Contenido en un vistazo

Tabla de Contenidos

Introducción

Bienvenidos al emocionante mundo de las aplicaciones con bases de datos para la Web. Este libro brinda las técnicas básicas para crear cualquier aplicación con bases de datos para uso en la Web, pero definitivamente le recomiendo que empiece con una que sea bastante simple. En este libro desarrollo dos ejemplos de aplicaciones, ambas escogidas para representar dos tipos de aplicaciones frecuentemente encontrados en la Web: catálogos de productos, y sitios restringidos a miembros o clientes que requieren que el usuario se suscriba y se registre por medio de una contraseña. Las aplicaciones de ejemplo son lo suficientemente complicadas, requieren más de un programa y usan un gran de variedad de datos y de técnicas de manipulación de datos. A la vez son lo suficientemente simples de entender y de adaptar a una variedad de sitios web. Después de dominar las aplicaciones simples, podrá ampliar el diseño básico para incluir toda la funcionalidad que pueda imaginar.

Sobre este libro

Considere este libro como su guía amigable para crear aplicaciones con bases de datos para la Web. Este libro está diseñado como texto de referencia, no como un tutor, de modo que no tiene que leerlo de principio a fin, a menos que así lo desee. Puede empezar a leer en cualquier punto del libro, en el Capítulo 1, en el Capítulo 9, dondequiera. He dividido la tarea de crear una aplicación con bases de datos para la Web en porciones manejables de información, así que revise la tabla de contenidos y localice el tema que le interesa. Si necesita conocer información de otro capítulo para entender el capítulo que está leyendo, yo haré referencia al número del capítulo.

Estos son algunos ejemplos de los temas que comento en este libro:

- Construir y usar una base de datos MySQL
- Cómo agregar PHP a archivos de HTML
- Usar las características del lenguaje PHP
- Usar formularios de HTML para obtener información de los usuarios
- Mostrar información de una base de datos en una página web
- Almacenar información en una base de datos

Convenciones utilizadas en este libro

Este libro incluye muchos ejemplos de instrucciones de programación en PHP, instrucciones de MySQL o código HTML. En este libro tales instrucciones se muestran en un tipo diferente de letra que se ve como la línea siguiente:

```
Una instrucción de programación en PHP
```

Además, a veces se incluyen dentro del texto de un párrafo trocitos o términos clave de PHP, MySQL y HTML. Cuando sucede, el texto especial en el párrafo también aparece en el tipo de letra mostrado, diferente del tipo de letra usado en el párrafo. Por ejemplo, `este texto` es un ejemplo de una instrucción de PHP que muestra el `texto exacto` dentro del texto del párrafo.

En los ejemplos, a menudo verá algunas palabras en letra cursiva. Las palabras en cursiva son tipos generales que deben ser reemplazados con los nombres apropiados específicamente a sus datos. Por ejemplo, cuando vea un ejemplo como el siguiente

```
SELECT campo1, campo2 FROM nombredetabla
```

sabrá que `campo1`, `campo2` y `nombredetabla`, dado que están en cursiva, deben reemplazarse con nombres reales. Cuando use estas instrucciones en un programa, posiblemente lo hará de la siguiente manera:

```
SELECT nombre, edad FROM Cliente
```

Además, podría ver tres puntos (...) después de una lista en una línea de ejemplo. No es necesario digitar los tres puntos. Los tres puntos solo significan que puede tener tantos elementos en la lista como desee. Por ejemplo, cuando vea la línea siguiente

```
SELECT campo1,campo2,... FROM nombredetabla
```

no necesita incluir los tres puntos en las instrucciones. Los tres puntos solo significan que su lista de campos puede ser mayor de dos. Significa que puede continuar con `campo3`, `campo4` y así sucesivamente. Por ejemplo, su instrucción podría ser

```
SELECT nombre, edad, estatura, talladelzapato FROM Cliente
```

De vez en cuando también verá algunas cosas en negrita. Póngales atención porque indican algo que quiero destacar o algo que debe digitar.

Lo que no necesita leer

Alguna información en este libro está marcada como Cosas técnicas con un icono a la izquierda. A veces encontrará estos temas técnicos en un recuadro: Considere esto como información que no necesita leer para crear una aplicación con bases de datos para la Web. Esta información adicional puede contener detalles más exhaustivos o, a veces, describe técnicas que requieren de más conocimiento técnico para ser ejecutadas. Algunos lectores podrían estar interesados en la información o los métodos técnicos adicionales, pero siéntase en libertad de ignorarlos si no los encuentra interesantes ni útiles.

Suposiciones ingenuas

Para poder escribir un libro más específico en vez de una enciclopedia, he asumido algunas cosas sobre usted, el lector. Asumo que conoce HTML y que ha creado sitios web con HTML. Por lo tanto, aunque uso HTML en muchos ejemplos, no explico HTML. Si no tiene conocimientos sobre HTML, encontrará este libro más difícil de usar. Le sugiero que lea un libro sobre HTML – tal como HTML 4 For Dummies, 4ta Edición, por Ed Tittel y Natanya Pitts, o HTML 4 For Dummies Quick Reference, 2a Edición, por Deborah S. Ray y Eric J. Ray (publicados por Wiley Publishing, Inc.) – y que, antes de empezar este libro, construya algunas páginas web como práctica. Le será particularmente útil tener algún conocimiento sobre formularios y tablas en HTML. Sin embargo, si es usted una persona impaciente, no le diré que es imposible seguir adelante sin saber algo sobre HTML. Quizá sea capaz de absorber lo suficiente sobre HTML de este libro para crear su propio sitio web. Si decide continuar sin saber HTML, le sugiero que tenga un libro sobre HTML a la mano para que le ayude cuando necesite averiguar algo sobre HTML que no expliquemos en este libro.

Si va a seguir adelante sin ninguna experiencia sobre páginas web, tal vez desconozca algunos conceptos básicos necesarios. Debe saber cómo crear y guardar archivos de solo texto con un editor como Notepad, o cómo guardar el archivo como solo texto desde su procesador de palabras (y no en el formato del procesador de palabras). También debe saber dónde colocar los archivos de texto que contienen el código (HTML o PHP) de sus páginas web de modo que las páginas estén a la disposición de todos los usuarios con acceso a su sitio Web, y debe saber cómo mover los archivos al lugar apropiado.

No necesita saber cómo diseñar o crear bases de datos ni cómo programar. Toda la información que necesita saber sobre las bases de datos y la programación se incluye en este libro.

Cómo está organizado este libro

Este libro se divide en seis partes, con varios capítulos en cada parte. El contenido va desde una introducción a PHP y MySQL, pasando por la instalación, la creación y el uso de bases de datos, hasta cómo escribir programas con PHP.

Parte I: Desarrollar una aplicación Web con Base de Datos usando PHP y MySQL

Esta parte brinda un resumen sobre el uso de PHP y MySQL en la creación de aplicaciones con bases de datos para la Web. Describe las ventajas de PHP, de MySQL y del uso conjunto de ambos. Averiguará cómo empezar, incluyendo lo que necesita, cómo conseguir PHP y MySQL y cómo someter su software a prueba. Luego aprenderá sobre el proceso de desarrollar la aplicación.

Parte II: Base de datos MySQL

Esta parte explica los detalles sobre cómo trabajar con las bases de datos MySQL. Aprenderá cómo crear una base de datos, cómo cambiar una base de datos y cómo mover datos hacia adentro y afuera de una base de datos.

Parte III: PHP

Esta parte proporciona los detalles sobre cómo escribir programas en PHP que le permitan a sus páginas web insertar información nueva, actualizar información existente o eliminar información de una base de datos MySQL. Aprenderá cómo usar las características de PHP que se utilizan para la interacción con las bases de datos y el procesamiento de formularios.

Parte IV: Aplicaciones

La Parte IV describe las aplicaciones con bases de datos para la Web como un todo. Aprenderá cómo organizar los programas PHP en una aplicación funcional que interactúe con la base de datos. Se brindan, describen y explican dos ejemplos completos de aplicaciones.

Parte V: Los diez más

Esta parte presenta algunas listas útiles de cosas importantes que se deben hacer y que no se deben hacer al desarrollar una aplicación con bases de datos para la Web.

Parte VI: Apéndices

Esta parte le proporciona las instrucciones para instalar PHP y MySQL a aquellos que necesitan instalar el software por sí mismos. El Apéndice C comenta la instalación y el uso de los servidores web, tales como Apache e IIS, para aquellos que necesitan instalar y administrar el servidor web por sí mismos.

Iconos usados en este libro

Los consejos presentan información adicional para un propósito específico. Los consejos le ahorran tiempo y esfuerzo, así que vale la pena revisarlos.

Siempre debe leer las advertencias. Las advertencias enfatizan acciones que debe realizar o que debe evitar para prevenir consecuencias desastrosas.

Este icono señala información y procedimientos que son más técnicos que otras secciones del libro. Esta información puede ser interesante y útil, pero no necesita entenderla para usar la información en este libro.

La función de este icono es ser algo así como una notita adhesiva: resalta información que vale la pena memorizar.

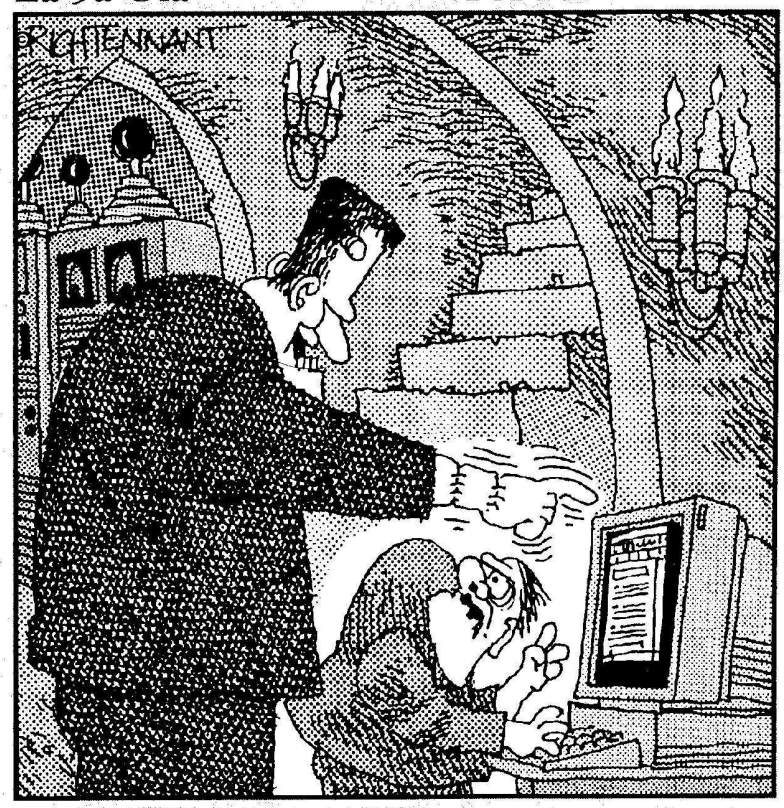
¿Y ahora adónde ir?

Este libro está organizado en el orden en el que deben hacerse las cosas. Si es usted un verdadero novato, probablemente deberá empezar con la Parte I, la cual describe cómo iniciar el proceso, incluyendo cómo diseñar las partes de su aplicación y cómo van a interactuar estas partes. Cuando implemente su aplicación, debe crear primero la base de datos MySQL, por lo cual comento MySQL antes que PHP. Después de haber entendido los detalles de MySQL y PHP, deberá unirlos en una aplicación completa, lo cual describo en la Parte IV. Si ya está familiarizado con cualquiera de las partes de este libro, puede ir directamente a la parte que necesita. Por ejemplo, si está familiarizado con el diseño de bases de datos, puede ir directamente a la Parte II, donde se describe cómo implementar el diseño en MySQL. Y si ya conoce MySQL bien, puede leer sólo la información sobre PHP en la Parte III.

Parte I

Desarrollar una aplicación Web con Base de Datos usando PHP y MySQL

La 5a Ola Por Rich Tennant



"Mira, ¡ya he buscado "cadáveres de bellezas reanimadas" tres veces y no he encontrado nada!"

En esta parte . . .

En esta parte, presento un panorama. Describo PHP y MySQL, cómo funciona cada uno de ellos y cómo funcionan juntos para convertir su aplicación Web con base de datos en una realidad. Después de describir sus herramientas, le muestro cómo configurar su ambiente de trabajo. Le presento las opciones que hay para tener acceso a PHP y MySQL y señalo qué buscar en cada ambiente.

Después de describir sus herramientas y sus opciones para desarrollar su ambiente, le brindo un panorama del proceso de desarrollo. Hablo sobre planeación, diseño y construcción de la aplicación.

Capítulo 1

Introducción a PHP y MySQL

En este capítulo

- Averiguar qué es una aplicación Web con bases de datos
- Echar un vistazo a PHP
- Descubrir cómo funciona MySQL
- Averiguar cómo funcionan juntos PHP y MySQL

Así que tiene que diseñar un sitio web interactivo. Tal vez su jefe acaba de ponerlo a cargo del catálogo de productos en línea de la compañía. O quizá desea desarrollar su propio negocio en la Web. O su hermana desea vender sus cuadros en línea. O se ofreció como voluntario para montar un sitio web exclusivo para los miembros de la asociación de acróbatas de su circo. Cualquiera que sea el motivo, habrá notado que la aplicación necesita almacenar información (por ejemplo, información sobre productos, información sobre cuadros, contraseñas de los miembros), y por lo tanto requiere de una base de datos. También habrá notado que la aplicación debe interactuar dinámicamente con el usuario; por ejemplo, el usuario selecciona un producto que desea ver o digita información sobre su membresía. Este tipo de sitio web es una aplicación Web con base de datos.

Asumo que anteriormente ha creado páginas web estáticas usando HTML (HyperText Markup Language), pero crear un sitio web interactivo es un reto nuevo, al igual que diseñar una base de datos. Tal vez le preguntó a tres gurús de la informática conocidos suyos qué debe hacer, y probablemente ellos le dijeron muchas cosas que no entendió. Pero en medio de la jerga técnica, seguramente habrá escuchado las palabras "rápido" y "fácil" y "gratis" mencionadas en las mismas oraciones que PHP y MySQL. Ahora desea saber más sobre cómo usar PHP y MySQL para desarrollar el sitio web que necesita.

PHP y MySQL trabajan muy bien juntos; es una sociedad dinámica. En este capítulo, aprenderá sobre las ventajas de cada uno, cómo funciona cada uno y cómo funcionan juntos para producir una aplicación Web con bases de datos.

¿Qué es una aplicación Web con bases de datos?

Una aplicación es un programa o un grupo de programas diseñados para ser usados por un usuario final (por ejemplo clientes, miembros, acróbatas circenses, etc.). Si el usuario final interactúa con la aplicación por medio de un explorador web, la aplicación es una aplicación para uso en la Web o simplemente una aplicación web. Si la aplicación web requiere de almacenamiento de información a largo plazo, por medio de una base de datos, entonces es una aplicación Web con base de datos. Este libro le brinda la información que necesita para desarrollar una aplicación Web con base de datos a la cual se pueda tener acceso usando exploradores web como Internet Explorer y Netscape.

Una aplicación Web con base de datos está diseñada para ayudarle al usuario a realizar una tarea. Puede ser una aplicación simple que muestre información en la ventana del explorador (por ejemplo, muestra ofertas de trabajo actuales cuando el usuario selecciona el título de un puesto) o puede ser un programa complicado con funcionalidad extendida (por ejemplo, la solicitud para ordenar un libro en Amazon.com o la solicitud para pujar en eBay).

De modo no sorprendente, una aplicación Web con base de datos consta de una base de datos y una aplicación — sólo dos partes:

- **Base de datos:** La base de datos es la memoria de largo plazo de su aplicación con base de datos para la Web. La aplicación no puede cumplir su propósito sin la base de datos. No obstante, por sí misma la base de datos no es suficiente.
- **Aplicación:** La parte de la aplicación es el programa o grupo de programas que realiza las tareas. Los programas crean la presentación visual que el usuario ve en la ventana del explorador; hacen que la aplicación sea interactiva al aceptar y procesar la información que el usuario digita en la ventana del explorador; almacenan información en la base de datos y extraen información de la base de datos. (La base de datos es inútil a menos que se puedan insertar y extraer datos de ella.)

Las páginas web que usted ha creado anteriormente solo con HTML son estáticas, lo cual significa que el usuario no puede interactuar con la página. Todos los usuarios ven la misma página web. Por otro lado, las páginas web dinámicas permiten al usuario interactuar con la página. Diferentes usuarios podrían ver páginas web diferentes. Por ejemplo, un usuario que está viendo el catálogo en línea de los productos de una tienda de muebles escoge ver información sobre sofás, mientras que otro usuario tal vez escoja ver información sobre mesitas para café. Para crear páginas web dinámicas, usted debe usar otro lenguaje además del HTML.

Un lenguaje ampliamente usado para crear páginas web dinámicas es JavaScript. JavaScript es útil para varios propósitos, tales como pasadas del (por ejemplo

para resaltar un botón de navegación cuando el usuario mueve el puntero del sobre él) o aceptar y validar información que los usuarios digitan en un formulario web. Sin embargo, no es útil para interactuar con una base de datos. No debe usarse JavaScript para mover la información del formulario web a la base de datos. Por el contrario, PHP es un lenguaje particularmente muy bien adaptado para interactuar con bases de datos. PHP puede aceptar y validar la información que los usuarios digitan en un formulario web y también puede mover la información a una base de datos. Los programas en este libro están escritos con PHP.

La base de datos

El corazón de una aplicación con base de datos para la Web es la base de datos, la cual es la memoria a largo plazo (ojalá más eficiente que mi memoria a largo plazo) que almacena la información de la aplicación. Una base de datos es un gabinete archivador electrónico que almacena información de una forma ordenada, de modo que uno pueda encontrarla cuando la necesita. Después de todo, guardar información sería inútil si no se pudiera encontrar. Una base de datos puede ser pequeña, con una estructura simple — por ejemplo, una base de datos que contenga los títulos y los nombres de los autores de todos los libros que usted posea. Pero una base de datos también puede ser enorme, con una estructura extremadamente compleja — tal como la base de datos que Amazon.com debe tener para guardar toda su información.

La información almacenada en las bases de datos viene en muchas variedades. El catálogo en línea de una compañía requiere de una base de datos para guardar información sobre todos los productos de la compañía. Un sitio web con membresía requiere de una base de datos para guardar información sobre los miembros. Un sitio web sobre empleos requiere de una base de datos (o tal vez de dos bases de datos) para almacenar la información sobre las ofertas de empleo e información sobre los currículum de los solicitantes. La información que usted planea almacenar quizá es similar a la información que hay guardada en miles de sitios web de Internet, o quizá sea información única a su aplicación.

Técnicamente, el término base de datos se refiere al archivo o grupo de archivos que contiene los datos reales. Se tiene acceso a los datos por medio de un conjunto de programas llamados SABD (Sistema Adminitrador de Bases de Datos o Database Management System). Casi todos los SABO actuales son Relacionales), en los cuales los datos se organizan y almacenan en un conjunto de tablas relacionadas.

En este libro, usamos MySQL porque es un SABD especialmente apropiado para uso en sitios web. MySQL y sus ventajas se discuten en la sección "MySQL, Mi base de datos" más adelante en este capítulo. Puede aprender cómo organizar y diseñar una base de datos MySQL en el Capítulo 3.

Listas de discusión electrónicas

Es posible encontrar muy buen soporte técnico en las listas de discusión electrónicas. Estas listas son grupos de personas que discuten temas específicos por medio del correo electrónico. Se pueden encontrar listas electrónicas para casi todos los temas en los que se pueda pensar: la lotería, filosofía antigua, cocina, los Beatles, razas de perros, política, etc. La discusión se realiza por medio del correo electrónico. El administrador de la lista mantiene una lista de distribución de direcciones electrónicas para los que quieran unirse a la discusión. Cuando envía un mensaje a la lista de discusión, su mensaje es enviado a toda la lista para que todos lo vean. Por ende, la discusión es un esfuerzo grupal y cualquiera puede responder a cualquier mensaje que le interese.

Las listas de discusión electrónicas tienen el apoyo de varios patrocinadores. Cualquier individuo u organización puede activar una lista. La mayoría de los proveedores de software tiene una o más listas dedicadas a su software. Las universidades tienen lista para las asignaturas académicas. Además, algunos sitios web como Yahoo! Groups y Topica administran listas de discusión. Los usuarios pueden crear una lista nueva o unirse a una lista existente por medio de la aplicación web.

Las listas electrónicas relacionadas con software son como la cueva del tesoro del soporte técnico. El número de usuarios de software suscritos a listas varía de un centenar a varios miles. Muchos tienen amplia experiencia con el software. A menudo los desarrolladores, programadores y personal de soporte técnico del proveedor del software están en la lista. Cualquiera que sea su pregunta o problema, alguien en la lista probablemente conoce la respuesta o la solución. No es probable que usted sea la primera persona en haber experimentado un problema. Cuando plantea una pregunta en una lista electrónica, la respuesta generalmente aparece en su casillero en unos pocos minutos. Además, la mayoría de las listas mantiene un archivo de discusiones anteriores para que usted pueda buscar respuestas a problemas específicos. Cuando es nuevo en el uso de cualquier software, puede averiguar mucho simplemente uniéndose a las listas de discusión para ese software y leyendo los mensajes durante algunos días.

Por supuesto, PHP y MySQL tienen listas de discusión electrónicas. De hecho, cada uno tiene varias listas de discusión sobre temas específicos, por ejemplo bases de datos y PHP. Puede encontrar los nombres de las listas electrónicas e instrucciones sobre cómo unirse a ellas en los sitios web de PHP y MySQL respectivamente.

La aplicación: Mover datos desde y hacia la base de datos

Para que la base de datos sea útil, usted debe poder introducirle y extraerle datos. Los programas son sus herramientas para lograrlo, pues interactúan con la base de datos para almacenar y recuperar datos. Un programa se conecta a la base de datos y hace una solicitud: "Tome estos datos y almacénelos en el lugar específico". Otro programa hace esta solicitud: "Encuentre los datos especificados y tráigamelos". Los programas de aplicación que interactúan con la base de datos corren cuando el usuario interactúa con la página web. Por ejemplo, cuando el usuario hace clic en el botón enviar después de completar un formulario en la web, un programa procesa la información en el formulario y la guarda en una base de datos.

MySQL, mi base de datos

MySQL es un SABD relacional rápido y fácil de usar utilizado para bases de datos en muchos sitios web. Desde el principio, la velocidad fue el atributo más importante para los desarrolladores. En aras de la velocidad, tomaron la decisión de ofrecer menos características que sus principales competidores (por ejemplo, Oracle y Sybase). Sin embargo, aunque MySQL tiene menos características que sus competidores comerciales, tiene todas las características necesarias para la gran mayoría de desarrolladores de bases de datos. Es más fácil de instalar y usar que sus competidores comerciales, y la diferencia en el precio favorece muchísimo a MySQL.

MySQL es desarrollado, mercadeado y mantenido por MySQL AB, que es una empresa sueca. La compañía ofrece dos tipos de licencias:

- **Software de fuente abierta:** MySQL está disponible por medio de una GNU GPL (General Public License/Licencia Pública General) sin ningún costo. Cualquiera que cumpla los requisitos para una GPL puede usar el software gratuitamente. Si está usando MySQL como base de datos en un sitio web (el tema de este libro), puede usar MySQL gratuitamente, incluso si está ganando dinero con su sitio web.
- **Licencia comercial:** MySQL está disponible con una licencia comercial para aquellos que prefieren esta alternativa a la GPL. Si un desarrollador desea usar MySQL como parte de un nuevo producto de software y desea vender el nuevo producto en vez de ponerlo a disposición bajo la GPL, el desarrollador debe comprar una licencia comercial. El costo es muy razonable.

Encontrar soporte técnico para MySQL no es difícil. Puede unirse a una de varias listas de discusión electrónicas que se ofrecen en el sitio web de MySQL: `www.mysql.com`. Incluso puede buscar en los archivos de la lista electrónica, los cuales contienen una gran base de conocimientos con preguntas y respuestas sobre MySQL. Si se siente más cómodo obteniendo soporte comercial, MySQL AB ofrece contratos de soporte técnico — con cinco niveles de soporte, que oscilan desde soporte directo por correo electrónico hasta soporte telefónico, con cinco niveles de precios.

Ventajas de MySQL

MySQL es una base de datos popular entre los desarrolladores Web. Su velocidad y pequeño tamaño la hacen ideal para un sitio Web. Súmele a eso el hecho de ser de código abierto, o sea gratis, y tiene allí la razón de su popularidad. Este es un informe minucioso de algunas de sus ventajas:

- **Es rápido.** La meta principal de los desarrolladores de MySQL fue la velocidad. En consecuencia, el software fue diseñado desde el principio con la velocidad en mente.

- **Es barato.** MySQL es gratis bajo la licencia GPL de código abierto, y el costo de una licencia comercial es muy razonable.
- **Es fácil de usar.** Puede construir una base de datos MySQL e interactuar con ella usando unas cuantas instrucciones simples en el lenguaje SQL, el cual es el lenguaje estándar para comunicarse con los SABD relacional. Consulte el Capítulo 4 para conocer toda la verdad sobre el lenguaje SQL.
- **Funciona en muchos sistemas operativos.** MySQL funciona en una gran variedad de sistemas operativos — Windows, Linux, Mac OS, la mayoría de las variedades de Unix (incluyendo Solaris, AIX y DEC Unix), FreeBSD, OS/2, Irix y otros.
- **Amplio soporte técnico.** Una base de usuarios grande brinda soporte gratuito por medio de listas electrónicas. Los desarrolladores de MySQL también participan en las listas electrónicas. También puede comprar soporte técnico a MySQL AB por un precio módico.
- **Es seguro.** El flexible sistema de autorización de MySQL permite algunos o todos los privilegios de la base de datos (por ejemplo, privilegios para crear una base de datos o borrar datos) a usuarios específicos o grupos de usuarios. Las contraseñas se encriptan.
- **Soporta bases de datos grandes.** MySQL maneja bases de datos de hasta 50 millones de filas o más. El límite predefinido para el tamaño de archivo de una tabla es de 4GB, pero se puede aumentar (si su sistema operativo lo puede manejar) hasta un límite de 8 millones de terabytes (TB), en teoría.
- **Se puede personalizar.** La licencia GPL de código abierto permite a los programadores modificar el software MySQL para adaptarlo a sus ambientes específicos.

Cómo funciona MySQL

El software de MySQL consta del servidor MySQL, varios programas utilitarios que ayudan en la administración de la base de datos MySQL y algún software de apoyo que necesita el servidor MySQL (usted no debe preocuparse por nada de esto ahora). El corazón del sistema es el servidor MySQL.

El servidor MySQL es el administrador del sistema de base de datos. Maneja todas las instrucciones de su base de datos. Por ejemplo, si desea crear una nueva base de datos, usted debe enviar un mensaje al servidor MySQL que diga "cree una base de datos nueva y llámela datosnuevos". Entonces el servidor MySQL crea un subdirectorio dentro de su directorio de datos, nombra el nuevo subdirectorio `nuevosdatos` y pone los archivos necesarios con el formato requerido en el subdirectorio `nuevosdatos`. Del mismo modo, para añadir datos a esa base de datos, usted envía un mensaje al servidor MySQL dándole los datos y diciéndole dónde quiere que añada los datos. Aprenderá cómo escribir y enviar mensajes a MySQL en la Parte II de este libro.

El servidor MySQL debe estar funcionando y esperando órdenes antes de que usted pueda transmitirle instrucciones. El servidor MySQL generalmente se configura para que inicie cuando inicia el PC y para que continúe funcionando todo el tiempo. Esta es la configuración usual para un sitio web. Sin embargo, no es necesario configurarlo para que inicie cuando inicia el PC. Si es necesario, puede iniciarlo manualmente en el momento en que desee ingresar a la base de datos. Cuando está funcionando, el servidor MySQL constantemente busca mensajes con instrucciones sobre lo que debe hacer.

Comunicación con el servidor MySQL

Toda su interacción con la base de datos se hace mediante mensajes enviados al servidor MySQL. Puede enviar mensajes al servidor MySQL de varias maneras, pero este libro se concentra en el envío de mensajes por medio de PHP. El software PHP tiene instrucciones específicas usadas para enviar instrucciones al servidor MySQL.

El servidor MySQL debe ser capaz de entender las instrucciones que recibe. Se comunica con él usando SQL (Structured Query Language / Lenguaje Estructurado de Consulta), que es el lenguaje estándar que entienden muchos SABD. El servidor MySQL entiende SQL. PHP no entiende SQL, pero no necesita hacerlo: PHP establece una conexión con el servidor MySQL y envía el mensaje en SQL por medio de dicha conexión. El servidor MySQL interpreta el mensaje en SQL y sigue las instrucciones. El servidor MySQL envía un mensaje de vuelta, indicando su estado y lo que hizo (o reportando un error si no pudo entender o seguir las instrucciones). Para conocer todos los detalles sobre cómo escribir y enviar mensajes en SQL a MySQL, consulte la Parte II de este libro.

PHP, un movilizador de datos

PHP, un lenguaje de programación diseñado específicamente para ser usado en la Web, es su herramienta para crear páginas web dinámicas. Como es rico en características que facilitan el diseño y la programación web, PHP se usa en más de 13 millones de dominios (según la encuesta de Netcraft en `www.php.net/usage.php`). Su popularidad continúa creciendo, lo cual significa que debe estar cumpliendo muy bien con su función.

PHP significa PHP: HyperText Preprocessor/Preprocesador de Hipertexto. Cuando Rasmus Lerdorf empezó a desarrollarlo, se llamaba Personal Home Page tools/Herramientas para una Página de Inicio Personal. Cuando se desarrolló a un lenguaje completo, el nombre se le cambió para que estuviera más a tono con su funcionalidad expandida.

La sintaxis del lenguaje PHP es parecida a la sintaxis de C, así que si tiene experiencia con C, se sentirá cómodo usando PHP. PHP es incluso más simple que C porque no usa algunos de los conceptos más difíciles de C. PHP tampoco incluye las capacidades de programación de bajo nivel de C porque PHP está diseñado para programar sitios web y no necesita esas capacidades.

La habilidad de PHP para interactuar con bases de datos es particularmente fuerte. PHP puede trabajar con prácticamente todas las bases de datos de las cuales haya escuchado hablar (y algunas de las que nunca ha oído nada). PHP maneja la conexión con la base de datos y la comunicación con ella. Usted no necesita conocer los detalles técnicos para conectarse a una base de datos o para intercambiar mensajes con ella. Sólo debe decirle a PHP el nombre de la base de datos y dónde está, y PHP se encarga de los detalles. Se conecta a la base de datos, pasa sus instrucciones a la base de datos y le trae de vuelta la respuesta de la base de datos.

Hay soporte técnico disponible para PHP. Puede unirse a una de varias listas de discusión electrónicas que se ofrecen en el sitio web de PHP (`www.php.net`), incluyendo una lista de bases de datos y PHP. Además, hay una interfaz hacia las listas de discusión en `news.php.net`, desde donde puede navegar o buscar en los mensajes.

Ventajas de PHP

La popularidad de PHP está aumentando rápidamente debido a sus múltiples ventajas:

- **Es rápido.** Como está empotrado en código HTML, el tiempo de respuesta es muy corto.
- **Es barato — gratis, de hecho.** PHP es prueba de que los almuerzos gratis sí existen y de que es posible recibir más de lo que se ha pagado.
- **Es fácil de usar.** PHP contiene muchas características y funciones especiales necesarias para crear páginas web dinámicas. El lenguaje PHP está diseñado para incluirse con facilidad en archivos en HTML.
- **Funciona en muchos sistemas operativos.** Funciona en una gran variedad de sistemas operativos: Windows, Linux, Mac OS y la mayoría de las variedades de Unix.
- **Amplio soporte técnico.** Una gran base de usuarios brinda soporte gratuito por medio de las listas de discusión electrónicas.
- **Es seguro.** El usuario no ve el código PHP.
- **Está diseñado para mantener bases de datos.** La funcionalidad de PHP fue diseñada para interactuar con bases de datos específicas. Le libera de la necesidad de conocer los detalles técnicos requeridos para comunicarse con una base de datos.
- **Se puede personalizar.** La licencia de código abierto les permite a los programadores modificar el software PHP y agregarle o modificar características necesarias para adaptarse a sus ambientes específicos.

