguelo en el servidor y pruébelo en el navegador web (véanse las Figuras 12.11, 12.12 y 12.13).

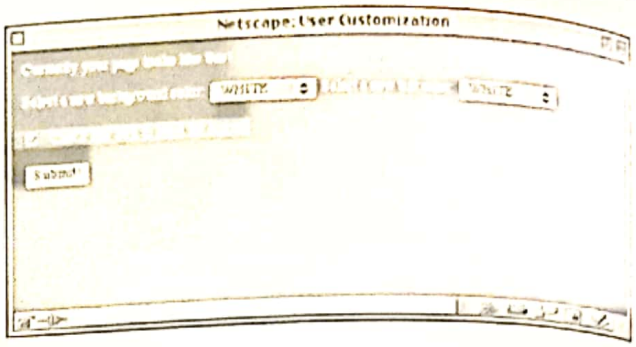


Figura 12.11. Hemos añadido una simple casilla de verificación al formulario HTML para darle al usuario la opción de restablecer la página. También borrará las cookies existentes.

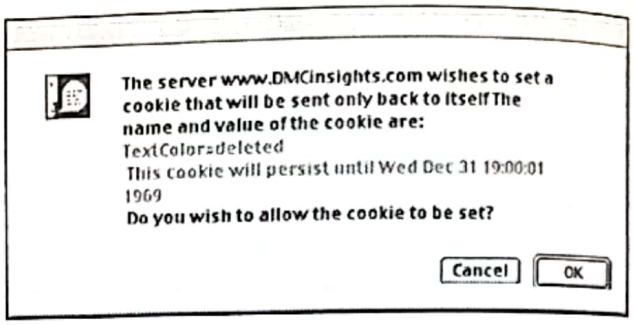


Figura 12.12. Cuando se utiliza la función setcookie() con un nombre pero no con un valor, se borrará la cookie existente con este nombre. La fecha de expiración en el pasado también garantiza la destrucción apropiada de la cookie existente.

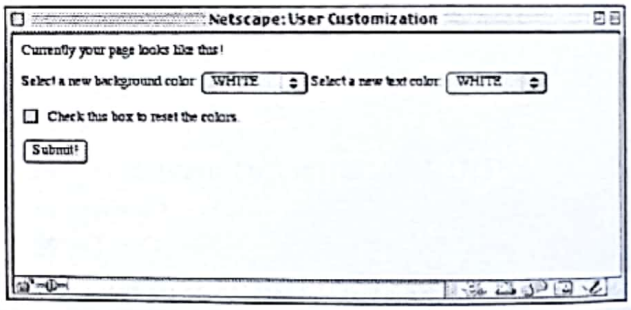


Figura 12.12. Cuando se utiliza la función setcookie() con un nombre pero no con un valor, se borrará la cookie existente con este nombre. La fecha de expiración en el pasado también garantiza la destrucción apropiada de la cookie existente.

CONSEJOS

- Para comprobar que las funciones setcookie() trabajan correctamente, asegúrese de activar Advertir antes de aceptar un cookie (en Netscape) o Preguntar para cada cookie (en Internet Explorer) en las preferencias del navegador (véase la Figura 12.1).
- La función setcookie() es una de las pocas en PHP que puede generar resultados distintos en navegadores diferentes, debido a que los navegadores tratarán las *cookies* de forma diferente. Asegúrese de probar los sitios web en varios navegadores y en plataformas distintas para asegurar la coherencia.

CREACIÓN DE APLICACIONES WEB 13

Ahora que ya ha aprendido lo más importante de la programación en PHP, es hora de trasladarlo a las aplicaciones web de estilo profesional. En este capítulo aprenderá las funciones y técnicas que puede utilizar para que los sitios web sean más profesionales, con más características y más fáciles de modificar.

Uso de include() y require()

Hasta este punto, ha escrito *scripts* cuya interfaz contenía bases de datos, manipulado formularios HTML y enviado *cookies*. Sin embargo, cuando comience a desarrollar sitios web de varias páginas, será casi imposible desarrollar cada atributo de un *script* de una sola vez. Ciertas funciones, como el diseño de HTML y las funciones de PHP, se utilizarán en cada página del sitio. Puede colocar esta información común en una página individual mientras desarrolla el sitio, pero cuando necesite cambiar algo, le pedirán que haga el cambio una y otra vez. Puede ahorrar mucho tiempo si crea páginas separadas para elementos en particular, y después los traslada a las páginas PHP principales utilizando una de estas dos funciones: include() o require().

```
include ("file.php");
require ("file.php");
```

Ambas funciones funcionan exactamente de la misma manera, con una única diferencia. Siempre se llamará a la función require(), sin importar dónde esté situada en el *script*, incluso si es el resultado de una condición que nunca es verdad. La función include(), por el contrario, sólo se activará si se solicita. Así, debería emplear require() si el archivo se debe incluir, y utilizar include() si el archivo puede o no incluirse dependiendo de las circunstancias que se manejen.

Pero ¿qué hacen estas funciones? Ambas, include() y require(), incorporarán el archivo al que hace referencia el archivo principal (para decirlo más claro, haremos referencia al archivo que tiene la línea include() o require() como archivo padre). Cualquier código que esté dentro del archivo incluido será tratado como HTML, a menos que esté dentro de unos paréntesis PHP. Todas las variables que existan en el documento padre antes de llamar a include() o a require(), estarán disponibles en el archivo incluido, y todas las variables que haya en el documento incluido estarán disponibles para el documento padre después de llamar a include() o a require().

Script 13.1. Éste es un archivo incluido básico, que va a crear la información <HEAD> de HTML, además de dar comienzo a la tabla del esquema. Como cada página tiene un título diferente, se va a asignar una variable para imprimir los valores particulares.

script

```
1  <HTML>
2  <HEAD>
3  <TITLE>
4  <?php
5  print ("$PageTitle");
6  ?>
7  </TITLE>
8  </HEAD>
9  <BODY>
10 <TABLE WIDTH="75%" ALIGN=CENTER BORDER=0>
11 <TR><TD ALIGN=CENTER><B>Welcome to the
   Site!</B></TD><TR>
12 <TR><TD ALIGN=LEFT><P>
```

Figura 13.1. Aunque esto no es muy elegante, debería demostrar adecuadamente cómo las páginas web se pueden dividir en varios componentes que luego pueden compartirse en múltiples páginas (véase la Figura 13.3).

Hay un buen número de razones para utilizar archivos incluidos. Puede colocar sus propias funciones definidas en un archivo común o colocar la información de acceso a la base de datos en un archivo de configuración. Primero, no obstante, vamos a colocar el diseño de HTML en los archivos incluidos para que pueda utilizarse en varias páginas.

Cómo emplear include() o require()

1. Cree un nuevo documento PHP en el editor de textos (véase el *Script* 13.1).

   ```
   <HTML>
   <HEAD>
   <TITLE>
   ```

   Aunque vaya a ser un archivo .php, sólo existe una sección de PHP, por lo que podrá codificar la mayor parte de la página como si fuese un *script* HTML estándar y evitar así el uso de numerosas instrucciones print().

2. Use PHP para imprimir el título de la página.

   ```
   <?php
   print ("$PageTitle");
   ?>
   ```

   La variable $PageTitle se asignará en el documento padre y después se empleará para establecer el título de la página que se visualizará en la parte superior de la ventana del navegador (véase la Figura 13.1).

3. Complete la información de la cabecera HTML.

   ```
   </TITLE>
   </HEAD>
   <BODY>
   ```

4. Cree una tabla para controlar el esquema de la página.

   ```
   <TABLE WIDTH="75%" ALIGN=CENTER
   → BORDER=0>
   <TR><TD ALIGN=CENTER><B>Welcome to
   → the Site!</B></TD><TR>
   <TR><TD ALIGN=LEFT><P>
   ```

El diseño global del sitio va a usar una tabla con tres filas. En la fila superior se incluirá un mensaje de bienvenida, aunque también podría tener un gráfico. La segunda fila será para el contenido específico de página, y la tercera contendrá el pie de página. Va a emplear el include de cabecera para crear la fila superior y continuar con la siguiente.

5. Guarde el *script* como header.php y cárguelo en el servidor.

   Escriba ahora el archivo de pie de página correspondiente.

6. Cree un nuevo documento PHP en el editor de textos (véase el *Script* 13.2).

7. Cierre la segunda fila de la tabla del esquema.

```
<P></TD></TR>
```

8. Codifique la tercera fila y la última, ésta con el copyright correspondiente.

```
<TR><TD ALIGN=CENTER><SMALL>
→ Copyright 2001</SMALL></TD></TR>
```

9. Cierre la tabla y la página HTML.

```
</TABLE>
</BODY>
</HTML>
```

10. Guarde el *script* como footer.php y cárguelo en el servidor.

    Una vez completados los dos archivos incluidos, puede comenzar a rellenar las páginas padre.

11. Cree un nuevo documento PHP en el editor de texto (véase el *Script* 13.3).

```
<?php
```

12. Asigne el nombre de la página a la variable $PageTitle.

```
$PageTitle = "Home Page";
```

    El valor de $PageTitle se usará para crear el título de la página que visualizará el navegador en la parte superior de la ventana. Como puede haber diferencias entre las páginas padre, crearemos una variable que la utilizará el archivo incluido.

Script 13.2. *Éste es un archivo incluido básico que completa la tabla del esquema y concluye la página HTML. Aquí también hemos colocado un copyrigh.*

script

```
1  <P></TD></TR>
2  <TR><TD ALIGN=CENTER><SMALL>Copyright
   2001</SMALL></TD></TR>
3  </TABLE>
4  </BODY>
5  </HTML>
```

Script 13.3. *Una vez que se han creado los dos archivos incluidos, la función require() los incorporará al archivo padre para crear una página entera rápidamente.*

script

```
1  <?php
2  $PageTitle = "Home Page";
3  require ("header.php");
4  print ("Content for the home page goes
   here.\n");
5  print ("<P>Click <A
   HREF=\"page2.php\">here</A> to go to the
   next page.\n");
6  require ("footer.php");
7  ?>
```

Script 13.4. Éste es otro archivo padre y, excepto por la línea 4, es exactamente idéntico al Script 13.3. Independientemente de que el sitio web tenga diez páginas o cien, todas podrán usar la misma plantilla básica.

script

```
1 <?php
2 $PageTitle = "Second Page";
3 require ("header.php");
4 print ("Content for the second page goes
  here.\n");
5 require ("footer.php");
6 ?>
```

13. Incluya el archivo de cabecera.

```
require ("header.php");
```

En este caso no tiene mayor importancia si emplea include() o require(), pero es mejor que emplee la última para garantizar la inclusión. Esta línea de programación trasladará todo el código de header.php hasta esta página. Asegúrese de llamar a esta línea después de que haya sido asignada $PageTitle, o el resto de la página no tendrá título.

14. Cree el contenido específico de la página.

```
print ("Content for the home page
→ goes here.\n");
print ("<P>Click <A HREF=
→ \"page2.php\">here</A> to go to
→ the next page.\n");
```

Por motivos de demostración, colocaremos sólo un mensaje y un enlace a la segunda página, en la que podrá incluir el contenido que desee, incluyendo un contenido generado dinámicamente mediante PHP.

15. Incluya el archivo de pie de página y cierre la página.

```
require ("footer.php");
?>
```

16. Guarde el *script* como index.php y cárguelo en el servidor.

Finalmente, escriba una segunda página que también utilice los archivos de diseño incluidos.

17. Cree un nuevo documento PHP en el editor de textos (véase el *Script* 13.4).

```
<?php
$PageTitle = "Second Page";
require ("header.php");
print ("Content for the second page
→ goes here.\n");
require ("footer.php");
?>
```

Para que esta página se diferencie de index.php, asigne a la variable $PageTitle un valor diferente y cambie la instrucción print().

18. Guarde el *script* como page2.php, cárguelo en el servidor y pruébelo junto a index.php en el navegador web (véanse las Figuras 13.1, 13.2 y 13.3).

CONSEJO

- Le recomendamos que emplee .php como extensión de archivo para los archivos incluidos, y no .inc, como hacen otros. Aunque no suele haber problemas con el archivo PHP padre, un usuario de la Web podría ver el código fuente de algún archivo .inc, pero nunca el código fuente de un archivo .php incluido (véanse las Figuras 13.4 y 13.5).

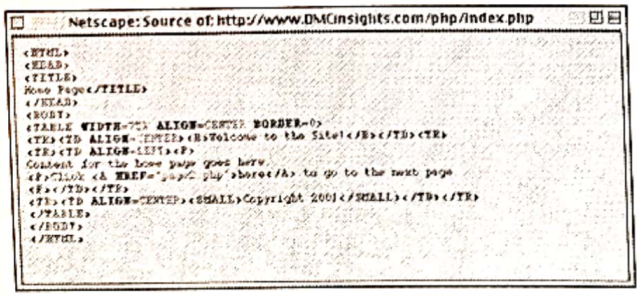


Figura 13.2. Nada en el código fuente de la página indica que esto se haya compilado sobre la marcha. Puede ver cómo los dos archivos incluidos se han traído como si fueran parte de un archivo padre.

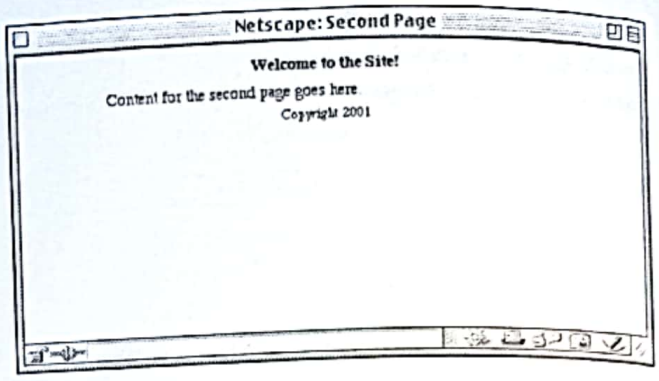


Figura 13.3. Esta página utiliza los mismos archivos de cabecera y pie que la Figura 13.1, pero el contenido y el título son distintos.

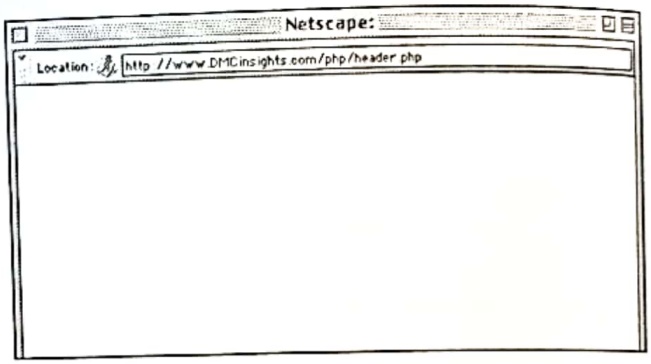


Figura 13.4. Cuando sus archivos incluidos tienen la extensión .php, los usuarios no pueden verlos directamente porque el servidor intentará ejecutar el código PHP, lo que en ocasiones dará como resultado páginas blancas. En otros casos, el navegador también puede revelar el principio del código HTML, pero, como medida de seguridad, ¡utilice la extensión .php!

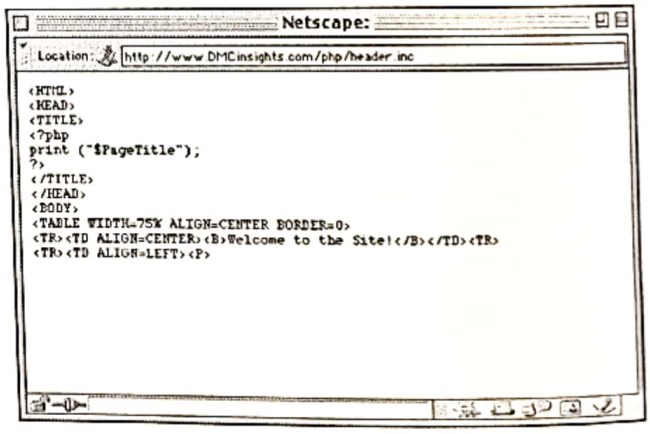


Figura 13.5. En comparación con la Figura 13.4 donde el archivo utilizaba la extensión .php, cuando se utiliza la extensión .inc (que es un hecho bastante común), un usuario furtivo puede ver su código fuente directamente visualizando la página. Si la información de acceso a las bases de datos y las contraseñas están almacenadas en un archivo incluido, esto podría suponer un gran riesgo de seguridad.

Tabla 13.1. La lista de posibilidades de formato para la función date() es amplia y no es fácil de recordar. Le recomendamos que tenga a mano una copia de esta lista mientras programa.

Opciones de formato con la función date()

Carácter	Formato
a	am o pm
A	AM o PM
d	día del mes con 2 dígitos y ceros a la izquierda: 01 a 31
D	día de la semana con 3 letras: Sun, Mon, etc.
F	mes, forma larga: January
g	hora del día en formato de 12 horas sin ceros a la izquierda: 1 a 12
G	hora del día en formato de 24 horas sin ceros a la izquierda: 0 a 23
h	hora del día en formato de 12 horas: 01 a 12
H	hora del día en formato de 24 horas: 00 a 23
i	minutos: 00 a 59
j	día del mes sin ceros a la izquierda: 1 a 31
l ('L' minúscula)	día de la semana, forma larga: Sunday
m	mes de 2 dígitos: 01 a 12
M	mes de 3 letras: Jan
n	mes con dígitos y sin ceros a la izquierda: 1 a 12
s	segundos: 00 a 59
S	Sufijos en inglés ordinario con 2 caracteres: th, nd, rd, etc.
t	número de días de un mes en concreto: 28 a 31
U	segundos desde el comienzo
w	día de la semana con un solo dígito: 0 (domingo) a 6 (sábado)
y	año de 2 dígitos: 01
Y	año de 4 dígitos: 2001
z	día del año: 0 a 365

Determinación de la fecha y la hora

Aunque ya ha usado la función date() en este libro, es una función lo suficientemente útil como para merecer un tratamiento más específico. Lo único que hace la función date() es devolver la información de fecha y hora en un formato que se basa en los argumentos que tiene, pero es sorprendente todo lo que puede hacer.

```
date("formato");
```

En el Capítulo 6 ha visto cómo date ("A"); devolvía los valores de AM o PM. Existe una amplia lista de posibles opciones de formato, como se indica en la Tabla 13.1. Esos diferentes parámetros también se pueden combinar, por lo que date ("l F j, Y"); devolvería, por ejemplo, Friday January 26, 2001.

La función date() puede llevar un argumento llamado Timestamp. Este argumento es un número que representa los segundos que han pasado desde la medianoche del 1 de enero de 1970, hasta el momento que haga referencia al periodo. Como ha podido ver en el Capítulo 12, la función time() devuelve los datos de fecha y hora del momento actual. La función mktime() puede devolver los datos de fecha y hora de un momento en concreto.

```
mktime (hora, minutos, segundos, mes,
→ día, año);
```

Por lo que el código $Timestamp = mktime (12, 30, 0, 12, 27, 1997); asignaría a $Timestamp el número de segundos transcurridos desde las 12:30 del 27 de diciembre de 1997. Ese número podría entonces alimentar la función date():por ejemplo, date("D", $Timestamp); devolvería Sat, que es el formato en tres letras del día de la semana.

Vamos a emplear las funciones date() y mktime() para crear un calendario *online*, donde verá cómo estas funciones se usan varias veces.

Cómo usar date()

1. Cree un nuevo documento PHP en el editor de textos (véase el *Script* 13.5).

```
<?php
```

2. Asigne un valor a la variable $PageTitle, e incluya el archivo de cabecera.

```
$PageTitle = "Calendar";
require ("header.php");
```

Continúe desarrollando más páginas como parte del mismo sitio web, para que empleen los mismos archivos incluidos que los que ha creado anteriormente en este mismo capítulo.

3. Asigne valores predeterminados a $Month y $Year, si no los tienen.

```
if ((!$Month) && (!$Year)) {
    $Month = date ("m");
    $Year = date ("Y");
}
```

La primera vez que el usuario llegue a esta página, no se habrán pasado ninguno de los valores de mes o año. PHP seguirá adelante y empleará el mes actual (por ejemplo, 1) y el año (por ejemplo, 2001), si éste es el caso.

4. Cree un *timestamp* que haga referencia a un mes y año específicos.

```
$Timestamp = mktime (0, 0, 0,
→ $Month, 1, $Year);
```

Para el *timestamp*, tiene que alimentar con ceros a la función mktime() para la hora, minutos y segundos y, después, uno para el día del mes. El mes y el año se calcularán basándose en los valores de la página.

5. Determine qué mes está en forma escrita.

```
$MonthName  =  date("F",
→ $Timestamp);
```

La función date() con una F mayúscula devuelve el nombre del mes (por ejemplo, January).

Script 13.5. Este script utiliza dos de las funciones de fecha y hora de PHP (date() y mktime()) para generar un calendario preciso. El formulario de la parte inferior visualiza los meses y los años.

script

```
1  <?php
2  $PageTitle = "Calendar";
3  require ("header.php");
4  // Si el valor de $Month y $Year no existe,
   serán el mes y año actuales.
5  if ((!$Month) && (!$Year)) {
6      $Month = date ("m");
7      $Year = date ("Y");
8  }
9  //  Calcula el mes examinado.
10 $Timestamp = mktime (0, 0, 0, $Month, 1,
   $Year);
11 $MonthName  =  date("F", $Timestamp);
12 //  Crea una tabla con el mes adecuado.
13 print ("<TABLE BORDER=0 CELLPADDING=3
   CELLSPACING=0 ALIGN=CENTER>");
14 print ("<TR BGCOLOR=BLUE><TD COLSPAN=7
   ALIGN=CENTER><FONT COLOR=WHITE><B>
   $MonthName $Year</B></FONT></TD></TR>");
15 print ("<TR BGCOLOR=BLUE>
   <TD ALIGN=CENTER WIDTH=20><B>
   <FONT COLOR=WHITE>Su</FONT></B></TD>
   <TD ALIGN=CENTER WIDTH=20><B>
   <FONT COLOR=WHITE>M</FONT></B></TD>
   <TD ALIGN=CENTER WIDTH=20><B>
   <FONT COLOR=WHITE>Tu</FONT></B></TD>
   <TD ALIGN=CENTER WIDTH=20><B>
   <FONT COLOR=WHITE>W</FONT></B></TD>
   <TD ALIGN=CENTER WIDTH=20><B>
   <FONT COLOR=WHITE>Th</FONT></B></TD>
   <TD ALIGN=CENTER WIDTH=20><B>
   <FONT COLOR=WHITE>F</FONT></B></TD>
   <TD ALIGN=CENTER WIDTH=20><B>
   <FONT COLOR=WHITE>Sa</FONT></B></TD>
   </TR>\n");
16 $MonthStart = date("w", $Timestamp);
17 if ($MonthStart == 0) {
18     $MonthStart = 7;
19 }
20 $LastDay = date("d", mktime (0, 0, 0,
   $Month+1, 0, $Year));
21 $StartDate = -$MonthStart;
22 for ($k = 1; $k <= 6; $k++) { // Print 6
   rows.
23     print ("<TR BGCOLOR=WHITE>");
24     for ($i = 1; $i <= 7; $i++) { // Use
       7 columns
```

Script 13.5. Continuación

```
script
        $StartDate++;
25      if (($StartDate <= 0) ||
26      ($StartDate > $LastDay)) {
            print ("<TD BGCOLOR=GREEN>
27           </TD>");
        } elseif (($StartDate >= 1) &&
28      ($StartDate <= $LastDay)) {
            print ("<TD ALIGN=CENTER>
29          $StartDate</TD>");
        }
30
31    }
32    print ("</TR>\n");
33 }
34 print ("</TABLE>\n");
35 // Crea el formulario
36 print ("<FORM ACTION=\"calendar.php\"
   METHOD=GET>\n");
37 print ("Select a new month to view:\n");
38 print ("<SELECT NAME=Month>
   <OPTION VALUE=1>January</OPTION>\n
   <OPTION VALUE=2>February</OPTION>\n
   <OPTION VALUE=3>March</OPTION>\n
   <OPTION VALUE=4>April</OPTION>\n
   <OPTION VALUE=5>May</OPTION>\n
   <OPTION VALUE=6>June</OPTION>\n
   <OPTION VALUE=7>July</OPTION>\n
   <OPTION VALUE=8>August</OPTION>\n
   <OPTION VALUE=9>September</OPTION>\n
   <OPTION VALUE=10>October</OPTION>\n
   <OPTION VALUE=11>November</OPTION>\n
   <OPTION VALUE=12>December</OPTION>\n
   </SELECT>\n");
39 print ("<SELECT NAME=Year>
   <OPTION VALUE=2001>2001</OPTION>\n
   <OPTION VALUE=2002>2002</OPTION>\n
   <OPTION VALUE=2003>2003</OPTION>\n
   </SELECT>\n");
40 print ("<INPUT TYPE=SUBMIT NAME=SUBMIT
   VALUE=\"Submit!\">\n");
41 print ("</FORM>\n");
42 require ("footer.php");
43 ?>
```

6. Cree una tabla para visualizar el calendario.

```
print ("<TABLE BORDER=0 CELLPADDING=3
 CELLSPACING=0 ALIGN=CENTER>");
```

7. Imprima un subtítulo empleando el mes y el año.

```
print ("<TR BGCOLOR=BLUE>
 <TD COLSPAN=7 ALIGN=CENTER>
 <FONT COLOR=WHITE><B>$MonthName
 $Year</B></FONT></TD></TR>");
```

8. Imprima los días de la semana.

```
print ("<TR BGCOLOR=BLUE>
 <TD ALIGN=CENTER WIDTH=20>
 <B><FONT COLOR=WHITE>Su</FONT></B>
 </TD><TD ALIGN=CENTER WIDTH=20>
 <B><FONT COLOR=WHITE>M</FONT></B>
 </TD><TD ALIGN=CENTER WIDTH=20>
 <B><FONT COLOR=WHITE>Tu</FONT></B>
 </TD><TD ALIGN=CENTER WIDTH=20>
 <B><FONT COLOR=WHITE>W</FONT></B>
 </TD><TD ALIGN=CENTER WIDTH=20>
 <B><FONT COLOR=WHITE>Th</FONT></B>
 </TD><TD ALIGN=CENTER WIDTH=20>
 <B><FONT COLOR=WHITE>F</FONT></B>
 </TD><TD ALIGN=CENTER WIDTH=20>
 <B><FONT COLOR=WHITE>Sa</FONT></B>
 </TD></TR>\n");
```

9. Determine el primer día de la semana de ese mes.

```
$MonthStart = date("w", $Timestamp);
```

Esta línea asignará el día de la semana, de forma numérica de 0 a 6, a la variable $MonthStart para el mes y el año del calendario. Como el primer día del mes se ha usado para establecer el *timestamp*, este código identificará ese día en particular. Necesitará hacer esto para ver si el mes comienza en lunes, martes, etc.

10. Asegúrese de que puede utilizar la variable $MonthStart.

```
if ($MonthStart == 0) {
    $MonthStart = 7;
}
```

Si $MonthStart es igual a 0, lo que significa que el mes comienza en domingo, tendrá problemas si emplea ese número cuando se ejecute el *script*, por lo que es mejor que cambie el 0 por un 7.

11. Determine el último día del mes.

```
$LastDay = date("d", mktime (0, 0, 0,
→ $Month+1, 0, $Year));
```

A $LastDay se le asignará el valor de un número, como 01, 22 ó 30, que es igual al último día del mes. Se determina empleando 0 como el día, y el mes siguiente ($Month + 1), como el mes siguiente en la función mktime(). Matemáticamente, el número de segundos del día 0 del mes siguiente será el mismo que el número de segundos del último día de este mes.

12. Seleccione una fecha de inicio basándose en el primer día del mes.

```
$StartDate = -$MonthStart;
```

Como el calendario va a empezar en domingo, necesitará determinar cuántos días se saltará antes del primer día del mes. Si el primer día del mes es martes, $MonthStart es igual a 2, lo que significa que $StartDate es igual a un 2 negativo, asegurando que se crean dos días en blanco antes de que comience la numeración.

13. Codifique un bucle que imprima las líneas del calendario (semanas).

```
for ($k = 1; $k <= 6; $k++) {
print ("<TR BGCOLOR=WHITE>");
```

El calendario tendrá 6 filas, por lo que la segunda sección de código se repetirá seis veces en el bucle for.

14. Cree un segundo bucle que imprima las columnas (días) del calendario.

```
for ($i = 1; $i <= 7; $i++) {
```

El calendario tendrá 7 columnas, una por cada día de la semana.

15. Aumente en uno el valor de $StartDate.

```
$StartDate++;
```

El valor de $StartDate se va a imprimir cada día para crear la fecha, por lo que se necesitará aumentar este valor en cada iteración del bucle.

16. Codifique una condición para cuando se imprima $StartDate.

```
if (($StartDate <= 0) ||
→ ($StartDate > $LastDay)) {
    print ("<TD
    → BGCOLOR=GREEN> 
    → </TD>");
} elseif (($StartDate >= 1) &&
→ ($StartDate <= $LastDay)) {
    print ("<TD ALIGN=CENTER>
    → $StartDate</TD>");
}
```

Primero, si $StartDate es menor que 0 (dicho de otro modo, el mes aún no ha comenzado), o mayor que el número de días del mes ($LastDay), la página va a imprimir un espacio verde vacío. Si $StartDate es por lo menos uno, pero todavía no es mayor que el número de días del mes ($LastDay), la página imprimirá ese mismo número.

17. Cierre el segundo bucle for, la fila de la tabla y el primer bucle for.

```
    }
    print ("</TR>\n");
}
```

18. Finalice la tabla.

```
print ("</TABLE>\n");
```

19. Cree un formulario HTML que envíe la página de nuevo a sí misma.

```
print ("<FORM
→ ACTION=\"calendar.php\"
→ METHOD=GET>\n");
```

20. Cree dos menús desplegables: uno para el mes y otro para el año.

```
print ("Select a new month to
→ view:\n");
print ("<SELECT NAME=Month>
→ <OPTION VALUE=1>January</OPTION>\n
→ <OPTION VALUE=2>February</OPTION>\n
→ <OPTION VALUE=3>March</OPTION>\n
→ <OPTION VALUE=4>April</OPTION>\n
→ <OPTION VALUE=5>May</OPTION>\n
→ <OPTION VALUE=6>June</OPTION>\n
→ <OPTION VALUE=7>July</OPTION>\n
→ <OPTION VALUE=8>August</OPTION>\n
→ <OPTION VALUE=9>September</OPTION>\n
→ <OPTION VALUE=10>October</OPTION>\n
→ <OPTION VALUE=11>November</OPTION>\n
→ <OPTION VALUE=12>December</OPTION>\n
→ </SELECT>\n");
print ("<SELECT NAME=Year>
→ <OPTION VALUE=2001>2001</OPTION>\n
→ <OPTION VALUE=2002>2002</OPTION>\n
→ <OPTION VALUE=2003>2003</OPTION>\n
→ </SELECT>\n");
```

Puede colocar los años que desee. Observe que los dos menús generan variables con los nombres $Month y $Year, una vez que se ha enviado el formulario. En ese momento, gracias a la condición que empieza en la línea 5, se van a usar esos valores en lugar del predeterminado.

21. Cree un botón de envío y cierre el formulario.

```
print ("<INPUT TYPE=SUBMIT NAME=
→ SUBMIT VALUE=\"Submit!\">\n");
print ("</FORM>\n");
```

22. Incluya el archivo de pie de página y cierre PHP.

```
require ("footer.php");
?>
```

23. Guarde el *script* como calendar.php, cárguelo en el servidor y pruébelo en el navegador web (véanse las Figuras 13.6 y 13.7).

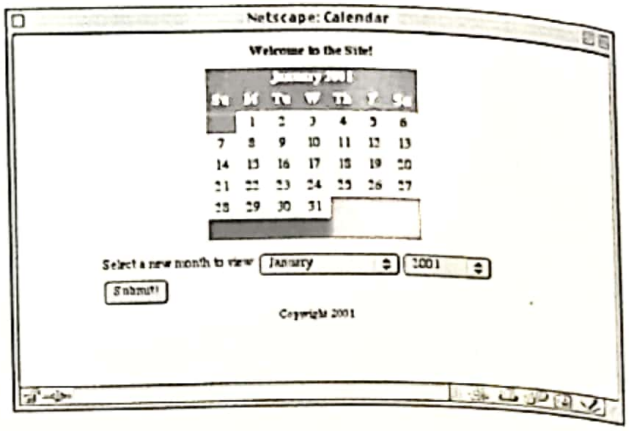


Figura 13.6. Al acceder por primera vez a la página del calendario, verá el mes actual. Utilice el formulario de la parte inferior para ver otros meses (véase la Figura 13.7).

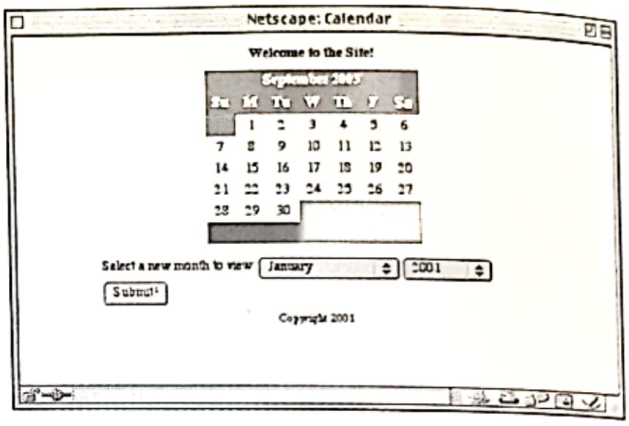


Figura 13.7. Independientemente del mes y el año seleccionados, este script siempre visualizará un calendario exacto.

CONSEJO

- En lugar de determinar el último día del mes como hemos hecho en la línea 20 (haciendo referencia a los cero segundos del primer día del mes siguiente), podría usar también el formato date ("t");, que devuelve el número de días de un mes. Si lo hace, el resto del *script* también tendría que alterarse un poco para generar un calendario correcto.

Utilización de las cabeceras HTTP

Una cabecera HTTP (Protocolo de transferencia de hipertexto) se emplea para enviar y devolver información entre el servidor y el cliente (el navegador web). Normalmente, esta información se encuentra en un formulario HTML, que es el motivo por el que las direcciones de las páginas web comienzan por http://.

Pero las cabeceras HTTP son lo suficientemente complicadas como para requerir un poco más de atención. Existen actualmente docenas y docenas de usos para ellas, y puede aprovecharse de todas si emplea la función header() de PHP. Le mostraremos los usos más frecuentes (redireccionar al usuario de una página a otra), aunque las especificaciones del consorcio de la World Wide Web para este tema son muy amplias (http://ww.w3.org/Protocols/rfc2616/rfc2616).

Para redireccionar al navegador del usuario con PHP, escriba el código:

```
header ("Location: page.php");
```

También puede usar la función header para enviar *cookies*, lo que es una buena copia de seguridad para la función setcookie(), que a veces tiene resultados muy poco coherentes de un navegador a otro.

```
header ("Set-cookie: name=value;
→ expires=expiration");
```

Lo más importante sobre el uso de header() es que se debe llamar antes de que se envíe nada al navegador web. Del mismo modo, tenga cuidado cuando emplee la función setcookie().

Para demostrar la redirección, vamos a crear un *script* de inicio de sesión que envíe al usuario a una página si emplea el nombre de usuario y contraseña correctos, o a otra si no lo hace.

Cómo usar la función header()

1. Cree un nuevo documento PHP en el editor de textos (véase el *Script* 13.6):

```
<?php
```

2. Asigne el título de la página e incluya un archivo de cabecera.

```
$PageTitle = "Login Page";
require ("header.php");
```

3. Cree una condición que imprima un mensaje si el usuario no se registra satisfactoriamente.

```
if ($Message == "Invalid") {
    print ("<B><CENTER><FONT COLOR=
    → RED>The username and password
    → you entered do not match what
    → is on file. Please try
    → again!</FONT>
    → </CENTER></B>\n");
}
```

Si el nombre de usuario y la contraseña no coinciden con los del archivo, el usuario será devuelto a esta página con $Message establecida a Invalid, mostrándose un mensaje de error.

4. Cree un formulario HTML que tenga un nombre de usuario y una contraseña.

```
print ("<FORM ACTION=
→ \"HandleLogin.php\"
→ METHOD=POST>\n");
print ("Username: <INPUT TYPE=TEXT
→ NAME=UserName><BR>\n");
print ("Password: <INPUT
→ TYPE=PASSWORD NAME=
→ Password><BR>\n");
print ("<INPUT TYPE=SUBMIT
→ NAME=SUBMIT VALUE=\"Submit!\">\n");
```

5. Incluya el archivo de pie de página y cierre la página PHP.

```
require ("footer.php");
?>
```

6. Guarde el *script* como login.php y cárguelo en el servidor.

Necesita crear la página que va a procesar login.php.