Cómo funciona PHP

PHP es un lenguaje de programación empotrado cuando se usa en páginas web. Esto quiere decir que el código PHP está empotrado en código HTML. Usted usa etiquetas de HTML para encapsular el lenguaje PHP que empotra en su archivo HTML, del mismo modo en que usaría otras etiquetas de HTML. Puede crear y editar páginas web que contengan PHP del mismo modo en que se crean y editan las páginas HTML comunes.

El software PHP funciona en conjunto con el servidor web. El servidor web es el software que entrega las páginas web al mundo. Cuando digita un URL en su explorador web, está enviando un mensaje al servidor web en ese URL en el cual le pide que le envíe un archivo en HTML. El servidor web responde enviando el archivo solicitado. Su explorador lee el archivo HTML y muestra la página web. También le pide a un servidor web que le envíe un archivo cada vez que hace clic en un vínculo en una página web. Además, el servidor web procesa un archivo cuando usted hace clic en el botón para enviar un formulario en una página web.

Cuando está instalado PHP, el servidor web se configura para esperar ciertas extensiones de archivo que contienen instrucciones en lenguaje PHP. A menudo la extensión es.`php o.phtml`, pero se puede usar cualquier extensión. Cuando el servidor web recibe una solicitud de un archivo con la extensión designada, envía las instrucciones HTML sin modificación, pero las instrucciones PHP las procesa el software PHP antes de enviarlas al solicitante.

Cuando se procesan las instrucciones PHP, el servidor web sólo envía el resultado al explorador Web. Las instrucciones en el lenguaje PHP no se incluyen en el resultado que se envía al explorador, así que el código PHP es seguro y transparente para el usuario. Por ejemplo, en esta simple instrucción en PHP:

```
<?php echo "<p>Hola Mundo"; ?>
```

`<?php` es la etiqueta de apertura PHP y `?>` es la etiqueta de cierre. `echo` es una instrucción en PHP que le dice a PHP que muestre el texto seguido a continuación. El software PHP procesa la instrucción PHP y el resultado es este:

```
<p>Hola Mundo
```

que es un instrucción común en HTML. Esta instrucción en HTML se entrega al explorador del usuario. El explorador interpreta la oración como código HTML y muestra una página web con un párrafo: Hola Mundo. La instrucción PHP no se entrega al explorador, de modo que el usuario nunca ve ninguna instrucción en PHP.

PHP y el servidor web deben trabajar muy unidos. PHP no está integrado a todos los servidores web, pero trabaja con muchos de los servidores web más populares. PHP fue desarrollado como un proyecto de la Fundación de Software Apache; en consecuencia, trabaja mejor con Apache. PHP también funciona con Microsoft IIS/PWS, iPlanet (anteriormente conocido como Netscape Enterprise Server) y otros.

Aunque PHP funciona con varios servidores web, funciona mejor con Apache. Si puede seleccionar o influir en la selección del servidor web usado en su organización, elija Apache. Por sí mismo, Apache es una buena elección. Es gratis, de código abierto, estable y popular. Actualmente sustenta a más del 60 por ciento de todos los sitios web, según la encuesta de servidores web en `www.netcraft.com`. Funciona con Windows, Linux, Mac OS y la mayoría de las variedades de Unix.

MySQL y PHP, la pareja perfecta

MySQL y PHP frecuentemente se usan juntos. A menudo se les llama el dúo dinámico. MySQL brinda la parte de la base de datos y PHP brinda la parte de la aplicación en su aplicación con bases de datos para la Web.

Ventajas de esta relación

Como pareja, MySQL y PHP tienen varias ventajas:

- **Son gratis.** En cuanto a lo rentable, es difícil ganarle a gratis.
- **Están diseñados para la Web.** Ambos fueron diseñados específicamente para usarse en sitios web. Ambos tienen un conjunto de características dedicadas a crear sitios web dinámicos.
- **Son fáciles de usar.** Ambos fueron diseñados para montar sitios web rápidamente.
- **Son rápidos.** Ambos fueron diseñados pensando en la velocidad como meta principal. Juntos brindan una de las maneras más rápidas de entregar páginas web dinámicas a los usuarios.
- **Se comunican bien entre si.** PHP incluye características específicas para comunicarse con MySQL. No necesita conocer los detalles técnicos; déjele ese trabajo a PHP.
- **Ambos cuentan con una amplia base de soporte.** Ambos tienen bases de usuarios amplias. Como generalmente se usan como pareja, a menudo tienen la misma base de usuarios. Hay muchas personas dispuestas a ayudarle, incluyendo a los miembros de las listas de discusión electrónicas con experiencia en el uso de MySQL y PHP juntos.
- **Se pueden personalizar.** Ambos son de código abierto, lo cual le permite a los programadores modificar el software de PHP y MySQL para adaptarlos a sus ambientes particulares.

Cómo funcionan juntos MySQL y PHP

PHP brinda la parte de la aplicación, y MySQL proporciona la parte de la base de datos de una aplicación con base de datos para la Web. Se usa el lenguaje PHP para escribir los programas que realizan las tareas de la aplicación. PHP es lo suficientemente flexible para realizar todas las tareas que requiera su aplicación. Se puede usar para tareas simples (como mostrar una página web) o para tareas complicadas (tales como aceptar y verificar datos digitados por un usuario en un formulario en HTML). Una de las tareas que su aplicación debe realizar es introducir información en la base de datos, y extraerla; y PHP incluye características específicas para usar al escribir programas que deban mover información hacia dentro y hacia afuera de una base de datos de MySQL.

Las instrucciones PHP se empotran en sus archivos HTML con etiquetas de PHP. Cuando las tareas que debe realizar la aplicación requieren almacenar o recuperar datos, usted usa instrucciones PHP específicamente diseñadas para interactuar con la base de datos MySQL. Se usa una instrucción PHP para conectarse a la base de datos correcta, diciéndole a PHP dónde está localizada la base de datos, su nombre y la contraseña necesaria para conectarse a ella. La base de datos no necesita estar en la misma máquina que su sitio web; PHP puede comunicarse con una base de datos a través de una red. Se usa otra instrucción PHP para enviar indicaciones a MySQL. Se envía un mensaje en SQL a través de la conexión dándole instrucciones a MySQL sobre la tarea que usted desea que se realice. MySQL devuelve un mensaje de estado que muestra si ha realizado la tarea exitosamente. Si hubo un problema, devuelve un mensaje de error. Si su mensaje en SQL pidió recuperar algunos datos, MySQL envía los datos que solicitó y PHP los almacena en una ubicación temporal donde estarán a su disposición.

Luego usted debe usar una o más instrucciones PHP para completar la tarea de la aplicación. Por ejemplo, puede usar instrucciones PHP para mostrar los datos que recuperó. O puede usar instrucciones PHP para mostrar un mensaje de estado en el explorador, informándole al usuario, por ejemplo, que los datos fueron guardados.

Dado que es un SABD relacional, MySQL puede guardar información muy compleja. Como lenguaje de programación, PHP puede realizar manipulaciones de datos muy complejas, ya sean datos que deben modificarse antes de ser guardados en la base de datos, o datos recuperados de una base de datos que deben modificarse antes de ser mostrados o usados para otra tarea. Juntos, PHP y MySQL pueden usarse para crear una aplicación con base de datos para la Web con un propósito muy complejo y sofisticado.

Mantenerse al día con los cambios en PHP y MySQL

PHP y MySQL son software de fuente abierta. Si usted solo ha usado software de los mayores desarrolladores de software — tales como Microsoft, Macromedia o Adobe

— se dará cuenta de que el software de código abierto es una especie totalmente diferente. Es desarrollado por grupos de programadores que escriben el código en su tiempo libre, por diversión y sin costo alguno. No hay oficinas corporativas.

El software de código abierto cambia con frecuencia, en lugar de una o dos veces al año, como hace el software comercial. Cambia cuando sus desarrolladores sienten que está listo para hacerlo. También cambia rápidamente como respuesta a problemas. Cuando se encuentra un problema serio — por ejemplo, una grieta en la seguridad — en unos cuantos días aparece una versión nueva que resuelve el problema. Eso sí, no recibirá panfletos llamativos ni verá coloridos anuncios en revistas durante un año antes del lanzamiento de una nueva versión. Por eso, si no hace un esfuerzo por mantenerse informado, podría perderse el lanzamiento de una versión nueva o desconocer un problema serio con su versión actual.

Visite regularmente los sitios web de PHP y MySQL. Debe conocer la información que se publica ahí. Únase a las listas de correo, que a menudo tienen mucho tráfico. Mientras empieza a conocer PHP y MySQL, la gran cantidad de mensajes electrónicos en las listas de discusión le traerán información valiosa a su casillero de correo electrónico; puede aprender mucho leyendo esos mensajes. Muy pronto estará en capacidad de ayudar a otros con base en su propia experiencia. Por lo menos suscríbase a lista de correo de avisos, que solo envía mensajes electrónicos ocasionalmente. Allí se anuncian los problemas importantes y las versiones nuevas. El mensaje electrónico que recibe de la lista de avisos contiene información que debe saber. Ahora mismo, antes de que se le olvide, entre en los sitios web de PHP y MySQL e inscríbase en una lista o dos en `www.php.net/mailing-lists.php` y `lists.mysql.com`.

Debe estar al tanto de algunos cambios significativos respecto de versiones anteriores de PHP, pues aplicaciones que funcionan bien en versiones anteriores podrían tener problemas al correr en una versión más nueva, y viceversa. Los siguientes son algunos cambios que debe conocer:

- **Versión 5.0.0:** Añadió soporte para MySQL 4.1. No incluye automáticamente soporte para MySQL 4.0; debe incluirse por medio de una opción cuando se instala PHP. Cambió el nombre del archivo del intérprete de PHP usado con un servidor web de `php a.php-cgi`.
- **Versión 4.3.1:** Arregló un problema de seguridad en la versión 4.3.0. No es aconsejable continuar corriendo sitios web con la versión 4.3.0 o anteriores.
- **Versión 4.2.0:** Cambió la configuración predeterminada de `register_globals` a `Off`. Es posible que aplicaciones que corran con versiones anteriores dependan de que `register_globals` esté configurado en `On` y podrían dejar de funcionar con la nueva configuración. Lo mejor es cambiar el código de la aplicación para que corra con `register_globals` configurado en `Off`.
- **Versión 4.1.0:** Introdujo las matrices superglobales. Las aplicaciones escritas con las superglobales (como lo describo en el Capítulo 6) no funcionarán en versiones anteriores. Antes de 4.1.0, debe usar las matrices al estilo antiguo, por ejemplo `$HTTP_POST_VARS`.

Capítulo 2

Preparar su ambiente de trabajo

En este capítulo

- Tener acceso a PHP y MySQL mediante los sitios web de su compañía y de empresas de hospedaje de sitios web
- Construir su propio sitio web empezando desde cero
- Poner a prueba a PHP y MySQL

Su primera tarea, tras haberse decidido a usar PHP y MySQL, es tener acceso a ellos. Quizá a usted le está esperando, listo con todas las herramientas que necesita, un ambiente de trabajo ya montado para el desarrollo de aplicaciones para la web. En caso contrario, será parte de su trabajo que usted mismo prepare este ambiente de trabajo. Tal vez su trabajo sea crear un sitio web completamente nuevo. En este capítulo describo las herramientas que necesita y cómo tener acceso a ellas.

Las herramientas necesarias

Para montar su sitio web dinámico, necesita tener acceso a las tres herramientas de software siguientes:

- **Un servidor web:** El software que lleva sus páginas web al mundo
- **MySQL:** El SABD relacional (Sistema Administrador de Base de Datos Relacional/Relational Database Management System) que almacenará la información de su aplicación con base de datos para la Web
- **PHP:** El lenguaje de programación que usará para escribir los programas que brindan la funcionalidad dinámica a su sitio web

Describo estas tres herramientas detalladamente en el Capítulo 1.

Encontrar un lugar para trabajar

Para crear sus páginas web dinámicas, necesita tener acceso a un sitio web que proporcione sus tres herramientas de software (vea la sección anterior). Todos los sitios web incluyen un servidor web, pero no todos los sitios web proporcionan MySQL y PHP. Estos son los ambientes más comunes en los cuales puede desarrollar su sitio web:

- **Un sitio web montado por una compañía en su propio PC:** La compañía — generalmente el departamento de TI (Tecnología de información) de la compañía — instala y administra el software del sitio web. Su trabajo, para los propósitos de este libro, es programar el sitio web, ya sea como empleado de la compañía o como contratista.
- **Un sitio web hospedado por una compañía de hospedaje web:** El sitio web se localiza en el PC de la compañía de hospedaje web. La compañía de hospedaje instala y mantiene el software del sitio web y proporciona espacio de su PC para que usted instale los archivos de HTML (HyperText Markup Language/Lenguaje de Marcado de Hipertexto) de un sitio web.
- **Un sitio web que aún no existe:** Usted planea instalar y mantener el software del sitio web por sí mismo. Podría ser un sitio web de su propiedad que está construyendo en su propio PC, o podría ser un sitio web que está instalando para un cliente en el PC del cliente.

Cuánto necesita entender sobre la administración y operación del software del sitio web depende del tipo de acceso al sitio que usted tenga. En las próximas secciones, describo estos ambientes con mayor detalle y explico cómo tener acceso a PHP y MySQL.

Un sitio web de la compañía

Cuando la compañía se encarga de manejar el sitio web, usted no necesita entender la instalación y la administración del software del sitio. La compañía es responsable de la operación del sitio web. En la mayoría de los casos, el sitio web ya existe y su trabajo es agregar algo, modificar o rediseñar ese sitio web ya existente. En unos cuantos casos, la compañía podría estar instalando su primer sitio web, y su trabajo es diseñar el sitio. Cualquiera sea su caso, su responsabilidad es escribir e instalar los archivos de HTML para el sitio web. Usted no es responsable de la operación del sitio.

Usted tiene acceso al software del sitio web mediante el departamento de TI de la compañía. El nombre de este departamento puede variar en diferentes empresas, pero su función es la misma: mantiene los PCs de la empresa funcionando y al día.

Si PHP y/o MySQL no están disponibles en el sitio web de la compañía, TI necesita instalarlos y ponerlos a su disposición. PHP y MySQL tienen muchas

opciones, pero TI tal vez no entienda cuál es la mejor opción, e incluso podría configurar algunas opciones de un modo inapropiado para sus propósitos. Si necesita cambiar las opciones de PHP o MySQL, debe solicitar a TI que haga el cambio; usted no podrá hacer el cambio por sí mismo. Por ejemplo, PHP debe estar instalado de modo que soporte MySQL, así que si PHP no se está comunicando correctamente con MySQL, TI probablemente tendrá que reinstalar PHP con el soporte para MySQL habilitado.

Para que el mundo pueda ver las páginas web de la compañía, los archivos de HTML deben estar en un lugar específico del PC. El servidor web que lleva las páginas web al mundo espera encontrar los archivos de HTML en un directorio específico. El departamento de TI debe darle acceso al directorio donde deben instalarse los archivos de HTML. En la mayoría de los casos, usted desarrollará y probará sus páginas web en una ubicación de prueba y luego transferirá los archivos completos a su hogar permanente. Dependiendo del acceso que le dé TI, usted podrá copiar los archivos desde el lugar de prueba hasta el lugar permanente, o bien tendrá que transferirlos por medio de FTP (Protocolo de Transferencia de Archivos/File Transfer Protocol), que es un método para copiar un archivo desde un PC a otro en una red). En algunos casos, por razones de seguridad, los encargados de TI no le darán acceso a la ubicación permanente y preferirán instalar los archivos en esa ubicación ellos mismos.

Para poder usar las herramientas de software para la Web y crear su sitio web dinámico, usted necesita que TI le dé la siguiente información:

- **La ubicación de las páginas web:** Debe saber dónde poner los archivos de las páginas web. TI debe darle el nombre y la ubicación del directorio donde se deben instalar los archivos. Además, debe saber cómo instalar los archivos: copiarlos, enviarlos por FTP, o usando otros métodos. Tal vez necesite una identificación de usuario y una contraseña para instalar los archivos.
- **El nombre predeterminado del archivo:** Cuando los usuarios apuntan sus exploradores hacia un URL, un archivo será enviado hacia ellos. El servidor web está configurado para enviar un archivo con un nombre específico cuando el URL señala un directorio. El archivo que se envía automáticamente es el archivo predeterminado. A menudo el archivo predeterminado se llama `index.htm` o `index.html`, pero a veces se usan otros nombres como `default.htm`. Pregunte a TI cómo debe nombrar el archivo predeterminado.
- **Una cuenta MySQL:** El acceso a las bases de datos MySQL se controla por medio de un sistema de nombres de cuentas y contraseñas. TI configurará una cuenta MySQL para usted con los permisos apropiados y también le dará el nombre de la cuenta MySQL y su contraseña. (En el Capítulo 5 explico con detalle las cuentas MySQL.)
- **La ubicación de las bases de datos MySQL:** Las bases de datos MySQL no tienen que estar ubicadas en el mismo PC que el sitio web. Si las bases de datos MySQL se ubican en un PC diferente al del sitio web, usted necesita saber el nombre del anfitrión (por ejemplo, `thor.companyname.com`) donde se encuentran las bases de datos.

- **La extensión del archivo PHP:** Cuando se instala PHP, al servidor web se le indica que debe esperar instrucciones de PHP en archivos con extensiones específicas. Con frecuencia, las extensiones usadas son `.php` o `.phtml`, pero se pueden usar otras extensiones. Las instrucciones PHP en los archivos que no tienen la extensión correcta no serán procesadas. Pregunte a TI cuál extensión debe usar para sus programas PHP.

Frecuentemente tendrá que interactuar con los encargados de TI, conforme surjan nuevas necesidades. Por ejemplo, tal vez necesite cambiar opciones, tal vez necesite información que le ayude a interpretar un mensaje de error, o tal vez necesite reportar un problema con el software del sitio web. Por eso tener una buena relación con ellos hará su vida más fácil. Acostumbre llevarles galletitas y donas.

Una compañía de hospedaje de sitios web

Una compañía de hospedaje de sitios web le brinda todo lo que necesita para montar un sitio web, incluyendo espacio en sus PCs y todo el software del sitio. Solo debe crear los archivos para sus páginas web y pasarlos a la ubicación especificada por la compañía de hospedaje.

Millones y millones de compañías ofrecen servicios de hospedaje en la Web. La mayoría cobra una tarifa mensual (generalmente muy pequeña), e incluso hay algunas gratuitas. (La mayoría, pero no todas, de las gratuitas tienen como requisito que usted publique anuncios). Generalmente, la tarifa mensual varía dependiendo de los recursos brindados para su sitio web. Por ejemplo, un sitio web con 2MB de espacio en disco para los archivos de su página web costaría menos que un sitio con 10MB de espacio en disco.

Cuando busque un lugar para hospedar su sitio web, asegúrese de que la compañía de hospedaje ofrezca lo siguiente:

- **PHP y MySQL:** No todas las compañías brindan estas herramientas. Tal vez tenga que pagar más por un sitio con acceso a PHP y MySQL; algunas veces tendrá que pagar una cuota adicional por usar bases de datos MySQL.
- **Una versión reciente de PHP:** A veces no se ofrecen las versiones más recientes de PHP. Ni siquiera debe considerar un sitio web que sólo tenga acceso a PHP 3. Necesita PHP 4 por lo menos. Preferiblemente debería tener acceso a PHP 5.

Al elegir una compañía de hospedaje debe tomar en cuenta estas otras consideraciones:

- **Confiabilidad:** Necesita una compañía de hospedaje en la cual pueda confiar: una que no vaya a quebrar repentinamente y desaparecer mañana, y una que no use PCs antiguos sostenidos con chicle y alambre para embalar, de esos que pasan más tiempo caídos que en servicio.

- **Velocidad:** Las páginas web que se descargan lentamente son un problema, porque los usuarios se ponen impacientes y se van a otra parte. Las páginas lentas pueden ser el resultado de una compañía de hospedaje que empezó su negocio con escasos recursos y que carece de equipo de buena calidad — o quizá la compañía de hospedaje sea tan exitosa que su equipo está sobrecargado por clientes nuevos. De cualquier modo, las compañías de hospedaje que despliegan las páginas web muy lentamente son inaceptables.
- **Soporte técnico:** Algunas compañías de hospedaje no tienen a nadie disponible para responder preguntas o resolver problemas. El soporte técnico a menudo se brinda sólo por correo electrónico, lo cual puede ser aceptable si el tiempo de respuesta es corto. A veces es aconsejable probar la calidad del soporte técnico de la compañía llamando al número de soporte o probando su tiempo de respuesta enviándoles un correo electrónico.
- **El nombre del dominio:** Cada sitio web tiene un nombre de dominio que los exploradores web usan para encontrar el sitio en la Web. Cada nombre de dominio se registra con el pago de una pequeña tarifa anual, de modo que solo un sitio pueda usarlo. Algunas compañías de hospedaje le permiten usar un nombre de dominio que usted ha registrado independientemente de la compañía de hospedaje; algunas le ayudan a registrar y usar un nombre de dominio nuevo; y algunas requieren que usted use un nombre de dominio definido por ellas. Por ejemplo, suponga que su nombre es Lola Diseñadora y desea que su sitio web se llame LolaDiseñadora. Algunas compañías de hospedaje le permitirán llamar a su sitio web `Lolamuestra.com`, pero algunas requerirán que su sitio se llame `Lolamuestra.empresa.com`, o `empresa.com/~Lolamuestra` o algo parecido. En general, su sitio web se verá más profesional si usted usa su propio nombre de dominio.
- **Respaldos:** Los respaldos son copias de los archivos de sus páginas web y de su base de datos que están almacenados en caso de que sus archivos o base de datos se pierdan o se dañen. Debe asegurarse de que la compañía haga copias de respaldo de su aplicación con frecuencia y rutinariamente. También debe averiguar cuánto tardarán en colocar los respaldos en su lugar para restaurar el funcionamiento normal de su sitio web después de un problema.
- **Características:** Seleccione las características con base en el propósito de su sitio web. Generalmente las compañía de hospedaje agrupan las características en planes, y más características = costos mayores. Algunas características que debe tomar en cuenta son:
  - **Espacio en disco:** ¿Cuántos MB/GB de espacio en disco requerirá su sitio web? Los archivos de medios, tales como gráficos o archivos de música pueden ser bastante grandes.
  - **Transferencia de datos:** Algunas compañías de hospedaje cobran por enviar páginas web a los usuarios. Si espera tener mucho tráfico en su sitio, será importante considerar este costo.
  - **Direcciones electrónicas:** Muchas compañías de hospedaje le brindan una cantidad de direcciones electrónicas para su sitio web. Por ejemplo,

si su sitio es `Lolamuestra.com`, usted podría permitirles a los usuarios que le envíen mensajes electrónicos a `yo@Lolamuestra.com`.

- **Software:** Las compañías de hospedaje ofrecen una buena variedad de software para el desarrollo de aplicaciones para la Web. PHP y MySQL son el software que comento en este libro. Algunas compañías de hospedaje podrían ofrecer otras bases de datos y algunas podrían ofrecer otras herramientas de desarrollo tales como extensiones de FrontPage, software para carritos de compras y validación de tarjetas de crédito.
- **Estadísticas:** A menudo es posible obtener estadísticas relacionadas con el tráfico en su sitio web, por ejemplo el número de usuarios, la hora de acceso, el acceso por página web, etcétera.

Nombres de dominio

Cada sitio web necesita una dirección única en la Web. Esa dirección única que usan los PCs para localizar un sitio web es la dirección IP, la cual es un arreglo de cuatro números entre 0 y 255, separados por puntos; por ejemplo, `172.17.204.2` o `192.163.2.33`.

Como las direcciones IP constan de números y puntos, no son fáciles de recordar. Afortunadamente, la mayoría de las direcciones IP tienen un nombre asociado que es mucho más fácil de recordar, tal como `amazon.com`, `www.irs.gov` o `miempresa.com`. Un nombre que sea la dirección de un sitio web es un nombre de dominio. Un dominio puede ser un PC o muchos PCs interconectados. Cuando un dominio se refiere a varios PCs, cada PC en el dominio puede tener su propio nombre. Un nombre que incluya el nombre de un PC individual, tal como `thor.miempresa.com`, identifica un subdominio.

Cada nombre de dominio debe ser único para que pueda servir como dirección. Por lo tanto, un sistema de registro de nombres de dominio garantiza que no haya dos ubicaciones que usen el mismo nombre de dominio. Cualquiera puede registrar un nombre de dominio siempre y cuando el nombre esté libre. Usted puede registrar un nombre de dominio en la Web. Primero, pruebe su nombre de dominio potencial para averiguar si está disponible. Si lo está, regístrelo bajo su nombre o el nombre de su compañía, pagando la tarifa correspondiente. El nombre será suyo y lo podrá usar; nadie más podrá usarlo. La tarifa estándar para registrar un nombre de dominio es $35 por año. No se debe pagar nunca más, pero a menudo hay gangas.

Muchos sitios web ofrecen la capacidad de registrar un nombre de dominio, incluyendo los sitios web de muchas compañías de hospedaje. Una búsqueda en Google (`www.google.com`) de registro de nombre de dominio da como resultado más de 3 millones de aciertos. Busque para asegurarse de encontrar el precio más bajo. Además, muchos sitios web le permiten digitar un nombre de dominio para saber quién lo registró. Estos sitios web hacen una búsqueda de bases de datos de nombres de dominio usando una herramienta llamada whois (quién es). Una búsqueda en Google de nombre de dominio whois resulta en 770,000 aciertos. Un par de lugares donde puede ir para hacer una búsqueda whois son Allwhois.com (`www.allwhois.com`) y BetterWhois.com (`www.betterwhois.com`).

Una desventaja de hospedar su sitio en una compañía de hospedaje comercial es que usted no tendrá control sobre el ambiente de desarrollo. La compañía de hospedaje de sitios web brinda el ambiente que funciona mejor para ella, probablemente un ambiente configurado para que el mantenimiento sea sencillo, de bajo costo y con un mínimo de retiro de clientes. La mayor parte de su ambiente lo configura la compañía y usted no lo puede cambiar. Solo puede rogarle a la compañía que lo cambie. La compañía no estará anuente a cambiar la configuración por temor a que un cambio pueda causar problemas en el sistema de la compañía o a otros clientes.

El acceso a las bases de datos MySQL se controla por medio de un sistema de cuentas y contraseñas que debe mantenerse manualmente, lo cual causa trabajo adicional para la compañía de hospedaje. Por esta razón, muchas compañías de hospedaje no ofrecen MySQL o cobran extra por él. Además, PHP tiene un millón de opciones que puede configurarse, desconfigurarse o recibir múltiples valores. La compañía de hospedaje decide las opciones de configuración con base en sus necesidades, lo cual puede ser o no ideal para sus propósitos.

Es bastante difícil investigar las compañías de hospedaje de sitios web partiendo desde un único punto — una búsqueda en Google.com de hospedaje de sitios web resulta en más de 6 millones de aciertos. La mejor manera de investigar compañías de hospedaje es pidiéndole recomendaciones a las personas que han tenido experiencia con esas compañías. Las personas que han usado una compañía de hospedaje pueden advertirle si el servicio es lento o si los PCs se caen a menudo. Después de recopilar unos cuantos nombres de compañías de hospedaje de clientes satisfechos, puede delimitar la lista a la que mejor se adapte a sus propósitos y sea más rentable.

Montar y manejar su propio sitio web

Si está iniciando un sitio web desde cero, necesitará comprender muy bien el software del sitio. Tiene que tomar varias decisiones relacionadas con el hardware y el software. Tiene que instalar un servidor web, PHP y MySQL, así como mantener, administrar y actualizar el sistema usted mismo. Tomar esta ruta requiere de más trabajo y más conocimiento. La ventaja es que tiene el control total sobre su ambiente de desarrollo de aplicaciones web.

Estos son los pasos generales que le conducirán a su sitio web dinámico (en las secciones siguientes explico estos pasos con más detalle):

1. **Prepare el PC.**
2. **Instale el servidor web.**
3. **Instale MySQL.**
4. **Instale PHP.**

Si está empezando desde cero, con nada más que un espacio vacío donde colocará el PC, empiece en el Paso 1. Si ya tiene un PC en funcionamiento, pero no tiene software de desarrollo web, empiece en el Paso 2. O si ya tiene un sitio web pero no tiene PHP ni MySQL instalados, empiece en el Paso 3.

Preparar el PC

Su primera decisión es escoger cuál plataforma de hardware y cuál sistema operativo usará. En la mayoría de los casos, escogerá un PC con Linux o Windows como sistema operativo. Estas son algunas ventajas y desventajas de estos sistemas operativos:

- **Linux:** Linux es de fuente abierta, así que es gratis. También tiene ventajas para ser usado como servidor web: corre por largos períodos de tiempo sin necesidad de ser reiniciado; y Apache, el servidor web más popular, funciona mejor con Linux que con Windows. Correr Linux en un PC es la opción de menor costo. La desventaja es que para muchas personas Linux es más difícil de instalar, configurar, administrar e instalarle software que Windows.
- **Windows:** A diferencia de Linux, Windows no es gratis. Sin embargo, las ventajas son que la mayoría de personas cree que Windows es más fácil de usar y, como es ampliamente utilizado, muchas personas le pueden ayudar si tiene problemas.

Asumiré que comprará un PC con el sistema operativo y el software instalados, listos para usar. Es más fácil encontrar un PC que venga con Windows instalado, pero sí se consiguen PCs con Linux instalado. Por ejemplo, actualmente Dell, IBM y Hewlett-Packard ofrecen PCs con Linux instalado.

Si está construyendo su propio hardware, necesita más información de la que le puedo brindar en el breve espacio de este libro. Si tiene el hardware y planea instalar el sistema operativo, Windows es más fácil de instalar, pero Linux se está haciendo cada día más fácil. Puede instalar Linux desde un CD, como Windows, pero a menudo debe proporcionar información o tomar decisiones que requieren de un conocimiento más profundo de su sistema. Si usted ya sabe cómo realizar las tareas de administración del sistema en Windows o en Linux (tal como instalar software y hacer respaldos), la solución más rápida es usar el sistema operativo que ya conoce.

Si va a usar PHP y MySQL debe considerar seriamente usar Linux. PHP es un proyecto de la Apache Software Foundation, así que corre mejor con el servidor de Apache. Y Apache corre mejor en Linux que en Windows. Por lo tanto, si el PC será usado principalmente para correr un sitio web con aplicaciones con bases de datos, Linux es perfectamente adecuado para sus propósitos.

Hay otras soluciones además de un PC con Windows o Linux, pero son menos populares:

- **Con base en Unix:** Hay otros sistemas operativos gratuitos basados en Unix disponibles para PCs, tales como FreeBSD (algunos personas lo prefieren a Linux) o una versión de Solaris que Sun permite descargar gratuitamente.

- **Mac:** Los equipos Mac pueden usarse como servidores web. Los Mac más nuevos vienen con PHP instalado. Instalar PHP y MySQL en Mac OS X es bastante simple. Sin embargo, hay menos usuarios de Mac, así que será difícil encontrar ayuda cuando la necesite. Un buen sitio es `www.phpmac.com`.

Cómo instalar el servidor web

Después de configurar el PC debe decidir cuál servidor web instalar. La respuesta casi siempre es Apache. Apache ofrece las siguientes ventajas:

- **Es gratis.** ¿Hace falta decir algo más?
- **Corre en una gran variedad de sistemas operativos.** Apache corre en Windows, Linux, Mac OS, FreeBSD y la mayoría de las variedades de Unix.
- **Es popular.** Aproximadamente el 60 por ciento de los sitios web en Internet usan Apache, según encuestas en `www.netcraft.com/survey` y `www.securityspace.com/s_survey/data/`. Esto no sería así si no funcionara bien. Además, esto significa que un gran número de usuarios podrá ayudarlo.
- **Es confiable.** Una vez que Apache esté instalado y funcionando, funcionará el tiempo que lo haga su PC. Es muy raro tener problemas de emergencia con Apache.
- **Se puede personalizar.** La licencia de fuente abierta permite a los programadores modificar el software de Apache, agregando o modificando módulos para adaptarlo a los ambientes específicos.
- **Es seguro.** Existe software gratuito disponible que convierte a Apache en un servidor SSL (Capa de "Sockets"/Terminal segura). La seguridad es esencial si va a usar el sitio para comercio electrónico.