Script 13.6. Puede que no piense hacerlo a menudo, pero incluir la impresión de un mensaje de error en el script de inicio tiene sentido, porque probablemente deberá hacer retroceder al usuario no autentificado.

script

```
1  <?php
2  $PageTitle = "Login Page";
3  require ("header.php");

4  if ($Message == "Invalid") {
5  print ("<B><CENTER><FONT COLOR=RED>
   The username and password you entered do
   not match what is on file. Please try
   again!</FONT></CENTER></B>\n");
6  }
7  print ("<FORM ACTION=\"HandleLogin.php\"
   METHOD=POST>\n");
8  print ("Username: <INPUT TYPE=TEXT
   NAME=UserName><BR>\n");
9  print ("Password: <INPUT TYPE=PASSWORD
   NAME=Password><BR>\n");
10 print ("<INPUT TYPE=SUBMIT NAME=SUBMIT
   VALUE=\"Submit!\">\n");
11 require ("footer.php");
12 ?>
```

Script 13.7. Este script validará al usuario y la contraseña empleando valores predeterminados, y redireccionando al cliente convenientemente. No debe haber espacios extraños fuera de PHP en este script, o el resto de las llamadas a header() no funcionarán.

script

```
1 <?php
2 if (($UserName == "Larry") &&
  ($Password == "LarryPass")) {
3 header ("Location:
  index.php?UserName=$UserName");
4 exit;
5 } else {
6 header ("Location: login.php?Message=
  Invalid");
7 exit;
8 }
9 ?>
```

7. Cree un nuevo documento PHP en el editor de textos (véase el *Script* 13.7):

```
<?php
```

8. Cree la condición que verificará la correcta coincidencia del nombre de usuario y la contraseña.

```
if (($UserName == "Larry") &&
 › ($Password == "LarryPass")) {
```

Esta condición comprueba si el nombre de usuario y la contraseña coinciden con los del archivo. Puede escribir el *script* para que verifique una base de datos y busque una contraseña para el nombre de usuario, lo que permitiría muchas combinaciones diferentes. Normalmente, por razones de seguridad, no debería escribir explícitamente esos valores en el *script*, porque es mejor recuperarlos de una base de datos o de un archivo de texto. No obstante, y con fines demostrativos, va a establecer esos valores en PHP.

9. Si la autentificación funciona, redireccione al usuario a la página principal.

```
header ("Location: index.php?
→ UserName=$UserName");
exit;
```

Esta línea de código enviará al usuario fuera de esta página, y a index.php. También pasará a esta página el nombre de usuario.

La instrucción exit; indica a PHP que deje de ejecutar cualquier línea de código de esta página, porque es lo que quiere hacer ahora que ha redireccionado al usuario.

10. Finalice la condición y redireccione al usuario hasta la página principal.

```
} else {
    header ("Location: login.php?
     › Message=Invalid");
    exit;
}
```

Si los valores enviados no coinciden, se le dará al usuario otra oportunidad de registrarse enviándole a login.php. El ?Message=Invalid que se ha añadido al URL, le dice a login.php que imprima el mensaje de error como estaba programado en el *Script* 13.6.

11. Cierre la página PHP.

    ```
    ?>
    ```

12. Guarde el *script* como HandleLogin.php y cárguelo en el servidor.

    Modifique ahora la página index.php para saludar al usuario.

13. Abra index.php en el editor de textos (véase el *Script* 13.3).

14. Cambie la línea 4 para que se lea lo siguiente (véase el *Script* 13.8):

    ```
    print ("Greetings, $UserName!\n");
    ```

15. Guarde el *script* como index.php, cárguelo en el servidor y pruebe todas las páginas, comenzando por login.php, en el navegador web (véanse las Figuras 13.8, 13.9 y 13.10).

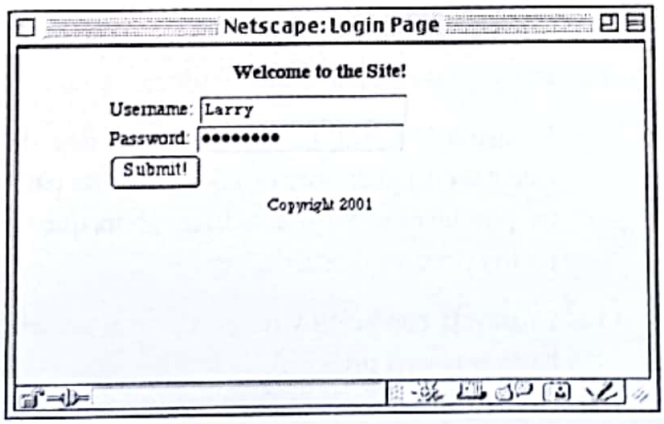


Figura 13.8. Sencilla página de inicio de sesión que toma un nombre de usuario y una contraseña.

Script 13.8. *Hemos añadido un saludo específico para el usuario en esta página para hacerlo más dinámico y personalizado. El valor de $UserName se ha pasado a la función header() en el Script 13.7*

```
script
1  <?php
2  $PageTitle = "Home Page";
3  require ("header.php");
4  print ("Greetings, $UserName!\n");
5  print ("<P>Click <A HREF=\"page2.php\">
   here</A> to go to the next page.\n");
6  require ("footer.php");
7  ?>
```

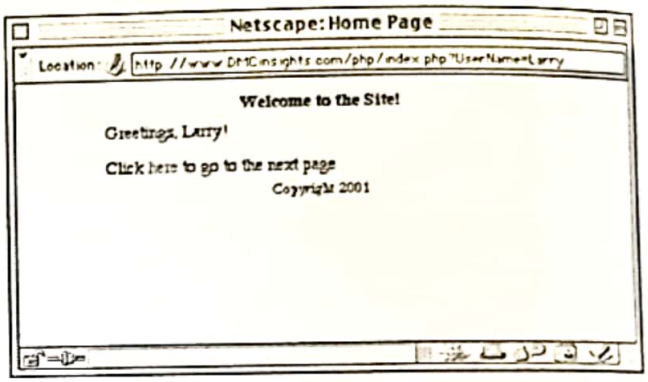


Figura 13.9. Tras una identificación satisfactoria, el usuario es dirigido a la página index.php, donde se le saludará por su nombre de usuario.

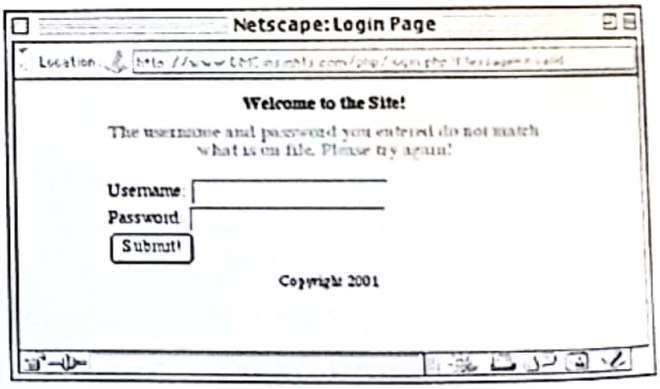


Figura 13.10. Si el nombre de usuario o la contraseña no coinciden con lo que hay en el archivo, el usuario es devuelto a la página login.php, donde verá este mensaje de error.

Script 13.9. En PHP puede enviar mensajes de correo electrónico con sólo la función mail(). Hemos cubierto la llamada a mail() con condiciones diferentes para asegurarnos de que el e-mail sólo se envía cuando sea necesario.

script

```
1  <?php
2  $PageTitle = 'Sending Emails';
3  require ("header.php");
4  if ($BeenSubmitted) {
5      if ($MailTo) {
6          if (mail($MailTo, $Subject,
           $Body, "From: $MailFrom")) {
7              print ("<B><CENTER><FONT
               COLOR=BLUE>Your email has
               been successfully sent!
               </FONT></CENTER></B>\n");
8          } else {
9              print ("<B><CENTER><FONT
               COLOR=RED>Your email was
               not successfully sent due
               to a system error!</FONT>
               </CENTER></B>\n");
10         }
11     } else {
12         print ("<B><CENTER><FONT COLOR=
           RED>Please enter the recipient's
           mail to address!</FONT></CENTER>
           </B>\n");
13     }
14 }
15 ?>
16 <FORM ACTION="email.php" METHOD=POST>
17 Recipient's Email Address: <INPUT TYPE=
   TEXT NAME="MailTo" SIZE="50"><BR>
18 Your Email Address: <INPUT TYPE=TEXT
   NAME="MailFrom" SIZE="50"><BR>
19 Email Subject: <INPUT TYPE=TEXT
   NAME="Subject" SIZE="80"><BR>
20 Email Body:<TEXTAREA NAME="Body"
   ROWS="10" COLS="50">
21 </TEXTAREA><P>
22 <INPUT TYPE=HIDDEN NAME=BeenSubmitted
   VALUE=TRUE>
23 <INPUT TYPE=SUBMIT NAME="SUBMIT"
   VALUE="Submit!">
24 </FORM>
25 <?
26 require ("footer.php");
27 ?>
```

Envío de un mensaje de correo electrónico

Entre las numerosas cosas que puede hacer con suma facilidad en PHP, se encuentra el envío de mensajes de correo electrónico.

```
mail ("mailto", "subject", "body");
```

La función mail() utiliza la aplicación de correo electrónico del servidor (como sendmail en Unix) para enviar los mensajes. Esta función puede tolerar otros argumentos mediante los que el usuario puede añadir más detalles al correo electrónico, como la dirección De, prioridad del mensaje de correo electrónico y copia de cortesía.

```
mail ("mailto", "subject", "body",
 "From: fromaddress");
```

Cómo enviar un mensaje de correo electrónico con PHP

1. Cree un nuevo documento PHP en el editor de textos (véase el *Script* 13.9):

   ```
   <?php
   ```

2. Asigne el título de la página e incluya el archivo de cabecera.

   ```
   $PageTitle = "Sending Emails";
   require ("header.php");
   ```

3. Cree una condición para procesar el formulario si está enviado.

   ```
   if ($BeenSubmitted) {
   ```

4. Compruebe que las direcciones de correo entrantes han sido enviadas y, si es así, envíe el mensaje de correo electrónico.

```
if ($MailTo) {
    if (mail($MailTo, $Subject,
    → $Body, "From: $MailFrom")) {
        print ("<B><CENTER>
        → <FONT COLOR=BLUE>
        → Your email has been
        → successfully sent!
        → </FONT></CENTER>
        → </B>\n");
    } else {
        print ("<B><CENTER>
        → <FONT COLOR=RED>
        → Your email was not
        → successfully sent due
        → to a system error!
        → </FONT></CENTER>
        → </B>\n");
    }
```

Hemos colocado la función mail() dentro de una condición para que se visualice un mensaje indicando si se ha enviado el mensaje correctamente o no.

5. Complete las condiciones.

```
    } else {
        print ("<B><CENTER>
        → <FONT COLOR=RED>Please
        → enter the recipient's mail
        → to address!</FONT>
        → </CENTER></B>\n");
    }
}
```

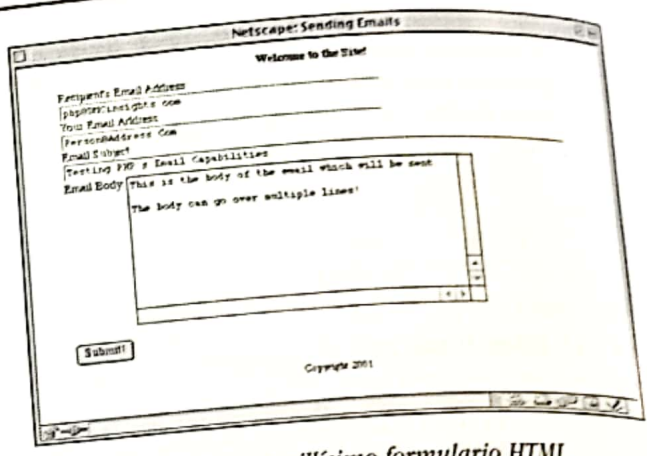


Figura 13.11. Este sencillísimo formulario HTML, combinado con la potencia de PHP, le permitirá enviar mensajes de correo electrónico desde su navegador.

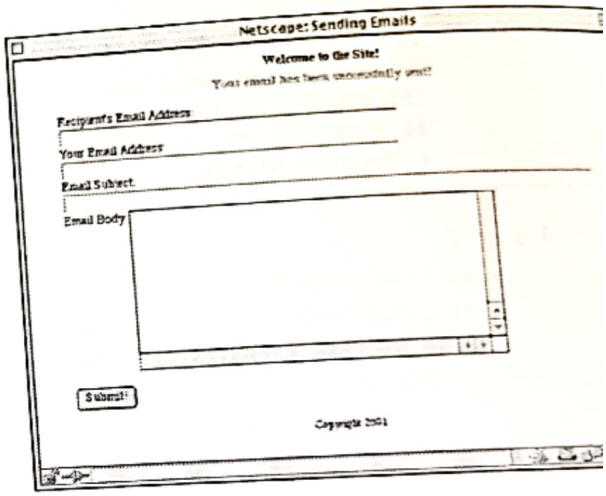


Figura 13.12. Si el script PHP ha podido enviar el de correo electrónico, un mensaje le informará de e

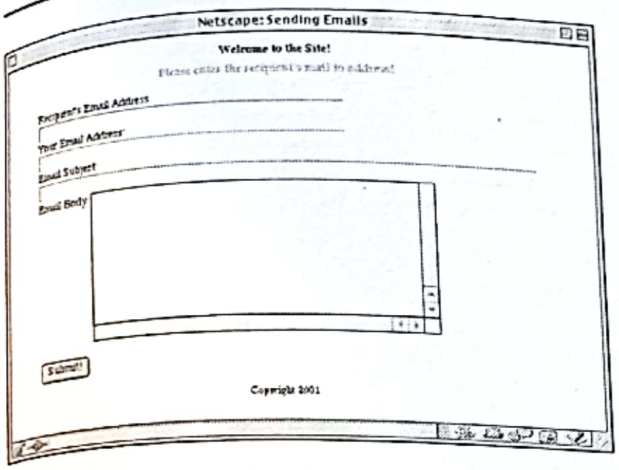


Figura 13.13. Si olvida introducir la dirección de correo electrónico del receptor, el script no intentará enviar el mensaje y, en su lugar, sólo imprimirá un mensaje de error.

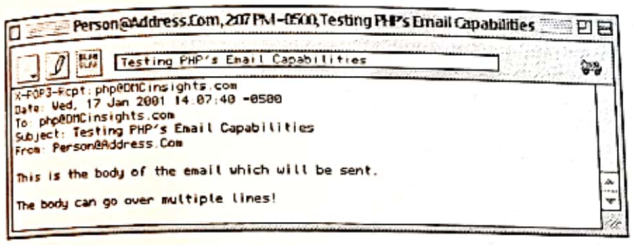


Figura 13.14. Éste es el mensaje recibido en Eudora (un cliente e-mail) después de enviar el formulario de la Figura 13.11.

6. Cierre la sección de PHP y cree un formulario HTML que tenga dos direcciones de correo electrónico, el asunto, el cuerpo y un valor oculto que determine si el formulario ha sido enviado.

```
?>
<FORMACTION="email.php" METHOD=POST>
Recipient's  Email Address:
→ <INPUT TYPE=TEXT NAME="MailTo"
→ SIZE="50"><BR>
Your Email Address: <INPUT TYPE=TEXT
→ NAME="MailFrom" SIZE="50"><BR>
Email Subject: <INPUT TYPE=TEXT
→ NAME="Subject" SIZE="80"><BR>
Email Body:<TEXTAREA NAME="Body"
→ ROWS="10" COLS="50">
</TEXTAREA><P>
<INPUT TYPE=HIDDEN NAME=BeenSubmitted
→ VALUE=TRUE>
<INPUT TYPE=SUBMIT NAME="SUBMIT"
→ VALUE="Submit!">
```

7. Cierre el formulario e incluya el archivo de pie de página.

```
</FORM>
<?
require ("footer.php");
?>
```

8. Guarde el *script* como email.php, cárguelo en el servidor y pruébelo en el navegador web (véanse las Figuras 13.11, 13.12, 13.13 y 13.14).

CONSEJO

- Es posible enviar correo electrónico con archivos adjuntos, aunque esto requiere un código más sofisticado (normalmente involucra a objetos). Afortunadamente, varios programadores han desarrollado soluciones funcionales que están disponibles para su uso. Consulte el Apéndice C para obtener más información.

DEPURACIÓN 14

Puede parecer irónico que el último capítulo del libro trate de un tema con el que probablemente se haya estado encontrando a cada paso, la depuración del código. Sin embargo, ya que ha aprendido mucho a lo largo del camino, la depuración le parecerá más natural, y este capítulo le ayudará a conocer algunos trucos. Desafortunadamente, no importa lo buen programador que sea, los ocasionales errores de descuido siempre le pillarán fuera de guardia, así que es mejor que esté preparado para cuando esto ocurra.

En este capítulo aprenderá varias técnicas para reducir, informar y gestionar errores en su código. Aunque evitar los errores sea importante en el desarrollo web, también tiene que saber cómo tratar los errores que inevitablemente se producirán al crear sitios web.

Errores comunes

El tipo de error más común con el que se encontrará es sintáctico, un fallo al completar los detalles finales, por así decirlo. Errores como éste terminarán en mensajes de error como el de la Figura 14.1. Éstos son los primeros tipos de errores que capturará PHP porque la sintaxis del código se evalúa antes de que la ejecución tenga lugar. Para evitar cometer este tipo de fallos al programar, asegúrese de:

- Terminar cada línea ejecutable de código con un punto y coma.
- Utilizar una comilla, paréntesis, corchete o llave de cierre para cada comilla, paréntesis, corchete o llave de apertura.
- Cree secuencias de escape, utilizando la barra invertida, con todas las comillas dobles dentro de la instrucción print().

Una cosa que también debería entender sobre los errores sintácticos es que sólo porque un mensaje de error PHP diga que el error ocurre en la línea 12, por ejemplo, no significa que el error esté realmente en esa línea. No es raro que haya diferencia entre lo que PHP piensa que es la línea 12 y lo que su editor de textos indica que es la línea 12. De modo que aunque la dirección de PHP sea útil para localizar un problema, entienda el número de línea enunciado más como un punto de inicio que como una posición absoluta.

El segundo tipo de error que encontrará resulta al intentar hacer algo que no se puede hacer. Estos errores ocurren, por ejemplo, cuando setcookie() o header() se llaman una vez que el navegador web ha recibido código HTML, cuando se llama a una función sin los argumentos apropiados o cuando se intenta escribir un archivo que no tiene los permisos adecuados. PHP descubre estos errores al intentar ejecutar el código (véase la Figura 14.2).

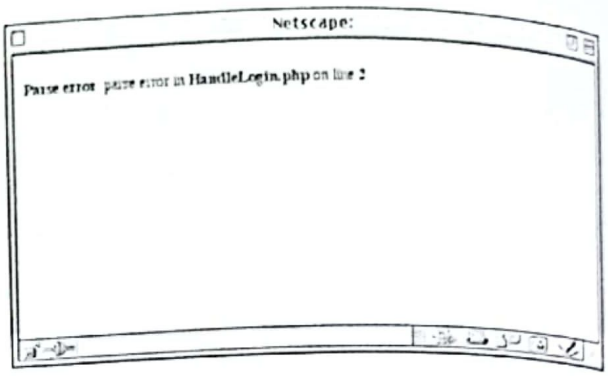


Figura 14.1. Los errores generados por fallos sintácticos son los más comunes en los scripts.

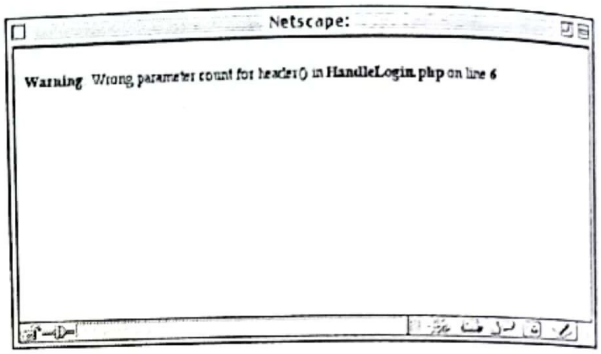


Figura 14.2. Los avisos son creados por PHP al intentar ejecutar el script y generalmente son provocados por el mal uso de las referencias a las funciones (por ejemplo, por escribir mal el nombre de una función o por utilizar un número de argumentos incorrecto al llamar a una función).

Script 14.1. La página original HandleLogin.php era ciertamente aceptable, pero se podría mejorar utilizando la función headers_sent().

script

```
1  <?php
2  if (($UserName == "Larry") &&
   ($Password == "LarryPass")) {
3      header ("Location:
       index.php?UserName=$UserName");
4      exit;
5  } else {
6      header ("Location:
       login.php?Message=Invalid");
7      exit;
8  }
9  ?>
```

Script 14.2. La función headers_sent() ayuda a evitar la confusión común de enviar una cabecera (o una cookie) una vez que el navegador haya recibido información (véase la Figura 14.3).

script

```
1
2  <?php
3  if ( headers_sent() ) {
4      print ("Cannot process your request
       due to a system error!\n");
5  } else {
6      if (($UserName == "Larry") &&
       ($Password == "LarryPass")) {
7          header ("Location:
           index.php?UserName=$UserName");
8          exit;
9      } else {
10         header ("Location:
           login.php?Message=Invalid");
11         exit;
12     }
13 }
14 ?>
```

El tercer tipo de errores que verá, y con demasiada frecuencia, no son errores PHP en absoluto, sino errores del programador, también llamados errores lógicos. (Algunos programadores dirían que todos los errores son del programador, porque PHP por sí mismo no puede cometer errores.) Una razón para que ocurran tales errores es que usted falle al utilizar el nombre de variable apropiado. Si hace eso, no verá un mensaje de error como los de las Figuras 14.1 y 14.2, sólo verá resultados extraños e impredecibles. Éstos pueden ser los más difíciles de discernir porque PHP no le dará ninguna pista sobre como cuál es el problema o cómo solucionarlo. ¡Nada excepto la comprobación exacta y un trabajo de investigación inteligente los solucionará!

Para el primer ejercicio vamos añadir una comprobación segura al *script* HandleLogin.php del Capítulo 13, para evitar la ocurrencia común de enviar una cabecera después de que el navegador web haya recibido algo de código HTML o un espacio en blanco.

Cómo evitar un error común

1. Abra HandleLogin.php en el editor de textos (véase el *Script* 14.1).
2. Cree una línea en blanco pulsando la tecla Intro antes de la etiqueta PHP inicial (véase el *Script* 14.2).

Esta línea en blanco, aunque pueda parecer insignificante, generará el mensaje de error de la Figura 14.3.

3. Después de la etiqueta PHP inicial, añada la siguiente condición:

```
if ( headers_sent() ) {
   print ("Cannot process your request
   → due to a system error!\n");
} else {
```

La función headers_sent() devuelve el valor TRUE si algo de código HTML, espacio en blanco, etc. ya se ha enviado al navegador. Si ha ocurrido esto, el intento de utilizar la función header() creará un enérgico aviso en el navegador del usuario. En lugar de que esto ocurra, si las cabeceras se hubieran enviado, esta página mostraría ahora un error genérico del sistema.

Si las cabeceras no se hubieran enviado, la función headers_sent() sería falsa y el resto de la página se ejecutaría de forma normal.

4. ¡No olvide cerrar la condición headers_sent() antes de cerrar la etiqueta PHP!

```
}
```

5. Guarde el *script* como HandleLogin.php, cárguelo en el servidor en el mismo directorio que login.php y pruebe ambas páginas en el navegador (véase la Figura 14.4).

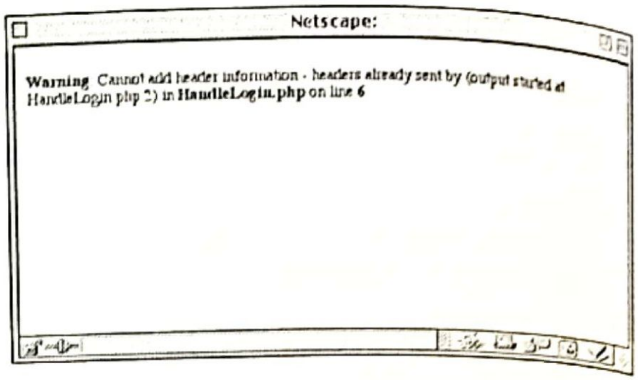


Figura 14.3. *El intento de establecer una cookie o de enviar una cabecera después de que el navegador ya ha recibido cualquier tipo de información produce un error parecido a éste.*

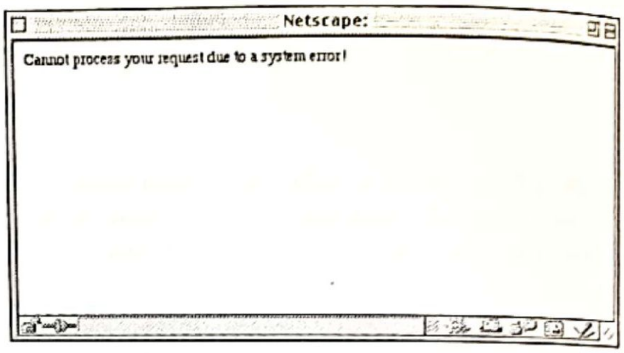


Figura 14.4. *Utilizando la función headers_sent(), puede imprimir un mensaje menos confuso para el usuario. Todavía no resuelve el problema, pero es bueno que los usuarios vean mensajes como el mostrado en la Figura 14.3.*

CONSEJO

- Algunos editores de textos, como BBEdit para Macintosh, incluyen utilidades para comprobar el equilibrio de paréntesis, corchetes y comillas.

Tabla 14.1.

Niveles de informe de error

Número	Constante	Significado
1	E_ERROR	Error fatal en tiempo de ejecución.
2	E_WARNING	Avisos en tiempo de ejecución no fatales.
4	E_PARSE	Análisis en tiempo de compilación.
8	E_NOTICE	Notificaciones en tiempo de ejecución.
16	E_CORE_ERROR	Errores fatales que ocurren durante el arranque de PHP (sólo en PHP 4).
32	E_CORE_WARNING	Avisos (errores no fatales) que ocurren durante el arranque de PHP (sólo en PHP 4).
64	E_COMPILE_ERROR	Errores fatales en tiempo de compilación (sólo en PHP 4).
128	E_COMPILE_WARNING	Avisos en tiempo de compilación (errores no fatales) (sólo en PHP 4).
256	E_USER_ERROR	Mensaje de error generado por el usuario (sólo en PHP 4).
512	E_USER_WARNING	Mensaje de atención generado por el usuario (sólo en PHP 4).
1024	E_USER_NOTICE	Mensaje de notificación generado por el usuario (sólo en PHP 4).
	E_ALL	Todo lo anterior, en la medida que esté soportado.

Informes de errores y registro

PHP tiene bastante buen soporte interno tanto para los informes de errores como para la manipulación de errores. Aunque estos dos conceptos no le ayudarán a minimizar sus errores, le ofrecerán otra forma de aprender de ellos.

La función error_reporting() en PHP se utiliza para especificar sobre qué tipo de errores PHP se debería informar. La línea error_report (0); desactiva el informe completamente. Todavía ocurrirán errores, sólo que no sabrá más sobre ellos. Por el contrario, la línea error_reporting (E_ALL); informará de todos los errores que ocurran, lo que PHP no hace necesariamente. (El manual de PHP enumera todos los posibles niveles de informe de error que se pueden configurar.) Utilizará esta función en la última sección del capítulo, "Uso de la instrucción die".

La función error_reporting() le indica a PHP sobre qué errores informar, mientras que la función error_log() instruye a PHP sobre cómo clasificar los errores que se produzcan.

```
error_log ("mensaje", "tipo",
→ "destino");
```

Cuando algo va mal, puede hacer que PHP le deje una anotación en un archivo (por ejemplo, registrar los errores en un archivo de registro) o incluso enviarle un mensaje de correo electrónico directamente. El uso prudente de la función error_log() permitirá a los administradores de web y a los programadores vigilar mejor lo que va a suceder en la Web. Para que PHP le envíe un mensaje de correo electrónico cuando ocurra un error, añada esta línea al código:

```
error_log ("message", "1",
→ "php@DMCinsights.com");
```

Modificaremos el *script* email.php del Capítulo 13 para que cree una anotación en un archivo cada vez que no pueda enviar satisfactoriamente un mensaje de correo electrónico.

Cómo usar la función error_log()

1. Abra email.php en su editor de textos (véase el *Script* 14.3).
2. Tras la línea 9, añada lo siguiente (véase el *Script* 14.4):

```
error_log ("Unable to send an email to
→ $MailTo from $MailFrom at " . time() .
→ "\n", 3, "errors.txt");
```

Script 14.3. *Éste es el script email.php del Capítulo 13. Actualmente no tiene la capacidad de registrar errores, lo que lo haría más útil.*

script

```
1  <?php
2  $PageTitle = "Sending Emails";
3  require ("header.php");
4  if ($BeenSubmitted) {
5      if ($MailTo) {
6          if (mail($MailTo, $Subject, $Body, "From: $MailFrom")) {
7              print ("<B><CENTER><FONT COLOR=BLUE>Your email has been successfully sent!
               </FONT></CENTER></B>\n");
8          } else {
9              print ("<B><CENTER><FONT COLOR=RED>Your email was not successfully sent due to a
               system error!</FONT></CENTER></B>\n");
10         }
11     } else {
12         print ("<B><CENTER><FONT COLOR=RED>Please enter the recipient's mail to
           address!</FONT></CENTER></B>\n");
13     }
14 }
15 ?>
16 <FORM ACTION="email.php" METHOD=POST>
17 Recipient's Email Address: <INPUT TYPE=TEXT NAME="MailTo" SIZE="50"><BR>
18 Your Email Address: <INPUT TYPE=TEXT NAME="MailFrom" SIZE="50"><BR>
19 Email Subject: <INPUT TYPE=TEXT NAME="Subject" SIZE="80"><BR>
20 Email Body:<TEXTAREA NAME="Body" ROWS="10" COLS="50">
21 </TEXTAREA><P>
22 <INPUT TYPE=HIDDEN NAME=BeenSubmitted VALUE=TRUE>
23 <INPUT TYPE=SUBMIT NAME="SUBMIT" VALUE="Submit!">
24 </FORM>
25 <?
26 require ("footer.php");
27 ?>
```

Cada vez que la función mail() no pueda trabajar, se registrará un mensaje sencillo en un archivo de registro indicando que $MailFrom no pudo enviar un mensaje a $MailTo en un momento en particular (establecido utilizando la función time()). Un administrador inteligente podría ver este archivo y seguir a los usuarios a quienes hubiera afectado o podría determinar cuándo dejó de trabajar correctamente la función mail().

Script 14.4. La versión modificada de email.php emplea la función error_log() para registrar cualquier problema que ocurra al enviar mensajes de correo electrónico. También puede utilizar error_log() para enviar por correo electrónico un mensaje, aunque ¡no podrá si el problema radica en enviar el mensaje!

script

```
1  <?php
2  $PageTitle = "Sending Emails";
3  require ("header.php");
4  if ($BeenSubmitted) {
5      if ($MailTo) {
6          if (mail($MailTo, $Subject, $Body, "From: $MailFrom")) {
7              print ("<B><CENTER><FONT COLOR=BLUE>Your email has been successfully sent!
               </FONT></CENTER></B>\n");
8          } else {
9              print ("<B><CENTER><FONT COLOR=RED>Your email was not successfully sent due to a
               system error!</FONT></CENTER></B>\n");
10             error_log ("Unable to send an email to $MailTo from $MailFrom at " . time() . "\n",
               3, "errors.txt");
11         }
12     } else {
13         print ("<B><CENTER><FONT COLOR=RED>Please enter the recipient's mail to
           address!</FONT></CENTER></B>\n");
14     }
15 }
16 ?>
17 <FORM ACTION="email.php" METHOD=POST>
18 Recipient's Email Address: <INPUT TYPE=TEXT NAME="MailTo" SIZE="50"><BR>
19 Your Email Address: <INPUT TYPE=TEXT NAME="MailFrom" SIZE="50"><BR>
20 Email Subject: <INPUT TYPE=TEXT NAME="Subject" SIZE="80"><BR>
21 Email Body:<TEXTAREA NAME="Body" ROWS="10" COLS="50">
22 </TEXTAREA><P>
23 <INPUT TYPE=HIDDEN NAME=BeenSubmitted VALUE=TRUE>
24 <INPUT TYPE=SUBMIT NAME="SUBMIT" VALUE="Submit!">
25 </FORM>
26 <?
27 require ("footer.php");
28 ?>
```

3. Grabe el *script* como email.php y cárguelo en el servidor.

   Ahora tendrá que crear un documento en blanco llamado errors.txt para que la función error_log() escriba en él.

4. Cree un nuevo documento en blanco en el editor de textos.

5. Guarde el documento como errors.txt y cárguelo en el servidor en el mismo directorio que email.php.

6. Configure los permisos de errors.txt para que todos tengan privilegios de escritura sobre él.

7. Pruebe email.php en el navegador (véanse las Figuras 14.5, 14.6 y 14.7).

Figura 14.5. Hemos reescrito la página email.php para registrar los problemas que pudieran surgir.

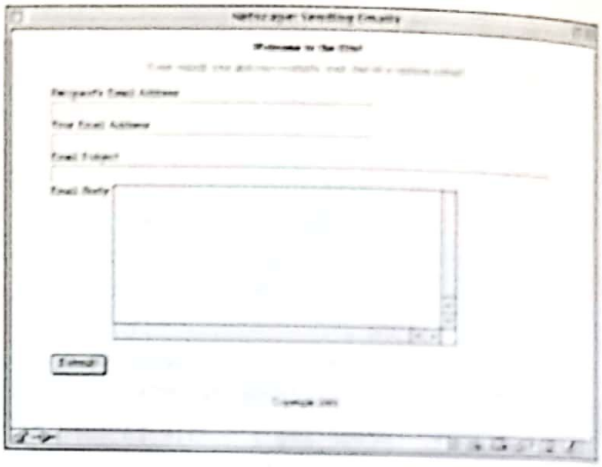


Figura 14.6. Todo lo que el usuario ve cuando hay un problema es un simple mensaje en la parte superior del formulario. Sin embargo, el administrador del sistema puede comprobar el registro de errores para recabar más información (véase la Figura 14.7).

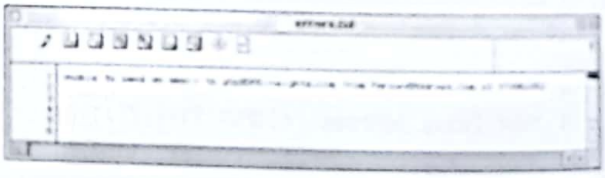


Figura 14.7. Cuando se realiza un registro de errores, se guarda aquella información que se piensa es adecuada. Este registro lista los valores de $MailTo y $MailFrom junto con una fecha.

CONSEJO

- Si la aplicación de correo electrónico que el servidor utiliza para enviar mensajes está funcionando correctamente, este *script* no registrará ningún error.

  Si quiere ver qué haría el *script* si no pudiera enviar un mensaje, cambie la línea 6 por esta otra:

```
if (!mail($MailTo, $Subject, $Body,
 "From: $MailFrom")) {
```

Depuración del trabajo

Cuando se llega a los tipos de error más variables, los errores del programador, PHP no le ayudará diciéndole en qué línea comenzar a mirar. De hecho, todo lo que probablemente verá es que los resultados no serán los que está esperando. Para resolver estos acertijos, tendrá que hacer un poco de trabajo de investigación para ver los errores que se cometieron y dónde se cometieron.

Hay tres herramientas y técnicas que le ayudan en su causa:

- Utilizar comentarios.
- Utilizar la instrucción print().
- Seguir la pista de las variables.

Al igual que puede utilizar comentarios para documentar sus *scripts*, también los puede utilizar para excluir líneas problemáticas. Si PHP le está dando un error en la línea 12, entonces convierta en comentario esa línea y compruebe si el error desaparece. Si no, entonces sabrá que el error está en algún otro sitio.

En los *scripts* más complicados, frecuentemente utilizamos la instrucción print() para dejarnos notas sobre lo que ocurre mientras el *script* se ejecuta. Cuando un *script* tiene varios pasos, puede que no sea fácil decir si el problema ocurre en el paso dos o en el paso cinco. Utilizando la instrucción print() puede reducir el problema al paso específico.

Finalmente, es bastante fácil que un *script* no funcione porque usted hizo referencia a la variable equivocada o a la variable correcta con el nombre equivocado. Para comprobar estas posibilidades, utilice la instrucción print() para conocer los valores de las variables mientras progresa el *script*. Entonces sabrá seguro si las variables están causando un problema.

Ilustraremos estas técnicas modificando la página HandleLogin.php anterior para mostrar cómo la depuraríamos si estuviéramos obteniendo resultados impredecibles.