Apache se instala automáticamente al instalar la mayoría de las versiones de Linux. Los Macs más recientes traen Apache instalado. Para la mayoría de las variedades de Unix, hay que descargar el código fuente de Apache y compilarlo uno mismo, aunque hay disponibles algunos binarios (programas ya compilados para sistemas operativos específicos). Para Windows, usted necesita instalar un archivo binario, preferiblemente en Windows NT/2000/XP, aunque Apache también corre en Windows 95/98/Me. Al momento de escribir esto, Apache 1.3.28 y 2.0.47 son actualmente las versiones estables. (Más información sobre las versiones de Apache está disponible en el Apéndice C.) Consulte el sitio web de Apache (`httpd.apache.org`) para mayor información, descargar software, documentación e instrucciones de instalación para diversos sistemas operativos. El sitio brinda amplia documentación, constantemente mejorada.

Hay otros servidores web disponibles. Microsoft ofrece IIS (Internet Information Server / Servidor de información de Internet), el cual es el segundo servidor web más popular en Internet, usado en aproximadamente el 27 por ciento de los sitios web. Sun ofrece iPlanet (anteriormente Netscape Enterprise Server) que sirve a menos de un 5 por ciento de Internet. Hay otros servidores web disponibles, pero sus bases de usuarios son aún más pequeñas.

Cómo instalar MySQL

Después de configurar el PC e instalar el servidor web, está listo para instalar MySQL. Debe instalar MySQL antes de instalar PHP, pues debe brindar la ruta al software de MySQL al instalar PHP.

Pero antes de instalar MySQL, asegúrese de que realmente necesita instalarlo. Quizá ya esté corriendo en su PC, o quizá esté instalado pero no esté corriendo. Muchas versiones de Linux, por ejemplo, instalan automáticamente MySQL. Esta es la manera de verificar si MySQL ya está corriendo en su PC:

- **Linux/Unix/Mac:** En la línea de comandos, digite lo siguiente:

  ```
  ps -ax
  ```

  El resultado será una lista de programas. Algunos sistemas operativos (generalmente variedades de Unix) tienen opciones diferentes para el comando `ps`. Si el comando anterior no produce una lista de programas que estén corriendo, digite **man ps** para ver cuáles opciones debe usar.

  En la lista de programas que aparece, busque uno llamado `mysqld`.

- **Windows:** Si MySQL está corriendo, deberá verlo en su bandeja de sistema en la parte inferior de su pantalla, posiblemente como un semáforo con una luz verde. Si no puede encontrar este icono lo más probable es que no esté corriendo.

Pero MySQL puede estar instalado aunque no esté corriendo, quizá no ha activado. Esta es la manera de revisar si MySQL está instalado en su PC:

- **Linux/Unix/Mac:** Digite lo siguiente:

  ```
  find / -name "mysql*"
  ```

  Si aparece un directorio llamado mysql, MySQL ya ha sido instalado.

- **Windows:** Busque el programa llamado WinMySQLadmin que, entre otras funciones, inicia y detiene a MySQL. Probablemente lo encontrará en el menú Inicio (escoja Start➪Programs). Si no, búsquelo en un directorio MySQL, que probablemente será `c:\mysql\bin`.

Si MySQL está instalado pero no se ha activado, esto es lo que debe hacer para activarlo:

- **Linux/Unix/Mac:**

  1. **Cámbiese al directorio** `mysql/bin`.

     Este es el directorio que habrá encontrado al verificar si MySQL estaba instalado.

  2. **Digite** safe_mysqld &.

     Cuando este comando termina, aparece el prompt.

3. **Verifique que el servidor MySQL se haya activado digitando** ps -ax.

   En la lista de programas que aparece, busque uno llamado `mysqld`.

✔ **Windows:**

1. **Inicie el programa** `WinMySQLadmin`.

   Si no lo puede encontrar en el menú, navegue hacia el programa, que probablemente estará en `c:\mysql\bin\winmysqladmin.exe` y haga doble clic en él.

2. **Haga clic derecho en la ventana WinMySQLadmin.**

   Aparecerá un submenú.

3. **Seleccione el elemento del menú para su sistema operativo: Win 9x o Win NT (que incluye Win 2000 y XP).**

4. **Haga clic en Start the Server.**

Si MySQL no está instalado en su PC, debe descargarlo de `www.mysql.com` e instalarlo. El sitio web brinda toda la información y el software que necesita. (Econtrará instrucciones de instalación detalladas en el Apéndice A.)

Cómo instalar PHP

Después de instalar MySQL, está listo para instalar PHP. Como mencioné anteriormente, debe instalar MySQL antes de instalar PHP porque necesita brindar la ruta al software de MySQL cuando instala PHP. PHP no podrá comunicarse con MySQL si al instalarse no se compila con soporte para MySQL.

Antes de instalar PHP, verifique si ya está instalado. Algunas versiones de Linux y Mac, por ejemplo, instalan automáticamente PHP. Para ver si PHP está instalado, busque en su disco cualquier archivo de PHP:

✔ **Linux/Unix/Mac:** Digite lo siguiente:

```
find / -name "php*"
```

✔ **Windows:** Use la opción Find (escoja Start⇨Find) para buscar php*.

Si encuentra archivos de PHP, PHP ya está instalado, y quizá no necesite reinstalarlo. Es posible que incluso si instaló MySQL usted mismo después de instalar PHP, lo haya instalado en el lugar donde PHP espera que esté. Pero mejor prevenir que lamentar: lleve a cabo la prueba que describo en la sección siguiente para ver si MySQL y PHP están funcionando juntos correctamente.

Si no encuentra ningún archivo PHP, PHP no está instalado. Para instalar PHP, debe tener acceso al servidor web de su sitio. Por ejemplo, cuando instala PHP con Apache, debe editar el archivo de configuración de Apache. Toda la información y el software que necesita los encontrará en el sitio web de PHP (`www.php.net`). En el Apéndice B encontrará instrucciones detalladas para su instalación.

Probando, probando, 1, 2, 3

Suponga que cree que PHP y MySQL están disponibles para ser usados, por alguna de las razones siguientes:

- El departamento de TI de su compañía o de la compañía de su cliente le dio toda la información que usted solicitó y le dijeron que todo está listo para continuar.
- La compañía de hospedaje de sitios web le dio toda la información que usted necesita y le dijo que todo está listo para continuar.
- Siguió todas las instrucciones e instaló PHP y MySQL usted mismo.

Ahora debe hacer algunas pruebas para asegurarse de que PHP y MySQL estén funcionando correctamente.

Someter PHP a prueba

Para comprobar si PHP está instalado y funcionando, siga estos pasos:

1. **Encuentre el directorio en el cual deben guardarse sus programas de PHP.**

   Este directorio y los subdirectorios dentro de él son su espacio web. Apache llama a este directorio el Documento Raíz. El espacio web predeterminado de Apache es htdocs en el directorio donde Apache está instalado. En IIS es `Inetpub\wwwroot`. En Linux podría ser `/var/www/html`. El espacio web se puede determinar en un directorio diferente configurando el servidor web (consulte el Apéndice C). Si está usando una compañía de hospedaje de sitios, el personal le suministrará el nombre del directorio.

2. **Cree el siguiente archivo en alguna parte de su espacio web con el nombre** `test.php`.

```
<html>
<head>
<title> Prueba PHP</title>
</head>
<body>
<p>Esto es una linea HTML
<p>
<?php
 echo "Esto es una linea PHP";
 phpinfo();
?>
</body></html>
```

   El archivo debe guardarse en su espacio web para que el servidor web pueda encontrarlo.

3. **Dirija su explorador al archivo test.php creado en el Paso 1. Es decir, digite el nombre de su servidor web (www.mifinaempresa.com) en la ventana de dirección del explorador.**

   Si su servidor web, PHP y el archivo `test.php` están en el mismo PC en el que está haciendo la prueba, puede digitar **localhost/test.php**.

   Para que el archivo sea procesado por PHP, usted debe tener acceso al archivo mediante el servidor web, y no escogiendo File⇨Open desde el menú de su explorador web.

   Deberá ver lo siguiente en el explorador web:

```
Esto es una linea HTML
Esto es una linea PHP
```

   Debajo de estas líneas debería ver una gran tabla que muestra toda la información asociada con PHP en su sistema. Muestra la información PHP, la ruta y los nombres de archivos, los valores de las variables y el estado de algunas opciones. La tabla la produce la línea `phpinfo()` en las instrucciones de la prueba. Siempre que tenga una pregunta sobre la configuración de PHP, puede usar la instrucción `phpinfo()` para mostrar esta tabla y revisar las configuraciones.

4. **Verifique los valores PHP de los valores que necesita.**

   Por ejemplo, necesita que el soporte para MySQL esté habilitado. Al recorrer la lista, encuentre la sección dedicada a MySQL y asegúrese de que el soporte para MySQL esté `On`.

5. **Si es necesario cambie los valores.**

   Si no tiene acceso administrativo a PHP, debe pedirle al administrador que cambie cualquier valor que deba ser cambiado. Usted mismo podrá cambiar los valores si fue usted quien instaló PHP y/o tiene acceso administrativo a PHP. (En el Apéndice B discuto cómo cambiar la configuración de PHP.)

Someter MySQL a prueba

Si ya sabe que PHP está corriendo bien, pruebe si tiene acceso a MySQL usando PHP. Solo siga estos pasos:

1. **Cree el siguiente archivo en alguna parte de su espacio web con el nombre `mysql_up.php`.**

   Puede descargar el archivo de mi sitio web en `janet.valade.com`.

```
<html>
<head><title>Test MySQL</title></head>
<body>
<!-- mysql_up.php -->
<?php
$host="hostname";
$user="mysqlaccount";
$password="mysqlpassword";

mysql_connect($host,$user,$password);
$sql="show status";
$result = mysql_query($sql);
if ($result == 0)
   echo "<b>Error ". mysql_errno(). ": "
        . mysql_error(). "</b>";
else
{
?>
<!-- Table that displays the results -->
<table border="1">
  <tr><td><b>Variable_name</b></td><td><b>Value</b>
      </td></tr>
  <?php
    for ($i = 0; $i < mysql_num_rows($result); $i++) {
      echo "<TR>";
      $row_array = mysql_fetch_row($result);
      for ($j = 0; $j < mysql_num_fields($result); $j++)
      {
        echo "<TD>". $row_array[$j]. "</td>";
      }
      echo "</tr>";
    }
  ?>
</table>
<?php } ?>
</body></html>
```

2. **Las líneas 6, 7 y 8 del programa deben cambiarse. Estas líneas son**

```
$host="host";
$user="mysqlaccount";
$password="mysqlpassword";
```

Cambie "`Lastname`" por el nombre del PC donde está instalado MySQL, por ejemplo `databasehost.micompañia.com`. Si la base de datos MySQL está en el mismo PC que su sitio web, puede usar `localhost` como el nombre del anfitrión.

Cambie `mysqlaccountname` y `mysqlpassword` a los valores apropiados. (Comento las cuentas y las contraseñas de MySQL en el Capítulo 5). Si su cuenta MySQL no requiere contraseña, no digite nada entre las comillas, como sigue:

```
$password="";
```

3. **Dirija su explorador a** `mysql_up.php`.

   Debería ver una tabla con una larga lista de nombres y valores de variables. No querrá ver un mensaje de error o un mensaje de advertencia. No se preocupe por el contenido de la tabla. Lo único importante es que aparezca la tabla, para así saber que su conexión a MySQL está funcionando correctamente.

   Si no aparece ningún mensaje de error o de advertencia, MySQL está funcionando bien. Si ve un mensaje de error o de advertencia, debe corregir el problema que está provocando el mensaje.

Los mensajes de error y advertencia generalmente son muy claros. El siguiente es un mensaje de error común.

```
MySQL Connection Failed: Access denied for user:
    'user73@localhost' (Using password: YES)
```

Lo que este mensaje quiere decir es que MySQL no aceptó el número de su cuenta MySQL o su contraseña MySQL. Note que el mensaje dice `YES` para `Using password`, pero por razones de seguridad no muestra la contraseña que usted digitó. Si intentó con una contraseña en blanco, el mensaje diría `NO`.

Si recibe un mensaje de error, vuelva a revisar el número de su cuenta y su contraseña. Recuerde que este es su número de cuenta MySQL, y no su número de cuenta para registrarse en el PC. Si no se puede conectarse con el número de cuenta y contraseña que tiene, deberá contactar al departamento TI o a la compañía de hospedaje que le dio su número de cuenta. (Consulte el Capítulo 5 para una discusión más exhaustiva sobre las cuentas y las contraseñas de MySQL.)

Capítulo 3

Desarrollar una aplicación Web con base de datos

En este capítulo

- Planear su aplicación
- Seleccionar y organizar sus datos
- Diseñar su base de datos
- Resumen de la construcción de su base de datos
- Resumen de la escritura de los programas para su aplicación

Desarrollar una aplicación con base de datos para la Web no se reduce a almacenar información en bases de datos MySQL y digitar programas en PHP. El desarrollo debe empezar con el planeamiento. Construir las partes de la aplicación se hace después del planeamiento. Los pasos del desarrollo son:

1. **Desarrollar un plan, enumerando las tareas que su aplicación debe realizar.**
2. **Diseñar la base de datos necesaria para las tareas de su aplicación.**
3. **Construir la base de datos MySQL, con base en el diseño de la base de datos.**
4. **Escribir los programas con PHP que realizarán las tareas de la aplicación.**

En este capítulo comento detalladamente estos pasos.

Planear su aplicación Web con base de datos

Antes de poner un dedo en el teclado para escribir un programa en PHP, debe planear su aplicación. Este es probablemente el paso más importante en el desarrollo de su aplicación. Es muy doloroso descubrir, especialmente justo después de haber terminado el último programa para su aplicación, que dejó

algo por fuera y que tendrá que empezar de nuevo desde el principio. ¡También será difícil para su PC (y para sus pies) si usted se libera de sus frustraciones pateándolo por toda la habitación!

Un buen planeamiento evita esos dolorosos retrocesos. Además, le ayuda a concentrarse en las funciones propias de su aplicación, lo cual evitará que usted escriba partes de la aplicación que hagan cosas realmente llamativas pero que no tengan ningún propósito real en la aplicación terminada. Y si más de una persona está trabajando en su aplicación, el planeamiento garantiza que al final todas las partes encajen.

Identificar lo que espera de la aplicación

El primer paso de la fase de planeamiento es identificar exactamente por qué está desarrollando su aplicación y qué espera de ella. Por ejemplo, su propósito principal podría ser:

- Recopilar los nombres y las direcciones de los usuarios de modo que pueda crear una lista de clientes.
- Darle a los usuarios información sobre sus productos, como en un catálogo para clientes.
- Vender productos en línea.
- Brindar soporte técnico a personas que ya poseen sus productos.

Después de haber identificado claramente el propósito general de su aplicación, haga una lista de lo que exactamente espera que haga la aplicación. Por ejemplo, si su meta es desarrollar una base de datos con los nombres y las direcciones de sus clientes para propósitos de mercadeo, la lista de tareas requeridas de la aplicación es bastante corta:

- Brindar un formulario para que lo llenen los clientes.
- Almacenar la información de los clientes en una base de datos.

Si su meta es vender productos en línea, la lista es un poco más larga:

- Brindar a los clientes información sobre sus productos.
- Motivar al cliente para que compre el producto.
- Proporcionar una manera para que el cliente ordene el producto en línea.
- Proporcionar un método para que el cliente pague por el producto en línea.
- Validar el pago de modo que sepa que realmente va a recibir el dinero.
- Enviar el pedido a la persona responsable de alistar y enviar el producto al cliente.

En este punto del proceso de planeamiento, las tareas que desea que su aplicación realice son todavía bastante generales. Puede cumplir cada una de estas tareas de muchas maneras diferentes. Ahora debe examinar las tareas de cerca y detallar con exactitud cómo las cumplirá la aplicación. Por ejemplo, si su meta es vender productos en línea, podría ampliar la lista anterior así:

- **Brindar a los clientes información sobre sus productos.**
  - Mostrar una lista de las categorías de productos. Cada categoría es un vínculo.
  - Cuando el cliente hace clic en el vínculo de una categoría, aparece la lista de productos de esa categoría. El nombre de cada producto es un enlace.
  - Cuando un cliente hace clic en el vínculo de un producto, aparece la descripción del producto.
- **Motivar al cliente para que compre el producto.**
  - Presentar descripciones de los productos muy bien escritas y que comuniquen sus cualidades obviamente superiores.
  - Usar fotos que favorezcan a sus productos.
  - Hacer que estén disponibles en línea folletos a color sobre los productos.
  - Ofrecer descuentos por cantidad.
- **Proporcionar una manera para que el cliente ordene el producto en línea.**
  - Incluir un botón en el que los clientes puedan hacer clic para indicar su intención de comprar el producto.
  - Incluir un formulario que recopile la información necesaria sobre el producto que el cliente está ordenando, tal como su tamaño, color, etc.
  - Calcular y mostrar el costo total de todos los artículos en el pedido.
  - Calcular y mostrar los costos de envío.
  - Calcular y mostrar el impuesto de ventas.
  - Incluir formularios para que los clientes digiten sus direcciones de envío y de cobro.
- **Proporcionar un método para que el cliente pague por el producto en línea.**
  - Incluir un botón en el que los clientes puedan hacer clic para pagar con una tarjeta de crédito.
  - Mostrar un formulario que recoja la información de la tarjeta de crédito de los clientes.
- **Validar el pago de modo que sepa que realmente va a recibir el dinero.**

  El método usual es enviar la información de la tarjeta de crédito del cliente a un servicio de procesamiento de tarjetas de crédito.
- **Enviar el pedido a la persona responsable de alistar y enviar el producto al cliente.**

Bastará con enviar la información del pedido por correo electrónico al departamento de envío.

En este punto usted ya debe tener una idea bastante clara de lo que desea de su aplicación con base de datos. Sin embargo, esto no significa que sus metas no pueden cambiar. (De hecho, sus metas probablemente cambiarán mientras desarrolla su aplicación y descubre posibilidades nuevas). Al inicio del proyecto empiece con un plan tan completo como le sea posible, de esa manera se mantendrá concentrado y evitará topar con callejones sin salida o desviarse de la meta original.

Tome en cuenta a los usuarios

Identificar lo que usted espera que haga su aplicación con base de datos para la Web es solo un aspecto del planeamiento. También debe tomar en cuenta lo que sus usuarios querrán de ella. Por ejemplo, digamos que su meta es recopilar una lista con los nombres y las direcciones de sus clientes para propósitos de mercadeo. ¿Estarán dispuestos sus clientes a entregar esa información?

Su aplicación tiene que cumplir un propósito para los usuarios al igual que para usted. Si no es así la ignorarán. Antes, por ejemplo, de que los usuarios estén dispuestos a darle sus nombres y direcciones, necesitan percibir que al darle esa información se beneficiarán de alguna manera. Estos son ejemplos de por qué estarían dispuestos los usuarios a registrar sus nombres y direcciones en su sitio:

- **Para recibir un boletín:** Para que lo consideren valioso, el boletín debe cubrir la industria relacionada con sus productos. Debe ofrecer noticias y tendencias, y no sólo servir como espacio de mercadeo para sus productos.
- **Para participar en una rifa con un buen premio:** ¿Quién va a rechazar la oportunidad de ganar unas vacaciones con todos los gastos pagados a Hawai, o una camioneta familiar completamente nueva?
- **Para recibir descuentos especiales:** Por ejemplo, periódicamente podría ofrecerles a sus clientes descuentos especiales por correo electrónico.
- **Para ser notificado sobre productos nuevos o mejoras en los productos, tan pronto estén disponibles:** Por ejemplo, a los clientes les podría interesar que se les notifique cuando haya actualizaciones de software listas para descargar.
- **Para tener acceso a información valiosa:** Por ejemplo, debe registrarse en el sitio web de The New York Times para tener acceso a sus artículos en línea.

Agregue, a continuación, las tareas desde la perspectiva de los clientes a la lista de tareas que usted quiere que la aplicación realice. Por ejemplo, suponga que ya identificó esta lista de tareas necesarias para montar en línea una tienda de ventas al detalle:

- Brindar un formulario para que lo llenen los clientes.
- Almacenar la información de los clientes en una base de datos.

Si toma en cuenta el punto de vista del cliente, la lista se amplía un poco:

- Presentar una descripción de las ventajas que tendrán los clientes si se registran en el sitio.
- Brindar un formulario para que lo llenen los clientes.
- Agregar las direcciones electrónicas de los clientes a la lista de distribución del boletín.
- Almacenar la información de los clientes en una base de datos.

Si su lista incluye las tareas que usted desea y las tareas que sus usuarios desean, tiene un plan para una aplicación web que valdrá la pena desarrollar, y que desde el punto de vista de sus usuarios valdrá la pena usar.

Hacer que el sitio sea fácil de usar

Además de planear lo que su aplicación web va a hacer, debe considerar cómo lo va a hacer. Hacer que su aplicación sea fácil de usar es importante: sus clientes no comprarán sus productos si no los pueden encontrar. Y si los clientes no encuentran la información que necesitan en un tiempo relativamente corto, buscarán en otro lugar. En la Web, los clientes siempre pueden irse fácilmente para otro lugar.

Hacer su aplicación fácil de usar es ingeniería de usabilidad. La usabilidad en la Web incluye cosas como:

- **Navegación:** Para el usuario debe ser inmediatamente obvio qué hay en su sitio y dónde está localizado.
- **Gráficos:** Los gráficos hacen que su sitio sea atractivo, pero los archivos de gráficos pueden ser muy lentos de mostrar.
- **Acceso:** Algunas decisiones sobre el diseño pueden hacer que su aplicación sea o no accesible a los usuarios con discapacidades tales como problemas visuales.
- **Exploradores:** Diferentes exploradores web (incluso versiones diferentes del mismo explorador) pueden mostrar el mismo archivo de HTML (Lenguaje de Marcado de Hipertexto/HyperText Markup Language) en forma diferente.

La usabilidad en la Web es un tema amplio e importante, aunque entrar en más detalles sobre este tema está más allá del alcance de este libro. Pero no tema, puede encontrar mucha información útil sobre la usabilidad en la Web — lo adivinó — en la Web. Asegúrese de revisar los sitios web de los expertos en usabilidad Jakob Nielsen (`www.useit.com`) y Jarod Spool (`http://world.std.com/~uieweb/`). Vincent Flanders también tiene un sitio divertido lleno de información útil sobre el diseño Web en `WebPagesThatSuck.com`. Hay libros sobre el tema que también pueden ayudarle, tal como Web Design For Dummies de Lisa Lopuck (Wiley Publishing, Inc.).

Dejar espacio para expansiones

Algo indudable sobre su aplicación web es que cambiará con el tiempo. Con el paso del tiempo podrían ocurrírsele nuevas funciones para ella, o simplemente deseará cambiarle algo. O tal vez el software para sitios web mejore de modo que su aplicación podría hacer cosas que no podía hacer cuando la montó por primera vez. Cualquiera que sea la razón, su sito web cambiará. Cuando planea su aplicación, debe tener presentes los cambios futuros.

Puede diseñar su aplicación en pasos, tomando en cuenta los cambios planeados. Puede desarrollar un plan en el cual construya una aplicación hoy que cumpla con sus más inmediatas necesidades y ponerla a disposición tan pronto como esté lista. Su plan puede incluir agregar funciones a la aplicación tan pronto como pueda desarrollarlas. Por ejemplo, puede construir un catálogo de productos y publicarlo en su sitio web apenas esté listo. En ese momento puede empezar a trabajar en la función para hacer pedidos en línea, la cuál agregará cuando esté lista.

No necesariamente podrá predecir todas las funciones que deseará tener en su aplicación en el futuro. Por ejemplo, podría diseñar su sitio web de viajes con secciones para todos los posibles destinos actuales, pero el futuro podría sorprenderle. ¿Viajes a Marte? ¿A Alpha Centauro? ¿A un universo alterno? Planee su aplicación con la flexibilidad necesaria para poder agregarle funcionalidad en el futuro.

Escríbalo

Escriba su plan. A menudo me va a escuchar diciendo esto. Hablo desde la dolorosa experiencia de no escribir el plan. Mientras lo desarrolla, su plan es lo más importante en su mente y está totalmente claro. Pero en unas cuantas semanas se sorprenderá al descubrir que se ha nublado absolutamente, mientras su atención se ocupaba de asuntos más urgentes. O dentro de un año querrá hacer algunos cambios en la aplicación y no recordará exactamente cómo la diseñó. O está trabajando con un socio para desarrollar una aplicación y descubre que su socio malentendió su explicación verbal y desarrolló funciones para la aplicación que no concuerdan con su plan. Puede evitar todos estos tipos de problemas si lo escribe todo.

Presentamos los dos ejemplos creados para este libro

En las dos secciones siguientes presento los dos ejemplos de aplicación con bases de datos para la Web que he creado para este libro. Me refiero a estos dos ejemplos a lo largo de todo el libro para demostrar aspectos sobre el diseño y desarrollo de aplicaciones.

Cosas para vender

El primer ejemplo es un catálogo de productos en línea. Usted es el dueño de una tienda de mascotas y desea que su catálogo brinde a los clientes información sobre las mascotas que tiene a la venta. Vender mascotas en línea no es viable, pero aún así usted está considerando la idea de permitir a los clientes "reservar" mascotas en línea, es decir, antes de que vengan a la tienda a comprarlas. Actualmente la aplicación simplemente es un catálogo en línea. Los clientes pueden revisar el catálogo en línea y luego ir a la tienda a comprar la mascota. La información sobre todas las mascotas se almacena en una base de datos y los clientes pueden buscar en la base de datos la información sobre mascotas específicas o tipos de mascotas.

Este es su plan para esta aplicación:

- **Permitirles a los clientes seleccionar sobre cuál mascota desean ver información.**

  Ofrezca dos métodos de selección:

  - **Seleccionar de una lista de vínculos:** Mostrar una lista de vínculos que son categorías de mascotas (por ejemplo, perros, gatos, dinosaurios, etc.). Cuando el cliente hace clic en el vínculo de la categoría, aparece una lista de mascotas. Cada mascota en la lista es un vínculo hacia una descripción de la mascota.
  - **Digitar los términos de la búsqueda:** Mostrar un formulario de búsqueda en el cual los clientes puedan digitar palabras que describen el tipo de mascota que están buscando. La aplicación busca en la base de datos las palabras que coinciden y muestra la información sobre las mascotas que concuerdan con las palabras de la búsqueda. Por ejemplo, un cliente puede digitar **gato** para ver una lista de todos los gatos disponibles. Cada gato en la lista es un vínculo hacia la descripción de ese gato.

- **Mostrar una descripción de la mascota cuando el cliente hace clic en el vínculo.**

  La descripción está guardada en una base de datos.

Sólo para Miembros

El segundo ejemplo de una aplicación con base de datos para la Web se relaciona con el ejemplo anterior de la tienda de mascotas. Además del catálogo en línea, también quiere poner una sección en el sitio web de su tienda de mascotas que sea sólo para miembros. Para entrar en esta área del sitio los clientes deben primero inscribirse, proporcionando sus nombres y direcciones. En esta sección "Sólo para miembros", los clientes pueden ordenar alimento para mascotas con descuento, averiguar sobre mascotas que han pedido pero que todavía no han llegado, y también tener acceso a artículos con noticias e información sobre mascotas y el cuidado de las mascotas.

Este es su plan para esta aplicación:

- **Mostar una descripción sobre la información y las características especiales que están disponibles en la sección Sólo para miembros.**
- **Incluir un área donde los clientes puedan inscribirse a la sección Sólo para miembros.**
  - **Incluir un vínculo hacia el área de inscripción.**
  - **Mostrar un formulario en el área de inscripción en el cual los clientes puedan digitar la información para inscribirse.**

    El formulario debe incluir espacio para el nombre de registro del usuario y una contraseña, así como cualquier otra información que desee recopilar.
  - **Validar la información que el usuario digitó.**

    Por ejemplo, verifique que el código postal tenga la longitud correcta, que la dirección electrónica esté en el formato correcto, etc.
  - **Almacenar la información en la base de datos.**
- **Incluir una sección para que ingresen los clientes que ya se han registrado en la sección "Sólo para miembros".**
  - **Mostrar un formulario de entrada que pida al cliente su nombre de usuario y la contraseña.**
  - **Comparar la contraseña y el nombre de usuario digitados por el cliente con los nombres de usuario y las contraseñas en la base de datos.**

    Si no se encuentra uno que concuerde, mostrar un mensaje de error.
- **Mostrar la página "Sólo para miembros" si el cliente se registra exitosamente.**

Diseñar la base de datos

Después de haber determinado con exactitud qué hará su aplicación con base de datos para la Web (consulte el inicio de este capítulo si todavía no lo ha hecho), ya está listo para diseñar la base de datos que guardará la información que necesita la aplicación. Diseñar la base de datos incluye identificar los datos que necesita y organizar los datos en la forma requerida por el software de la base de datos.

Escoger los datos

Primero tiene que identificar cuál información debe estar en su base de datos. Con base en la lista de tareas que desea que la aplicación realice, determine cuál información necesita para completar cada una de esas tareas.

Estos son algunos ejemplos:

- Un catálogo en línea necesita una base de datos que contenga información sobre los productos.
- Una aplicación para hacer pedidos en línea necesita una base de datos que pueda guardar la información sobre el cliente y la información sobre los pedidos.
- Un sitio web de viajes necesita una base de datos con información sobre destinos, reservaciones, tarifas, horarios, etcétera.

En muchos casos, su aplicación incluirá una tarea que recopile información de los usuarios. Tendrá que balancear su necesidad de recopilar toda la información potencialmente útil en la que pueda pensar, contra la renuencia de sus usuarios a dar información personal — así como su deseo de evitar formularios en los que habría que invertir mucho tiempo para poder contestarlos. Una manera es pedir información opcional. Hay usuarios a quienes no les importará digitarla, pero los usuarios que no estén de acuerdo podrán dejar los campos en blanco. Otra posibilidad es ofrecer un incentivo: mientras más largo sea el formulario más atractivo debe ser el incentivo que necesitará para motivar a los usuarios a llenar el formulario. Un usuario podría estar dispuesto a llenar un formulario muy corto para participar en una rifa que ofrezca como premio dos tiquetes para un estreno cinematográfico. Pero si el formulario es largo y complicado, el precio debe ser más valioso, tal como un viaje gratis a California que incluya un tour a un estudio de cine en Hollywood.

En la aplicación del primer ejemplo, sus clientes recorren el catálogo en línea buscando información sobre las mascotas que tal vez quisieran comprar. A usted le conviene que los clientes vean información que los motive a comprar la mascota. La información que debiera estar disponible en la base de datos para ser vista por los clientes es:

- **El nombre de la mascota**

  Por ejemplo poodle, unicornio y así sucesivamente
- **Una descripción de la mascota**
- **Una foto de la mascota**
- **El costo de la mascota**

En la aplicación del segundo ejemplo, la sección Sólo para miembros, le conviene guardar información sobre los miembros inscritos. La información que debiera guardar en la base de datos es:

- El nombre del miembro
- La dirección del miembro
- El número telefónico del miembro
- El número de fax del miembro
- La dirección electrónica del miembro

Tómese tiempo para desarrollar una lista completa de la información que necesita guardar en su base de datos. Aunque puede cambiar y agregar información a su base de datos después de haberla desarrollado, es más fácil incluir la información desde el principio. Además, si añade información a la base de datos posteriormente — después de que está en uso — los primeros usuarios en la base de datos tendrán su información incompleta. Por ejemplo, si en un momento dado cambia su formulario y empieza a preguntar la edad del usuario, no tendrá la edad de las personas que llenaron el formulario antes y que todavía están en la base de datos.

Organizar los datos

MySQL es un SABD relacional (Sistema Administrativo de Base de Datos), lo cual significa que los datos se organizan en tablas. (Consulte el Capítulo 1 para mayor información sobre MySQL.) Usted puede establecer relaciones entre las tablas en la base de datos.

Organizar datos en tablas

Las tablas de un SABD relacional se organizan como cualesquiera otras tablas que usted ya conoce, es decir, en filas y columnas, como se muestra en la Figura 3-1. Un campo es el lugar en el cual se cruzan una fila y una columna particulares, es decir, una celda individual.