Cómo depurar un *script*

1. Abra HandleLogin.php en el editor de textos (véase el *Script* 14.2).

2. Borre la línea en blanco inicial creada anteriormente para hacer que la etiqueta PHP inicial sea la primera línea de la página (véase el *Script* 14.5).

```
<?php
```

3. Después de la condición headers_sent() (línea 2), añada una instrucción print():

```
if ( headers_sent() ) {
   print ("Headers have been sent. Not
   → attempting to verify.<P>\n");
   print ("Cannot process your request
   → due to a system error!\n");
```

Si el problema es debido a que ya se han enviado las cabeceras, al preguntar esto le quedaría más claro.

4. Después de else (línea 5), añada tres instrucciónes print() más.

```
} else {
   print ("Headers have not been sent.
   → Attempting to verify.<P>\n");
   print ("UserName is $UserName.
   → <P>");
   print ("Password is $Password.
   → <P>");
```

La primera instrucción print() tiene correlación con la añadida arriba, indicando que el problema no está relacionado con header().

Las dos siguientes instrucciones print le dirán exactamente qué valores se recibieron para $UserName y $Password. Puede seguir el valor de las variables a través de un *script* imprimiéndolas en los momentos clave.

Script 14.5. Si la página HandleLogin.php estuviera actuando de forma extraña y dándole resultados que no pudiera entender, podría modificarla así con el fin de depurarla. La combinación de usar instrucciones print(), seguir las variables y convertir en comentarios líneas de código le ayudará a solucionar rápidamente cualquier problema.

script

```
1  <?php
2  if ( headers_sent() ) {
3      print ("Headers have been sent.
       Not attempting to verify.<P>\n");
4      print ("Cannot process your request
       due to a system error!\n");
5  } else {
6      print ("Headers have not been sent.
       Attempting to verify.<P>\n");
7      print ("UserName is $UserName. <P>");
8      print ("Password is $Password. <P>");
9      if (($UserName == "Larry") &&
       ($Password == "LarryPass")) {
10         print ("<P>Match!");
11 // header ("Location: index.php?
   UserName=$UserName");
12         exit;
13     } else {
14         print ("<P>Not a Match!");
15 // header ("Location: login.php?
   Message=Invalid");
16         exit;
17     }
18 }
19 ?>
```

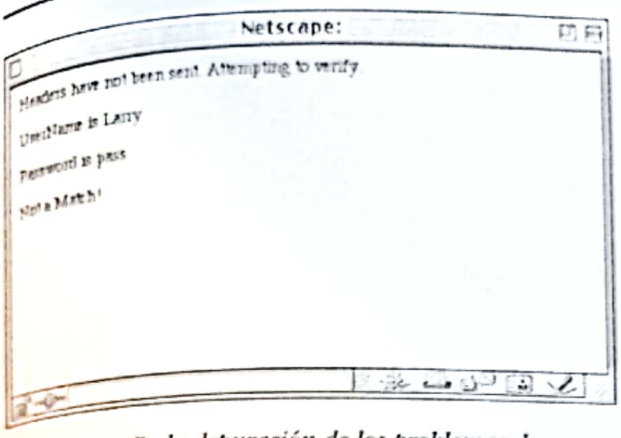


Figura 14.8. En la depuración de los problemas de un script, la impresión de mensajes detallando el proceso por el que atraviesa PHP puede ser inestimable. Aquí puede determinar que el problema no está causado por las cabeceras o por una referencia errónea a las variables.

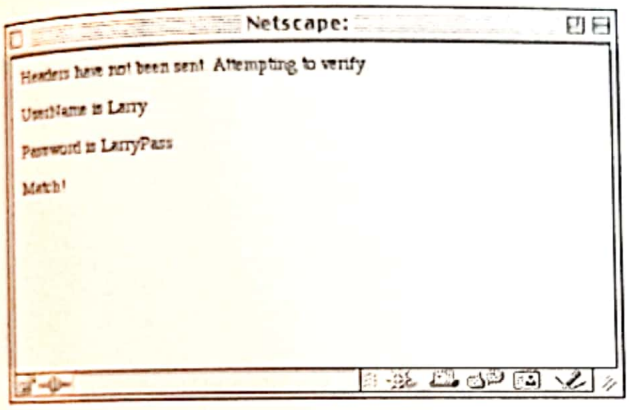


Figura 14.9. De nuevo, en pantalla se reciben las especificaciones de lo que HandleLogin.php está haciendo. Si esta página provoca errores, puede arrinconarlos bajo las líneas header().

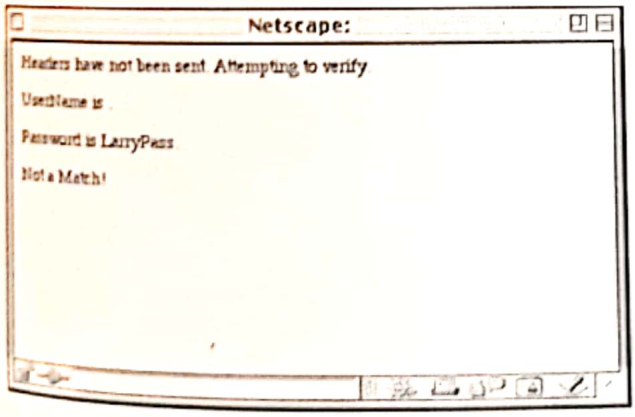


Figura 14.10. Esto es muy común. La razón de que no haya una coincidencia es que se está haciendo referencia a un nombre de variable erróneo. Observe que no existe un valor para UserName. Esto significa que hay una diferencia entre lo que utilizan login.php y HandleLogin.php para referirse a este valor.

5. Después de la segunda condición if (línea 9), añada otra instrucción print() y ponga como comentario la línea header().

```
if (($UserName == "Larry") &&
→ ($Password == 'LarryPass')) {
    print ("<P>Match!");
    header ("Location: index.php?
    → UserName=$UserName");
    exit;
} else {
```

Imprimiendo la palabra Match! sabrá si el sistema verificó o no el nombre de usuario y la contraseña, lo que le puede permitir excluir eso como una posible causa de error.

Como está depurando el *script*, puede eliminar el efecto de la llamada a header() situando dos barras inclinadas al comienzo de la línea para que sea un comentario. Cuando esté preparado, lo único que tiene que hacer es eliminar las barras de la línea para que deje de ser un comentario y el comando se restablecerá.

6. Una vez más, añada otra instrucción print() y comente la función header() después de else (línea 13).

```
        print ("<P>Not a Match!");
        header ("Location: login.php?
        → Message=Invalid");
        exit;
    }
}
?>
```

7. Guarde el *script* como HandleLogin.php, cárguelo en el servidor en el mismo directorio que login.php y pruebe ambas páginas en el navegador (véanse las Figuras 14.8, 14.9 y 14.10).

Uso de la instrucción die

Un último tipo de error que encontrará es el error circunstancial. Este error sólo ocurre cuando algo va mal en el servidor. Afortunadamente, son menos frecuentes que los errores de programación. Por ejemplo, si la base de datos de MySQL no se está ejecutando, cada función relacionada con la base de datos generará un error. O bien, como vio con el ejemplo email.php, si la aplicación que envía mensajes de correo electrónico del servidor no está funcionando (en un servidor Unix), eso también causaría un problema. Aunque estos errores del sistema estén más allá del control del programador, debería recordar esta posible ocurrencia cuando codifique.

Puede detener estos errores en sus rastreos utilizando la instrucción die, que indica a PHP que cese la ejecución de un *script* cuando suceda un error. También puede imprimir un mensaje de error en el navegador o en la llamada a una función.

Por ejemplo,

```
$Link = mysql_connect ($Host, $User,
→ $Password) or die ("Couldn't connect
→ to the database.");
```

Si, por cualquier razón, PHP no puede conectar con la base de datos, el *script* dejará de ejecutarse y se imprimirá el mensaje "Couldn't connect to the database".

```
$Link = mysql_connect ($Host, $User,
→ $Password) or die (mysql_error());
```

En este ejemplo, si PHP no pudiera conectar con la base de datos, die llamaría a la función mysql_error(). Esta función imprime el error real que genera la base de datos MySQL.

Los ejemplos anteriores funcionan porque si PHP determina que la primera parte de la condición (es decir, mysql_connect) es FALSE, se promulgará la segunda parte de la condición (que es la instrucción die) para garantizar que la línea completa se evalúa como TRUE. En otras palabras, la línea dice que si la conexión a la base de datos no funciona, haga esto. Si la conexión a la base de datos se realiza, esto es todo lo que se necesita.

Como ejemplo de esto, añadirá la instrucción die a un *script* de base de datos muy sencillo que conecta

Script 14.6. Para los scripts que utilizan varias líneas de código, cada una de las cuales se basa en la conclusión satisfactoria de la precedente, la instrucción die es inestimable reduciendo y manipulando errores. Ya que die administrará errores potenciales, el informe de errores interno de PHP se ha desactivado (compare la Figura 14.11 con la Figura 14.13).

script

```
1  <HTML>
2  <HEAD>
3  <TITLE>Die!</TITLE>
4  <BODY>
5  <?php
6  // Establece las variables para el acceso
   a la base de datos:
7  $Host = "localhost";
8  $User = "username";
9  $Password = "password";
10 $DBName = "database";
11 error_reporting(0);
12 $Link = mysql_connect ($Host, $User,
   $Password) or die("Couldn't connect to
   the database!");
13 mysql_select_db ($DBName, $Link) or
   die(mysql_error());
14 mysql_close ($Link);
15 ?>
16 Testing the die statement!
17 </BODY>
18 </HTML>
```

con MySQL y después intenta seleccionar la base de datos.

Cómo utilizar la instrucción die

1. Cree un nuevo documento PHP en el editor de textos.

2. Comience con la cabecera HTML estándar (véase el *Script* 14.6):

```
<HTML>
<HEAD>
<TITLE>Die!</TITLE>
<BODY>
```

3. Abra la sección PHP con la etiqueta PHP inicial y después configure las variables para el acceso a la base de datos.

```
<?php
$Host = "localhost";
$User = "username";
$Password = "password";
$DBName = "database";
```

4. Desactive el informe de errores.

```
error_reporting(0);
```

Ya que la instrucción die administrará los errores que sucedan, desactivará el método predeterminado de PHP de informar de los errores para evitar la redundancia.

5. Conecte con MySQL.

```
$Link = mysql_connect ($Host, $User,
› $Password) or die("Couldn't connect
› to the database!");
```

Esta línea de código indica a PHP que intente conectar a MySQL. Si no pudiera hacerlo por cualquier razón, se imprimiría el mensaje "Couldn't connect to the database" y se detendría la ejecución del *script*. Es importante detener la ejecución inmediatamente, porque si PHP no puede conectar con la base de datos, las dos líneas siguientes de código, que se basan en esa conexión, provocarían mensajes de error. (En realidad, desactivando error_reporting, no se informará de ningún error aunque los errores

ocurran. Sin tener en cuenta este hecho, es mejor detener los *scripts* con problemas en el primer instante posible.)

6. Seleccione la base de datos.

```
mysql_select_db ($DBName, $Link) or
-> die(mysql_error());
```

Aquí, si PHP no puede seleccionar la base de datos (determinada por $DBName), cesará la ejecución de la página y se imprimirá el error generado por MySQL.

7. Cierre el enlace con MySQL y la sección PHP.

```
mysql_close ($Link);
?>
```

8. Cree un mensaje sencillo y termine la página HTML.

```
Testing the die statement!
</BODY>
</HTML>
```

Hemos añadido esta línea de texto para que la página tenga algún contenido. Esta página se podría modificar fácilmente para enviar una base de datos.

9. Guarde el *script* como die.php, cárguelo en el servidor y pruébelo en el navegador (véanse las Figuras 14.11, 14.12 y 14.13).

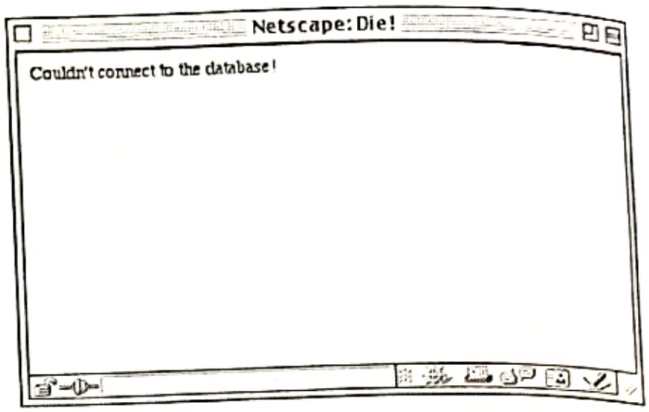


Figura 14.11. *Desactivando el informe de errores y utilizando la instrucción die, es posible presentar al usuario un mensaje de error más significativo. Además, el script cesará la ejecución de la página si no pudiera completar este primer paso.*

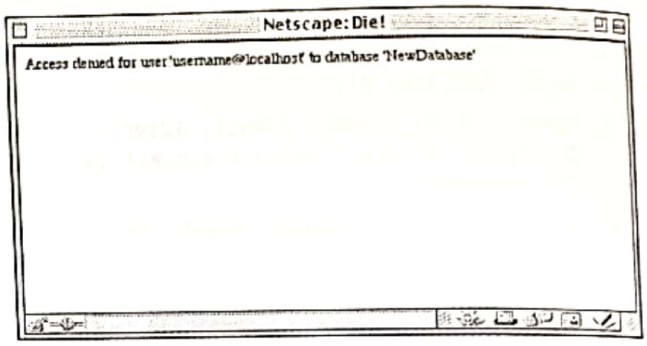


Figura 14.12. *Aquí, la instrucción die llama a la función mysql_error(), que imprime el mensaje de error actual generado por la base de datos de MySQL.*

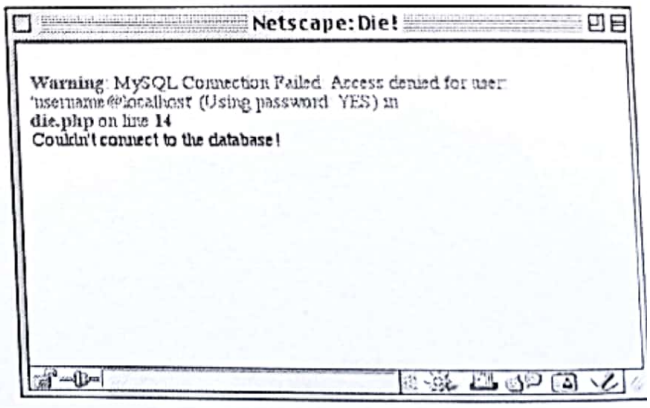


Figura 14.13. *Si el informe de errores no está desactivado, el usuario final verá tanto el mensaje de error de PHP como el mensaje creado por la instrucción die. Compare esto con la Figura 14.11, donde el informe de errores está desactivado.*

CONSEJOS

- Puede prevenir los mensajes de error utilizando el símbolo @ antes de cualquier llamada a una función. Llamado el operador de control de error, el símbolo @ hará caso omiso de los errores provocados por la función que le preceda. Se utiliza así:

```
$Link = @mysql_connect ($Host, $User,
→ $Password);
```

  A diferencia de die, @ no le da la opción de imprimir un mensaje o llamar a una función.

- También puede utilizar die para llamar a sus propias funciones. Por ejemplo, si tiene una función creada con el fin de imprimir mensajes de error en un formato particular, die podría llamarla.

```
$Link = mysql_connect ($Host, $User,
→ $Password) or die (print_message
→ ("Couldn't connect."));
```

INSTALACIÓN Y CONFIGURACIÓN

Muchos usuarios de PHP, especialmente los que ya conocen el lenguaje, no necesitan aprender a instalarlo y configurarlo en un servidor. Sin embargo, si decidiese tomar este camino, verá que no es muy complicado y su precio, gratis, hace de él una opción más que apetecible. Cuando haga la instalación de PHP, asegúrese de que la versión soporta las funciones que desea y si dichas funciones están disponibles para el uso en programación.

Existen dos puntos básicos que hay que plantearse antes de iniciar la instalación. Primero, ¿con qué sistema operativo funciona el servidor? Segundo, ¿qué aplicación del servidor web empleará? En esta sección le mostraremos cómo se instala PHP en una variante de Unix (por ejemplo, Linux) y en los servidores de Windows. Respecto al servidor de web, Apache es la aplicación más utilizada en los servidores Unix, y probablemente utilizará ésta o IIS (*Internet Information Server*) en las computadoras con Windows.

Después de estas consideraciones, existen otras como: qué base(s) de datos soporta, la capacidad para crear imágenes, archivos PDF o Shockwave, interactividad XML, etc. Se debería tener en cuenta cada una de estas propiedades antes de instalarlo.

El manual de PHP, especialmente en la versión *online*, cubre la instalación con una gran cantidad de detalles. En esta sección le daremos una lista básica de la instalación, tanto en Linux como en una computadora con Windows 2000, que, exceptuando cualquier imprevisto, deberían ser suficientes para la mayoría de los usuarios.

Instalación de PHP y Apache en un servidor Linux

La popularidad de Linux como sistema operativo, especialmente para usarlo como servidor web, ha aumentado enormemente. El hecho de ser gratis y muy estable ha sido un gran tanto a su favor. Sin embargo, la dificultad al estructurar una computadora con Linux, todavía puede ser un obstáculo que franquear, aunque ahora este problema ha mejorado sensiblemente.

Para poder efectuar cualquier tipo de instalación en un servidor Linux, debe tener la contraseña raíz y un acceso. El servidor no tiene necesariamente que estar *on-line*, aunque puede trabajar perfectamente con una computadora, o crear una minired conectándose con otra computadora para comunicarse entre ellas.

Cómo instalar Apache y PHP

1. Descargue las versiones más estables y actuales a un directorio común, como /usr/local/ (véase la Figura A.1).

   Los archivos de Apache se encuentran en http://www.Apache.org.

2. Si los archivos tienen la extensión .gz, utilice estas líneas para descomprimirlos mediante unzip (pulse Intro después de escribir cada línea individualmente):

   ```
   gunzip php-4.0.4pl1.tar.gz
   gunzip apache_1.3.14.tar.gz
   ```

   Aunque para esta demostración nos referiremos a versiones específicas de PHP y Apache que hemos instalado, asegúrese de que cuando haga la instalación, hace referencia a su propia versión (por ejemplo, php-3.0.3pl1).

3. Desempaquete los archivos:

   ```
   tar -xvf php-4.0.4pl1.tar
   tar -xvf apache_1.3.14.tar
   ```

   Tendrá que escribir estas dos líneas para obtener los archivos puros. Esta acción cambiará cada paquete de una versión comprimida a la carpeta que contenga todos los archivos específicos. Tras escribir cada línea, podrá visualizar mucha información en pantalla (véase la Figura A.2).

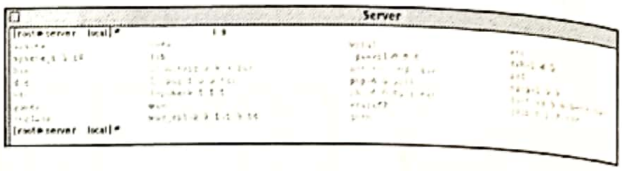


Figura A.1. El comando ls lista todos los archivos del directorio actual. PHP y Apache están en esta carpeta junto con otras bibliotecas y aplicaciones.

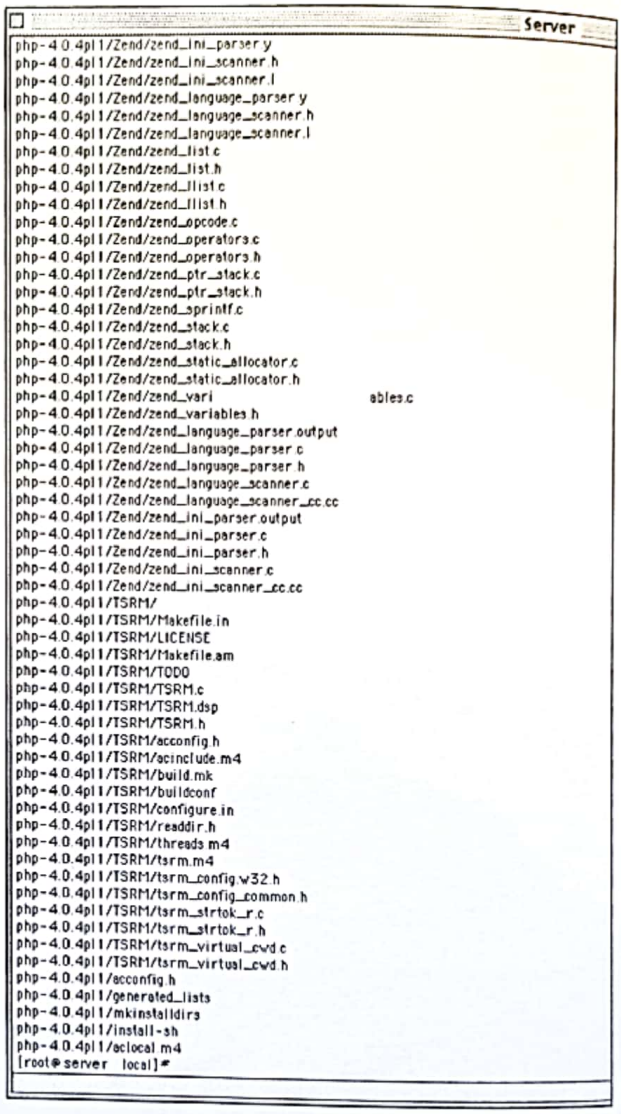


Figura A.2. Cuando utilice tar para expandir un paquete, verá todos los archivos que contiene este último.

```
+-----------------------------------------------------------
| You now have successfully built and installed the      |
| Apache 1.3 HTTP server. To verify that Apache actually |
| works correctly you now should first check the         |
| (initially created or preserved) configuration files   |
|                                                        |
| /usr/local/apache/conf/httpd.conf
|                                                        |
| and then you should be able to immediately fire up     |
| Apache the first time by running:                      |
|                                                        |
| /usr/local/apache/bin/apachectl start
|                                                        |
| Or when you want to run it with SSL enabled use:       |
|                                                        |
| /usr/local/apache/bin/apachectl startssl
|                                                        |
| Thanks for using Apache.     The Apache Group          |
|                        http://www.apache.org/          |
+-----------------------------------------------------------
[root@server  apache_1.3.14]#
```

Figura A.3. Después de configurar, construir e instalar Apache, le mostrará este mensaje indicándole que todo funciona correctamente.

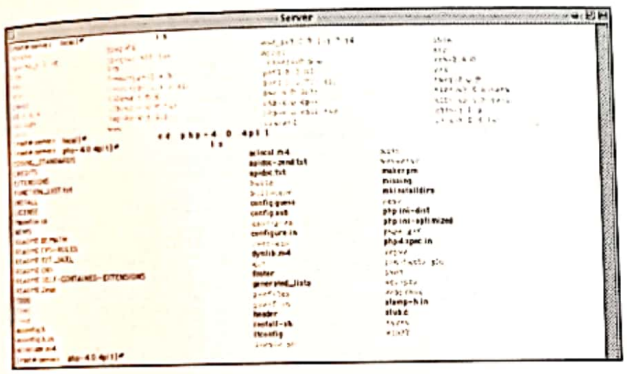


Figura A.4. Todas las carpetas y archivos descomprimidos se encuentran ahora ubicados en la carpeta PHP.

```
----------+
| License:
|
| This software is subject to the PHP License, available in
this  |
| distribution in the file LICENSE.  By continuing this inst
allation |
| process, you are bound by the terms of this license agreem
ent.  |
| If you do not agree with the terms of this license, you mu
st abort |
| the installation process at this point.
|
+-----------------------------------------------------------
----------+

Thank you for using PHP.

[root@server  php-4.0.4pl1]#
```

Figura A.5. La configuración, construcción e instalación satisfactorias de PHP finalizan con la licencia y los agradecimientos.

4. Vaya a la carpeta Apache que acaba de crear, e inicie la configuración (véase la Figura A.3).

```
cd ../apache_1.3.14
./configure -prefix=/www
```

El comando cd le permite cambiar de carpeta. Cuando esté en la carpeta Apache, puede preconfigurarla.

5. Vaya a la carpeta PHP (véase la Figura A.4) y ejecute esta configuración.

```
cd ../php-4.0.4pl1
./configure -with-apache=../
→ apache_1.3.14 -enable-track-vars
```

Los parámetros de la configuración que utilice dependerán de las funciones que quiere que soporte PHP. Cada comando de configuración de la clasificación necesita que se indique el directorio donde se puede encontrar. En este ejemplo queremos que PHP funcione con Apache, por lo que nos dirigiremos donde se encuentra la carpeta Apache. La opción enable-track-vars es necesaria para que PHP emplee las formas de HTML.

6. Realice la instalación de PHP (véase la Figura A.5).

```
make
make install
```

Estas líneas de código instalarán la configuración de PHP para que pueda utilizarse con Apache.

7. Vuelva a la carpeta Apache, ejecute la configuración e instálelo.

```
cd ../apache_1.3.14
./configure -activate-module=
→ rc/modules/php4/libphp4.a
make
make install
```

Ahora que ya ha configurado made e instalado Apache, ya puede ejecutarlo.

8. Copie el archivo php.ini en la nueva ubicación.

```
cp /usr/local/php-4.0.4pl1/
→php.ini-dist /usr/local/lib/php.ini
```

El archivo php.ini indica el modo de trabajar de PHP. Se encuentra en el paquete como php.ini-dist. Para que Apache pueda utilizarlo, debe copiarlo al directorio adecuado y renombrarlo.

9. Inicie Apache.

```
bin/apachect1 start
```

Como cualquier otra aplicación, hay que iniciar Apache para que funcione. Si está en la carpeta Apache, esta línea iniciará el servidor web.

10. Haga una prueba para ver si Apache y PHP están trabajando con el URL adecuado y con el navegador web (véase la Figura A.6).

Si el sevidor está *online*, puede visitar el URL http://www.DMCinsights.com. Si no lo está, diríjase a http://localhost/ o a la dirección IP que se ha asignado a la computadora.

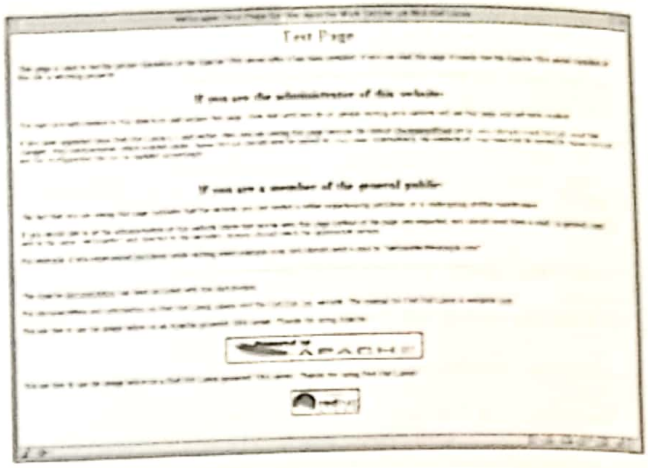


Figura A.6. Apache utiliza esta página predeterminada para indicar que se está ejecutando correctamente.

CONSEJOS

- Puede probar la instalación de PHP empleando el archivo test.php que se ha creado en la introducción del Capítulo 1.
- Como Linux puede funcionar en computadoras con un hardware muy anticuado (disponemos de una computadora con Linux, 16MB de RAM, un 1 GB de disco duro y un procesador de 90 Mhz), crear un servidor web desde una computadora antigua con Windows es perfectamente razonable y además un modo perfecto para aprender.

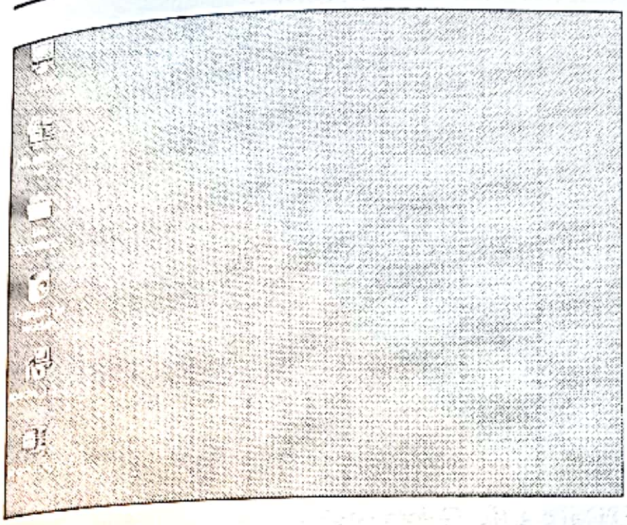


Figura A.7. Hemos descargado los archivos necesarios para la instalación en una computadora con Windows.

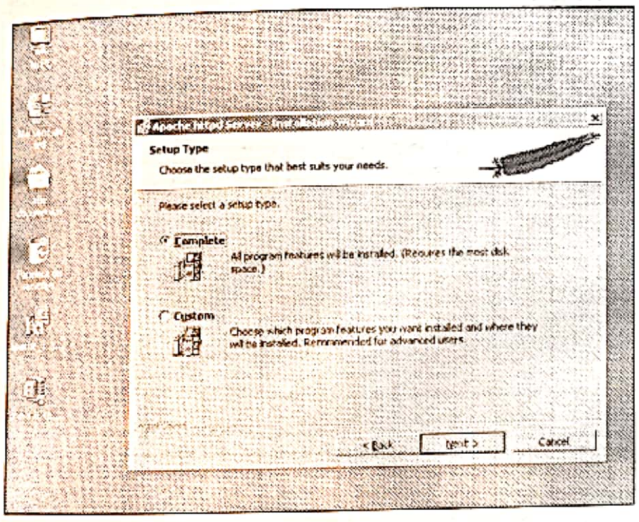


Figura A.8. Haga doble clic en el icono Apache para iniciar el proceso de instalación.

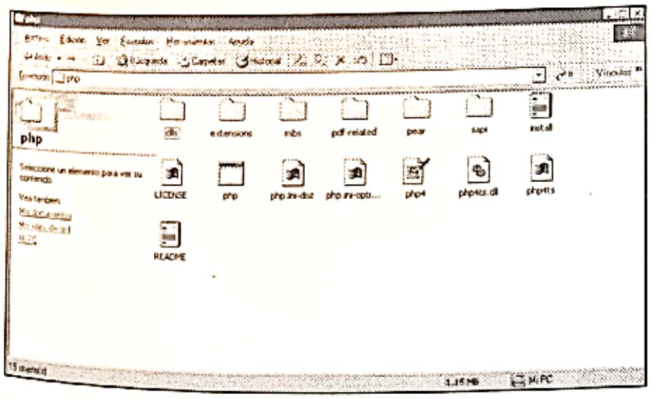


Figura A.9. Los archivos PHP descomprimidos se han colocado dentro de la carpeta C:\php.

Instalación de PHP y Apache en un servidor Windows 2000

Conseguir que PHP y Apache funcionen en una computadora con Windows (no tiene por qué ser Windows 2000) es más fácil en cierto modo que utilizar Linux, si puede usar una interfaz gráfica. Tampoco tiene por qué ser necesariamente el administrador del servidor. La instalación y la prueba posterior resultan más fáciles si se efectúan en la misma computadora.

Puede instalar PHP en Windows 95, 98, ME, NT o 2000. Sólo tiene que asegurarse de descargar los archivos adecuados de los sitios web de Apache y PHP. Incluso existen archivos Windows PHP específicos para facilitar la instalación.

Cómo instalar Apache y PHP

1. Descargue las versiones más estables y actuales de PHP y Apache a la computadora (véase la Figura A.7).
2. Descargue mediante Unzip los archivos PHP a una carpeta lógica (como C:/php). Necistará la utilidad zip para descargar mediante unzip el paquete que ha descargado.
3. Instale Apache ejecutando la aplicación de la instalación (véase la Figura A.8).
4. Copie el archivo php.ini-dist de la carpeta PHP (véase la Figura A.9) a la carpeta del sitema adecuada (por ejemplo C:/windows). Guarde el archivo como php.ini.
5. Inicie el servidor Apache.

   La manera más fácil de iniciar el servidor es dirigirse al menú Inicio, Programas, Servidor Web Apache, Iniciar Apache.
6. Haga una prueba para ver si Apache y PHP funcionan con el navegador web.

   La manera más fácil de probar Apache es teclear http://localhost/ en la ventana del navegador web. También podrá pobar si ha instalado correctamente PHP, empleando el archivo test.php.

CONSEJO

- Puede conseguir que PHP funcione con otros servidores web, como por ejemplo IIS, en una computadora con Windows.

Configuración

Aunque lo mejor es que configure PHP durante la instalación, al finalizarla puede cambiar varios parámetros. El archivo php.ini, que ha copiado a un directorio adecuado durante la instalación, contiene varios parámetros que muestran cómo funciona PHP (véanse las Figuras A.10 y A.11). Puede efectuar cambios si edita dichos parámetros y reinicia la aplicación del servidor web. Si tiene problemas, puede volver a integrar el archivo php.ini original, copiando el archivo php.ini-dist que se encuentra en el paquete original.

Para que PHP soporte bases de datos, genere imágenes rápidamente, utilice mcrypt(), etc., tendrá que descargar otras librerías y paquetes y hacer que funcionen con PHP declarando su ubicación durante la instalación, o alterando después el archivo php.ini. Vea el archivo y el manual de PHP para obtener más información sobre este punto.

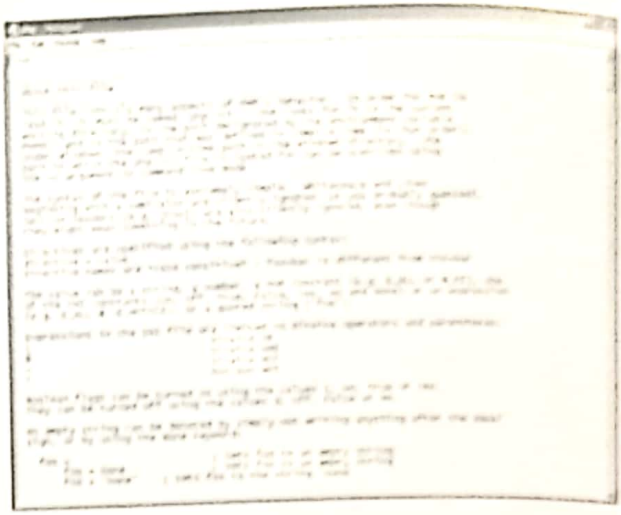


Figura A.10. El documento php.ini es el archivo más importante para determinar cómo funciona PHP.

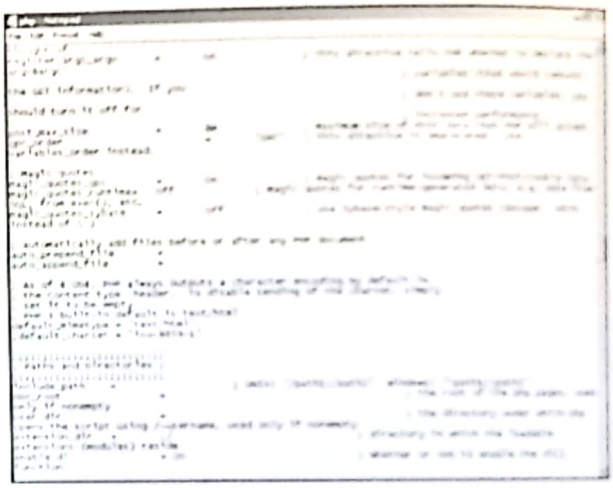


Figura A.11. En el archivo php.ini se muestran páginas y páginas de parámetros, así como instrucciones sobre cómo modificarlos.

SEGURIDAD B

La seguridad es un aspecto de la web (desde el hardware hasta el software, pasando por la programación) que no se debe olvidar. Aunque no hay justificación para asumir que cada sitio sea el objetivo de un ataque, no hay ninguna razón para dejar que sus proyectos sean infectados. Hemos mencionado dos consideraciones sobre la seguridad en este libro y en el Apéndice A. Vamos a pasar rápidamente a tratar algunas medidas de seguridad que puede adoptar mientras programa, además de dirigirle a otros recursos sobre seguridad.

Criptografía y SSL

La criptografía es el proceso que cambia el formato de los datos (por ejemplo, encriptarlos), para que sean más difíciles de leer. Criptografías, como PGP, que está disponible gratuitamente para los usuarios en http://www.pgp.com (véase la Figura B.1), emplea claves públicas y privadas para codificar y descodificar información. Otros sistemas de criptografía, cómo la función crypt(), integrada en PHP, encriptarán datos pero no los desencriptarán. Descubrirá más temas relacionados con crypt() en la sección dedicada a las cadenas del manual de PHP.