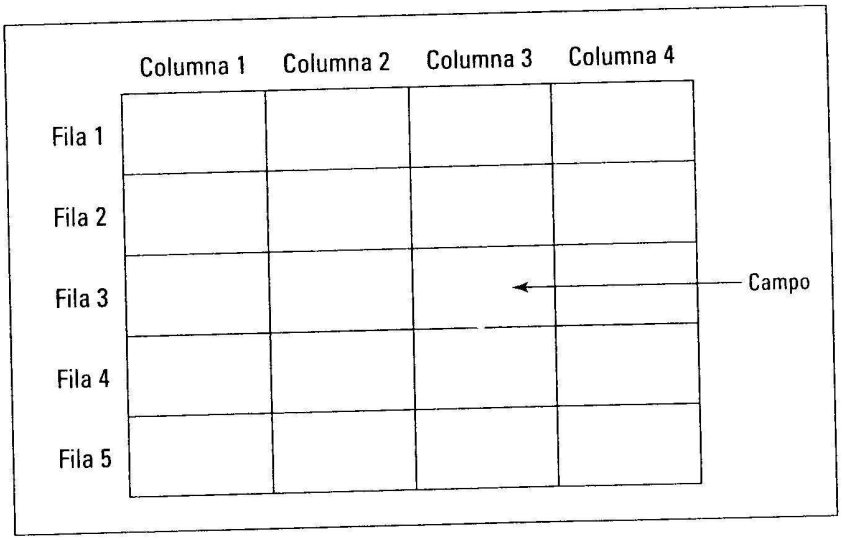


Figura 3-1: Los datos en MySQL se organizan en tablas.

Cada tabla se concentra en un objeto (una cosa) sobre el cual se desea guardar información. Estos son algunos ejemplos de objetos:

- Clientes
- Productos
- Compañías

- Animales
- Ciudades
- Salas
- Libros
- PCs
- Formas
- Documentos
- Proyectos
- Semanas

Se debe crear una tabla para cada objeto. El nombre de la tabla debe identificar claramente los objetos que contiene con una palabra o un término descriptivos. El nombre debe ser una cadena de caracteres sin espacios. El nombre de la tabla puede contener letras, números, líneas de subrayado (_) o signos de dólares ($). Se acostumbra nombrar la tabla en singular. Así, un nombre para una tabla de clientes podría ser `Cliente`, y una tabla con los pedidos de los clientes se podría llamar `PedidodelCliente`. El uso de mayúsculas y minúsculas es significativo en Linux/Unix pero no en Windows: PedidodelCliente y pedidodelcliente son lo mismo para Windows, pero no para Linux o Unix.

En la jerga de las bases de datos, un objeto es una entidad, y una entidad tiene atributos. En la tabla, cada fila representa una entidad, y las columnas contienen los atributos de cada entidad. Por ejemplo, en una tabla de clientes cada fila contiene información sobre un único cliente. Algunos de los atributos en las columnas podrían ser nombre, apellido, número telefónico, edad y otros datos por el estilo.

Estos son los pasos a seguir para organizar sus datos en tablas:

1. **Nombre su base de datos.**

   Asigne un nombre a la base de datos de su aplicación. Por ejemplo, una base de datos que contenga información sobre las familias en un vecindario podría llamarse `DirectorioFamilias`.

2. **Identifique los objetos.**

   Revise la lista de información que desea guardar en la base de datos. (Si todavía no la ha hecho, consulte la sección "Escoger los datos" anteriormente en este capítulo). Analice su lista e identifique los objetos. Por ejemplo, posiblemente la base de datos `DirectorioFamilias` necesitaría almacenar lo siguiente:

   - Nombre de cada miembro de la familia
   - Dirección de la casa
   - Número telefónico
   - Edad de cada miembro de la familia
   - El cereal que cada miembro de la familia prefiere para desayunar

Cuando analice esta lista cuidadosamente, se dará cuenta de que está almacenando información sobre dos objetos: la familia y los miembros de la familia. O sea, la dirección y el número telefónico son para la familia en general, pero el nombre, la edad y el cereal favorito son para miembros individuales de la familia.

3. **Defina y nombre una tabla para cada objeto.**

   Por ejemplo, la base de datos `DirectorioFamilias` necesita una tabla llamada `Familia` y una tabla llamada `MiembrodelaFamilia`

4. **Identifique los atributos de cada objeto.**

   Analice su lista de información e identifique los atributos que necesita guardar sobre cada objeto. Desglose la información a guardar en las partes más pequeñas posibles, dentro de lo razonable. Por ejemplo, cuando guarde el nombre de una persona en una tabla, puede desglosar el nombre en nombre y apellido. Hacer esto le permitirá ordenar los datos por apellido, lo cual sería más difícil de hacer si el nombre y el apellido estuvieran guardados juntos. De hecho, podría desglosar el nombre en nombre, segundo nombre, primer y segundo apellido, aunque no son muchas las aplicaciones que necesitan usar el segundo nombre en forma separada.

5. **Defina y nombre las columnas para cada uno de los atributos que identificó en el Paso 4.**

   Dé a cada columna un nombre que identifique claramente la información que hay en esa columna. Los nombres de las columnas deben ser de una sola palabra, sin espacios. Por ejemplo, podría nombrar columnas así: `nombre` y `apellido` o `primer_nombre` y `apellido`.

   Algunas palabras están reservadas para el uso específico de MySQL o SQL y no se pueden usar como nombres de columnas. Estas palabras se usan en instrucciones SQL o están reservadas para uso futuro. Por ejemplo `ADD`, `ALL`, `AND`, `CREATE`, `DROP`, `GROUP`, `ORDER`, `RETURN`, `SELECT`, `SET`, `TABLE`, `USE`, `WHERE` y muchas, muchas más no se pueden usar como nombres de columnas. Para ver la lista completa de palabras reservadas, consulte el manual en línea de MySQL en `www.mysql.com/doc/en/Reserved_words.html`.

6. **Identifique la clave primaria.**

   Cada fila en una tabla necesita un identificador único. En una misma tabla no puede haber dos filas exactamente iguales. Cuando diseñe su tabla, decida cuál columna tiene el identificador único, llamado llave primaria. La llave primaria puede estar conformada por la combinación de más de una columna. En muchos casos, los atributos de sus objetos no tendrán un identificador único. Por ejemplo, una tabla de clientes podría no tener un identificador único porque dos clientes pueden tener el mismo nombre. Cuando no hay una columna de identificador único, debe agregar una columna específicamente para que sea la llave primaria. Frecuentemente, una columna con una secuencia numérica se usa para este propósito. Por ejemplo, en la Figura 3-2 la clave primaria es el campo `cliente_id` porque cada cliente tiene un número de identificación único.

cliente_id	nombre	apellido	telefono
27895	John	Smith	555-5555
44555	Joe	Lopez	555-5553
23695	Judy	Chang	555-5552
27822	Jubal	Tudor	555-5556
29844	Joan	Smythe	555-5559

Figura 3-2: Un ejemplo de la tabla `Cliente`.

7. **Defina los valores predeterminados.**

   Puede definir los valores predeterminados que MySQL asignará a los campos cuando no se digiten datos en ellos. No es imprescindible tener valores predeterminados, pero a menudo es útil. Por ejemplo, si su aplicación guarda direcciones que incluyen el país, puede especificar EEUU como predeterminado. Si el usuario no digita ningún país, se usará EEUU.

8. **Identifique columnas con datos obligatorios.**

   Puede definir columnas a las que no se les permitirá estar vacías (también llamadas `NULL`). Por ejemplo, la columna que contiene su clave primaria no puede estar vacía. Eso significa que MySQL no va a crear la fila si no se guarda ningún valor en la columna. El valor puede ser un espacio en blanco o una cadena vacía (por ejemplo, ""), pero debe guardarse algún valor en la columna. Puede determinar que otras columnas, además de la que contenga la clave primaria, indiquen que hay un error cuando están vacías.

Las bases de datos bien diseñadas guardan cada trozo de información en un solo lugar. Guardarlo en más de un lugar es ineficiente y crea problemas si hay que cambiar la información. Si cambia la información en un lugar pero olvida cambiarla en el otro lugar, su base de datos tendrá serios problemas.

Si descubre que está guardando los mismos datos en varias filas, probablemente necesitará reorganizar sus tablas. Por ejemplo, suponga que está guardando datos sobre libros, incluyendo la dirección de la casa editorial. Cuando digita los datos, descubre que está digitando la dirección de la misma editorial en muchas filas. Un modo más eficiente de almacenar estos datos sería guardar la información sobre los libros en una tabla y la información sobre la editorial del libro en una tabla aparte. Puede definir dos tablas: `Libro` y `Editorial`. En la tabla Libro, tendrá las columnas `titulo`, `autor`, `fecha_pub` y `precio`. En la tabla `Editorial` tendrá columnas tales como `nombre`, `DireccionCalle`, `ciudad`, etc.

Cómo crear relaciones entre tablas

Hay tablas en las bases de datos que se relacionan entre sí. Con mucha frecuencia, una fila en una tabla estará relacionada con varias filas en otra tabla. Se necesita una columna para conectar las filas relacionadas en tablas diferentes. En muchos casos, habrá que incluir una columna en una tabla para guardar los datos que concuerden con los datos en la columna de llave primaria de otra tabla.

Un tipo de aplicación común que necesita una base de datos con dos tablas relacionadas es una aplicación para pedidos de clientes. Por ejemplo, una tabla contiene la información sobre el cliente, tal como nombre, dirección, teléfono, etc. Cada cliente puede tener de cero a muchos pedidos. Puede guardar la información sobre sus pedidos en la tabla con la información del cliente, pero tendría que crearse una fila completamente nueva cada vez que el cliente hiciera un pedido, y cada fila nueva tendría que contener toda la información del cliente. Sería mucho más eficiente guardar los pedidos en una tabla aparte. La tabla `Pedidos` tendría una columna que contenga la clave primaria de una fila en la tabla `Cliente`, de modo que el pedido se relacione con la fila correcta en la tabla `Cliente`. La relación se muestra en las tablas de las Figuras 3-2 y 3-3.

La tabla `Cliente` en este ejemplo luce como en la Figura 3-2 (consulte la sección anterior). Observe el identificador único (`cliente_id`) para cada cliente.

La tabla `Pedido` relacionada se muestra en la Figura 3-3. Observe que tiene la misma columna `cliente_id` que aparece en la tabla `Cliente`. De este modo, la información del pedido en la tabla `Pedido` se conecta con el nombre y número telefónico del cliente con el que se relaciona en la tabla `Cliente`.

En este ejemplo, las columnas que relacionan la tabla `Cliente` y la tabla `Pedido` tienen el mismo nombre. Podrían tener nombres diferentes siempre y cuando los datos en las columnas fueran los mismos.

Num_orden	cliente_id	num_item	costo
87-222	27895	cat-3	200.00
87-223	27895	cat-4	225.00
87-224	44555	horse-1	550.00
87-225	44555	dog-27	210.00
87-226	27895	bird-1	50.00

Figura 3-3: Un ejemplo de la tabla `Pedido`.

Diseñar las bases de datos de los ejemplos

En las dos secciones siguientes, diseño las dos bases de datos para los dos ejemplos de aplicaciones usados en este libro.

Proceso del diseño del Catálogo de mascotas

Suponga que quiere mostrar la siguiente lista de información cuando los clientes busquen en su catálogo de mascotas:

- **El nombre de la mascota**

  Por ejemplo, poodle, unicornio y así sucesivamente.
- **Una descripción de la mascota**
- **Una foto de la mascota**
- **El costo de la mascota**

En el plan del Catálogo de mascotas, aparece una lista de categorías de mascotas. Esto implica que cada mascota debe clasificarse en una categoría de mascotas y que la categoría de mascotas debe almacenarse en la base de datos.

Se debe diseñar la base de datos `CatalogodeMascotas` siguiendo los pasos presentados en la sección "Organizar los datos en tablas", anteriormente en este capítulo:

1. **Nombre su base de datos.**

   La base de datos para el Catálogo de mascotas se llama `CatalogodeMascotas`.

2. **Identifique los objetos.**

   La lista de la información es:

   - **El nombre de la mascota** (por ejemplo, poodle, unicornio, y así sucesivamente.)
   - **Una descripción de la mascota**
   - **Una foto de la mascota**
   - **El costo de la mascota**
   - **La categoría de la mascota**

   Toda esta información es sobre mascotas, así que el único objeto para esta lista es `Mascota`.

3. **Defina y nombre una tabla para cada objeto.**

   La aplicación para el Catálogo de mascotas necesita una tabla llamada `Mascota`.

4. **Identifique los atributos de cada objeto.**

   Ahora debe considerar la información en detalle:

   - **El nombre de la mascota:** Un solo atributo; por ejemplo poodle, unicornio y así sucesivamente. Sin embargo, es muy posible que su tienda de mascotas tenga más de un poodle a la venta al mismo tiempo. Por lo tanto, su tabla necesita un identificador único que sirva como clave primaria.
   - **El número de identificación de la mascota:** Un número secuencial asignado a cada mascota cuando se añade a la tabla. Este número es la clave primaria.
   - **La descripción de la mascota:** Dos atributos: la descripción escrita de la mascota tal como aparecería en el catálogo impreso y el color de la mascota.
   - **La foto de la mascota:** El nombre de la ruta hacia un archivo gráfico que contenga una hermosa foto de la mascota.
   - **El costo de la mascota:** La cantidad en efectivo que la tienda pide por la mascota.
   - **La categoría de la mascota:** Dos atributos: el nombre de la categoría a la que pertenece la mascota — por ejemplo perro, caballo, dragón — y una descripción de la categoría.

   Sería ineficiente incluir dos tipos de información en la tabla `Mascota`:

   - La información de la categoría incluye una descripción de la categoría. Como cada categoría puede incluir varias mascotas, incluir la descripción de la categoría en la tabla `Mascota` daría como resultado que la misma descripción aparezca en varias filas. Es más eficiente definir la Categoría de la mascota como un objeto con su propia tabla.
   - Si la mascota viene en varios colores, toda la información de la mascota se repetiría en una fila aparte para cada color. Es más eficiente definir Color de la mascota como un objeto con su propia tabla.

   Las tablas agregadas se llaman `TipodeMascota` y `ColordelaMascota`.

5. **Defina y nombre las columnas.**

   La tabla `Mascota` tiene una fila para cada mascota. Las columnas de la tabla `Mascota` son

   - `Idmascota`: Un número secuencial único asignado a cada mascota.
   - `Nombremascota`: Nombre de la mascota.
   - `Tipomascota`: El nombre de la categoría. Esta es la columna que conecta la mascota con la fila correcta en la tabla `TipodeMascota`.
   - `Descripcionmascota`: La descripción de la mascota.
   - `Precio`: El precio de la mascota.
   - `Pix`: El nombre del archivo gráfico que contiene una foto de la mascota.

La tabla TipodeMascota tiene una fila para cada categoría de mascotas. Tiene las siguientes columnas:

- Tipomascota: El nombre de la categoría de un tipo de mascota. Esta es la clave primaria para esta tabla. Observe que la tabla Mascota tiene una columna con el mismo nombre. Esta columna vincula esta tabla con la tabla Mascota.
- Descripciontipo: La descripción del tipo de mascota.

La tabla ColordelaMascota tiene una fila para cada color de mascota. Tiene las siguientes columnas:

- Nombremascota: El nombre de la mascota. Esta es la columna que conecta la fila de color con la fila correcta en la tabla Mascota.
- Colormascota: El color de la mascota.

6. **Identifique la clave primaria.**
   - La clave primaria de la tabla Mascota es Idmascota.
   - La clave primaria de la tabla TipodeMascota es Tipomascota.
   - La clave primaria de la tabla ColordelaMascota es Nombremascota y Colormascota juntas.
7. **Defina los valores predeterminados.**

   No se definen valores predeterminados para ninguna de las tablas.
8. **Identifique las columnas con datos obligatorios.**

   A las siguientes columnas no se les permitirá nunca estar vacías:
   - Idmascota
   - Nombremascota
   - Colormascota
   - Tipomascota

   Estas son las columnas con claves primarias. No debe permitirse que haya una fila sin estos valores en las tablas.

Proceso del diseño del área Sólo para miembros

Suponga que ha creado la siguiente lista de información que desea guardar cuando los clientes se registren en la sección Sólo para miembros de su sitio web:

- Nombre del miembro
- Dirección del miembro
- Número telefónico del miembro

- Número de fax del miembro
- Dirección electrónica del miembro

Además, también le convendría guardar la fecha en que un miembro se inscribe y rastrear la frecuencia con la que cada miembro entra a la sección Sólo para miembros.

Diseñe la base de datos Sólo para miembros siguiendo los pasos presentados en la sección "Organizar los datos en tablas", anteriormente en este capítulo:

1. **Nombre su base de datos.**

   La base de datos para la sección Sólo para miembros se llama `DirectoriodeMiembros`.

2. **Identifique los objetos.**

   La lista de información es

   - Nombre del miembro
   - Dirección del miembro
   - Número telefónico del miembro
   - Número de fax del miembro
   - Dirección electrónica del miembro
   - Fecha de inscripción del miembro
   - Entradas del miembro

   Toda esta información pertenece a los miembros, así que el único objeto para esta lista es `miembro`.

3. **Defina y nombre una tabla para cada objeto.**

   La base de datos `DirectoriodeMiembros` necesita una tabla llamada `Miembro`.

4. **Identifique los atributos de cada objeto.**

   Fíjese detalladamente en la lista de información:

   - **Nombre del miembro:** Dos atributos: nombre y apellido.
   - **Dirección del miembro:** Cuatro atributos: dirección física, ciudad, estado y código postal. Como actualmente usted solo tiene tiendas de mascotas en Estados Unidos, puede asumir que la dirección del miembro es una dirección en el formato de direcciones postales de Estados Unidos.
   - **Número telefónico del miembro:** Un atributo.
   - **Número de fax del miembro:** Un atributo.
   - **Dirección electrónica del miembro:** Un atributo.
   - **Fecha de inscripción del miembro:** Un atributo.

Varias piezas de información se relacionan con las entradas del miembro en el área especial:

- El ingreso en la sección Sólo para miembros requiere de un nombre de registro y una contraseña. Estos dos ítemes deben estar almacenados en la base de datos.
- La forma más fácil de rastrear las entradas de los miembros es almacenando la fecha y la hora en que el usuario entró en la sección Sólo para miembros.

Como cada miembro puede tener muchas entradas, habrá que almacenar muchas fechas y horas de entrada. Por lo tanto, en lugar de definir el tiempo de entrada como un atributo del miembro, defina la entrada como un objeto relacionado con el miembro, pero con su propia tabla.

La tabla agregada se llama `Entrada`. El atributo de un objeto de entrada es su tiempo de entrada (el tiempo incluye la fecha).

5. **Defina y nombre las columnas.**

La tabla `Miembro` tiene una fila para cada miembro. Las columnas para la tabla `Miembro` son:

- `Nombreentrada`

Cada nombre debe ser único. Los programas en la aplicación deben asegurarse de que no haya dos nombres de entrada iguales.

- `Clave`
- `Fechadecreacion`
- `Nombre`
- `Apellido`
- `Calle`
- `Ciudad`
- `Estado`
- `CodigoPostal`
- `email`
- `Telefono`
- `Fax`

La tabla `Entrada` tiene una fila para cada entrada: o sea, cada vez que un miembro entra en la sección Sólo para miembros. Tiene las columnas siguientes:

- `Nombreentrada`: El nombre de registro del miembro que entró. Esta es la columna que vincula esta tabla con la tabla `Miembro`. Este es un valor único en la tabla `Miembro` pero no es un valor único en esta tabla.
- `Tiempoentrada`: La fecha y hora de entrada.

6. **Identifique la clave primaria.**
   - La clave primaria para la tabla `Miembro` es `Nombreentrada`.
   - La clave primaria para la tabla `Entrada` es `Nombreentrada` y `Tiempo_entrada` juntos.
7. **Defina los valores predeterminados.**

   No hay valores predeterminados para ninguna de las tablas.
8. **Identifique las columnas con datos obligatorios.**

   Las siguientes columnas nunca deben estar vacías:
   - `Nombreentrada`
   - `Contraseña`
   - `Tiempoentrada`

   Estas columnas son las columnas con la clave primaria. En las tablas nunca debe permitirse que haya una fila sin estos valores.

Tipos de datos

MySQL almacena información en formatos diferentes con base en el tipo de información que usted le dice a MySQL que debe esperar. MySQL permite que tipos diferentes de datos se usen de maneras diferentes. Los tipos principales de datos son datos de caracteres, numéricos y de fecha/hora.

Datos de caracteres

Los datos más comunes son los datos de caracteres, es decir, datos que se guardan en cadenas de caracteres y que solo se pueden manipular en cadenas. La mayor parte de la información que guardará será de este tipo, por ejemplo datos como el nombre del cliente, su dirección, número telefónico, descripción de la mascota, etc. Los datos de caracteres se pueden mover e imprimir. Dos cadenas de caracteres se pueden poner juntas (concatenadas); se puede seleccionar una subcadena de una cadena más larga, y una cadena se puede sustituir por otra.

Los datos de caracteres se pueden almacenar en un formato de longitud fija o un formato de longitud variable.

- **Formato de longitud fija:** En este formato, MySQL reserva un espacio fijo para los datos. Si los datos son más largos que el largo fijado, solo se guardarán los caracteres que quepan en ese largo, y el resto de los caracteres no se guardará. Si la cadena es más corta que la longitud fijada, el espacio extra se deja vacío y se desperdicia.

- **Formato de longitud variable:** En este formato, MySQL guarda la cadena en un campo que tiene la misma longitud que la cadena. Aún así debe especificar la longitud para la cadena, pero si la cadena es más corta que la longitud especificada, MySQL usa el espacio requerido en lugar de dejar el espacio extra vacío. Si la cadena es más larga que el espacio especificado, los caracteres extra no se almacenan.

Si la longitud de una cadena de caracteres varía muy poco, use el formato de longitud fija. Por ejemplo, una longitud de 10 funciona para todos los códigos postales, incluyendo los que tienen el número postal+4. Si el código postal no incluye el número postal+4, solo quedarán cinco espacios vacíos. Sin embargo, si su cadena de caracteres puede variar más que unos cuantos caracteres, use el formato de longitud variable para ahorrar espacio. Por ejemplo, la descripción de la mascota podría ser Murciélago pequeño o podría ocupar varias líneas. Así que sería mejor guardar esta descripción en un formato de longitud variable.

Datos numéricos

Otro tipo común de datos son los numéricos, es decir, datos que se guardan como números. Los números decimales (por ejemplo 10.5, 2.34567, 23456.7) se pueden guardar así como los números enteros (por ejemplo, 1, 2, 248). Cuando los datos se guardan numéricamente, se pueden usar en operaciones numéricas tales como suma, resta, elevar al cuadrado, etc. Sin embargo, si los datos no van a usarse en operaciones numéricas es mejor guardarlos como cadenas de caracteres, porque el programador los usará como cadenas de caracteres. No hará falta ninguna conversión. Por ejemplo, probablemente no tendrá que sumar los dígitos en los números telefónicos de los usuarios, así que el número telefónico debiera guardarse como una cadena de caracteres.

MySQL guarda números positivos y negativos, pero usted puede pedirle a MySQL que guarde solo números positivos. Si sus datos no van a ser negativos, guarde los datos `unsigned` (sin signo, es decir, sin usar los signos + o — antes del número). La población de una ciudad, por ejemplo, o el número de páginas en un documento nunca serán un número negativo.

Datos de fecha y hora

Un tercer tipo común de datos son los de fecha y hora. Los datos guardados como fecha se pueden mostrar en diversos formatos de fecha. También se pueden usar para determinar el lapso de tiempo entre dos fechas o dos horas, o entre una fecha u hora específicas y alguna fecha y hora arbitrarias.

Datos de enumeración

Algunas veces los datos solo pueden tener un número limitado de valores. Por ejemplo, una columna podría tener como únicos valores posibles `si` y `no`. MySQL ofrece un tipo de datos llamados de enumeración para usar con este tipo de datos. Usted le indica a MySQL cuáles valores pueden guardarse en la columna (por ejemplo `si, no`), y MySQL no guardará ningún otro valor.

Nombres de los tipos de datos en MySQL

Cuando usted crea una base de datos debe decirle a MySQL, usando los nombres específicos de MySQL para los tipos de datos, cuáles tipos de datos debe esperar en una columna en particular. La Tabla 3-1 muestra los tipos de datos en MySQL usados con más frecuencia en las aplicaciones con bases de datos para la Web.

Tabla 3-1 Tipos de datos en MySQL

Tipos de datos en MySQL	*Descripción*
`CHAR(length)`	Cadena de caracteres de longitud fija.
`VARCHAR(length)`	Cadena de caracteres de longitud variable. La cadena más larga que se puede guardar es `length`, que debe estar entre 1 y 255.
`TEXT`	Cadena de caracteres de longitud variable con una longitud máxima de 64KB de texto.
`INT(length)`	Número entero cuyo rango va de —2147483648 a +2147483647. El número que se puede mostrar está limitado por `length`. Por ejemplo, si `length` es 4, solo los números de —999 a 9999 se podrán mostrar, aunque los números más altos se guardarán.
`INT(length) UNSIGNED`	Número entero cuyo rango va de 0 a 4294967295. `length` es el tamaño del número que se podrá mostrar. Por ejemplo, si `length` es 4 solo se mostrarán los números hasta 9999, aunque los números más altos se guardarán.
`DECIMAL(length,dec)`	Número decimal en el cual `length` es el número de caracteres que se usarán para mostrar el número, incluyendo el punto de los decimales, los signos y exponentes, y dec es el número máximo de decimales permitidos. Por ejemplo, 12.34 tiene length de 5 y dec de 2.

Tipos de datos en MySQL	*Descripción (continuación)*
DATE	Valor de la fecha con año, mes y día. Muestra el valor como AAAA-MM-DD (por ejemplo, 2001-04-03).
TIME	Valor de la hora con hora, minutos y segundos. Muestra la hora como HH:MM:SS.
DATETIME	La fecha y hora guardadas juntas. Aparece como AAAA-MM-DD HH:MM:SS.
ENUM ("val1","val2"...)	Sólo pueden guardarse los valores en la lista. En la lista se puede incluir un máximo de 65535 valores.

MySQL permite muchos otros tipos de datos, pero se necesitan con menos frecuencia. Para una descripción de todos los tipos de datos disponibles, consulte la documentación sobre MySQL en: `www.mysql.com/doc/C/o/Column_types.html`.

Escríbalo

Insisto en mi usual necedad: Escríbalo. Probablemente usted pasó bastante tiempo tomando decisiones de diseño para su base de datos. En este punto, las decisiones están fijadas con firmeza en su mente. No cree que pueda olvidarlas. Sin embargo, suponga que de pronto surge una crisis y tiene que dejar de lado este proyecto durante dos meses. Tendría que analizar sus datos y tomar todas las decisiones de diseño nuevamente. Eso, claro, puede evitarlo si escribe sus decisiones ahora mismo.

Documente la organización de las tablas, los nombres de las columnas y todas las demás decisiones de diseño. Una buena idea de formato es hacerlo en un documento que describa cada tabla en el formato de tabla, con una fila para cada columna y una columna para cada decisión de diseño. Por ejemplo, sus columnas serían nombre de la columna, tipo de datos y descripción.

Un vistazo a los diseños de las bases de datos de los ejemplos

Esta sección contiene los diseños de las bases de datos de los dos ejemplos de aplicación con bases de datos para la Web.

Tablas de la base de datos de Cosas para vender

El diseño de la base de datos para la aplicación del Catálogo de mascotas incluye tres tablas: `Mascota`, `TipodeMascota` y `ColordelaMascota`. Las Tablas 3-2 hasta 3-4 muestran la organización de estas tablas. La definición de la tabla no es inalterable; MySQL le permite cambiar las tablas con mucha facilidad. Por ejemplo, si fija el tipo de datos para una variable en `CHAR(20)` y se da cuenta de que no es lo suficientemente larga, fácilmente podrá cambiar el tipo de dato.

El diseño de la base de datos es como sigue:

Nombre de la base de datos: CatalogodeMascotas

Tabla 3-2 Tabla 1 de la Base de datos: Mascota

Nombre de la variable	*Tipo*	*Descripción*
`Idmascota`	`INT(5)`	Número secuencial para la mascota (llave primaria)
`Nombremascota`	`CHAR(25)`	Nombre de la mascota
`Tipomascota`	`CHAR(15)`	Categoría de la mascota
`Descripcion mascota`	`VARCHAR(255)`	Descripción de la mascota
`precio`	`DECIMAL(9,2)`	Precio de la mascota
`pix`	`CHAR(15)`	Nombre de la ruta hacia el archivo gráfico que contiene la foto de la mascota.

Tabla 3-3 Tabla 2 de la Base de datos: TipodeMascota

Nombre de la variable	*Tipo*	*Descripción*
`Tipomascota`	`CHAR(15)`	Nombre de la categoría de la mascota (llave primaria)
`Descripcion mascota`	`VARCHAR(255)`	Descripción de la categoría

Tabla 3-4 Tabla 3 de la Base de datos: ColordelaMascota

Nombre de la variable	*Tipo*	*Descripción*
Nombremascota	CHAR(25)	Nombre de la mascota (llave primaria 1)
Colormascota	CHAR(15)	Nombre del color (llave primaria 2)

Tablas para la base de datos Sólo para miembros

El diseño de la base de datos para la aplicación de Sólo para miembros incluye dos tablas, llamadas Miembro y Entrada. Las Tablas 3-5 y 3-6 documentan la organización de estas tablas. La definición de las tablas no está fijada de modo inalterable; MySQL le permite cambiar las tablas con facilidad. Si fija el tipo de datos para una variable en CHAR(25) y se da cuenta de que no es lo suficientemente larga, es fácil cambiar el tipo de datos.

El diseño de la base de datos es el siguiente:

Nombre de la base de datos: DirectoriodeMiembros

Tabla 3-5 Tabla 1 de la Base de datos: Miembro

Nombre de la variable	*Tipo*	*Descripción*
Nombre entrada	VARCHAR(20)	Nombre para entrar especificado por el usuario (llave primaria)
Clave	CHAR(255)	Contraseña especificada por el usuario
Fecha creacion	DATE	Fecha en que el miembro se inscribió y registró su cuenta
Apellido	VARCHAR(50)	Apellido del miembro
Nombre	VARCHAR(40)	Nombre del miembro
Calle	VARCHAR(50)	Dirección física del miembro
Ciudad	VARCHAR(50)	Ciudad del miembro
Estado	CHAR(2)	Estado del miembro
Codigo postal	CHAR(10)	Código postal del miembro

Nombre de la variable	*Tipo*	*Descripción (Continuación)*
email	VARCHAR(50)	Dirección electrónica del miembro
Telefono	CHAR(15)	Número telefónico del miembro
Fax	CHAR(15)	Número de fax del miembro

Tabla 3-6 Tabla 2 de la Base de datos: Entrada

Nombre de la variable	*Tipo*	*Descripción*
Nombre entrada	CHAR(20)	Nombre para entrar especificado por el usuario (llave primaria 1)
Tiempo entrada	DATETIME	Fecha y hora de entrada (llave primaria 2)

Desarrollar la aplicación

Después de haber creado el plan enumerando las tareas que su aplicación debe realizar y de haber desarrollado el diseño de la base de datos, ya está listo para crear su aplicación. Primero debe construir la base de datos; luego escribirá los programas en PHP. Está a muy pocos momentos de tener trabajando una aplicación con base de datos para la Web. Bueno, quizá estoy exagerando un poco, pero ciertamente está progresando.

Construir la base de datos

Construir la base de datos significa convertir la base de datos diseñada en el papel a una base de datos que funcione. Construir la base de datos es algo independiente de los programas PHP que su aplicación usará para interactuar con la base de datos. Se puede tener acceso a la base de datos usando lenguajes de programación diferentes de PHP, por ejemplo Perl, C o Java. La base de datos guarda los datos de manera independiente.

Debe construir la base de datos antes de escribir los programas en PHP. Los programas en PHP se escriben para insertar y extraer los datos de la base de datos, así que no podrá desarrollarlos ni probarlos hasta que la base de datos esté disponible.

El diseño de la base de datos nombra y define las tablas que constituyen la base de datos. Para construir la base de datos, usted debe comunicarse con MySQL mediante el lenguaje SQL. Le dice a MySQL que cree la base de datos y que le agregue tablas. Le dice a MySQL cómo organizar las tablas de datos y qué formato usar para guardar los datos. Las instrucciones detalladas sobre cómo construir la base de datos se brindan en el Capítulo 4.