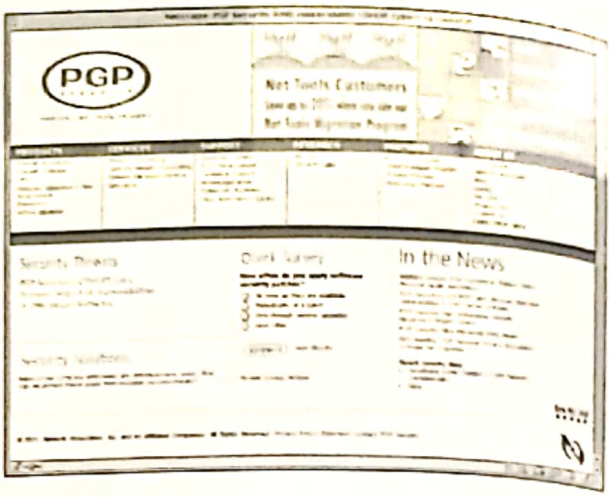


Figura B.1. PGP está disponible en versiones gratuita y comercial. Permite enviar y recibir datos encriptados.

Para conseguir un mejor nivel de criptografía en los sitios, tendrá que descargar la librería mcrypt de http://mcrypt.hellug.gr/ y configurar después la instalación de PHP con el soporte de mcrypt. La inclusión de esta librería le permitirá utilizar las funciones de mcrypt(), que pueden encriptar y desencriptar información. Visite el URL anterior y la sección mcrypt del manual de PHP para obtener más información.

La criptografía es sólo parte de una solución para la seguridad que sólo puede emplearse una vez que los datos han sido recibidos por el servidor. También puede que necesite beneficiarse de las conexiones SSL en los sitios web. SSL (*Secure Sockets Layer*, Capa de *sockets* segura) es un método de seguridad que transmite información entre un cliente (el navegador web) y el servidor. La utilización de las conexiones SSL (indicadas en el prefijo https:// de un URL) es una aplicación que debe estar dedicada al comercio electrónico. Puede especificar las *cookies* que va a enviar en una conexión SSL, ajustando los parámetros adecuados cuando emplee la función setcookie(). Verifique el ISP o el administrador del servidor para comprobar si las conexiones SSL son soportadas por la computadora que está usando.

CONSEJO

- Las contraseñas que se emplean con la aplicación PHP deberían estar siempre encriptadas. Si el servidor que emplea no soporta mcrypt(), use crypt() para encriptar la contraseña durante el inicio de la sesión, y después verifique la contraseña encriptada que está almacenada.

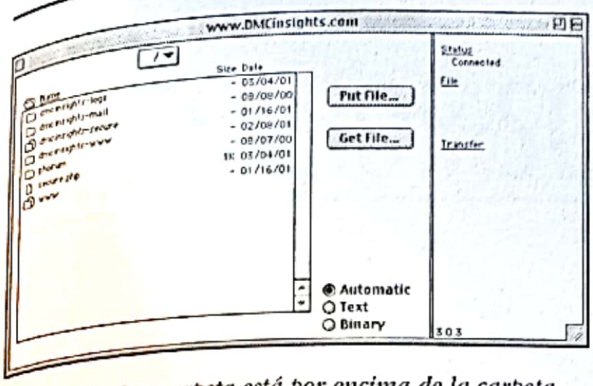


Figura B.2. Esta carpeta está por encima de la carpeta raíz del documento web (www o dmcinsights-www) y, por tanto, no disponible para el usuario a través del navegador. Aun así, puede utilizarla para almacenar y acceder a documentos sensibles, como secure.php.

Escritura de código PHP seguro

PHP no tiene la misma seguridad que la que puede encontrar cuando use *scripts* CGI o ASP. Existen varias consideraciones que debe recordar mientras programe.

La primera recomendación es que los archivos que contengan información sensible, como contraseñas, se ubiquen fuera de la raíz del documento web. Cada aplicación del servidor web utiliza una carpeta raíz predeterminada para dichos documentos. Se puede acceder a los elementos que estén dentro de la carpeta a través de un URL, pero no a los elementos que estén ubicados sobre la carpeta predeterminada. No obstante, se pueden usar todavía en PHP con la línea:

```
require ("../secure.php");
```

La línea de código anterior incluirá el archivo secure.php, que está ubicado en una carpeta sobre el documento actual (véase la Figura B.2). No se puede acceder directamente al archivo a través de la Web, aunque esté fuera de la raíz predeterminada del documento.

La segunda recomendación consta de dos partes, que consiguen que el usuario envíe los datos de los formularios HTML. Primero, debe recordar que tiene que emplear siempre el método de formulario POST (lo contrario a GET), cuando transfiera información sensible. Esto se debe a que el método GET añadiría los datos enviados al URL, haciéndolos visibles en la ventana del navegador web.

Segundo, debe ser muy prudente cuando utilice datos enviados, porque pueden convertirse en un mirador a través del cual los usuarios sin escrúpulos podrían causar estragos en el sistema. Los usuarios pueden insertar JavaScript o códigos ejecutables en los sitios utilizando un formulario HTML. Este código podría enviar información sensible, bases de datos alteradas, etc. La manera más fácil de apropiarse de esta especie de malversación, es verificar todos los datos entrantes utilizando las expresiones regulares que ha visto en el Capítulo 8. El Apéndice C le dirigirá a sitios que contienen artículos sobre estos puntos específicos, además de varios ejemplos de códigos de expresiones regulares.

Recursos de seguridad

Si va a realizar un trabajo de desarrollo web complicado, debería considerar seriamente aprender más sobre la seguridad en la Web que los puntos que le enseñamos en este apéndice.

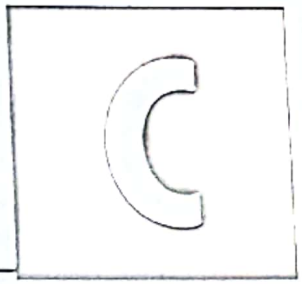
Existen literalmente docenas de sitios web que puede visitar para mantenerse informado de los lanzamientos pertinentes relacionados con la seguridad. Los cuatro más destacables, en nuestra opinión son:

- *Computer Response Emergency Team* (http://www.cert.org).
- *Security Focus* (http://www.security-focus.com).
- *Packet Storm* (http://packetstorm.securify.com).
- *World Wide Web Consortium* (http://www.w3.org/Security/Faq/www-security-faq.html).

Existen también muchos libros disponibles que tratan sobre los temas de seguridad en los servidores web de Linux o de Windows NT.

Respecto a PHP, no se olvide de leer la sección de seguridad en el manual. Revise también la sección de seguridad de la documentación de la base de datos que emplea en el servidor. A veces, como en los manuales de MySQL, se incluyen trucos específicos respecto al uso de PHP y MySQL.

RECURSOS SOBRE PHP



Este libro se ha escrito para dar a los programadores de PHP de nivel inicial y medio un buen fundamento en el que basar su aprendizaje. Debido al enfoque de este libro, se han omitido o encubierto algunos temas. En este apéndice enumeramos algunos recursos útiles de Internet relacionados con PHP, discutimos brevemente dónde obtener más información sobre bases de datos y otros temas avanzados, al tiempo que incluimos unas cuantas tablas, antiguas y nuevas.

El manual de PHP

Cada programador PHP debería adquirir inmediatamente una versión del manual de PHP antes de comenzar a trabajar con el lenguaje. El manual está disponible en el sitio oficial de PHP, http://www.php.net/docs.php (véase la Figura C.1), así como en otras localizaciones. Puede descargar el manual en casi doce idiomas en cualquiera de estos formatos: PDF, HTML, texto sin formato o compatible con Palm. El sitio oficial también tiene una versión comentada del manual disponible en http://www.php.net/manual/en/ donde otros usuarios han añadido útiles notas y comentarios que pueden solucionar alguno de los problemas que encuentre al utilizar una función particular.

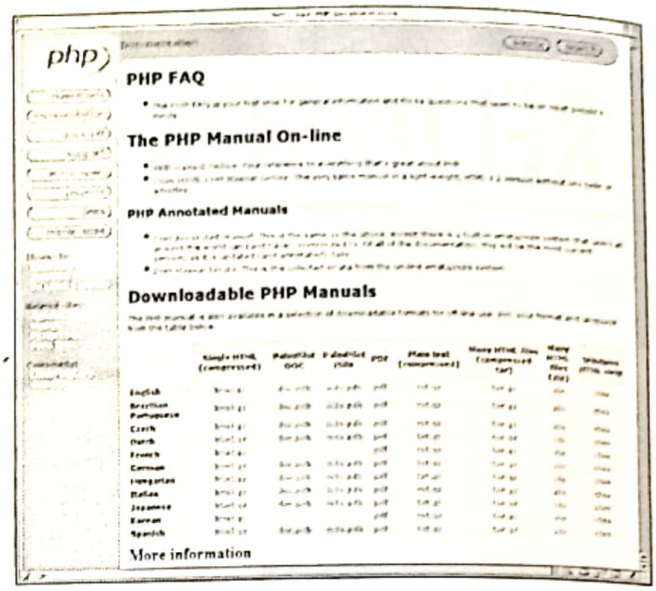


Figura C.1. La página principal de PHP dispone de distintas versiones del manual de PHP, listas para su descarga. Es recomendable que guarde una versión en su computadora a modo de referencia rápida y que la consulte si tiene problemas con una función determinada.

El manual comienza con cuestiones de instalación y seguridad. Después continúa con construcciones de lenguaje básicas y algunos elementos PHP. La parte principal del manual detalla las funciones internas de PHP, organizadas por temas. Lo que no queda claro para el programador PHP principiante es el formato que utiliza el manual para describir estas funciones. Líneas corrientes son:

```
array file (string filename [, int
→ use_include_path])
int mysql_close ([int link_identifier])
void exit(void)
double round (double val)
```

Es importante entender que la primera palabra de cada línea se refiere al tipo de valor que devolverá la función. La función file() devuelve una tabla, mysql_close() devuelve un entero, exit no devuelve nada y round() devuelve un número de punto flotante (o doble). El manual también enumera los argumentos (y tipos de argumentos) que toma una función, con argumentos opcionales enumerados dentro de corchetes. Por ejemplo, exit no toma argumentos, round() toma un número de punto flotante y mysql_close() toma un identificador de enlace opcional.

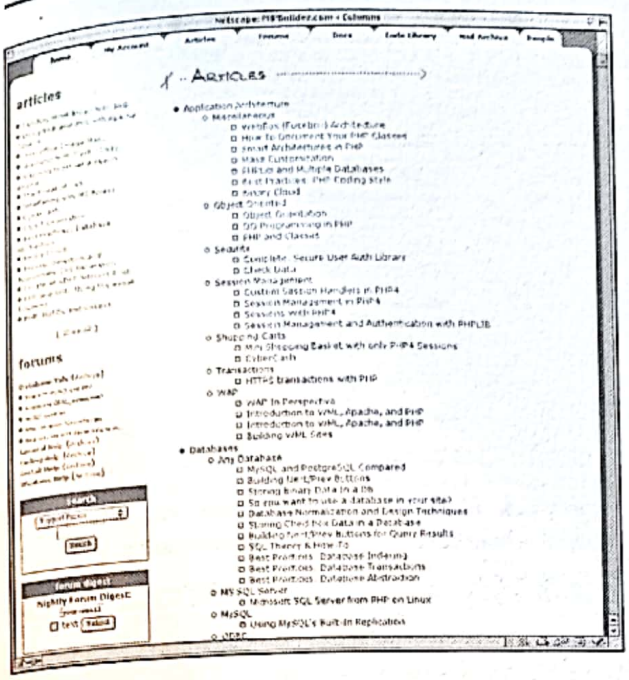


Figura C.2. PHPBuilder cuenta con una larga lista de artículos que explican cómo realizar determinadas tareas utilizando PHP. Abarca desde las tareas más básicas (por ejemplo, documentación del código) hasta las más avanzadas (por ejemplo, programación orientada a objetos).

Sitios web y grupos de noticias

Mencionaremos sólo algunos de los muchos sitios web útiles a los que puede volver cuando programe y descubrir los que más le gusten. La mayoría de ellos también contienen enlaces a otros sitios relacionados con PHP.

El primer sitio, y más obvio, a señalar es PHP.net (http://www.php.net), el sitio oficial de PHP.

En segundo lugar, debería familiarizarse con Zend.com (http://www.zend.com), la página principal de los creadores de PHP versión 4. El sitio contiene numerosas descargas, además de abundantes recursos.

Para más información sobre temas específicos, PHPBuilder (http://www.phpbuilder.com) es un buen sitio para visitar. El sitio tiene docenas de artículos explicativos de cómo realizar tareas utilizando PHP (véase la Figura C.2). Por encima de todo esto, PHPBuilder tiene foros de soporte y una librería de códigos donde los programadores han descargado *scripts* de muestra.

PHPstart4all (http://php.start4all.com/) es un buen recurso para aquellos que empiezan a utilizar PHP. Contiene enlaces que le vuelven a dirigir a: sitios y seminarios PHP, *hosts* PHP (incluyendo *hosts* gratuitos), libros PHP disponibles, proyectos de código abierto PHP y librerías de *scripts*.

De forma similar, el PHP Rescource Index (http://php.resourceindex.com/) le introducirá en seminarios *online*, *scripts* disponibles y librerías de códigos (véase la Figura C.3). Se pueden encontrar más librerías de códigos en:

- WeberDev

  (http://www.weberdev.com/maincat.php3?categoryID=106&category=PHP).

- HotScripts (http://www.hotscripts.com/PHP/).

Una última referencia que mencionaremos es el PHP Coding Standard, disponible en http://utvikler.start.no/code/php_coding_standard.html. El estándar es un documento que hace recomendaciones para programar en PHP en términos de formato y sintaxis apropiados de nombres de variables, estructuras de control, etc. Aunque no debería sentirse obligado a cumplir con estas normas, son recomendaciones sólidas y bien pensadas que pueden ayudar a minimizar los errores cuando programe.

Si tiene acceso a grupos de noticias, puede utilizarlos como una gran caja de resonancia para ideas, así como un lugar para que las respuestas más difíciles sean contestadas. Por supuesto, siempre puede ofrecer al grupo su propia destreza para quienes la necesiten. El mayor grupo de noticias sobre PHP en inglés es alt.php. Puede acceder a alt.php a través de su ISP o mediante una organización Usenet de pago por uso. Hay grupos de noticias disponibles en otros idiomas además del inglés.

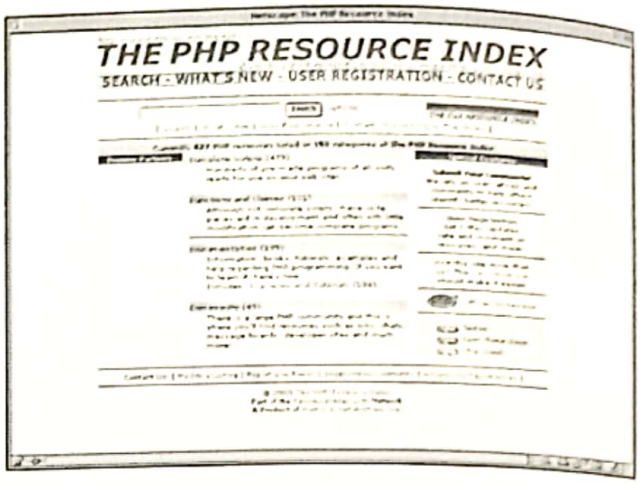


Figura C.3. El sitio web PHP Resource Index y otros parecidos mantienen impresionantes depósitos de ejemplos de códigos y scripts listos para su uso.

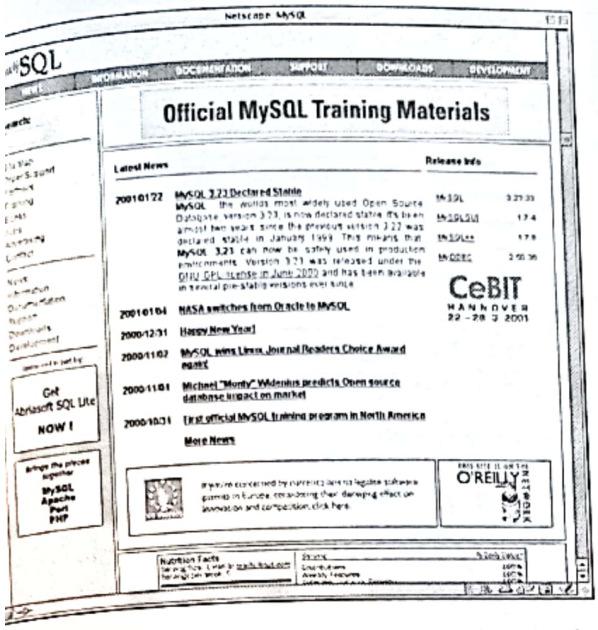


Figura C.4. El sitio web de MySQL cuenta con la mayoría de la documentación que necesitará, así como con los archivos descargables del propio MySQL.

Recursos de bases de datos

Los recursos de bases de datos que le serán más útiles dependen, obviamente, del sistema de gestión de bases de datos (DBMS, *database management system*) que utilice. La base de datos más común utilizada con PHP es MySQL, aunque PostgreSQL sea la segunda y PHP soporte la mayoría de las bases de datos estándar.

Para aprender más acerca del empleo de MySQL, comience con el sitio oficial de MySQL (http://www.mysql.com) (véase la Figura C.4). Puede descargar el manual de MySQL desde allí para usarlo como referencia mientras trabaje. También hay disponibles muchos libros específicos de MySQL que van desde el nivel inicial de **Sams' Teach Yourself MySQL in 21 Days** por Mark Maslakowski y Tony Butcher, al más técnico **MySQL & mSQL**, escrito por Randy Jay Yarger, George Reese y Tim King, y publicado por O'Reilly, Prentice Hall.

La información del uso de las bases de datos PostgreSQL se puede reunir primero visitando su sitio web (http://www.postgresql.org). Un libro popular sobre esta base de datos es **PostgreSQL: Introduction and Concepts**, por Bruce Momjian.

Puede que también quiera considerar un libro general sobre diseño de bases de datos, como **Database Design for Mere Mortals**, por Michael J. Hernandez. Aunque libros como éste no ahondarán en las especificaciones de utilizar MySQL o PostgreSQL, le ayudarán a entender mejor los razonamientos que hay detrás de la creación y utilización de bases de datos. Como PHP se puede usar con distintas bases de datos, adquirir un sentido más amplio sobre el tema puede ser primordial.

Materias avanzadas

Aunque este libro le hará empezar a utilizar PHP adecuadamente, puede que haya algunas materias que quiera investigar más a fondo.

El primer tema es crear objetos y clases utilizando PHP. Mientras mejora su capacidad de programar, y especialmente mientras construye una librería de códigos fuente, ser capaz de crear y utilizar objetos puede mejorar su velocidad de programación y minimizar los errores de ejecución. Además, aprender sobre los objetos usando PHP puede facilitar la posterior adquisición de otros lenguajes de programación orientados a objetos, como Java (a pesar de la existencia de objetos en PHP, no es un lenguaje orientado a objetos). Puede encontrar varios buenos tutoriales *online* sobre objetos en PHP, incluyendo uno en Zend.com, http://www.zend.com/zend/tut/class-intro.php (véase la Figura C.5).