Escribir los programas

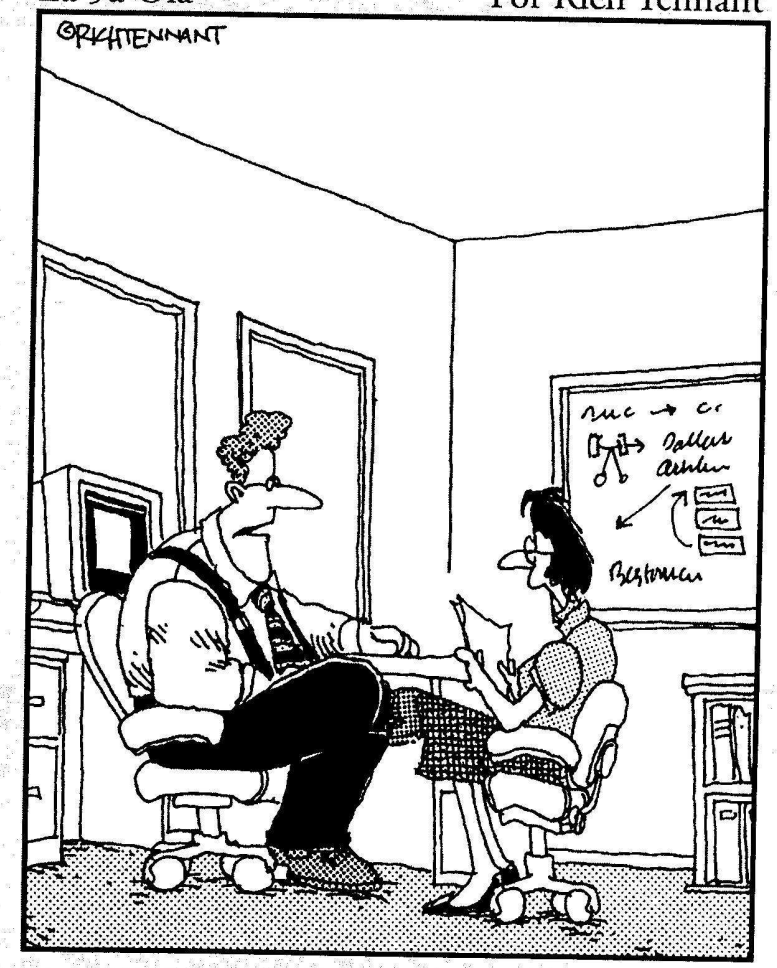
Sus programas realizan las tareas necesarias de su aplicación con base de datos para la Web. Crean lo que se muestra al usuario en la ventana del navegador. Hacen que su aplicación sea interactiva, pues aceptan y procesan la información digitada por el usuario en la ventana del navegador. Almacenan información en la base de datos y extraen información de la base de datos. La base de datos es inútil a menos que puede insertarle y extraerle datos.

El plan desarrollado (tal como indiqué en las secciones anteriores de este capítulo) esquematiza los programas que debe escribir. Generalmente cada tarea del plan requerirá de un programa. Si su plan dice que su aplicación mostrará un formulario, usted necesita un programa para mostrar el formulario. Si su plan dice que su aplicación guardará los datos de un formulario, necesita un programa que obtenga los datos del formulario y los ponga en la base de datos.

El lenguaje PHP fue desarrollado específicamente para escribir aplicaciones web interactivas. Incluye la funcionalidad necesaria para hacer que la escritura de los programas sea lo menos dolorosa posible. Tiene métodos que se incluyeron en el lenguaje específicamente para tomar datos de formularios. Tiene métodos para poner los datos en las bases de datos MySQL, y tiene métodos para extraer datos de una base de datos MySQL. En la Parte III de este libro explico detalladamente las instrucciones necesarias para escribir programas con PHP.

Parte II
La base de datos MySQL

La 5a Ola Por Rich Tennant



"Nuestra política de respuesta automatizada ante una pérdida de datos en toda la compañía es notificar a la gerencia, respaldar los datos existentes y vender el 90% de mis acciones de la compañía".

En esta parte . . .

Esta parte le brinda los detalles relacionados con el trabajo con una base de datos MySQL. Aprenderá cómo usar SQL para comunicarse con MySQL. Además, descubrirá cómo crear una base de datos, cómo cambiarla y cómo extraer e insertar datos en una base de datos.

Capítulo 4

Construcción de la base de datos

En este capítulo

- Usar SQL para hacer solicitudes a MySQL
- Crear una base de datos nueva
- Agregar información a una base de datos existente
- Consultar la información en una base de datos existente
- Eliminar información de una base de datos existente

Después de terminar el diseño de su base de datos (consulte el Capítulo 3 si aún no lo ha hecho), estará listo para convertirlo en una base de datos funcional. En este capítulo aprenderá cómo construir una base de datos con base en su diseño, y cómo mover datos hacia dentro y hacia fuera de ella.

El diseño le asigna un nombre a la base de datos y define las tablas que la conforman. Para poder construir la base de datos debe comunicarle a MySQL el nombre de la base de datos y la estructura de las tablas. Posteriormente deberá comunicarse con MySQL para agregar datos a la base de datos (o solicitarle información). El lenguaje usado para comunicarse con MySQL es SQL. En este capítulo, explico cómo crear consultas en SQL y cómo usarlas para construir bases de datos nuevas, así como para interactuar con bases de datos ya existentes.

Cómo comunicarse con MySQL

El servidor MySQL es el administrador de su base de datos:

- Crea bases de datos nuevas.
- Sabe dónde están guardadas las bases de datos.
- Guarda y recupera información guiado por las solicitudes (consultas) que recibe.

Para hacer una solicitud que MySQL pueda entender, se deben construir consultas en SQL y enviarlas al servidor MySQL. (Para una descripción más detallada del servidor MySQL, consulte el Capítulo 1). Las dos secciones siguientes puntualizan cómo hacerlo.

Construir consultas en SQL

SQL (Structured Query Language/Lenguaje de consulta estructurado) es el lenguaje informático usado para comunicarse con MySQL. SQL es prácticamente inglés; está hecho principalmente de palabras en inglés, colocadas juntas en cadenas de palabras muy parecidas a oraciones en inglés. En general (afortunadamente) no es necesario comprender ningún lenguaje técnico misterioso para escribir consultas SQL que funcionen.

La primera palabra de una consulta es su nombre, el cual es una palabra de acción (un verbo) que le dice a MySQL qué quiere hacer usted. Las consultas que discuto en este capítulo son `CREATE`, `DROP`, `ALTER`, `SHOW`, `INSERT`, `LOAD`, `SELECT`, `UPDATE` y `DELETE`. Este vocabulario básico basta para crear bases de datos en sitios web, e interactuar con ellas.

Al nombre de la consulta le siguen palabras y frases — algunas obligatorias y otras opcionales — que le dicen a MySQL cómo realizar la acción. Por ejemplo, siempre debe indicarle a MySQL qué debe crear, y siempre debe decirle en cuál tabla insertar datos o en cuál seleccionar datos.

La siguiente es una consulta SQL típica. Como puede ver, usa palabras en inglés:

```
SELECT apellido FROM Miembro
```

Esta consulta recupera todos los apellidos guardados en la tabla llamada `Miembro`. Por supuesto, consultas más complicadas (como la siguiente) se parecen menos al inglés:

```
SELECT apellido, nombre FROM Miembro WHERE estado="CA" AND
    ciudad="Fresno" ORDER BY apellido
```

Esta consulta recupera todos los apellidos y nombres de los miembros que viven en Fresno y los coloca en orden alfabético por el apellido. Esta consulta se parece menos al inglés pero todavía es bastante clara.

Los siguientes son algunos puntos generales que se deben recordar cuando se construye una consulta SQL, como se ilustró en el ejemplo anterior:

- **Mayúsculas:** En este libro, pongo las palabras del lenguaje SQL completamente en mayúsculas; los elementos de información variable (como los

nombres de las columnas) usualmente reciben etiquetas que están en su mayor parte o totalmente en minúsculas. Lo hice así para que le fuera más fácil leer los ejemplos, no porque MySQL necesite este formato. No importa si las palabras en SQL están en mayúscula o minúscula; para MySQL `select` es igual a `SELECT` y `from` es igual a `FROM`. Por otro lado, si su sistema operativo es Unix o Linux sí es importante si los nombres de las tablas, los nombres de las columnas y cualquier otra información variable está en mayúscula o minúscula. Cuando se usa Unix o Linux, para MySQL los nombres de las columnas deben coincidir con exactitud, de modo que la mayúscula o minúscula de los nombres de las columnas tiene que estar correcta: apellido no es igual a Apellido. No obstante, Windows no es tan quisquilloso como Unix y Linux; desde su perspectiva, apellido y Apellido son iguales.

- **Espacios:** Las palabras en SQL tienen que separarse por uno o más espacios. No importa cuántos espacios use; es lo mismo usar 20 espacios o solo 1. SQL tampoco pone ninguna atención al final de la línea. En SQL usted puede empezar una nueva línea en cualquier punto de la instrucción, o escribir toda la instrucción en una sola línea.
- **Comillas:** Observe que en la consulta anterior `CA` y `Fresno` están encerradas en comillas dobles ("). CA y Fresno son series de caracteres llamadas cadenas de texto o cadenas de caracteres. (Explico detalladamente las cadenas más adelante en este capítulo.) En el ejemplo, le está pidiendo a MySQL que compare las cadenas de texto en la consulta SQL con las cadenas de texto almacenadas en la base de datos. Las cadenas de texto se encierran entre comillas. Cuando compara números (enteros, por ejemplo) almacenados en columnas numéricas, no hace falta encerrar los números entre comillas. (En el Capítulo 3 explico los tipos de datos que pueden almacenarse en una base de datos MySQL).

Enviar consultas en SQL

Este libro es sobre PHP y MySQL como pareja. Por lo tanto, no describiré la multitud de formas en que pueden enviarse consultas SQL a MySQL, muchas de las cuales no tienen nada que ver con PHP. En su lugar, proporciono un programa PHP simple que usted puede usar para ejecutar consultas en SQL. (Para conocer los detalles sobre PHP y cómo escribir programas en PHP, consulte la Parte III de este libro).

El programa `mysql_send.php` tiene una función simple: ejecutar consultas y mostrar los resultados. Digite el programa en el directorio donde está desarrollando su aplicación web (o descárguelo desde mi sitio web en `janet.valade.com`), cambie la información en las líneas 9 — 19, y luego dirija su explorador hacia el programa. La Lista 4-1 muestra el programa.

Lista 4-1: Programa PHP para enviar consultas en SQL a MySQL

```
<!-- Program: mysql_send.php

     Desc:   PHP program that sends an SQL query to the

               MySQL server and displays the results.

-->

<html>

<head><title>SQL Query Sender</title></head>

<body>

<?php

$host="hostname";

$user=" mysqlaccountname ";

$password="mysqlpassword";

/* Section that executes query */

if(@$_GET['form'] == "yes")

{

  mysql_connect($host,$user,$password);
  mysql_select_db($_POST['database']);
  $query = stripSlashes($_POST['query']);
  $result = mysql_query($query);
  echo "Database Selected: <b>{$_POST['database']}</b><br>
        Query: <b>$query</b><h3>Results</h3><hr>";
  if($result == 0)
     echo "<b>Error ".mysql_errno().": ".mysql_error().
          "</b>";
  elseif (@mysql_num_rows($result) == 0)
     echo("<b>Query completed. No results returned.
          </b><br>");
  else
  {
   echo "<table border='1'>
          <thead>
           <tr>";
            for($i = 0;$i < mysql_num_fields($result);$i++)
            {
             echo "<th>".mysql_field_name($result,$i).
                  "</th>";
            }
```

```
    echo "  </tr>
          </thead>
          <tbody>";
          for ($i = 0; $i < mysql_num_rows($result); $i++)
          {
            echo "<tr>";
             $row = mysql_fetch_row($result);
             for($j = 0;$j<mysql_num_fields($result);$j++)
             {
               echo("<td>" . $row[$j] . "</td>");
             }
            echo "</tr>";
          }
    echo "</tbody>
        </table>";
  }  //end else
  echo "
   <hr><br>
   <form action=\"{$_SERVER['PHP_SELF']}\" method=\"POST\">
     <input type='hidden' name='query' value='$query'>
     <input type='hidden' name='database'
            value={$_POST['database']}>
     <input type='submit' name=\"queryButton\"
            value=\"New Query\">
     <input type='submit' name=\"queryButton\"
            value=\"Edit Query\">
   </form>";
  unset($form);
  exit();
}  // endif form=yes

/* Section that requests user input of query */
@$query=stripSlashes($_POST['query']);
if (@$_POST['queryButton'] != "Edit Query")
{
  $query = " ";
}
?>

<form action="<?php echo $_SERVER['PHP_SELF'] ?>?form=yes"
      method="POST">
 <table>
  <tr>
   <td align=right><b>Type in database name</b></td>
   <td><input type="text" name="database"
              value=<?php echo @$_POST['database'] ?> ></td>
  </tr>
  <tr>
   <td align="right" valign="top">
        <b>Type in SQL query</b></td>
   <td><textarea name="query" cols="60"
               rows="10"><?php echo $query ?></textarea>
   </td>
```

```
</tr>
  <tr>
   <td colspan="2" align="center"><input type="submit"
       value="Submit Query"></td>
  </tr>
 </table>
</form>
</body></html>
```

Debe cambiar las líneas 9, 10 y 11 del programa antes de poder usarlo. Estas líneas son

```
$host="hostname";
$user="mysqlaccountname";
$password="mysqlpassword";
```

Cambie `hostname` al nombre del PC donde MySQL está instalado: por ejemplo, `servidorbasedatos.miempresa.com`. Si la base de datos MySQL está instalada en el mismo PC que su sito web, puede usar `localhost` como el `hostname`.

Cambie `mysqlaccountname` y `mysqlpassword` al nombre de la cuenta y de la contraseña que le dio el administrador de MySQL para que tuviera acceso a su base de datos MySQL. Si instaló MySQL usted mismo, una cuenta nombrada `root` sin contraseña se instala automáticamente. A veces se instala una cuenta con nombre de cuenta y de contraseña en blanco. Puede usar la cuenta `root` o la cuenta en blanco, pero es mucho mejor si instala una cuenta para uso específico de su aplicación web. (Me ocupo detalladamente de las cuentas y las contraseñas MySQL en el Capítulo 5.)

Una cuenta llamada `root` sin ninguna contraseña no es segura. Debe asignarle una contraseña lo más pronto posible. Una cuenta con el nombre de cuenta y el de contraseña en blanco es todavía menos segura. Cualquiera podría tener acceso a su base de datos sin necesidad de saber el nombre de la cuenta ni la contraseña. Si existe, debe borrar esta cuenta (consulte el Capítulo 5).

Si su cuenta MySQL no requiere de una contraseña, no digite nada entre las comillas dobles, así:

```
$password="";
```

Después de digitar en `mysql_send.php` los nombres correctos de hostname, de la cuenta y de la contraseña, estos son los pasos generales que debe seguir para ejecutar una consulta SQL:

1. **Dirija su explorador a** `mysql_send.php`.

   Verá la página web mostrada en la Figura 4-1.

2. **Digite la consulta SQL en el recuadro de texto grande.**

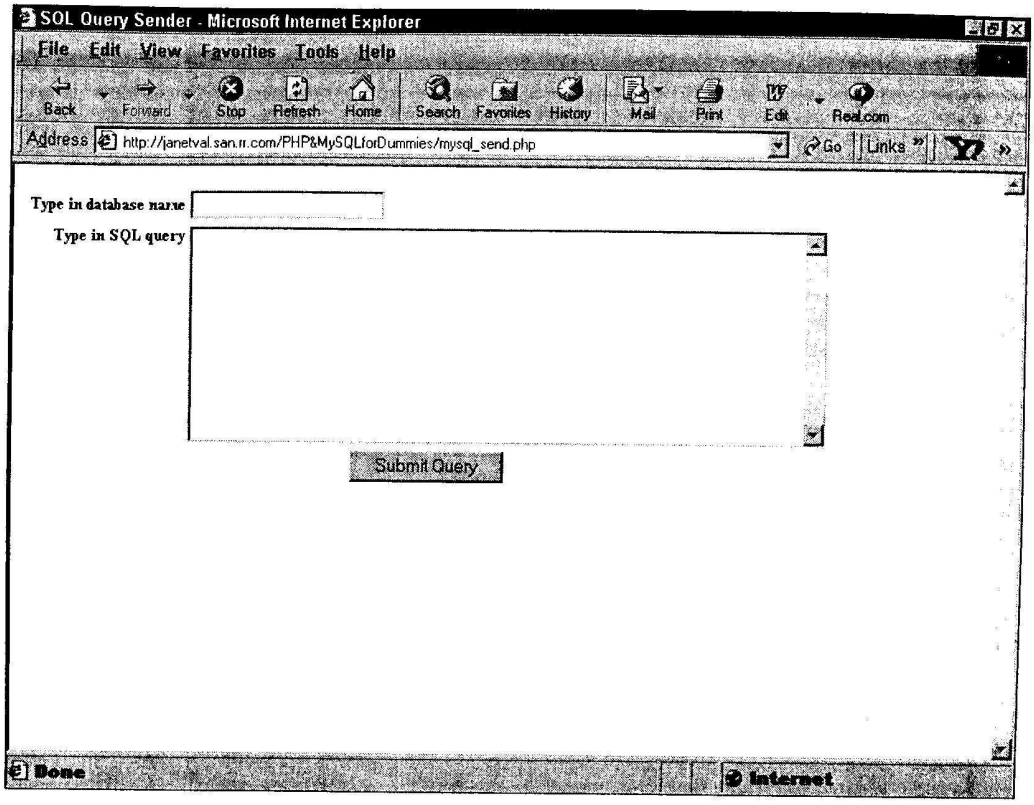


Figura 4-1: Página web de consulta SQL producida por `mysql_send.php`

3. **Si la consulta lo requiere, digite el nombre de la base de datos en el primer recuadro de texto.**

   Explico los detalles de cómo escribir consultas específicas en SQL en las siguientes secciones de este capítulo.

4. **Haga clic en el botón "Submit Query".**

   La consulta se ejecuta y aparece una página que muestra los resultados. Si su consulta tenía un error, aparecerá un mensaje de error.

Puede probar el programa `mysql_send.php` digitando esta consulta de prueba en el Paso 2 de los pasos anteriores:

```
SHOW DATABASE
```

En esta consulta no es necesario digitar el nombre de una base de datos, así que puede saltarse el Paso 3. Cuando hace clic en el botón Submit query en el Paso 4, aparecerá una lista de las bases de datos existentes. En la mayoría de los casos, verá una base de datos llamada `Test`, que automáticamente se instala cuando se instala MySQL. Además, probablemente verá una base de datos llamada `mysql`, que MySQL usa para almacenar la información que necesita, tal como los nombres de las cuentas, las contraseñas y los permisos. Incluso si no hay bases de datos existentes, su consulta SQL se ejecutará correctamente. Si ocurre un problema, aparecerá un mensaje de error. Los mensajes de error de MySQL son habitualmente muy útiles para encontrar el problema.

Una forma más rápida de enviar consultas en SQL al servidor MySQL

Cuando se instala MySQL, un programa simple basado en texto llamado mysql (o a veces monitor terminal o monitor) también se instala. Los programas que se comunican con servidores se llaman software cliente; como se comunica con el servidor MySQL, este programa es un cliente. Cuando digita las solicitudes en SQL en este cliente, la respuesta es devuelta al cliente y aparece en pantalla. El programa monitor puede enviar consultas a través de una red; no tiene que estar corriendo en la máquina donde está almacenada la base de datos.

Para enviar consultas en SQL a MySQL usando el cliente mysql, siga estos pasos:

1. **Localice el cliente mysql.**

   De manera predeterminada, el programa cliente `mysql` se instala en el subdirectorio bin, dentro del directorio donde se instaló MySQL. En Unix/Linux el directorio predeterminado es `/usr/local/mysql/bin o /usr/local/bin`. En Windows el directorio predeterminado es `c:\mysql\bin`. Sin embargo, el cliente puede haber sido instalado en un directorio diferente. O, si usted no es el administrador de MySQL, posiblemente no tenga acceso al cliente mysql. Si no sabe dónde está instalado MySQL o no puede correr el cliente, pídale al administrador que ponga el cliente en algún lugar donde lo pueda correr o que le dé una copia para poner en su PC.

2. **Inicie el cliente.**

   En Unix/Linux, digite la ruta y el archivo (por ejemplo `/usr/local/mysql/bin/mysql`). En Windows, abra una ventana de comando y luego digite el nombre de la ruta y del archivo (por ejemplo `c:\mysql\bin\mysql.exe`). Oprima Enter después de digitar el nombre de la ruta y del archivo, a menos que esté usando los parámetros mostrados en el Paso 3.

3. **Si está iniciando el cliente `mysql` para entrar en una base de datos en otra parte de la red, use los siguientes parámetros después del comando `mysql`:**

   `-h host`: `host` es el nombre de la máquina donde se ubica MySQL.

   `-u user`: `user` es el nombre de su cuenta MySQL.

   `-p`: Este parámetro le pide la contraseña de su cuenta MySQL.

   Por ejemplo, si está en el directorio donde se ubica el cliente `mysql`, el comando se vería así:

   ```
   mysql -h mysqlhost.mycompany.com -u root -p
   ```

   Oprima Enter después de digitar el comando.

4. **Digite su contraseña cuando le sea pedida.**

   El cliente `mysql` se inicia y usted verá algo parecido a esto:

   ```
   Welcome to the MySQL monitor. Commands end with ; or \g.
   Your MySQL connection id is 459 to server version: 4.0.13
   Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
   mysql>
   ```

(continuación)

(continúa)

5. **Seleccione la base de datos que desea usar.**

   En la línea de comando `mysql`, digite lo siguiente:

   ```
   use databasename
   ```

   Use el nombre de la base de datos que desea consultar.

6. **En la línea de comando `mysql`, digite su consulta SQL, seguida de un punto y coma (;), y luego presione la tecla Enter.**

   El cliente `mysql` continuará solicitándole información y no ejecutara la consulta hasta que usted digite un punto y coma. La respuesta a la consulta aparecerá en pantalla.

7. **Para salir del cliente mysql, digite `quit` en la línea de comando y oprima la tecla Enter.**

Construir una base de datos

Una base de datos tiene dos partes: una estructura que guarda los datos y los datos mismos. En las siguientes secciones explico cómo crear la estructura de la base de datos. Primero se crea una base de datos vacía, sin ninguna estructura, y luego se le agregan las tablas.

Las consultas SQL usadas para trabajar con la estructura de la base de datos son `CREATE`, `ALTER`, `DROP` y `SHOW`. Para usar estas consultas debe tener una cuenta MySQL con permiso para crear, alterar y botar bases de datos y tablas. Consulte en el Capítulo 5 mayor información sobre las cuentas MySQL.

Crear una base de datos nueva

Para crear una base de datos nueva y vacía, use la siguiente consulta SQL:

```
CREATE DATABASE nombrebasededatos
```

donde `nombrebasededatos` es el nombre que usted le da a la base de datos. Por ejemplo, estas dos consultas en SQL crean las bases de datos de los ejemplos usados en este libro:

```
CREATE DATABASE CatalogodeMascotas
CREATE DATABASE DirectoriodeMiembros
```

Algunas compañías de hospedaje de sitios web no le permitirán crear bases de datos nuevas. Le darán una base de datos para que la use con MySQL, y solo podrá crear tablas en esta base de datos. Puede tratar de solicitar otra base de datos, pero necesitará una muy buena razón. A MySQL y PHP no les importa

que todas sus tablas estén en una sola base de datos, en lugar de organizadas en varias bases de datos con nombres significativos. Solo para los humanos es más fácil seguirle la pista a los proyectos cuando están organizados.

Para comprobar por usted mismo que la base de datos fue verdaderamente creada, use esta consulta SQL:

```
SHOW DATABASES
```

Después de crear una base de datos vacía, puede agregarle tablas. (Más adelante en este capítulo describo cómo agregar tablas a una base de datos).

Cómo borrar una base de datos

Puede borrar cualquier base de datos con esta consulta SQL:

```
DROP DATABASE nombrebasededatos
```

Use DROP cuidadosamente, pues es irreversible. Después de que una base de datos se borra, se perdió para siempre. Y todos los datos que estaban en ella también se pierden.

Cómo agregar tablas a una base de datos

Puede agregar tablas a cualquier base de datos, ya sea a una base de datos nueva y vacía creada recientemente, o a una base de datos existente que ya tenga tablas y datos en ella. Se usa la consulta CREATE para agregar tablas a la base de datos.

En los ejemplos de diseño de bases de datos que introduje en el Capítulo 3, la base de datos catalogosdemascotas está diseñada con tres tablas: Mascota, TipodeMascota y ColordelaMascota. La base de datos DirectoriodeMiembros está diseñada con dos tablas: Miembro y Entrada. Como las tablas se crean en una base de datos, debe indicar el nombre de la base de datos donde desea crear la tabla. O sea, cuando use el formulario mostrado en la Figura 4-1, debe digitar el nombre de la base de datos en el campo superior. Si no lo hace verá el mensaje de error No Database Selected.

La consulta para agregar una tabla empieza con

```
CREATE TABLE nombredetabla
```

Luego sigue una lista de nombres de columnas con sus definiciones. La información de cada columna se separa de la información de la siguiente columna mediante una coma. La lista completa debe encerrarse entre paréntesis. Cada nombre de columna debe seguirse de su tipo de datos (explico los tipos de datos detalladamente en el Capítulo 3) y cualquier otra definición requerida. Estas son algunas definiciones que puede usar:

- NOT NULL: Esta columna debe tener un valor; no puede estar vacía.
- DEFAULT value: Si no se indica ningún otro valor, este valor se almacena en la columna cuando se crea la fila.
- AUTO_INCREMENT: Use esta definición para crear un número secuencial. Conforme se agregan filas, el valor de esta columna aumentará en un número entero desde la última fila que se agregó. Puede pasar por alto el número automático asignándole un valor específico a la columna.
- UNSIGNED: Use esta definición para indicar que los valores en este campo numérico nunca serán números negativos.

El último elemento en una consulta CREATE TABLE indica cuál columna o combinación de columnas es el identificador único para la fila, es decir, la clave primaria. Cada fila de una tabla debe tener un campo o una combinación de campos diferente para cada fila. Dos filas no pueden tener la misma clave primaria. Si intenta agregar una fila con la misma clave primaria que una fila ya existente en la tabla, obtendrá un mensaje de error y la fila no se agregará. El diseño de la base de datos identifica la clave primaria (tal como describo en el Capítulo 3). Debe especificar la clave primaria usando el siguiente formato:

```
CREATE TABLE Miembro (
  loginName     VARCHAR(20) NOT NULL PRIMARY KEY,
  createDate    DATE        NOT NULL);
PRIMARY KEY(nombre_columna)
```

El *nombrecolumna* se encierra entre paréntesis. Si está usando una combinación de columnas como clave primaria, incluya todos los nombres de columnas, separados por comas. Por ejemplo, podría designar la clave primaria para la tabla Entrada en la base de datos DirectoriodeMiembros usando la siguiente consulta:

```
PRIMARY KEY (Nombre_entrada,Tiempo_entrada)
```

La Lista 4-2 muestra la consulta CREATE TABLE usada para crear la tabla Miembro de la base de datos DirectoriodeMiembros. Si quisiera, podría digitar esa consulta en una sola línea. A MySQL no le importa cuántas líneas use. Sin embargo, el formato mostrado en la Lista 4-2 es más fácil de leer. Este formato amigable a los humanos también le ayuda a detectar errores de digitación.

Lista 4-2: Consulta SQL para crear una tabla

```
CREATE TABLE Miembro (
  Nombreentrada      VARCHAR(20) NOT NULL,
  Fechacreacion            DATE        NOT NULL,
  clave    CHAR(255)   NOT NULL,
  Apellido                 VARCHAR(50),
  Nombre             VARCHAR(40),
  calle                    VARCHAR(50),
  ciudad                   VARCHAR(50),
  estado                   CHAR(2),
  codigopostal       CHAR(10),
```

```
 email                          VARCHAR(50),
  telefono                      CHAR(15),
  fax                           CHAR(15),
PRIMARY KEY(loginName) )
```

Observe que la lista de los nombres de las columnas en la Lista 4-2 está encerrada entre paréntesis (uno en la primera línea y otro en la última línea) y que hay una coma después de la definición de cada columna.

Recuerde que, como indiqué en el Capítulo 3, no debe usar ninguna palabra reservada por MySQL en los nombres de las columnas. Si lo hace, MySQL le dará un mensaje de error parecido a este:

```
You have an error in your SQL syntax near 'order var(20))' at line 1
```

Observe que este mensaje muestra la definición de la columna que a SQL no le gustó y la línea donde encontró la definición ofensiva. Sin embargo, el mensaje no dice mucho sobre cuál es el problema. El `error in your SQL syntax` (error en la sintaxis de SQL) al cual se refiere es haber usado la palabra `order`, reservada por MySQL, como nombre de una columna.

Después de crear una tabla puede solicitar verla, revisar su estructura o eliminarla.

- **Para ver las tablas agregadas a una base de datos, use esta consulta SQL:**

  ```
  SHOW TABLES
  ```

- **También puede ver la estructura de una tabla con esta consulta:**

  ```
  SHOW COLUMNS FROM nombre_tabla
  ```

- **Puede eliminar cualquier tabla con esta consulta:**

  ```
  DROP TABLE nombre_tabla
  ```

Use `DROP` cuidadosamente, pues es irreversible. Después de borrar una tabla, se pierde para siempre. Y todos los datos que estaban en ella también se pierden.

Cómo cambiar la estructura de la base de datos

Su base de datos no está escrita en piedra. Puede cambiar el nombre de la tabla usando la consulta `ALTER`; también puede agregar, eliminar o renombrar una columna, o cambiar el tipo de datos y cualquier otro atributo de la columna.

El formato básico de esta consulta es `ALTER TABLE` nombretabla, seguido por los cambios específicos que está solicitando. La Tabla 4-1 muestra los cambios que es posible hacer.

Tabla 4-1 Cambios que se pueden hacer con la consulta ALTER

Cambio	*Descripción*
ADD *nombre_columna* definicion	Agrega una columna; definicion incluye el tipo de datos y definiciones opcionales
ALTER *nombre_columna* SET DEFAULT valor	Elimina el valor predeterminado de una columna
ALTER *nombre_columna* DROP DEFAULT	Elimina el valor predeterminado de una columna
CHANGE *definicion* nombre_columna *nuevo_nombre_columna*	Cambia la definición de una columna y renombra la columna; definicion incluye el tipo de datos y definiciones opcionales
DROP *nombre_columna*	Borra una columna, incluyendo los datos que contenga. Los datos no se pueden recuperar.
MODIFY *nombre_columna* definicion	Cambia la definición de una columna; definicion incluye el tipo de datos y definiciones opcionales.
RENAME *nuevo_nombre_columna*	Renombra una tabla

Cambiar una base de datos no es nada raro. Hay muchas razones por las cuales podría querer cambiar algo en su base de datos. Por ejemplo, suponga que definió la columna Apellido con VARCHAR(20) en la tabla Miembro de la base de datos DirectoriodeMiembros. En ese momento, 20 caracteres parecieron suficientes para un apellido. Pero acaba de recibir un memorando anunciando que el nuevo CEO es John Schwartzheimer-Losertman. ¡Ay! MySQL truncará su nombre dejando solo las primeras 20 letras, un nombre poco-menos-que-deseable para su nuevo jefe. Así que debe hacer más ancha su columna, y pronto. Envíe esta consulta para cambiar la columna en un segundo:

```
ALTER TABLE Member MODIFY Apellido VARCHAR(50)
```

Mover datos hacia dentro y hacia fuera de una base de datos

Una base de datos vacía es como un frasco de galletas vacío: algo muy poco atractivo. Y sondear una base de datos vacía no es más interesante o fructífero que sondear un frasco de galletas vacío. Una base de datos solo es útil en relación con la información que guarda.

Una base de datos debe ser capaz de recibir información para almacenar y de entregar información cuando se le pide. Por ejemplo, la base de datos `DirectoriodeMiembros` debe ser capaz de recibir la información de los miembros, y también debe ser capaz de entregar la información que guarda cuando se le solicita. Por ejemplo, si desea averiguar la dirección de un miembro en particular, la base de datos debe entregarle esta información apenas se la pida.

Su base de datos MySQL responde a cuatro tipos de solicitudes:

- **Agregar información:** Agregar una fila a una tabla.
- **Actualizar información:** Cambiar información en una fila existente. Esto incluye agregar datos a un campo en blanco dentro de una fila ya existente.
- **Recuperar información:** Buscar datos. Este tipo de solicitud no elimina los datos de la base de datos.
- **Eliminar información:** Borrar datos de la base de datos.

A veces sus preguntas requerirán información de más de una tabla. Por ejemplo, la pregunta "¿Cuánto cuesta un dragón verde?" requiere información de la tabla `Mascota` y de la tabla `Color`. Fácilmente puede hacer esta pregunta en una sola consulta `SELECT`, combinando las tablas.

En las secciones siguientes comento cómo recibir y entregar información, y también cómo combinar tablas.

Agregar información

Todas las bases de datos necesitan datos. Una posibilidad es que usted quiera agregar datos a su base de datos para que los usuarios puedan verla, por ejemplo en el Catálogo de Mascotas que introduje en el Capítulo 3. Otra posibilidad es crear una base de datos vacía para que los usuarios pongan datos en ella, haciendo que la información sólo esté disponible para usted, como en el ejemplo del Directorio de Miembros. En cualquiera de estos escenarios hay que agregar datos a la base de datos.

Si sus datos están todavía en papel usted puede digitarlos directamente en una base de datos MySQL, una fila a la vez, usando consultas SQL. Sin embargo, si tiene muchos datos, este proceso podría hacerse muy tedioso y tomaría mucho tiempo digitar la información. Suponga que debe agregar a su base de datos información sobre 1,000 productos. Asumiendo que digite a la velocidad de la luz y que pueda digitar una fila por minuto, serían 16 horas de digitar rápidamente — o de editar rápidamente, en todo caso. Se puede hacer, pero no es divertido. Por otra parte, suponga que tiene que incluir a los 5,000 miembros de una organización en una base de datos y que toma cinco minutos digitar la información de cada uno. Esto tomaría más de 400 horas de digitación, ¡quién tiempo para eso!

Si tiene que digitar grandes cantidades de datos, considere algunas alternativas. A veces es una buena opción escanear los datos. O quizá necesite suplicar por ayuda, o pedirla prestada o contratarla. En muchos casos será más rápido digitar los datos en un gran archivo de texto que digitar fila por fila en consultas individuales de SQL.

La consulta LOAD de SQL puede leer datos de un archivo grande de texto (o incluso de un archivo de texto pequeño). O sea, si sus datos ya están en un archivo de PC, puede trabajar con ese archivo; no tiene que digitar todos los datos de nuevo. Incluso si los datos están en un archivo cuyo formato no es de texto (por ejemplo en un archivo de Excel, Access u Oracle), usualmente es posible convertir el archivo a un gran archivo de texto que luego puede ser leído por su base de datos MySQL. Si los datos todavía no están en un archivo de PC y hay mucha información, probablemente será más rápido digitar los datos en el PC en un gran archivo de texto y transferirlo a MySQL como segundo paso.

La mayoría de los archivos de texto pueden ser leídos por MySQL, pero algunos formatos son más fáciles de leer que otros. Si está planeando digitar los datos en un gran archivo de texto, lea la sección "Cómo agregar una enorme cantidad de datos" para averiguar cuál es el mejor formato para su archivo de texto. Por supuesto que si los datos ya están en su PC tendrá que trabajar con el archivo tal como está.

Cómo agregar una fila a la vez

La consulta INSERT se usa para agregar una fila a una base de datos. Esta consulta le dice a MySQL en cuál tabla agregar la fila y cuáles son los valores para los campos en la fila. La forma general de la consulta es

```
INSERT INTO nombre_tabla (nombre_columna, nombre_columna,....., nombre_columna)
    VALUES (valor, valor,.....,valor)
```

Las siguientes reglas se aplican a la consulta INSERT:

- **Los valores deben enumerarse en el mismo orden en que se enumeran los nombres de las columnas.** El primer valor en la lista de valores se inserta en la columna que se nombra primero en la lista de columnas; el segundo valor en la lista de valores se inserta en la columna que se nombra de segunda en la lista de columnas y así sucesivamente.
- **Se permite una lista de columnas parcial.** No tiene que enumerar todas las columnas. Las columnas que no se enumeran recibirán su valor predeterminado o, si no tienen un valor predeterminado, se dejarán en blanco.
- **No es obligatorio usar una lista de columnas.** Si está digitando valores para todas las columnas, no necesita la lista de columnas del todo. Si no se enumeran columnas, MySQL buscará valores para todas las columnas, en el orden en el que aparecen en la tabla.
- **La lista de columnas y la lista de valores debe ser de la misma longitud.** Si la lista de columnas es más larga o más corta que la lista de valores, recibirá un mensaje de error como este: `Column count doesn't match value count.`

La siguiente consulta INSERT agrega una fila a la tabla Miembro:

```
INSERT INTO Miembro (Nombreentrada,Fechacreacion, clave,Apellido,
                     calle,ciudad,estado,codigo_postal,email,telefone ,fax)
       VALUES ("bigguy","2001-Dec-2","secret","Smith",
               "1234 Happy St","Las Vegas","NV","88888",
               "gsmith@GSmithCompany.com","(555) 555-5555","")
```

Observe que Nombre no aparece en la lista de los nombres de columnas. No se digita ningún valor en el campo Nombre. Si Nombre se define como NOT NULL, MySQL no lo permitiría, pero como Nombre no está definido como NOT NULL, no hay problema. Además, si la definición para Nombre incluye un valor predeterminado, se le asignaría ese valor predeterminado, pero como no es así, el campo se deja vacío. Observe que el valor almacenado para fax es una cadena vacía; a MySQL no le molestan las cadenas vacías.

Para ver los datos que digitó y asegurarse de que los digitó correctamente, use una consulta SQL que recupere los datos de la base de datos. Describo estas consultas detalladamente en "Cómo recuperar información" más adelante en este capítulo. En resumen, la siguiente consulta recupera todos los datos en la tabla Miembro:

```
SELECT * FROM Miembro
```

Cómo agregar una enorme cantidad de datos

Si tiene que agregar una gran cantidad de datos que ya están en un archivo de PC, puede transferir los datos del archivo existente a su base de datos MySQL. La consulta SQL que lee los datos de un archivo de texto es LOAD. La consulta LOAD requiere que especifique la base de datos.

Como los datos en una base de datos están organizados en filas y columnas, el archivo de texto leído debe indicar dónde empiezan y terminan los datos para cada columna y dónde termina la fila. Para indicar columnas, un caracter específico separa los datos de cada columna. Como valor predeterminado, MySQL busca un caracter de tabulación para separar los campos. Sin embargo, si la tabulación no funciona para su archivo de datos, puede escoger un caracter diferente para separar los campos y decirle a MySQL, en la consulta, que un caracter diferente del de tabulación separa los campos. También como valor predeterminado, se espera que el final de una línea sea el final de una fila, aunque puede escoger un caracter específico para indicar el final de una línea si es necesario. Un archivo de datos para la tabla Mascota se ve así:

```
Unicornio<TAB>caballo<TAB>cuerno en espiral<Tab>5000.00<Tab>/pix/unicorn.jpg
Pegaso<TAB>caballo<TAB>alado<Tab>8000.00<Tab>/pix/pegasus.jpg
Leon<TAB>gato<TAB>grande;melena en
cuello<Tab>2000.00<Tab>/pix/lion.jpg
```

Un archivo de datos con tabuladores entre los campos es un archivo delimitado por tabuladores. Otro formato común es un archivo delimitado por comas, donde las comas separan los campos. Si sus datos están en otro formato de archivo, debe convertirlo a un archivo delimitado.

Para convertir datos de otro formato de archivo a un archivo delimitado, consulte el manual del software correspondiente o hable con un experto local que entienda el formato actual de los datos. Muchos programas, como Excel, Access u Oracle, le permiten ordenar los datos en un archivo delimitado. En un archivo de texto, tal vez pueda convertirlo al formato delimitado usando la función de buscar y reemplazar de un editor o procesador de palabras. Si tiene un archivo verdaderamente problemático, quizá necesite buscar la ayuda de un experto o de un programador.

La forma básica de una consulta LOAD es

```
LOAD DATA INFILE "nombrearchivodatos" INTO TABLE nombre_tabla
```

Esta forma básica puede ir seguida de frases opcionales si desea cambiar algunos de los delimitadores predeterminados. Las opciones son

```
FIELDS TERMINATED BY 'caracter'
FIELDS ENCLOSED BY 'caracter'
LINES TERMINATED BY 'caracter'
```

Suponga que tiene el archivo de datos para la tabla Mascota, mostrada anteriormente en esta sección, pero los campos están separados por comas y no por el tabulador. El nombre del archivo de datos es mascotas.dat, y está localizado en el mismo directorio que la base de datos. La consulta SQL para leer los datos en la tabla es

```
LOAD DATA INFILE "mascotas.dat" INTO TABLE Mascota
        FIELDS TERMINATED BY ','
```

Para poder usar la consulta LOAD DATA INFILE, la cuenta MySQL debe tener el privilegio FILE en el anfitrión del servidor. En el Capítulo 5 comento los privilegios de las cuentas MySQL.

Para ver los datos que cargó, y asegurarse de que están correctos, use una consulta SQL que recupere los datos de la base de datos. En la sección siguiente describo detalladamente estos tipos de consultas. En resumen, use la siguiente consulta para ver y revisar todos los datos de la tabla:

```
SELECT * FROM Mascota
```

Cómo recuperar información

El único propósito de guardar información es tenerla a disposición cuando se necesite. Una base de datos vive para responder preguntas. ¿Cuáles mascotas están a la venta? ¿Quiénes son los miembros? ¿Cuántos miembros viven en Arkansas? ¿Hay un lagarto a la venta? ¿Cuánto cuesta un dragón? ¿Cuál es el número telefónico de Goliath Smith? Y así sucesivamente. Se usa la consulta SELECT para hacerle preguntas a la base de datos.

La siguiente consulta INSERT agrega una fila a la tabla Miembro:

```
INSERT INTO Miembro (Nombreentrada,Fechacreacion, clave,Apellido,
                     calle,ciudad,estado,codigo_postal,email,telefone ,fax)
       VALUES ("bigguy","2001-Dec-2","secret","Smith",
               "1234 Happy St","Las Vegas","NV","88888",
               "gsmith@GSmithCompany.com","(555) 555-5555","")
```

Observe que Nombre no aparece en la lista de los nombres de columnas. No se digita ningún valor en el campo Nombre. Si Nombre se define como NOT NULL, MySQL no lo permitiría, pero como Nombre no está definido como NOT NULL, no hay problema. Además, si la definición para Nombre incluye un valor predeterminado, se le asignaría ese valor predeterminado, pero como no es así, el campo se deja vacío. Observe que el valor almacenado para fax es una cadena vacía; a MySQL no le molestan las cadenas vacías.

Para ver los datos que digitó y asegurarse de que los digitó correctamente, use una consulta SQL que recupere los datos de la base de datos. Describo estas consultas detalladamente en "Cómo recuperar información" más adelante en este capítulo. En resumen, la siguiente consulta recupera todos los datos en la tabla Miembro:

```
SELECT * FROM Miembro
```

Cómo agregar una enorme cantidad de datos

Si tiene que agregar una gran cantidad de datos que ya están en un archivo de PC, puede transferir los datos del archivo existente a su base de datos MySQL. La consulta SQL que lee los datos de un archivo de texto es LOAD. La consulta LOAD requiere que especifique la base de datos.

Como los datos en una base de datos están organizados en filas y columnas, el archivo de texto leído debe indicar dónde empiezan y terminan los datos para cada columna y dónde termina la fila. Para indicar columnas, un caracter específico separa los datos de cada columna. Como valor predeterminado, MySQL busca un caracter de tabulación para separar los campos. Sin embargo, si la tabulación no funciona para su archivo de datos, puede escoger un caracter diferente para separar los campos y decirle a MySQL, en la consulta, que un caracter diferente del de tabulación separa los campos. También como valor predeterminado, se espera que el final de una línea sea el final de una fila, aunque puede escoger un caracter específico para indicar el final de una línea si es necesario. Un archivo de datos para la tabla Mascota se ve así:

```
Unicornio<TAB>caballo<TAB>cuerno en espiral<Tab>5000.00<Tab>/pix/unicorn.jpg
Pegaso<TAB>caballo<TAB>alado<Tab>8000.00<Tab>/pix/pegasus.jpg
Leon<TAB>gato<TAB>grande;melena en
cuello<Tab>2000.00<Tab>/pix/lion.jpg
```

Un archivo de datos con tabuladores entre los campos es un archivo delimitado por tabuladores. Otro formato común es un archivo delimitado por comas, donde las comas separan los campos. Si sus datos están en otro formato de archivo, debe convertirlo a un archivo delimitado.

Para convertir datos de otro formato de archivo a un archivo delimitado, consulte el manual del software correspondiente o hable con un experto local que entienda el formato actual de los datos. Muchos programas, como Excel, Access u Oracle, le permiten ordenar los datos en un archivo delimitado. En un archivo de texto, tal vez pueda convertirlo al formato delimitado usando la función de buscar y reemplazar de un editor o procesador de palabras. Si tiene un archivo verdaderamente problemático, quizá necesite buscar la ayuda de un experto o de un programador.

La forma básica de una consulta `LOAD` es

```
LOAD DATA INFILE "nombrearchivodatos" INTO TABLE nombre_tabla
```

Esta forma básica puede ir seguida de frases opcionales si desea cambiar algunos de los delimitadores predeterminados. Las opciones son

```
FIELDS TERMINATED BY 'caracter'
FIELDS ENCLOSED BY 'caracter'
LINES TERMINATED BY 'caracter'
```

Suponga que tiene el archivo de datos para la tabla `Mascota`, mostrada anteriormente en esta sección, pero los campos están separados por comas y no por el tabulador. El nombre del archivo de datos es `mascotas.dat`, y está localizado en el mismo directorio que la base de datos. La consulta SQL para leer los datos en la tabla es

```
LOAD DATA INFILE "mascotas.dat" INTO TABLE Mascota
        FIELDS TERMINATED BY ','
```

Para poder usar la consulta `LOAD DATA INFILE`, la cuenta MySQL debe tener el privilegio `FILE` en el anfitrión del servidor. En el Capítulo 5 comento los privilegios de las cuentas MySQL.

Para ver los datos que cargó, y asegurarse de que están correctos, use una consulta SQL que recupere los datos de la base de datos. En la sección siguiente describo detalladamente estos tipos de consultas. En resumen, use la siguiente consulta para ver y revisar todos los datos de la tabla:

```
SELECT * FROM Mascota
```

Cómo recuperar información

El único propósito de guardar información es tenerla a disposición cuando se necesite. Una base de datos vive para responder preguntas. ¿Cuáles mascotas están a la venta? ¿Quiénes son los miembros? ¿Cuántos miembros viven en Arkansas? ¿Hay un lagarto a la venta? ¿Cuánto cuesta un dragón? ¿Cuál es el número telefónico de Goliath Smith? Y así sucesivamente. Se usa la consulta `SELECT` para hacerle preguntas a la base de datos.

La consulta SELECT más simple y básica es

```
SELECT * FROM nombre_tabla
```

Esta consulta recupera toda la información de la tabla. El asterisco (*) es un comodín que significa todas las columnas.

La consulta SELECT puede ser mucho más selectiva. Las palabras y frases en SQL en la consulta SELECT pueden señalar exactamente la información necesaria para responder su pregunta. Puede especificar cuál información desea, cómo desea que se organice y cuál es la fuente de información:

- **Puede solicitar únicamente la información (las columnas) que necesita para responder su pregunta.** Por ejemplo, puede solicitar solo el nombre y los apellidos para crear una lista de miembros.
- **Puede solicitar la información en algún orden en particular.** Por ejemplo, puede solicitar que la información se clasifique en orden alfabético.
- **Puede solicitar información de objetos seleccionados (las filas) en su tabla.** (En el Capítulo 3 explico qué son los objetos en las bases de datos). Por ejemplo, puede solicitar el nombre y el apellido solo de aquellos miembros cuyas direcciones estén en Florida.

Cómo recuperar información específica

Para recuperar información específica, enumere las columnas que contienen la información que desea. Por ejemplo:

```
SELECT nombre_columna, nombre_columna, nombre_columna,...
          FROM nombre_tabla
```

Esta consulta recupera los valores de todas las filas para las columnas indicadas. Por ejemplo, la siguiente consulta recupera todos los nombres y apellidos almacenados en la tabla Miembro:

```
SELECT Apellido,Nombre FROM Miembro
```

Puede realizar operaciones matemáticas con las columnas cuando las selecciona. Por ejemplo, puede usar la siguiente consulta SELECT para sumar dos columnas:

```
SELECT col1+col2 FROM nombre_tabla
```

O puede usar la siguiente consulta:

```
SELECT precio,precio*1.08 FROM Mascota
```

El resultado es el precio y el precio con el impuesto de ventas del 8 por ciento sumado. Puede cambiar el nombre de una columna cuando la selecciona, así:

```
SELECT precio,precio*1.08 AS precioConImpuesto FROM Mascota
```

La cláusula AS le dice a MySQL que le dé el nombre precioConImpuesto a la segunda columna recuperada. Así, la consulta recupera dos columnas de datos: precio y precioConImpuesto.

En algunos casos usted no deseará ver los valores en una columna, pero sí saber algo sobre la columna. Por ejemplo, tal vez quiere saber el valor más bajo en la columna o el valor más alto en la columna. La Tabla 4-2 enumera parte de la información disponible sobre las columnas.

Tabla 4-2 Información que se puede seleccionar

Formato SQL	*Descripción de la información*
AVG (*nombrecolumna*)	Indica el promedio de todos los valores en nombrecolumna
COUNT (*nombrecolumna*)	Indica el número de filas en las cuales nombrecolumna no está en blanco.
MAX (*nombrecolumna*)	Indica el valor más alto en nombrecolumna
MIN (*nombrecolumna*)	Indica el valor más bajo en nombrecolumna
SUM (*nombrecolumna*)	Indica la suma de todos los valores en nombrecolumna

Por ejemplo, la consulta para averiguar el precio más alto en la tabla Mascota es

```
SELECT MAX(precio) FROM Mascota
```

Las palabras de SQL como MAX() y SUM() son funciones. SQL ofrece muchas otras funciones además de las que aparecen en la Tabla 4-2. Algunas funciones, como las de la Tabla 4-2, brindan información sobre una columna. Otras funciones cambian cada valor seleccionado. Por ejemplo, SQRT() indica la raíz cuadrada de cada valor en la columna, y DAYNAME() indica el nombre del día de la semana para cada valor en la columna fecha, en lugar de la fecha verdadera almacenada en la columna. Hay más de 100 funciones que pueden usarse en las consultas SELECT. Para conocer las descripciones de todas las funciones, vea la documentación de MySQL en www.mysql.com/documentation.

Cómo recuperar datos en un orden específico

Tal vez quiera recuperar los datos en algún orden en particular. Por ejemplo, en la tabla Miembro, tal vez quiera que los miembros sean organizados en orden alfabético por apellido. O, en la tabla Mascota, tal vez quiera que las mascotas sean agrupadas por tipo de mascota.

En la consulta SELECT, ORDER BY y GROUP BY afectan el orden en que los datos se presentan:

- ORDER BY: Para ordenar información, use la frase

  ORDER BY *nombre_columna*

  Los datos son organizados por nombrecolumna en orden ascendente. Por ejemplo, si el nombrecolumna es Apellido, los datos se presentarán en orden alfabético ordenados por el apellido.

  Puede organizarlos en orden descendente agregando la palabra DESC antes del nombre de la columna. Por ejemplo:

  ```
  SELECT * FROM Miembro ORDER BY DESC Apellido
  ```

- GROUP BY: Para agrupar información, use la siguiente frase:

  GROUP BY *nombrecolumna*

  Las filas que tienen el mismo valor de nombre_columna se agrupan juntas. Por ejemplo, use esta consulta para agrupar las filas que tienen un mismo valor de TipodeMascota:

  ```
  SELECT * FROM Mascota GROUP BY TipodeMascota
  ```

Puede usar GROUP BY y ORDER BY en una misma consulta.

Cómo recuperar datos de una fuente específica

Lo más usual es que no quiera toda la información de una tabla. Solo quiere información de objetos específicos de una base de datos: o sea, las filas. Hay tres palabras SQL usadas habitualmente para especificar la fuente de la información:

- WHERE: Le permite solicitar información de objetos de una base de datos con ciertas características. Por ejemplo, puede solicitar los nombres de los miembros que viven en California, o puede pedir sólo las mascotas que sean gatos.
- LIMIT: Le permite limitar el número de filas de las cuales se extraerá la información. Por ejemplo, puede solicitar toda la información de las primeras tres filas en la tabla.
- DISTINCT: Le permite solicitar información de solo una fila de filas idénticas. Por ejemplo, en la tabla Entrada puede solicitar el Nombre_entrada, pero especificar no duplicar nombres, delimitando así la respuesta a un registro para cada miembro. Esto respondería a la pregunta "¿Ha entrado el miembro alguna vez?", en lugar de la pregunta "¿Cuántas veces ha entrado el miembro?"

La cláusula WHERE de la consulta SELECT le permite hacer selecciones bastantes complicadas. Por ejemplo, suponga que su jefe le pide una lista de todos los miembros cuyos apellidos empiezan con B, que viven en Santa Bárbara y que tienen un 8 en su número telefónico o de fax. Estoy seguro de que hay muchos usos para ese tipo de lista. Puede complacer a su jefe y obtener esta lista con la consulta SELECT, usando la cláusula WHERE.

El formato básico de la cláusula WHERE es

```
WHERE expresion AND|OR expresion AND|OR expresion ...
```

expresion especifica un valor que debe compararse con los valores guardados en la base de datos. Solo se seleccionan las filas que contengan información que concuerde con la expresión. Puede usar tantas expresiones como necesite, cada una separada por AND u OR. Cuando usa AND, las dos expresiones conectadas por AND (es decir, tanto la expresión que está antes de AND como la expresión que está después de AND) deben concordar para seleccionar la fila. Cuando se usa OR, sólo una de las expresiones conectadas por OR debe concordar para seleccionar esa fila.

Algunas expresiones comunes se muestran en la Tabla 4-3.

Tabla 4-3 Expresiones de la cláusula WHERE

Expresión	*Ejemplo*	*Resultado*
columna = valor	codigopostal= "12345"	Sólo selecciona las filas donde 12345 esté almacenado en la columna llamada código_postal
columna > valor	codigopostal > "50000"	Sólo selecciona las filas donde el código postal es 50001 o mayor
columna >= valor	codigopostal >= "50000"	Sólo selecciona las filas donde el código postal es 50000 o mayor
columna < valor	codigopostal < "50000"	Sólo selecciona las filas donde el código postal es 49999 o menor
columna <= valor	codigopostal <= "50000"	Sólo selecciona las filas donde el código postal es 50000 o menor
columna BETWEEN valor1 AND valor2	codigopostal BETWEEN "20000" AND "30000"	Sólo selecciona las filas donde el código postal es mayor a 19999 pero menor a 30001
columna IN (valor1, valor2,...)	codigopostal IN("90001", "30044")	Sólo selecciona las filas donde el código postal es 90001 o 30044

Expresión	*Ejemplo*	*Resultado (continuación)*
`columna NOT IN (valor1,valor2,...)`	`codigopostal NOT IN ("90001", "30044")`	Sólo selecciona las filas donde el código postal es cualquier código postal con excepción de 90001 o 30044
`columna LIKE valor` valor puede contener el comodín % (que concuerda con cualquier cadena) y _ (que concuerda con cualquier caracter)	`codigopostal LIKE "9%"`	Selecciona todas las filas donde el código postal empieza con 9
`columna NOT LIKE valor` – valor puede contener el comodín % (que concuerda con cualquier cadena) y _ (que concuerda con cualquier caracter)	`zip NOT LIKE "9%"`	Selecciona las filas donde el código postal no empieza con 9

Puede combinar cualesquiera de las expresiones de la Tabla 4-3 con AND y OR. En algunos casos deberá usar paréntesis para aclarar los criterios de selección. Por ejemplo, puede usar la siguiente consulta para responder la urgente necesidad de su jefe por encontrar todas las personas en el Directorio de Miembros cuyos nombres empiecen con B, que vivan en Santa Bárbara y que tengan un 8 en su número telefónico o de fax:

```
SELECT Apellido,Nombre FROM Miembro
       WHERE Apellido LIKE "B%"
         AND ciudad = "Santa Barbara"
         AND (telefono  LIKE "%8%" OR fax LIKE "%8%")
```

Observe los paréntesis en la última línea. Sin ellos no obtendrá los resultados que le pidió su jefe. Sin los paréntesis, cada conector se procesaría en orden del primero al último, lo que daría como resultado una lista que incluiría todos los miembros cuyos nombres empiezan con B y que viven en Santa Bárbara y cuyos números telefónicos tienen un 8 en ellos, y todos los miembros cuyo número de fax tiene un 8 en ellos independientemente de si viven o no en Santa Bárbara e independientemente de si su nombre empieza o no con B. Cuando se procesara el último `OR`, se seleccionarían los miembros cuyas características concordaran con la expresión antes de `OR` o la expresión después de `OR`. La expresión antes de `OR` se conecta con las experiencias previas por los `AND` previos y por ello no es independiente, pero la expresión después de `OR` sí es independiente, lo cual resultaría en la selección de todos los miembros con un 8 en su número de fax.

LIMIT especifica cuántas filas se pueden recuperar. La forma para LIMIT es

```
LIMIT numeroinicio,numerofilas
```

La primera fila que desea recuperar es numeroinicio, y el número de filas que desea recuperar es numerofilas. Si numeroinicio no se especifica, se asume que es 1. Para seleccionar sólo los tres primeros miembros que viven en Texas, use esta consulta:

```
SELECT * FROM Miembro WHERE estado="TX" LIMIT 3
```

Algunas consultas SELECT encontrarán registros idénticos, pero en este ejemplo usted sólo quiere ver uno — no todos — de los registros idénticos. Para evitar que la consulta recupere todos los registros idénticos, agregue la palabra DISTINCT inmediatamente después de SELECT.

Cómo combinar información de tablas

En las secciones anteriores de este capítulo, asumí que toda la información estaba en una sola tabla. Sin embargo, usted podría querer combinar información de tablas diferentes. Fácilmente puede hacerlo en una sola consulta.

Se pueden usar dos palabras en una consulta SELECT para combinar información de dos o más tablas:

- UNION: Se recuperan filas de una o más tablas y se almacenan juntas, una después de la otra, en un solo resultado. Por ejemplo, si su consulta seleccionó 6 filas de una tabla y 5 filas de otra tabla, el resultado contendrá 11 filas.
- JOIN: Las tablas se combinan una junto a la otra y la información se recupera de ambas tablas.

UNION

Se usa UNION para combinar los resultados de dos o más consultas de selección. Los resultados de cada consulta se añaden al conjunto de resultados seguidos de los resultados de la consulta anterior. El formato de la consulta UNION es el siguiente:

```
SELECT consulta UNION ALL SELECT consulta ...
```

Puede combinar tantas consultas SELECT como necesite. Una consulta SELECT puede incluir cualquier formato SELECT válido, incluyendo cláusulas WHERE, cláusulas LIMIT, etc. Las reglas para las consultas son

- Todas las consultas SELECT deben seleccionar el mismo número de columnas.
- Las columnas seleccionadas en las consultas deben contener el mismo tipo de datos.

Expresión	*Ejemplo*	*Resultado (continuación)*
`columna NOT IN (valor1,valor2,...)`	`codigopostal NOT IN ("90001", "30044")`	Sólo selecciona las filas donde el código postal es cualquier código postal con excepción de 90001 o 30044
`columna LIKE valor` valor puede contener el comodín % (que concuerda con cualquier cadena) y _ (que concuerda con cualquier caracter)	`codigopostal LIKE "9%"`	Selecciona todas las filas donde el código postal empieza con 9
`columna NOT LIKE valor` – valor puede contener el comodín % (que concuerda con cualquier cadena) y _ (que concuerda con cualquier caracter)	`zip NOT LIKE "9%"`	Selecciona las filas donde el código postal no empieza con 9

Puede combinar cualesquiera de las expresiones de la Tabla 4-3 con AND y OR. En algunos casos deberá usar paréntesis para aclarar los criterios de selección. Por ejemplo, puede usar la siguiente consulta para responder la urgente necesidad de su jefe por encontrar todas las personas en el Directorio de Miembros cuyos nombres empiecen con B, que vivan en Santa Bárbara y que tengan un 8 en su número telefónico o de fax:

```
SELECT Apellido,Nombre FROM Miembro
       WHERE Apellido LIKE "B%"
         AND ciudad = "Santa Barbara"
         AND (telefono  LIKE "%8%" OR fax LIKE "%8%")
```

Observe los paréntesis en la última línea. Sin ellos no obtendrá los resultados que le pidió su jefe. Sin los paréntesis, cada conector se procesaría en orden del primero al último, lo que daría como resultado una lista que incluiría todos los miembros cuyos nombres empiezan con B y que viven en Santa Bárbara y cuyos números telefónicos tienen un 8 en ellos, y todos los miembros cuyo número de fax tiene un 8 en ellos independientemente de si viven o no en Santa Bárbara e independientemente de si su nombre empieza o no con B. Cuando se procesara el último OR, se seleccionarían los miembros cuyas características concordaran con la expresión antes de OR o la expresión después de OR. La expresión antes de OR se conecta con las experiencias previas por los AND previos y por ello no es independiente, pero la expresión después de OR sí es independiente, lo cual resultaría en la selección de todos los miembros con un 8 en su número de fax.

LIMIT especifica cuántas filas se pueden recuperar. La forma para LIMIT es

```
LIMIT numeroinicio,numerofilas
```

La primera fila que desea recuperar es numeroinicio, y el número de filas que desea recuperar es numerofilas. Si numeroinicio no se especifica, se asume que es 1. Para seleccionar sólo los tres primeros miembros que viven en Texas, use esta consulta:

```
SELECT * FROM Miembro WHERE estado="TX" LIMIT 3
```

Algunas consultas SELECT encontrarán registros idénticos, pero en este ejemplo usted sólo quiere ver uno — no todos — de los registros idénticos. Para evitar que la consulta recupere todos los registros idénticos, agregue la palabra DISTINCT inmediatamente después de SELECT.

Cómo combinar información de tablas

En las secciones anteriores de este capítulo, asumí que toda la información estaba en una sola tabla. Sin embargo, usted podría querer combinar información de tablas diferentes. Fácilmente puede hacerlo en una sola consulta.

Se pueden usar dos palabras en una consulta SELECT para combinar información de dos o más tablas:

- UNION: Se recuperan filas de una o más tablas y se almacenan juntas, una después de la otra, en un solo resultado. Por ejemplo, si su consulta seleccionó 6 filas de una tabla y 5 filas de otra tabla, el resultado contendrá 11 filas.
- JOIN: Las tablas se combinan una junto a la otra y la información se recupera de ambas tablas.

UNION

Se usa UNION para combinar los resultados de dos o más consultas de selección. Los resultados de cada consulta se añaden al conjunto de resultados seguidos de los resultados de la consulta anterior. El formato de la consulta UNION es el siguiente:

```
SELECT consulta UNION ALL SELECT consulta ...
```

Puede combinar tantas consultas SELECT como necesite. Una consulta SELECT puede incluir cualquier formato SELECT válido, incluyendo cláusulas WHERE, cláusulas LIMIT, etc. Las reglas para las consultas son

- Todas las consultas SELECT deben seleccionar el mismo número de columnas.
- Las columnas seleccionadas en las consultas deben contener el mismo tipo de datos.

El conjunto de resultados contendrá todas las filas de la primera consulta seguidas de todas las filas de la segunda consulta y así sucesivamente. Los nombres de las columnas usados en el conjunto de resultados son los nombres de las columnas de la primera consulta SELECT.