Otros dos temas que le recomendaríamos investigar, la utilización de funciones para que devuelvan valores múltiples y variables variables, se tratan en el manual de PHP. Lo anterior se muestra en la sección "Funciones", que explica cómo puede utilizar una tabla que devuelva varios valores desde dentro de una función. Lo último se puede encontrar en el área "Variables" del manual y muestra cómo puede crear, asignar un valor a y recuperar un valor desde variables, sobre la marcha. Es un tema complicado de entender y utilizar pero puede llevar su programación a un nivel mucho más sofisticado.

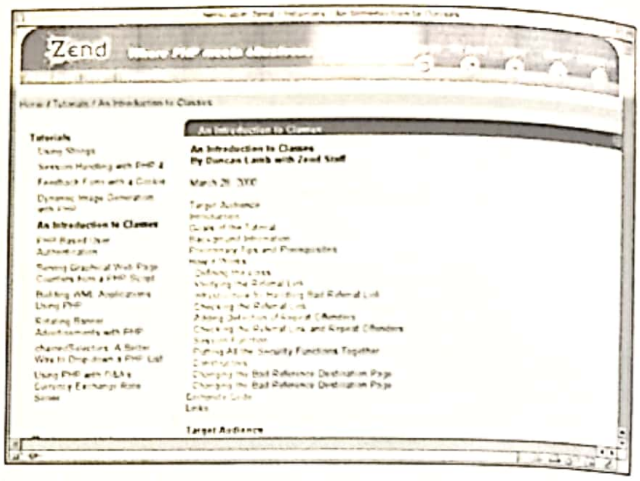


Figura C.5. El tutorial Zend.com sobre programación orientada a objetos en PHP es muy completo y fácil de comprender.

La cuestión de los permisos de archivo se trató brevemente en el Capítulo 10.

Si utiliza un servidor Unix y desea saber más, consulte el manual de Unix (puede hacerlo con el comando man) o el excelente libro **Guía de aprendizaje Perl y CGI, 2.ª edición**, de Elizabeth Castro, publicado por Prentice Hall. En sus apéndices habla de los permisos, así como de la seguridad y alguna otra información básica sobre Unix. Quienes utilicen servidores Windows pueden conocer más sobre permisos de archivo a través del sitio web de Microsoft (http://www.microsoft.com) o de los archivos Windows Help.

Para finalizar, a lo largo de este libro hemos comentado que puede ampliar el uso de expresiones regulares a otros elementos PHP empleando códigos existentes. Visite cualquiera de los sitios web mencionados anteriormente en este apéndice para ver el código de muestra que puede pegar en su programa. Otros programadores ya han desarrollado patrones para la coincidencia con expresiones regulares y *scripts* avanzados que le permiten hacer cosas tales como enviar adjuntos en un mensaje de correo electrónico usando PHP; simplemente usando estos patrones y *scripts* o creando su *script* basándose en ellos puede ahorrarle tiempo y enseñarle diferentes formas de conseguir sus objetivos.

Tablas

A lo largo de este libro hemos incluido muchas tablas de información útil. En este apéndice unimos estas tablas en una localización como una referencia conveniente, y hemos incluido una nueva, la que muestra la precedencia de los operadores (véanse las Tablas C.1 a C.6).

Tabla C.1. Ésta es una lista parcial de la precedencia de los operadores de mayor a menor (por ejemplo, la multiplicación tiene mayor precedencia que la adición).

Precedencia de operadores
! ++ –
* / %
+ - .
<<=>>=
== != ===
&&
||
= += -= *= /= .= %=
and
xor
or

Tabla C.2. Reimpresión de la lista de caracteres especiales usados para definir patrones de expresiones regulares que aparece en el Capítulo 8.

Caracteres especiales de expresión regular	
Carácter	Correspondencia
.	cualquier carácter.
^a	comienza por a.
a$	termina por a.
a+	al menos una a.
a?	cero o una a.
\n	nueva línea.
\t	tabulador.
\	carácter de secuencia de escape.
(ab)	ab juntos.
a|b	a o b.
a{2}	aa.
a{1,}	a, aa, aaa, etc.
a{1,3}	a, aa, aaa.
[a-z]	cualquier letra minúscula.
[A-Z]	cualquier letra mayúscula.
[0-9]	cualquier dígito.

Tabla C.3. No olvide que PHP ya tiene definidas varias clases de caracteres para que los use al crear patrones de expresiones regulares. Éstas son las principales.

Clases predefinidas de expresiones regulares	
Clase	Correspondencia
[[:alpha:]]	cualquier letra
[[:digit:]]	cualquier dígito
[[:alnum:]]	cualquier letra o dígito
[[:space:]]	cualquier espacio en blanco
[[:upper:]]	cualquier letra mayúscula
[[:lower:]]	cualquier letra minúscula
[[:punct:]]	cualquier signo de puntuación

Tabla C.4 El parámetro que utilice al abrir un archivo determina exactamente lo que el PHP puede hacer con el archivo, escribir en él, leerlo, etc.

Modos de archivo	
Modo	Permite
r	Sólo leer del archivo.
w	Sólo escribir en el archivo, pero creará el archivo si éste no existe o descartará los contenidos existentes antes de empezar a escribir (si el archivo no existe).
a	Añade nuevos datos al final del archivo y crea el archivo si éste no existe.
r+	Leer y escribir en el archivo.
w+	Leer y escribir en el archivo, pero creará el archivo si éste no existe o descartará los contenidos existentes antes de escribir (si el archivo no existe).
a+	Leer y escribir en el archivo y crear el archivo si éste no existe. Los nuevos datos se añadirán al final del archivo.

Tabla C.5. Resulta habitual que nunca podamos recordar los distintos formatos de fecha. Mantenga esta tabla cerca cuando utilice la función date().

Opciones de formato con la función date()	
Carácter	Formato
a	am o pm
A	AM o PM
d	día del mes de 2 dígitos con ceros delante: 01 a 31
D	día de la semana con 3 letras: Sun, Mon, etc.
F	mes, formato largo: January
g	hora del día en formato de 12 horas sin ceros delante: 1 a 12
G	hora del día en formato de 24 horas sin ceros delante: 0 a 23
h	hora del día en formato de 12 horas: 01 a 12
H	hora del día en formato de 24 horas: 00 a 23
i	minutos: 00 a 59
j	día del mes sin ceros delante: 1 a 31
l ('L' minúscula)	día de la semana, formato largo: Sunday
m	mes de 2 dígitos: 01 a 12
M	mes de 3 letras: Jan
n	mes en dígitos sin ceros delante: 1 a 12
s	segundos: 00 a 59
S	Sufijos ordinales en inglés de 2 caracteres: th, nd, rd, etc.
t	número de días en un mes dado: 28 a 31
U	segundos desde el comienzo
w	día de la semana de un solo dígito: 0 (domingo) a 6 (sábado)
y	año de 2 dígitos: 01
Y	año de 4 dígitos: 2001
z	día del año: 0 a 365

Tabla C.6. Estas opciones de informe de error vienen bien al desarrollar inicialmente un sitio y una vez que ya está en marcha.

Niveles de informe de error		
Número	Constante	Significado
1	E_ERROR	Error fatal en tiempo de ejecución.
2	E_WARNING	Avisos en tiempo de ejecución no fatales.
4	E_PARSE	Análisis en tiempo de compilación.
8	E_NOTICE	Notificaciones en tiempo de ejecución.
16	E_CORE_ERROR	Errores fatales que ocurren durante el arranque de PHP (sólo en PHP 4).
32	E_CORE_WARNING	Avisos (errores no fatales) que ocurren durante el arranque de PHP (sólo en PHP 4).
64	E_COMPILE_ERROR	Errores fatales en tiempo de compilación (sólo en PHP 4).
128	E_COMPILE_WARNING	Avisos en tiempo de compilación (errores no fatales) (sólo en PHP 4).
256	E_USER_ERROR	Mensaje de error generado por el usuario (sólo en PHP 4).
512	E_USER_WARNING	Mensaje de atención generado por el usuario (sólo en PHP 4).
1024	E_USER_NOTICE	Mensaje de notificación generado por el usuario (sólo en PHP 4).
	E_ALL	Todo lo anterior, en la medida que esté soportado.

ÍNDICE ALFABÉTICO

Símbolos

!=, 72
=, 13
$, 19, 22
&&, 74
*, 40, 71
*/, 14
-, 41, 45,71
—, 71
. (punto), 54
.html, extensión de archivo, 30
/*, 14
/, 21, 71
//, 13
; (punto y coma), 3, 4
@, 247
[], 96
\, 9, 22
\\, 129
\n, 12
||, 74
+, 38, 41, 45, 56, 71
++, 71
<!--, 15
<, 72
<=, 72
=, 25, 71
==, 71
-->, 15
>, 72
>=, 72

A

abs(), función, 49, 50
ACTION, atributo, 30, 165, 168
addslashes(), función, 58
alternancia, 121
AND, 74
Apache, 26, 250-253
aplicaciones web
    cabeceras HTTP, 225-228
    descripción, 213
    envío de un mensaje de correo electrónico, 229-231
    fecha y hora, 219-224
archivos
    descripción, 155
    escribir en un archivo externo, 159-162
    escritura en archivos, 158
    lectura desde archivos, 163-167
    modos de archivo, 158, 266
    permisos, 156
    renombrar, 177-178
array(), función, 97, 116
arrays, 24
    acceso a los elementos, 103-104
    adición de elementos, 99-100
    clasificación, 106-108
    corchetes ([]), 96
    creación, 97-98
        desde un formulario, 113-114
    descripción, 95-96
    imprimir los valores, 104-105
    multidimensionales, 115-116
    transformaciones entre cadenas y arrays, 109-112
    unir dos arrays, 101-102
arsort(), función, 107
asort(), función, 111
AUTO_INCREMENT, 191, 195

B

bases de datos
    conexión y creación, 185-188
    creación de una tabla, 189-192
    descripción, 183
    envío de datos, 193
    introducir datos desde un formulario HTML, 193-196
    MySQL, 184, 190
    recuperación de datos, 197-200
    SQL Server, 183
    SQL, 184

BBEdit, 6, 15, 118, 162, 236
break, instrucción, 82

C

cabeceras HTTP, 225-228
cadenas
    codificación y descodificación, 56-58
    concatenación, 54
    descripción, 51
    encriptación y desencriptación, 59-61
    recorte, 52-53
    sustracción de partes, 62-63
    transformaciones entre cadenas y arrays, 109-112
calculadora de costes, 40-42
cargar en el servidor por FTP, 5
CGI, 27, 183
chmod, 157
cliente FTP, 157
código seguro, escritura de, 257
códigos especiales
     , 11
comentarios, 13-15
condiciones, 68-70
cookie, 52
    adición de parámetros, 207-208
    creación, 203
    descripción, 201-202
    eliminación, 210-212
    fecha de expiración, 208-209
    lectura, 203
    recuperar con PHP, 204-206
copy(), función, 173
count(), función, 100
CREATE, 191
criptografía, 256
crypt(), función, 60, 256

D

date(), función, 79, 81, 88, 219
    opciones de formato, 267
DBMS, 183, 185
decrypt(), función, 60
depuración
    del trabajo, 241-243
    descripción, 233
    errores comunes, 234-236
    informes de errores, 237-240
die, instrucción, 244-247
dirección de e-mail, 38
directorios
    creación, 168-172
    descripción, 168
    manipulación de cargas de archivos, 173
    panel de control, 178-182
    renombrar, 177-178
DOCUMENT_ROOT, variable, 26
Dreamweaver, 2

E

each(), función, 104
echo(), función, 8
encrypt(), función, 60, 61, 64
envío de datos al navegador, 3
ereg(), función, 122
eregi(), función, 122, 123-125
error_log(), función, 237-240
error_reporting(), función, 237-240
errores
    comunes, 234-236
    constantes de error, 237
    informes, 237-240
    niveles de informe, 267
estructuras de control
    descripción, 67
    do...while, bucle, 91
    else, 77-78
    elseif, 79-81
    for, bucle, 92-94
    if, 68-70, 73, 91
    switch, 82-87
        distinción entre mayúsculas y minúsculas, 87
    while, bucle, 88-91
etiquetas
    %>, 2
    ?>, 2, 14, 30, 136
    <%, 2
    <?, 2
    <?PHP, 2, 14, 30, 135, 139
    B, 9
    BODY, 205
    BR, 11, 29, 111
    CENTER, 9
    FONT, 9
    FORM, 28, 32, 173
    SELECT, 91
    TEXT, 29, 33, 86, 113, 169

TEXTAREA, 29, 33, 114
explode(), función, 109-112, 165
expresiones regulares
agrupaciones, 126
caracteres especiales, 121, 266
clases, 127, 266
comparación de patrones, 122-125
definición, 118
descripción, 117
literales, 119
metacaracteres, 120-121
patrón sencillo, 199
patrones
complejos, 126-127
de búsqueda y sustitución, 130
referencia regresiva, 129
extensiones de archivo, 2

F

FALSE, concepto, 68, 74
file(), función, 163
formularios
campos, 28
creación, 28-30
entrada manual de datos, 35
recepción de datos, 33-34
validación, 125
y bases de datos, 193
y variables, 27
FTP, 5, 6, 169, 172
funciones
argumentos, 138-140
creación, 134-135
descripción, 133
devolución de un valor, 141-145
función con valores predeterminados, 152-153
llamada, 135-137
matemáticas internas, 48-50
valores de argumentos predeterminados, 150-151
y variables, 146-148
fwrite(), función, 158

G

Get, método, 31-32, 36, 38, 56, 168
sin un formulario HTML, 36
global, instrucción, 146-150

H

header(), función, 225
HIDDEN, tipo, 34, 84, 85
HOSTNAME, variable, 26
HREF, 56, 57
HTML, 56
comillas, 9
enviar al navegador utilizando PHP, 9
formularios, 27-38, 83, 86, 88, 89
HTML y PHP, diferencias, 2
HTTP_USER_AGENT, 26
HTTPS, 208

I

implode(), función, 109-112
imprimir un mensaje sencillo, 7
include(), 214-218
INPUT, 85, 113, 173
INSERT, 193, 195
ISP, 2, 60, 157
isset(), función, 70

J

join(), función, 112

K

krsort(), 106

L

ltrim(), función, 53

M

mail(), función, 229-231
mcrypt(), función, 256
md5(), función, 61, 63, 64
METHOD, 31-32
añadir un método al script, 32
MIME, 176
mktime(), 219
MySQL, 184, 190
mysql_connect(), función, 184

N

NAME, elemento, 33, 113
navegador
envío de datos, 3

envío de HTML, 9
envío de texto, 7
NOT, 74
números
formateo, 43
incremento y disminución, 45
operaciones básicas, 40-42

O

ODBC, 188
operadores, 46, 71
comparación, 71-73
lógicos, 74-76
precedencia, 266
prioridad, 47
OR, 74
Origen de página, opción, 12
OSTYPE, variable, 26

P

PASSWORD, valor, 169
Perl, 27, 45
pg_connect(), función, 184
PHP
aplicaciones web, 213-231
archivos y directorios, 155-182
arrays, 95-116
bases de datos, 183-200
cadenas, 51-66
cargar
archivos, 173-176
por FTP en el servidor, 5
comentarios, 13-15
comprobación del script, 5
cookies, 201-212
creación de un script, 2
depuración, 233-247
envío de HTML al navegador, 9
estructuras de control, 67-94
expresiones regulares, 117-132
formularios, 27-38
instalación y configuración, 249-254
línea de código, definición, 4
mayúsculas y minúsculas, distinción entre, 4
números, 39-50
recursos, 259-265
seguridad, 255-258
uso de espacios en blanco, 10-12
variables, 17-26
PHP y HTML, diferencias, 2
PHP_SELF, variable, 26
phpinfo(), función, 3, 6
POST, método, 31-32, 168
print(), función, 4, 7, 8, 9, 17, 42, 43, 103, 138, 215, 241-243
printf(), función, 8, 43-44, 143, 145

R

rand(), función, 49, 65
recursos de PHP, 259-265
rename(), función, 177
require(), función, 214-218, 257
Reset
botón, 30
función, 107
round(), función, 50
rsort(), función, 106
rtrim(), función, 53

S

scripts
añadir un método, 32
comprobación, 6
creación, 2
depuración, 241-243
probar en el navegador, 6
secuencia de escape, 23
seguridad, 255-258
SELECT, 197
setcookie(), función, 203, 210-212, 225
shuffle(), función, 106
sintaxis básica, 2
Solaris, 26
Sólo texto, formato de archivo, 2
sort(), función, 106
sprintf(), función, 44, 143, 145
SQL Server, 183
srand(), función, 49, 65
SSL, 256
stripslashes(), función, 58
strlen(), función, 63-64, 66, 100
strtok(), función, 62
SUBMIT, tipo, 29
substr(), función, 63, 65

T

telnet, 157, 169
TextPad, 118

time(), función, 207
trim(), función, 52-53, 54
TRUE, concepto, 68, 74
TYPE, atributo, 169

U

unlink(), función, 173
urlencode(), función, 56-58, 60

V

variables, 55
  ámbito, 146-148
  arrays, 24
  asignación de valores, 25, 37
  cadenas, 21-23
  comillas, 23
  declaración, 19
  definición, 18
  descripción, 17
  globales, 146
  incrementar, 45
  mayúsculas y minúsculas, 19
  números, 20-21
  predefinidas, 26
  sintaxis, 19
  tipos, 20
  y formularios, 27
  y funciones, 146-148
Ver código fuente, opción, 12

X

XOR, 74

W

Windows 2000, 253
Windows NT, 26, 157, 183

¿Necesita aprender PHP?
¡Inténtelo con una Guía de aprendizaje!

ABARCA VERSIONES 3 y 4

Larry Ullman

PHP

- Con párrafos concisos y comentarios directos que explican lo que debe saber.
- Un método sencillo y visual de aprender PHP, utilizando imágenes que le guiarán por el lenguaje y le mostrarán lo que debe hacer.
- Esta guía funciona también como un libro de referencia. Busque lo que necesita y empiece a trabajar.

- **Larry Ullman** es el director de Tecnologías de medios digitales en DMC Insights, Inc. y presta sus servicios como programador de PHP. Está especializado en el desarrollo de aplicaciones web y sitios web dinámicos, y enseña PHP a través de distintos medios.

Pearson Educación
www.pearsoneducacion.com

Peachpit Press

Nivel de Usuario
Principiante Intermedio Avanzado

Categoría
PROGRAMACIÓN/INTERNET

ISBN 84-205-3231-2
9 788420 532318