El arreglo de consultas SELECT puede seleccionar diferentes columnas de la misma tabla, pero no son comunes las situaciones en las que desea una tabla nueva con una columna en una tabla seguida por otra columna de la misma tabla. Es mucho más probable que necesite combinar columnas de tablas diferentes. Por ejemplo, tal vez tenga una tabla de miembros que han renunciado al club y una tabla aparte de miembros activos. Puede obtener una lista de todos los miembros, activos e inactivos, con la siguiente consulta:

```
SELECT Apellido,Nombre FROM Miembro UNION ALL
    SELECT apellido,nombre FROM MiembroAntiguo
```

El resultado de esta consulta es el apellido y el nombre de todos los miembros activos seguidos de los apellidos y nombres de todos los miembros que han renunciado.

Dependiendo de cómo organizó sus datos, podría tener nombres repetidos. Por ejemplo, tal vez un miembro renunció y su nombre está en la tabla MiembroAntiguo; pero se volvió a inscribir, así que su nombre se agregó a la tabla Miembro. Si no quiere duplicados, no incluya la palabra ALL. Si no se incluye ALL las líneas repetidas no se agregarán a los resultados.

Puede usar ORDER BY con cada consulta SELECT, como comenté en la sección anterior, o puede usar ORDER BY con una consulta UNION para ordenar todas las filas en el conjunto de resultados. Si desea que ORDER BY se aplique a todo el conjunto de resultados, y no sólo a la consulta a la que sigue, use un paréntesis, como se indica a continuación:

```
(SELECT Apellido FROM Miembro UNION ALL
     SELECT Apellido FROM MiembroAntiguo) ORDER BY Apellido
```

La instrucción UNION se introdujo en MySQL 4.0. No está disponible en MySQL 3.

Join

Combinar tablas una junta a otra se llama join o conjunción. Las tablas se combinan mediante la coincidencia de datos en una columna: la columna que tienen en común. La tabla de resultados combinada producida por una conjunción contiene todas las columnas de ambas tablas. Por ejemplo, si una tabla tiene dos columnas (Idmiembro y estatura), y la segunda tabla tiene dos columnas (Idmiembro y peso), una conjunción da como resultado una tabla con cuatro columnas: Idmiembro (de la primera tabla), estatura, Idmiembro (de la segunda tabla) y peso.

Hay dos tipos comunes de conjunción: una conjunción interna y una conjunción externa. La diferencia entre una conjunción interna y una conjunción externa es el número de filas incluidas en la tabla de resultados. La tabla de resultados produci-

da por una conjunción interna sólo contiene las filas que existían en ambas tablas. La tabla combinada producida por una conjunción externa contiene todas las filas que existían en una tabla con blancos en las columnas de las filas que no existen en la segunda tabla. Por ejemplo, si `tabla1` contiene una fila para Joe y una fila para Sally, y la `tabla2` contiene sólo una fila para Sally, una conjunción interna contendría solo una fila: la fila para Sally. Sin embargo, una conjunción externa contendría dos filas — una fila para Joe y una fila para Sally — aunque la fila para Joe tendría un campo en blanco en peso.

La tabla de resultados de la conjunción externa contiene todas las filas de una tabla. Si cualquiera de las filas de esa tabla no existe en la segunda tabla, las columnas de la segunda tabla estarán vacías. Por supuesto, los contenidos de la tabla de resultados los determina cuál tabla contribuye con todas sus filas, lo cual requiere que la segunda tabla coincida con ellas. Dos tipos de conjunción externa controlan cuál tabla determina las filas y cuál coincide con ella: una `LEFT JOIN` (conjunción izquierda) y una `RIGHT JOIN` (conjunción derecha).

Se usan diferentes consultas `SELECT` para una conjunción interna y los dos tipos de conjunción externa. La siguiente consulta es una conjunción interna:

```
SELECT listanombrecolumna FROM tabla1,tabla2
                    WHERE tabla1.col2 = tabla2.col2
```

Y estas consultas son conjunciones externas:

```
SELECT listanombrecolumna FROM tabla1 LEFT JOIN tabla2
       ON tabla1.col1=tabla2.col2

SELECT listanombrecolumna FROM tabla1 RIGHT JOIN tabla2
       ON tabla1.col1=tabla2.col2
```

En las tres consultas, `tabla1` y `tabla2` son las tablas a unir. Se pueden unir más de dos tablas. En ambas consultas, `col1` y `col2` son los nombres de las columnas que se asocian para unir las tablas. Las tablas se asocian con base en los datos en estas columnas. Estas dos columnas pueden tener el mismo nombre o nombres diferentes. Las dos columnas deben contener el mismo tipo de datos.

Como ejemplo de las conjunciones interna y externa, considere un breve formulario del Catálogo de Mascotas. Una tabla es `Mascota`, en la cual las dos columnas `NombreMascota` y `TipoMascota` guardan los siguientes datos:

```
NombreMascota  TipoMascota

Unicornio      Caballo
Pegaso         Caballo
Leon           Gato
```

La segunda tabla es Color, en la cual las dos columnas `NombreMascota` y `ColorMascota` guardan los siguientes datos:

```
NombreMascota  ColorMascota

Unicornio      blanco
Unicornio      plateado
Pez            Dorado
```

Suponga que necesita hacer una pregunta que requiere información de ambas tablas. Si hace una conjunción interna con la siguiente consulta:

```
SELECT * FROM Mascota,Color WHERE Mascota.NombreMascota = Co-
          lor.NombreMascota
```

obtendrá la siguiente tabla de resultados con cuatro columnas: `NombreMascota (de Mascota)`, `TipoMascota`, `NombreMascota (de Color)` y `ColorMascota`.

```
NombreMascota  TipoMascota  NombreMascota ColorMascota

Unicornio          Caballo   Unicornio   blanco
Unicornio          Caballo   Unicornio   plateado
```

Observe que solo `Unicornio` aparece en la tabla de resultados, porque solo `Unicornio` estaba en las dos tablas originales antes de la conjunción. Por otra parte, suponga que hace una conjunción externa izquierda con la siguiente consulta:

```
SELECT * FROM Mascota LEFT JOIN Color
         ON Pet.NombreMascota=Color.NombreMascota
```

Se obtiene la siguiente tabla de resultados, con las mismas cuatro columnas — `NombreMascota (de Mascota)`, `TipoMascota`, `NombreMascota (de Color)` y `ColorMascota` — pero con filas diferentes:

```
NombreMascota  TipoMascota  NombreMascota ColorMascota

Unicornio          Caballo     Unicornio  blanco
Unicornio          Caballo     Unicornio  plateado
Pegaso             Caballo     <NULL>     <NULL>
Leon               Gato        <NULL>     <NULL>
```

Esta tabla tiene cuatro filas. Tiene las dos primeras filas que la conjunción interna, pero tiene dos filas adicionales — filas que están a la izquierda en la tabla `TipoMascota` pero no en la tabla `Color`. Observe que las columnas de la tabla `Color` están en blanco en las dos últimas filas.

Y, en tercer lugar, suponga que hace una conjunción externa derecha con la siguiente consulta:

```
SELECT * FROM Mascota RIGHT JOIN Color
         ON Pet.NombreMascota=Color.NombreMascota
```

Obtendrá la siguiente tabla de resultados, con las mismas cuatro columnas pero filas diferentes:

```
NombreMascota  TipoMascota  NombreMascota ColorMascota

Unicornio        Caballo    Unicornio     blanco
Unicornio        Caballo    Unicornio     plateado
<NULL>           <NULL>       Pez         dorado
```

Observe que estos resultados contienen todas las filas de la tabla `Color` a la derecha, pero no de la tabla `Mascota`. Observe los espacios en blanco en las columnas de la tabla `Mascota`, la cual tiene una fila para Pez.

Las conjunciones de las que he hablado hasta ahora encuentran entradas coincidentes en las tablas. A veces es útil averiguar cuáles filas en una tabla no tienen entradas coincidentes en otra tabla. Por ejemplo, suponga que desea saber quién no ha entrado nunca en la sección Solo para miembros. Como tiene una tabla con el nombre de entrada del miembro y otra tabla con las fechas en las que entran los usuarios, puede hacer aquella pregunta usando las dos tablas. Hacer una conjunción y ver todas las concordancias para tratar de ver quién falta sería imposible si hay un gran número de usuarios en la tabla. Sin embargo, puede averiguar cuáles nombres de entrada no tienen entradas registradas en la tabla de entrada haciendo la siguiente consulta:

```
SELECT Nombreentrada FROM Miembro LEFT JOIN Entrada
      ON Miembro.Nombreentrada=Entrada.Nombreentrada
      WHERE Entrada.Nombreentrada IS NULL
```

Esta consulta le dará una lista de todos los nombres de entrada en `Miembro` que no estén en la tabla `Entrada`.

Cómo actualizar información

Cambiar información en una fila existente es actualizar la información. Por ejemplo, tal vez necesite cambiar la dirección de un miembro porque se mudó, o tal vez tenga que agregar un número de fax que un miembro dejó en blanco cuando inicialmente digitó la información.

La consulta `UPDATE` es muy directa:

```
UPDATE nombretabla SET columna=valor,columna=valor,...
       WHERE clausula
```

En la cláusula `SET` usted enumera las columnas que deben actualizarse y los valores nuevos que deben insertarse. Enumere todas las columnas que quiera cambiar en una sola consulta. Sin una cláusula `WHERE`, los valores de las colum-

nas se cambiarán en todas las filas. Pero con la cláusula WHERE puede especificar cuáles filas actualizar. Por ejemplo, use esta consulta para actualizar una dirección en la tabla Miembro:

```
UPDATE Miembro SET calle="3333 Giant St",
                   telefono ="555-555-5555"
               WHERE Nombreentrada="bigguy"
```

Cómo eliminar información

Mantenga la información en su base de datos al día borrando la información obsoleta. Puede eliminar una fila de una tabla usando la consulta DELETE:

```
DELETE FROM nombretabla WHERE clausula
```

Tenga muchísimo cuidado al usar DELETE. Si usa una consulta DELETE sin una cláusula WHERE, borrará todos los datos en la tabla. Repito: todos los datos. Una vez más: todos los datos. Los datos no se podrán recuperar. Esta función de la consulta DELETE está justo al inicio de mi lista no-intente-esto –en-su-casa.

Puede borrar una columna de una tabla usando la consulta ALTER:

```
ALTER TABLE nombretabla DROP nombrecolumna
```

O, por supuesto, puede eliminar toda la tabla y empezar de nuevo con:

```
DROP TABLE nombretabla
```

o

```
DROP DATABASE nombrebasededatos
```

Capítulo 5

Cómo proteger los datos

En este capítulo

- Comprender la seguridad de los datos en MySQL
- Agregar nuevas cuentas MySQL
- Modificar cuentas existentes
- Cambiar contraseñas
- Hacer respaldos
- Reparar datos
- Restaurar datos

Sus datos son esenciales en su aplicación con base de datos para la Web. Almacenar y/o presentar datos son las actividades principales de su aplicación web. Usted ha invertido una considerable cantidad de tiempo desarrollando su base de datos, la cual contiene información importante digitada por usted o por sus usuarios. Por lo tanto, debe protegerla. En este capítulo le muestro cómo hacerlo.

Controlar el acceso a sus datos

Usted controlar el acceso a la información en su base de datos. Debe decidir quién puede ver los datos y quién puede cambiarlos. Imagine lo que sucedería si sus competidores pudieran cambiar la información de su catálogo de productos en línea o copiar su lista de clientes. Muy pronto lo dejarían fuera del negocio. Es obvio que necesita resguardar sus datos.

MySQL brinda un sistema de seguridad para proteger sus datos. Nadie puede tener acceso a su base de datos sin una cuenta. Todas las cuentas MySQL tienen los siguientes atributos:

- Un nombre
- Un hostname: la máquina desde la cual puede la cuenta tener acceso al servidor MySQL
- Una contraseña
- Un conjunto de permisos

Para tener acceso a sus datos, una persona debe usar un nombre de cuenta válido y conocer la contraseña asociada con esa cuenta. Además, esa persona debe conectarse desde un PC autorizada a conectarse con su base de datos mediante esa cuenta específica.

Después de que al usuario se le otorga el acceso a la base de datos, lo que esa persona haga con los datos depende de los permisos que se hayan establecido para esa cuenta. A cada cuenta se le permite o se le prohíbe realizar operaciones en su base de datos, tales como `SELECT`, `DELETE`, `INSERT`, `CREATE`, `DROP`, etc. Las configuraciones que especifican qué puede hacer una cuenta se llaman privilegios o permisos. Puede configurar una cuenta con todos los permisos, sin ningún permiso o con cualquier rango entre esos extremos. Por ejemplo, para un catálogo en línea de productos, a usted le conviene que los clientes puedan ver la información en el catálogo pero que no puedan cambiarla.

Cuando un usuario intenta conectarse a MySQL y ejecutar una consulta, MySQL controla el acceso a los datos en dos etapas:

1. **Verificación de la conexión:** MySQL verifica la validez del nombre de la cuenta y de la contraseña y verifica que la conexión provenga de un host autorizado a conectarse al servidor MySQL usando la cuenta en cuestión. Si todo está en orden, MySQL acepta la conexión.
2. **Verificación de la solicitud:** Después de que MySQL acepta la conexión, verifica si la cuenta tiene los permisos necesarios para ejecutar la consulta especificada. Si es así, MySQL ejecuta la consulta.

Cualquier consulta enviada a MySQL puede fallar porque la conexión es rechazada en el primer paso, o porque la consulta no es permitida en el segundo paso. En estos casos aparecerá un mensaje de error que ayuda a identificar la fuente del problema.

En las siguientes secciones describo más detalladamente las cuentas y los permisos.

Comprender los nombres de las cuentas y los hostnames

Juntos, el nombre de la cuenta y el hostname (el nombre del PC autorizado a conectarse con la base de datos) identifican una cuenta única. Pueden existir dos cuentas con el mismo nombre de cuenta pero con hostnames diferentes, y pueden tener contraseñas y permisos diferentes. Sin embargo, no puede haber dos cuentas con el mismo nombre y el mismo hostname.

El servidor MySQL aceptará conexiones de una cuenta MySQL sólo cuando se conecte desde el hostname. Cuando se prepara una consulta `GRANT` o `REVOKE` (las describo más adelante en este capítulo), se identifica la cuenta MySQL usando tanto el nombre de la cuenta como el hostname, en el siguiente formato: `nombrecuenta@hostname` (por ejemplo, `root@localhost`).

El nombre de la cuenta MySQL no tiene ninguna relación con el nombre de usuario en Unix/Linux o Windows (también llamado a veces nombre de entrada). Si está usando una cuenta MySQL administrativa llamada `root`, esta no se relaciona con el nombre de entrada `root` en Unix/Linux. Cambiar el nombre de entrada de MySQL no afecta de ninguna manera el nombre de entrada de Unix/Linux o Windows, y viceversa.

Los hostnames y los nombres de las cuentas MySQL se definen como sigue:

- **Un nombre de cuenta puede tener hasta 16 caracteres de longitud.** Puede usar caracteres especiales en los nombres de las cuentas, por ejemplo espacios o un guiones (-). Sin embargo, no puede usar comodines en el nombre de la cuenta.
- **Un nombre de cuenta puede estar en blanco.** Si existe una cuenta en MySQL con el nombre en blanco, cualquier nombre de cuenta será válido para esa cuenta. Un usuario podría usar cualquier nombre de cuenta para conectarse con su base de datos, esto si el usuario se conecta desde un hostname al que se le permite conectarse a la cuenta con el nombre en blanco y si usa la contraseña correcta, de ser necesaria. Puede usar una cuenta con el nombre en blanco si quiere permitir que usuarios anónimos se conecten con la base de datos.
- **El hostname puede ser un nombre o una dirección IP.** Por ejemplo, puede ser un nombre como `thor.mycompany.com`, o una dirección IP (protocolo de Internet) como 192.163.2.33. La máquina en la cual está instalado el servidor MySQL es `localhost`.
- **Pueden usarse comodines en el hostname.** Puede usar el signo de porcentaje (%) como comodín; % funciona para cualquier hostname. Si agrega una cuenta para `george@%`, alguien que use el nombre de cuenta george puede conectarse al servidor MySQL desde cualquier PC.
- **El hostname puede estar en blanco.** Un hostname en blanco equivale a usar % para el hostname.

Es posible configurar una cuenta con el nombre de la cuenta en blanco y con el hostname en blanco. Una cuenta tal permitiría a cualquier persona conectarse con el servidor MySQL usando cualquier nombre de cuenta desde cualquier PC. Una cuenta con el nombre de cuenta en blanco y un signo de porcentaje (%) en el hostname funcionaria del mismo modo. No es muy probable que quiera tener tal tipo de cuenta. A veces esta cuenta se instala cuando se instala MySQL, pero en esos casos no se le da ningún privilegio, de modo que no podrá hacer nada.

Cuando MySQL es instalado, automáticamente instala una cuenta con todos los privilegios: `root@localhost`. Esta cuenta se instala sin contraseña. Cualquiera que esté conectado al PC donde está instalado MySQL tiene acceso a MySQL y puede hacer cualquier cosa con él usando el nombre de cuenta `root`. (Obviamente, `root` es un nombre de cuenta muy conocido, así que esta cuenta no es muy segura. Si usted es el administrador de MySQL, agréguele inmediatamente una contraseña a esta cuenta).

En algunos sistemas operativos se instalan automáticamente otras cuentas, además de `root@localhost`. En Windows, por ejemplo, una cuenta llamada `root@%` podría instalarse sin contraseña de protección. Esta cuenta root con todos los privilegios puede ser usada por cualquier persona desde cualquier máquina. Debe eliminar esta cuenta inmediatamente, o al menos asignarle una contraseña.

Aprender más sobre contraseñas

A cada cuenta se le asigna una contraseña. Si a una cuenta no se le asigna ninguna, la contraseña estará en blanco, y eso significa que no requerirá de contraseña para usarse. MySQL no impone límites a la longitud de una contraseña, pero a veces algún otro software en su sistema limitará la longitud a ocho caracteres. Si es así, los caracteres después del octavo serán eliminados.

Como seguridad adicional, MySQL encripta las contraseñas antes de almacenarlas. Eso significa que las contraseñas no se almacenan en los caracteres reconocibles en los cuales se digitaron. Esta medida de seguridad garantiza que nadie pueda ver y conocer las contraseñas almacenadas.

Desafortunadamente, algunas malas personas tratarán de tener acceso a sus datos adivinando cuál es su contraseña. Usan un software que trata de conectarse usando rápida y sucesivamente diferentes contraseñas, una práctica llamada descifrамiento (cracking). A continuación presento algunas recomendaciones para escoger una contraseña que sea tan difícil de descifrar como sea posible:

- Use de seis a ocho caracteres.
- Incluya uno o más de cada uno de los siguientes: letras en mayúscula, letras en minúscula, números y signos de puntuación.
- No use el nombre de su cuenta ni ninguna variación del nombre de su cuenta.
- No incluye ninguna palabra que esté en un diccionario.
- No incluya un nombre.
- No use un número telefónico ni una fecha.

Una buena contraseña es difícil de adivinar, no incluye palabras de ningún diccionario (incluyendo diccionarios en otras lenguas) y es fácil de recordar. Si es muy difícil de recordar tendría que escribirla, lo cual contradice el propósito de usar una contraseña. Una manera de crear una buena contraseña es usar los primeros caracteres de una de sus frases favoritas. Por ejemplo, podría usar la frase "¡Todos para uno! ¡Uno para todos!" para hacer esta contraseña:

```
¡Tpu!¡Upt!
```

Esta contraseña no incluye números, pero puede arreglar eso usando el número 4 en vez de la letra p. Entonces la contraseña sería:

```
¡T4u!¡U4t!
```

O podría usar el número 1 en vez de la letra u para representar la palabra uno. Entonces la contraseña sería

```
¡T41!¡14t!
```

Esta contraseña es definitivamente difícil de adivinar. Otra forma de incorporar números en su contraseña es sustituyendo 1 (uno) por l (ele) o sustituir 0 (cero) por la letra o.

Echemos un vistazo a los permisos

Los permisos de las cuentas los usa MySQL para especificar quién puede hacer qué. Cualquier persona que use una cuenta válida puede conectarse al servidor MySQL, pero sólo podrá hacer las cosas aprobadas por los permisos asignados a esa cuenta. Por ejemplo, es posible configurar una cuenta para que los usuarios puedan seleccionar datos, pero no insertar ni actualizar datos.

Se puede asignar permisos a bases de datos, tablas o columnas específicas. Por ejemplo, se puede configurar una cuenta que permita al usuario seleccionar datos de todas las tablas en la base de datos, pero que le permita insertar datos sólo en una tabla y actualizar datos sólo en una columna de una tabla en particular.

Los permisos se conceden usando la consulta `GRANT` y se deniegan usando la consulta `REVOKE`. Las consultas `GRANT` o `REVOKE` deben enviarse usando una cuenta que tenga permiso para ejecutar instrucciones `GRANT` o `REVOKE` en la base de datos. Si trata de enviar una consulta `GRANT` o una consulta `REVOKE` usando una cuenta que no tenga permiso para hacerlo, obtendrá un mensaje de error. Por ejemplo, si trata de conceder un permiso para usar un comando de selección y envía la consulta usando una cuenta que no tiene permiso para conceder permisos, posiblemente verá el siguiente mensaje:

```
grant command denied
```

Los permisos se pueden otorgar y remover individualmente o todos a la vez. La Tabla 5-1 enumera algunos de los permisos que puede asignar o revocar.

Tabla 5-1 Permisos para cuentas MySQL

Permiso	*Descripción*
ALL	Todos los permisos
ALTER	Puede alterar la estructura de las tablas

Permiso	Descripción (continuación)
CREATE	Puede crear bases de datos o tablas nuevas
DELETE	Puede borrar filas en las tablas
DROP	Puede borrar bases de datos o tablas
FILE	Puede leer y escribir archivos en el servidor
GRANT	Puede cambiar los permisos en una cuenta MySQL
INSERT	Puede insertar filas nuevas en las tablas
SELECT	Puede leer datos de las tablas
SHUTDOWN	Puede apagar el servidor MySQL
UPDATE	Puede cambiar datos en una tabla
USAGE	No tiene ningún permiso

Otorgar `ALL` no es una buena idea porque incluye permisos para operaciones administrativas, tales como apagar el servidor MySQL. No es probable que quiera que cualquiera además de usted tenga tales privilegios.

Cómo configurar cuentas MySQL

Al crear una cuenta nueva se especifica la contraseña, el nombre de los PCs que pueden tener acceso a la base de datos usando esta cuenta, y los permisos; pero en cualquier momento puede cambiar estas configuraciones. Toda la información de la cuenta se almacena en una base de datos llamada `mysql`, creada automáticamente al instalar MySQL. Para agregar una cuenta nueva o cambiar información sobre cualquier cuenta, debe usar una cuenta que tenga los permisos apropiados en la base de datos `mysql`.

Necesita por lo menos una cuenta para tener acceso al servidor MySQL. Cuando se instala, MySQL automáticamente configura algunas cuentas, incluyendo una cuenta con todos los permisos, llamada `root`. Si tiene acceso a MySQL por medio del sitio web de su compañía o de una compañía de hospedaje de sitios web, el administrador MySQL de la compañía debe darle la cuenta; la cuenta probablemente no se llamará root y es posible que no tenga todos los permisos.

En lo que resta de esta sección, describo cómo agregar y borrar cuentas y cómo cambiar las contraseñas y los permisos de las cuentas. Si recibió su cuenta del departamento TI de su compañía o de la compañía de hospedaje de sitios

La base de datos de seguridad de MySQL

Cuando es instalado, MySQL automáticamente crea una base de datos llamada mysql. Toda la información usada para proteger sus datos se guarda en esta base de datos, incluyendo los nombres de las cuentas, los hostnames, las contraseñas y los permisos.

Los permisos se almacenan en columnas. El formato de cada nombre de columna es `permission_priv`, donde permission es cualquiera de los permisos mostrados en la Tabla 5-1. Por ejemplo, la columna que contiene los permisos ALTER se llama `alter_priv`. El valor en cada columna de permiso es Y o N, lo cual significa sí (yes) o no. De este modo, por ejemplo en la tabla de usuarios (descrita en la lista siguiente), habría una fila para una cuenta y una columna para `alter_priv`. Si el campo de una cuenta bajo `alter_priv` contiene Y, la cuenta se puede usar para ejecutar consultas ALTER. Si `alter_priv` contiene N, la cuenta no tiene permiso para ejecutar consultas ALTER.

La base de datos mysql guarda permisos en las cinco tablas siguientes:

- Tabla `user`: Esta tabla almacena los permisos que aplican a todas las bases de datos y todas las tablas. Contiene una fila para cada cuenta válida con el nombre del usuario, el hostname y la contraseña. El servidor MySQL rechazará cualquier conexión proveniente de una cuenta que no exista en esta tabla
- Tabla `db`: Esta tabla almacena permisos que aplican a una base de datos en particular. Contiene una fila para la base de datos, que da permisos a un nombre de cuenta y hostname. La cuenta debe existir en la tabla user para que se le pueda conceder permisos. Los permisos concedidos en la tabla user anulan los permisos asignados en esta tabla. Por ejemplo, si la tabla user tiene una fila para la cuenta designer con privilegios INSERT, designer puede insertar datos en todas las bases de datos. Si una fila en la tabla db muestra N bajo INSERT para la cuenta designer en la base de datos CatálogodeMascotas, la tabla user anula ese valor y designer podrá insertar datos en la base de datos CatalogodeMascotas.
- Tabla `host`: Esta tabla controla el acceso a una base de datos dependiendo del host. La tabla host opera con la tabla db. Si una fila en la tabla db tiene el campo host vacío, MySQL verifica la tabla host para ver si la tabla db tiene ahí una fila. De esta manera, usted puede permitir el acceso a db desde algunos hosts pero no de otros. Por ejemplo, suponga que tiene dos bases de datos: db1 y db2. La base de datos db1 tiene información muy delicada, y quiere que sólo ciertas personas la vean. La base de datos db2 tiene información que quiere que todos vean. Si tiene una fila en la tabla db para db1 con el campo host en blanco, puede tener dos filas para db1 en la tabla host. Una fila puede conceder todos los permisos a los usuarios que se conecten desde un host específico, mientras que la otra fila puede denegarles privilegios a los usuarios que se conecten desde cualquier otro host.
- Tabla `tables_priv`: Esta tabla almacena permisos aplicados a tablas específicas.
- Tabla `columns_priv`: Esta tabla almacena permisos aplicados a columnas específicas.

Usted puede ver y cambiar las tablas directamente en mysql si está usando una cuenta con los permisos necesarios. Puede usar consultas SQL tales como SELECT, INSERT, UPDATE y otras. Si tiene acceso a MySQL por medio de empleador, un cliente o una compañía de hospedaje de sitios web, es poco probable que le den una cuenta con los permisos necesarios.

web, podría recibir un error cuando trate de enviar alguna de las consultas `GRANT` o `REVOKE` antes descritas. Si su cuenta tiene restricciones para ejecutar alguna de las consultas que necesite hacer, debe solicitar una cuenta con más permisos o pedirle al administrador MySQL que agregue una cuenta nueva o que haga los cambios que usted requiera.

Identificar las cuentas existentes

Para averiguar cuáles cuentas existen para su base de datos, necesita una cuenta que tenga los permisos necesarios. Trate de ejecutar la siguiente consulta en la base de datos llamada `mysql`:

```
SELECT * FROM user
```

Debiera obtener una lista de todas las cuentas. Sin embargo, si tiene acceso a MySQL por medio de su compañía o de una compañía de hospedaje de sitios web, probablemente usted no tiene los permisos necesarios. En ese caso, probablemente recibirá un mensaje de error como este:

```
No Database Selected
```

Este mensaje significa que su cuenta no tiene permiso para seleccionar la base de datos `mysql`. O quizá recibirá un mensaje de error que diga que no tiene permiso para usar `SELECT`. Aunque este mensaje es molesto, en algún sentido es bueno, pues muestra que la compañía tiene buenas medidas de seguridad. No obstante, en otro sentido es malo pues usted no podrá ver cuáles privilegios tiene su cuenta. Deberá preguntarle al administrador de MySQL o averiguarlo usted mismo intentando consultas diferentes y viendo si se le permite ejecutarlas o no.

Cómo agregar cuentas nuevas y cambiar permisos

La forma preferida para tener acceso a MySQL desde PHP es configurando una cuenta específicamente para este propósito, sólo con los permisos imprescindibles. En esta sección, describo cómo agregar cuentas nuevas y cómo cambiar permisos. Si su cuenta se la dio el departamento TI de su compañía o una compañía de hospedaje de sitios web, tal vez no tenga los permisos necesarios para crear una cuenta nueva. Si no los tiene, no podrá agregar una cuenta ejecutando la consulta `GRANT`, y tendrá que solicitar una segunda cuenta para usarla con PHP.

Si necesita solicitar una segunda cuenta, obtenga una cuenta con permiso restringido (si es posible), pues su aplicación web será más segura si la cuenta usada en sus programas PHP no tiene más privilegios de los necesarios.

La consulta GRANT se usa para configurar cuentas nuevas, para cambiar contraseñas o para agregar permisos a cuentas ya existentes. Si la cuenta ya existe, la consulta GRANT cambia la contraseña o agrega permisos. Si la cuenta aún no existe, la consulta GRANT agrega una cuenta nueva.

Este es el formato general para una consulta GRANT:

```
GRANT permiso (columnas) ON nombretabla
    TO nombrecuenta@hostname IDENTIFIED BY 'clave'
```

Puede usar esta consulta GRANT para crear una cuenta nueva o para cambiar una cuenta existente. Debe llenar la información siguiente:

- *permiso (columnas):* Debe indicar por lo menos un permiso. Puede limitar cada permiso a una o más columnas indicando el nombre de la columna en paréntesis después del permiso. Si no aparece el nombre de la columna, el permiso se concede a todas las columnas en la(s) tabla(s). Puede indicar tantos permisos/columnas como necesite, separados por comas. Los permisos posibles se indican en la Tabla 5-1. Por ejemplo, una consulta GRANT podría empezar así:

  ```
  GRANT select (Apellido,Nombre), update,
        insert (fechanacimiento) ...
  ```

- *nombretabla:* Indica sobre cuál(es) tabla(s) se concede el permiso. Se requiere por lo menos de una tabla. Puede indicar varias tablas, separándolas por comas. Los valores posibles para nombretabla son

  - *nombretabla:* Toda la tabla llamada nombretabla en la base de datos actual. Puede usar un asterisco (*) para indicar todas las tablas en la base de datos actual. Si usa un asterisco y no selecciona una base de datos, el privilegio se concederá a todas las tablas en todas las bases de datos.
  - *nombrebasedatos.nombre.tabla:* Toda la tabla llamada nombretabla en nombrebasedatos. Si desea indicar todas, puede usar un asterisco (*) para el nombre de la base de datos o el nombre de la tabla. Usar *.* concede el permiso a todas las tablas en todas las bases de datos.

- *nombrecuenta@hostname:* Si la cuenta ya existe, recibe los permisos indicados. Si la cuenta no existe, se añade. La cuenta se identifica con el nombrecuenta y hostname como pareja. Si una cuenta existe con el nombre de cuenta especificado pero un hostname diferente, la cuenta existente no se cambia, sino que se crea una nueva.

- *clave:* Es la contraseña que agrega o cambia. No es obligatorio usar una contraseña. Si no quiere agregar ni cambiar una contraseña para esta cuenta, no incluya la frase IDENTIFIED BY 'clave'.

La consulta GRANT para agregar una nueva cuenta para usar en los programas PHP para la base de datos CatalogodeMascotas podría ser

```
GRANT select ON CatalogodeMascotas.* TO phpuser@localhost
        IDENTIFIED BY 'A41!14a!'
```

Cómo agregar y cambiar contraseñas

Puede agregar o cambiar contraseñas usando la consulta GRANT. Puede incluir como requisito una contraseña cuando agrega una cuenta nueva, tal como describí en la sección anterior. Si una cuenta ya existe, puede cambiar su contraseña usando la siguiente consulta GRANT:

```
GRANT permiso ON * TO nombrecuenta@hostname
                IDENTIFIED BY 'clave'
```

Debe llenar la información siguiente:

- *permiso:* Debe indicar por lo menos un permiso en una consulta GRANT. Si el permiso ya ha sido concedido, no cambia.
- *nombrecuenta@hostname:* Si la cuenta ya existe, recibe el permiso y la contraseña indicados. Si la cuenta no existe, la cuenta se agrega. La cuenta se identifica con el nombrecuenta y el hostname como pareja. Si una cuenta existe con el nombre de cuenta especificado pero con un hostname diferente, la cuenta no cambia, sino que se crea una nueva.
- *clave:* La contraseña en la consulta GRANT reemplaza la contraseña existente. Si suministra una contraseña vacía usando

  ```
  IDENTIFIED BY ''
  ```

  la contraseña existente se reemplaza con un blanco, dejando la cuenta sin contraseña. Consulte la sección "Aprender más sobre contraseñas", anteriormente en este capítulo, para leer algunos consejos sobre cómo escoger una buena contraseña.

Eliminar permisos

Para revocar permisos use la consulta REVOKE. El formato general es

```
REVOKE permiso (columnas) ON nombretabla
    FROM nombrecuenta@hostname
```

Debe llenar la información siguiente:

- *permiso (columnas):* Debe indicar por lo menos un permiso. Puede revocar cada permiso de una o más columnas indicando el nombre de la columna entre paréntesis después del permiso. Si no se indica el nombre de la columna, el permiso se revocará de todas las columnas en la(s) tabla(s). Puede indicar tantos permisos/columnas como necesite, separándolos por comas. Los permisos posibles se enumeran en la Tabla 5-1. Por ejemplo, una consulta REVOKE podría empezar así:

  ```
  REVOKE select (Nombre,Apellido), update, insert (fechanacimiento) ...
  ```

- *nombretabla:* Indica a cuáles tablas se les está revocando el permiso. Se debe indicar al menos una tabla. Puede enumerar varias tablas, separándolas por comas. Los valores posibles para nombretabla son
  - *nombretabla:* Toda la tabla llamada nombretabla en la base de datos actual. Puede usar un asterisco (*) para indicar todas las tablas. Si usa un asterisco cuando no ha seleccionado la base de datos actual, el privilegio se revocará de todas las tablas en todas las bases de datos.
  - *nombrebasedatos. nombretabla:* Toda la tabla llamada nombretabla en nombrebasedatos. Si desea indicar todas, puede usar un asterisco (*) para el nombre de la base de datos o para el nombre de la tabla. Usar *.* revoca el permiso en todas las tablas en todas las bases de datos.
- *nombrecuenta@hostname:* La cuenta se identifica con el nombrecuenta y el hostname como pareja. Si una cuenta existe con el nombre de cuenta especificado pero un hostname diferente, la consulta REVOKE fallará y usted recibirá un mensaje de error.

Eliminar cuentas

Generalmente no es necesario eliminar una cuenta. Si creó una cuenta con permisos que no desea que tenga, sólo cambie los permisos. Si no quiere usar una cuenta elimine todos los permisos, de modo que con esa cuenta no se pueda hacer nada. Para eliminar todos los permisos de una cuenta, use una consulta REVOKE con la siguiente sintaxis:

```
REVOKE all ON *.* FROM nombrecuenta@hostname
```

Para realmente poder eliminar una cuenta, necesita una cuenta con los permisos necesarios en la base de datos mysql. Necesita usar una consulta DELETE en la tabla del usuario en la base de datos mysql. Si desea más información sobre la estructura de la base de datos de seguridad mysql, consulte el recuadro "La base de datos de seguridad de MySQL", unas páginas atrás en este capítulo. Tenga cuidado al usar una consulta DELETE, pues con el formato incorrecto podría eliminar la cuenta equivocada o incluso todas las cuentas. Lea los comentarios sobre el comando DELETE al final del Capítulo 4.

Respaldar sus datos

Usted debe tener por lo menos una copia de su valiosa base de datos. Los desastres no son frecuentes, pero sí ocurren. El PC donde está almacenada su base de datos puede fallar y perder sus datos, el archivo puede corromperse, el edificio puede quemarse, etcétera. Las copias de respaldo de su base de datos previenen la pérdida de datos en tales desastres.

Al menos debe tener una copia de respaldo de su base de datos, almacenada en un lugar aparte, y no en el mismo lugar en el que está la copia que está usando actualmente. Tener más de una copia — tal vez unas tres — es generalmente una buena idea.

- Almacene una copia en una ubicación fácilmente accesible, incluso podría ser en el mismo PC, para reemplazar rápidamente una base de datos en uso que se haya dañado.
- Almacene una segunda copia en otro PC; le será útil en caso de que el PC falle y la primera copia de respaldo no esté disponible.
- Almacene una tercera copia en un lugar físico completamente diferente, pues existe la remota posibilidad de que el edificio verdaderamente se queme. Esta tercera copia no será necesaria si la segunda copia de respaldo se almacena vía red en un PC ubicado en otro lugar físico.

Si no tiene acceso a un PC fuera del sitio donde pueda respaldar su base de datos, copie su respaldo en un medio portátil, tal como una cinta o un CD y guárdelo fuera del sitio. Ciertas compañías almacenarán sus respaldos en sus oficinas a cambio de una mensualidad, o simplemente puede poner el respaldo en su bolsillo y llevárselo a casa.

Si usa MySQL en el PC de otra persona, tal como el PC de su jefe o de una compañía de hospedaje de sitios web, las personas que le proporcionan acceso son responsables de los respaldos. Deben contar con procedimientos automáticos para hacer respaldos de su base de datos. Cuando evalúa compañías de hospedaje de sitios web, una buena pregunta que no debe olvidar hacer es cuáles son sus procedimientos de respaldo. Lo que le interesa saber es con cuánta frecuencia hacen copias de respaldo y dónde las guardan. Si cree que sus datos no están seguros, solicite que se hagan cambios o adiciones a los procedimientos de respaldo.

Si es el administrador de MySQL, usted es el responsable de hacer los respaldos. MySQL brinda un programa llamado `mysqldump` que puede usar para hacer las copias de respaldo; `mysqldump` crea un archivo de texto con todas las instrucciones SQL necesarias para recrear toda su base de datos. El archivo contiene las instrucciones `CREATE` para cada tabla y las instrucciones `INSERT` para cada fila de datos en las tablas. Puede restaurar su base de datos ejecutando el conjunto de instrucciones MySQL. Puede restaurarla en su ubicación actual o, de ser necesario, puede restaurarla en otro PC.

Siga estos pasos para hacer una copia de respaldo de su base de datos en Linux/Unix/Mac:

1. **Cámbiese al subdirectorio bin en el directorio donde está instalado MySQL.**

   Por ejemplo, digite **cd /usr/local/mysql/bin.**

2. **Digite lo siguiente:**

```
mysqldump --user=nombrecuenta --password=clave
    nombrebasedatos >ruta/nombrearchivorespaldo
```

donde

- *nombrecuenta* es el nombre de la cuenta MySQL que está usando para respaldar la base de datos.
- *clave* es la contraseña de la cuenta.
- *nombrebasedatos* es el nombre de la base de datos que desea respaldar.
- *ruta/nombrearchivorespaldo* es la ruta al directorio donde desea almacenar los respaldos y el nombre del archivo en el que se almacenará el resultado SQL.

La cuenta que use tiene que tener permiso para seleccionar. Si la cuenta no requiere de una contraseña, puede dejar de lado toda la opción de contraseña.

Puede digitar el comando en una línea, sin presionar Enter. O puede digitar una barra inversa (\), presionar Enter y luego continuar el comando en otra línea.

Por ejemplo, para respaldar la base de datos `CatalogodeMascotas`, el comando podría ser

```
mysqldump --user=root --password=bigsecret
        CatalogodeMascotas \
>/usr/local/mysql/backups/RespaldoCatalogodeMascotas
```

Nota: La cuenta Linux/Unix con la que esté registrado tiene que tener permiso para guardar archivos en el directorio de respaldo.

Para hacer una copia de respaldo de su base de datos en Windows, siga estos pasos:

1. **Abra una ventana de comando.**

   Por ejemplo, escoja Start⇨Programs⇨MS-DOS.

2. **Cámbiese al subdirectorio bin en el directorio donde está instalado MySQL.**

   Por ejemplo, digite **cd c:\mysql\bin.**

3. **Digite lo siguiente:**

   ```
   mysqldump.exe --user=nombrecuenta --password=contraseña
     nombrebasedatos >ruta\nombrearchivorespaldo
   ```

   donde

   - *nombrecuenta* es el nombre de la cuenta MySQL que está usando para respaldar la base de datos.
   - *contraseña* es la contraseña de la cuenta.
   - *nombrebasedatos* es el nombre de la base de datos que desea respaldar.
   - *ruta\nombrearchivoresplado* es la ruta al directorio donde desea almacenar los respaldos y el nombre del archivo en que se almacenará el resultado SQL.

La cuenta que use necesita tener permiso para seleccionar. Si la cuenta no requiere de una contraseña, puede dejar de lado toda la opción de contraseña.

Debe digitar el comando `mysqldump` en una sola línea sin oprimir Enter.

Por ejemplo, para respaldar la base de datos `CatalogodeMascotas`, el comando podría ser

```
mysqldump.exe --user=root CatalogodeMascotas
    >RespaldoCatalogodeMascotas
```

Los respaldos deben hacerse en un horario predeterminado, por lo menos una vez al día. Si su base de datos cambia frecuentemente, sería mejor respaldarla más a menudo. Por ejemplo, podría ser conveniente respaldar el directorio de respaldos cada hora y respaldar todo en otro PC una vez al día.

Restaurar sus datos

En algún punto una de las tablas de su base de datos podría dañarse y dejar de funcionar. Es inusual, pero sucede. Un problema de hardware, por ejemplo, o que el PC se apague inesperadamente, puede causar que las tablas se corrompan. Algunas veces una anomalía en los datos que confunda a MySQL puede corromper las tablas. En algunos casos, una tabla corrompida puede hacer que su servidor MySQL deje de funcionar.

Este es un típico mensaje de error que indica que una tabla se ha corrompido:

```
Incorrect key file for table: 'nombretabla'.
```

En algunos casos es posible reparar la(s) tabla(s) de datos corrompidas usando un utilitario de reparación incluido con MySQL. Si el utilitario de reparación no logra restaurar la(s) tabla(s) corrompida(s) a su funcionamiento normal, no todo está perdido; también puede reemplazar la(s) tabla(s) corrompidas con los datos almacenados en una copia de respaldo. En algunos casos la base de datos se podría perder completamente. Por ejemplo, si el PC donde reside su base de datos deja de funcionar y no se puede arreglar, su base de datos actual se perderá, pero sus datos no se perderán para siempre; puede reemplazar el PC dañado con un PC nuevo y restaurar su base de datos con la copia de respaldo.

Cómo reparar tablas

A menudo es posible arreglar una base de datos dañada. MySQL provee un utilitario llamado `myisamchk` para reparar tablas. Si está usando MySQL en el PC de su empleador o en el de su cliente, o a través de una compañía de hospedaje de sitios web, debe contactar al administrador de MySQL para que corra `myisamchk` por usted.

Si usted es el administrador de MySQL, puede correr `myisamchk` usted mismo. Para usar `myisamchk` en Linux/Unix/Mac, siga estos pasos:

1. **Cámbiese al subdirectorio bin en el directorio donde está instalado MySQL.**

   Por ejemplo, digite **cd /usr/local/mysql/bin.**

2. **Detenga el servidor MySQL digitando esta consulta:**

   ```
   mysqladmin -u nombrecuenta -p shutdown
   ```

   donde `-u nombrecuenta` especifica el nombre de la cuenta que usará para conectarse a MySQL.

   La cuenta debe tener permiso para apagar. Si la cuenta no requiere de una contraseña, deje de lado `-p`. Si incluye `-p`, se le pedirá su contraseña.

3. **Digite lo siguiente:**

   ```
   myisamchk -r ruta/nombrebasedatos/nombretabla.MYI
   ```

   Incluya la ruta completa a su directorio de datos, seguida por el nombre de la base de datos, el nombre de la tabla y `.MYI`. Puede usar un asterisco (*) como comodín. Por ejemplo, para reparar todas las tablas en la base de datos `CatalogodeMascotas`, podría digitar

   ```
   myisamchk -r../data/CatalogodeMascotas/*.MYI
   ```

   La opción -r es la opción para recuperar. Después de digitar la instrucción verá como resultado, en la pantalla, cuáles tablas se están revisando.

4. **Arranque el servidor MySQL digitando lo siguiente:**

   ```
   mysqladmin -u nombrecuenta -p start
   ```

Para usar `myisamchk` en Windows, siga estos pasos:

1. **Abra una ventana de comando.**

   Por ejemplo, escoja Start⇨Programs⇨MS-DOS.

2. **Cámbiese al subdirectorio bin en el directorio donde está instalado MySQL.**

   Por ejemplo, digite **cd c:\mysql\bin.**

3. **Detenga el servidor MySQL digitando**

   ```
   mysqladmin -u nombrecuenta -p shutdown
   ```

   donde `nombrecuenta` es el nombre de una cuenta con permiso para apagar. Si la cuenta no requiere de una contraseña, deje por fuera -p. Si incluye -p se le pedirá su contraseña.

4. **Digite lo siguiente:**

   ```
   myisamchk -r ruta/nombrebasedatos/nombretabla.MYI
   ```

   Incluya la ruta completa a su directorio de datos, seguida del nombre de la base de datos, el nombre de la tabla y `.MYI`. Puede usar un asterisco (*) como comodín. La opción -r es la opción para recuperar. Por ejemplo, para reparar todas las tablas en la base de datos `CatalogodeMascotas`, podría digitar

```
myisamchk -r..\data\CatalogodeMascotas\*.MYI
```

Después de digitar esta instrucción verá como resultado, en la pantalla, cuáles tablas se están revisando. Espere a que `myisamchk` termine.

5. **Arranque su servidor MySQL digitando lo siguiente;**

```
mysqladmin -u nombrecuenta -p start
```

Si después de correr este comando su tabla todavía no funciona, pruebe a correr el utilitario myisamchk nuevamente usando la opción -o en lugar de la opción -r. La opción -o es un proceso de reparación más antiguo y mucho más lento que la opción -r, pero maneja algunos casos que la opción -r no puede manejar.

Cómo restaurar de una copia de respaldo

Si después de reparar sus datos no se reactiva el funcionamiento normal de su base de datos, o si su base de datos no está disponible del todo, como en el caso de un fallo del PC, puede reemplazar la(s) tabla(s) de su base de datos actual con la base de datos almacenada en la copia de respaldo. La copia de respaldo contiene una "fotografía" de los datos tal como estaban en el momento en que se hizo la copia. Los cambios hechos a la base de datos desde que se hizo la copia de respaldo no estarán incluidos; usted deberá recrear esos cambios manualmente.

De nuevo, si su acceso a MySQL es por medio del departamento TI o de una compañía de hospedaje de sitios web, debe pedir al administrador de MySQL que restaure su base de datos con la copia de respaldo. Si usted es el administrador de MySQL puede restaurarla usted mismo.

Tal como describí en el Capítulo 4, se construye una base de datos creando la base de datos y luego agregándole tablas. El respaldo que usted crea con el utilitario `mysqldump` es un archivo que contiene todas las instrucciones SQL necesarias para reconstruir todas las tablas, pero no contiene las instrucciones necesarias para crear la base de datos.

Quizá su base de datos no exista del todo, o quizá exista con una o más tablas dañadas. Puede restaurar toda la base de datos o una tabla individual. Siga estos pasos para restaurar una sola tabla:

1. **Si la tabla existe, borre la tabla con la siguiente consulta SQL:**

```
DROP TABLE nombretabla
```

donde nombretabla es la tabla que desea borrar.

2. **Dirija su explorador hacia mysql_send.php.**

Consulte en el Capítulo 4 una descripción de `mysql_send.php`.

3. **Copie la consulta CREATE para la tabla desde el archivo de respaldo al formulario en la ventana del explorador.**

   Por ejemplo, escoja Edit➪Copy y Edit➪Paste.

4. **Digite el nombre de la base de datos en la cual está restaurando la tabla.**

   El formulario muestra dónde debe digitar el nombre de la base de datos.

5. **Haga clic en Submit.**

   Una nueva página web muestra los resultados de la consulta.

6. **Haga clic en New Query.**

7. **Copie una consulta INSERT para la tabla desde el archivo de respaldo al formulario en la ventana del explorador.**

   Por ejemplo, escoja Edit➪Copy y Edit➪Paste.

8. **Digite el nombre de la base de datos en la cual está restaurando la tabla.**

   El formulario muestra dónde debe digitar el nombre de la base de datos.

9. **Haga clic en Submit.**

   Una nueva página web muestra los resultados de la consulta.

10. **Haga clic en New Query.**

11. **Repita los Pasos 7 — 10 hasta que todas las consultas INSERT del archivo de respaldo hayan sido enviadas.**

Si sus consultas `INSERT` para la tabla son tantas que si se enviaran una a una no se terminaría nunca — o si simplemente hay demasiadas tablas — puede enviar todas las consultas en el archivo de respaldo de una sola vez siguiendo estos pasos:

1. **Si alguna de las tablas en el archivo de respaldo existe, bórrela con la siguiente consulta SQL:**

   ```
   DROP TABLE nombretabla
   ```

   donde `nombretabla` es la tabla que desea borrar.

2. **Cámbiese al subdirectorio bin en el directorio donde está instalado MySQL.**

   En Linux/Unix/Mac:

   Digite el comando cd para cambiar al directorio correcto (por ejemplo, digite **cd /usr/local/mysql/bin).**

   En Windows:

   a. Abra una venta de comando.

      Por ejemplo, escoja Start➪Programs➪Accessories➪Command Prompt.

   b. Digite un comando cd para cambiarse al directorio correcto (por ejemplo, digite **cd c:\mysql\bin).**

3. **Digite el comando que envía las consultas SQL en el archivo de respaldo.**

En Linux/Unix/Mac:

Digite

```
mysql -u nombrecuenta -p nombrebasedatos < ruta/nombrearchivorespaldo
```

donde `nombrecuenta` es una cuenta con permiso para crear. Si la cuenta no requiere de contraseña, deje por fuera la -p. Si usa la -p se le pedirá su contraseña. `nombrebasedatos` es la base de datos existente en la cual quiere crear todas las tablas. Use la ruta y el nombre del archivo completos para el archivo de respaldo. Por ejemplo, un comando para restaurar la base de datos `CatalogodeMascotas` podría ser

```
mysql -u root -p CatalogodeMascotas < /usr/archivosrspaldo/Catalogode-
    Mascotas.bak
```

En Windows:

Digite

```
mysql -u nombrecuenta -p nombrebasedatos < ruta\nombrerespaldo
```

donde `nombrecuenta` es una cuenta con permiso para crear. Si la cuenta no requiere de contraseña, deje por fuera la -p. Si usa la -p se le pedirá la contraseña. `nombrebasedatos` es la base de datos existente donde desea construir todas las tablas. Use la ruta y el nombre del archivo completos para el archivo de respaldo. Por ejemplo, un comando para restaurar la base de datos `CatalogodeMascotas` podría ser

```
mysql -u root -p CatalogodeMascotas < c:\mysql\bak\CatalogodeMascotas.bak
```

Las tablas podrían tardar un poco en restaurarse. Espere que el comando termine. Si ocurre algún problema aparecerá un mensaje de error. Si no hay problemas no verá ningún mensaje. Cuando el comando termina, aparece la línea de comando.

Para restaurar sólo tablas seleccionadas del archivo de respaldo, prepare un archivo que contenga únicamente las consultas para las tablas seleccionadas que desea restaurar. Luego siga los Pasos 1 — 3 de la lista anterior. En el paso 3 digite el nombre de la ruta o el nombre del archivo con el subconjunto de consultas que desea en sustitución de todo el archivo de respaldo.

Si la base de datos no existe del todo, debe crearla antes de usar las consultas en el archivo de respaldo para reconstruir todas las tablas. Para restaurar la base de datos cuando no hay queda nada de ella, use los pasos siguientes:

1. **Agregue las siguientes dos líneas a la parte superior del archivo de respaldo:**

```
CREATE DATABASE nombrebasedatos;
use nombrebasedatos;
```

donde `nombrebasedatos` es el nombre de la base de datos que desea restaurar. Por ejemplo, los comandos para la base de datos `CatalogodeMascotas` son

```
CREATE DATABASE CatalogodeMascotas;
use CatalogodeMascotas;
```

Nota: Asegúrese de añadir un punto y coma (;) al final de cada línea.

2. **Cámbiese al subdirectorio bin en el directorio donde está instalado MySQL.**

   En Linux/Unix/Mac:

   Digite un comando cd para cambiarse al directorio correcto (por ejemplo, digite **cd /usr/local/mysql/bin).**

   En Windows:

   a. Abra una ventana de comando.

      Por ejemplo, escoja Start⇨Programs⇨Accessories⇨Command Prompt.

   b. Digite un comando cd para cambiarse al directorio correcto (por ejemplo, digite **cd c:\mysql\bin).**

3. **Digite el comando que envía las consultas SQL en el archivo de respaldo.**

   En Linux/Unix/Mac:

   Digite esto:

   ```
   mysql -u nombrecuenta -p < ruta/nombrearchivorespaldo
   ```

   donde `nombrecuenta` es una cuenta con permiso para crear. Si la cuenta no requiere de una contraseña, deje por fuera la -p. Si usa la -p se le pedirá la contraseña. Use la ruta y el nombre del archivo completos para el archivo de respaldo. Por ejemplo, un comando para restaurar la base de datos podría ser

   ```
   mysql -u root -p < /usr/archivosrespaldo/CatalogodeMascotas.bak
   ```

   En Windows:

   Digite esto:

   ```
   mysql -u nombrecuenta -p < ruta\nombrearchivorespaldo
   ```

   donde `nombrecuenta` es una cuenta con permiso para crear. Si la cuenta no requiere de contraseña, deje por fuera la -p. Si usa la -p se le pedirá la contraseña. Use la ruta y el nombre del archivo completos para el archivo de respaldo. Por ejemplo, un comando para restaurar la base de datos `CatalogodeMascotas` podría ser

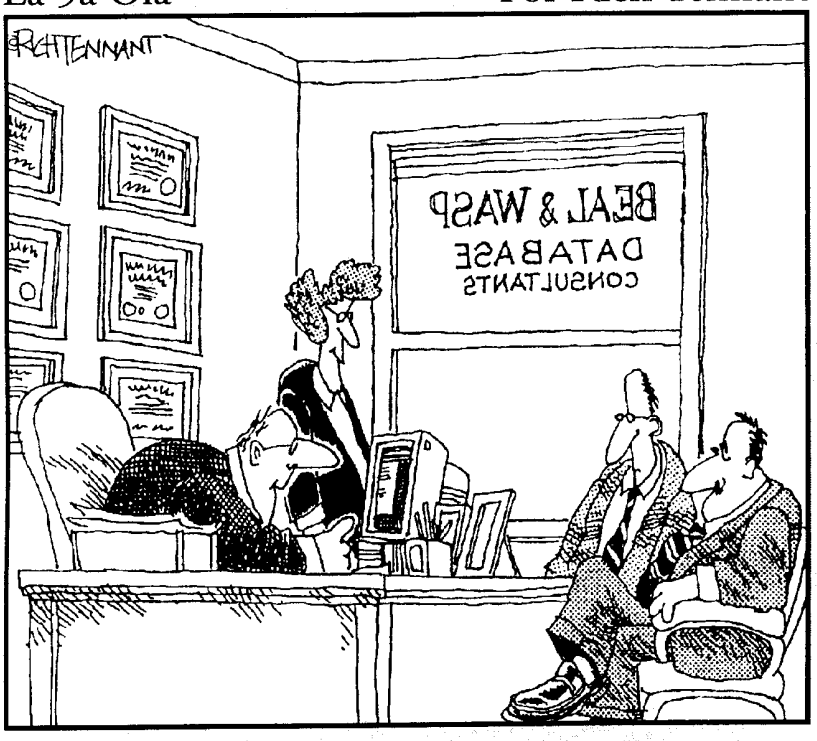
   ```
   mysql -u root -p < c:\mysql\bak\CatalogodeMascotas.bak
   ```

Las tablas podrían tardar un poco en restaurarse. Espere que el comando termine. Si ocurre algún problema aparecerá un mensaje de error. Si no hay problemas no verá ningún mensaje. Cuando el comando haya terminado, aparecerá la línea de comando.

Si ha seguido estos pasos, su base de datos estará restaurada con todos los datos que contenía en el momento en que se hizo la copia de respaldo. Si los datos cambiaron después de que se hizo la copia, los cambios se habrán perdido. Por ejemplo, si se agregaron más datos después de que se hizo la copia de respaldo, los datos nuevos no pueden restaurarse. Si sabe cuáles cambios se hicieron, inclúyalos manualmente en la base de datos restaurada.

Parte III
PHP

La 5a Ola Por Rich Tennant



"Su base de datos ya no se puede reparar, pero antes de darle nuestra recomendación de respaldo, déjeme hacerle una pregunta. ¿Cuántas fichas cree que cabrían en las paredes de su sala de cómputo?

En esta parte. . .

En esta parte, aprenderá cómo usar PHP para su aplicación con base de datos para la Web. Estos son algunos de los temas que se describen:

- Cómo agregar PHP a archivos HTML
- Las características de PHP que son útiles para construir una aplicación con base de datos para la Web
- Cómo usar las características de PHP
- Cómo usar formularios para recopilar información de los usuarios
- Cómo mostrar información de una base de datos en una página web
- Cómo almacenar datos en una base de datos
- Cómo mover información de una página web a otra

Aprenderá todo lo que necesita saber para escribir los programas PHP que usted necesita.

Capítulo 6

PHP General

En este capítulo

- Agregar secciones PHP a los archivos HTML
- Escribir enunciados PHP
- Usar variables PHP
- Comparar valores en las variables PHP
- Documentar sus programas

Los programas son la parte de aplicación de su aplicación con base de datos para la Web. Los programas realizan las tareas. Crean y muestran las páginas web, aceptan y procesan la información que envían los usuarios, almacenan la información en la base de datos, extraen la información de la base de datos y llevan a cabo todas las demás tareas necesarias.

PHP, el lenguaje que usted usa para escribir sus programas, es un lenguaje de scripting diseñado específicamente para usarse en la Web. Es su herramienta para crear páginas web dinámicas. Tiene características diseñadas para ayudarle a programar las tareas que necesitan las aplicaciones web dinámicas.

En este capítulo, describo las reglas generales para escribir programas PHP, reglas que aplican a todos los enunciados PHP. Piense en estas reglas como las reglas generales de gramática y puntuación. En los capítulos restantes de la Parte III, usted aprenderá sobre características y enunciados específicos de PHP y sobre cómo escribir programas PHP para cumplir con tareas específicas.

Agregar una sección PHP a una página HTML

PHP es un socio de HTML (Lenguaje de Marcado de Hipertexto: HyperText Markup Language, en inglés) que expande sus capacidades. Le permite a un programa HTML hacer cosas que no puede hacer por sí mismo. Por ejemplo, los programas en HTML pueden mostrar páginas web, y el HTML tiene características que le permiten a usted formatear dichas páginas web. El HTML también le da la posibilidad

Cómo el servidor web procesa archivos PHP

Cuando un explorador es conducido hacia un archivo HTML regular con una extensión.html o.htm, el servidor web envía el archivo, tal y como está, al explorador. Éste procesa el archivo y muestra la página web descrita por las etiquetas HTML en el archivo. Cuando un buscador es conducido hacia un archivo PHP (con una extensión .php), el servidor web busca las secciones en PHP en el archivo y las procesa, en lugar de simplemente enviar el archivo tal y como está al explorador. Los pasos que el servidor web usa para procesar un archivo PHP son los siguientes:

1. El servidor web empieza a escanear el archivo en modo HTML. Asume que los enunciados están en HTML y los envía al explorador sin procesar.
2. El servidor web continúa en modo HTML hasta toparse con una etiqueta PHP de apertura (<?php).
3. Al toparse con una etiqueta PHP de apertura, el servidor web cambia al modo PHP. A esto se le llama a veces *escapar de HTML*. El servidor web luego asume que todos los enunciados son enunciados PHP y ejecuta los enunciados PHP. Si hay output, éste es enviado por el servidor al explorador.
4. El servidor web continúa en el modo PHP hasta topar con una etiqueta PHP de cierre (?>).
5. Cuando el servidor web encuentra una etiqueta PHP de cierre, regresa al modo HTML. Continúa escaneando y el ciclo se repite desde el Paso 1.

de desplegar gráficos en sus páginas web y reproducir archivos de música. Pero el HTML solo no le permite interactuar con la persona que visualiza la página Web.

El HTML es casi interactivo. O sea, los formularios en HTML permiten a los usuarios digitar la información que la página web está diseñada para recopilar; sin embargo, usted no puede tener acceso a esa información sin usar un lenguaje diferente a HTML. PHP procesa la información de los formularios sin necesidad de un programa aparte, y da espacio para realizar otras tareas interactivas también.

Las etiquetas HTML se usan para incorporar los enunciados en lenguaje PHP a los programas HTML. El archivo del programa tiene una extensión.php. El administrador PHP puede definir otras extensiones, tales como .phtml,.php4 o.php5, aunque.php es la más común. Entonces, en este libro, yo asumiré que la extensión.php es para los programas PHP. Los enunciados en lenguaje PHP se encierran en etiquetas PHP con la siguiente forma:

```
<?php      ?>
```

A veces se puede usar una versión más corta de las etiquetas PHP. Puede intentar usar <? and ?> sin el php. Si las etiquetas cortas están activadas, puede ahorrarse algo de digitación.

PHP procesa todos los enunciados entre las dos etiquetas PHP. Una vez que la sección PHP se ha procesado, se descarta. O bien, si los enunciados PHP producen output, la sección PHP es reemplazada por dicho output. El explorador no ve la sección PHP; sólo su output, si lo hay. Para más información sobre este proceso, revise el cuadro: "Cómo el servidor web procesa archivos PHP".

Como ejemplo, voy a empezar con un programa que despliega `¡Hola, Mundo!` en la ventana del explorador. (Es una especie de tradición: el primer programa que se escribe en cualquier lenguaje es el programa ¡Hola, Mundo! Tal vez usted haya escrito un programa ¡Hola, Mundo! la primera vez que aprendió HTML.) La Lista 6-1 muestra un programa HTML que despliega `¡Hola, Mundo!` en la ventana del buscador.

Lista 6-1: Programa ¡Hola, Mundo! en HTML

```
<html>
<head><title>Hola, Mundo Program</title></head>
<body>
<p>¡Hola, Mundo!
</body>
</html>
```

Si dirige su explorador hacia este programa HTML, verá una página web que dice

```
¡Hola, Mundo!!
```

en la ventana del explorador.

La Lista 6-2 muestra un programa PHP que hace exactamente lo mismo: muestra `¡Hola, Mundo!` en la ventana del buscador.

Lista 6-2: Programa ¡Hola, Mundo! en PHP

```
<html>
<head><title>Hola, Mundo Program</title></head>
<body>
<?php
  echo "<p>¡Hola, Mundo!"
?>
</body>
</html>
```

Si dirige su explorador hacia este programa, muestra exactamente la misma página web que el programa HTML en la Lista 6-1.

No vea el archivo directamente con su explorador. Es decir, no escoja File➪ Open➪Browse en el menú de su explorador para navegar hasta el archivo y hacer clic en él. Debe dirigirse hacia el archivo digitando su URL, como lo describo en el Capítulo 2. Si ve el código PHP aparecer en la ventana del buscador en lugar del output esperado, es probable que usted no se haya dirigido hacia el archivo usando su URL.

En este programa PHP, la sección PHP es

```
<?php
  echo "<p>¡Hola, Mundo!"
?>
```

Las etiquetas PHP encierran sólo un enunciado: un enunciado `echo`. El enunciado `echo` es un enunciado PHP que usará con mucha frecuencia. Simplemente produce como output el texto entre comillas dobles.

No hay ninguna regla que diga que usted debe digitar el PHP en líneas separadas. Se podría incluir perfectamente el PHP en el archivo en una sola línea, como sigue:

```
<?php echo "<p>¡Hola, Mundo!" ?>
```

Cuando la sección PHP se procesa, se reemplaza con el output. En este caso, el output es

```
<p>¡Hola, Mundo!
```

Si reemplaza la sección PHP en la Lista 6-2 con el output anterior, el programa ahora luce exactamente igual que el programa HTML en la Lista 6-1. Si dirige su explorador a cualquiera de los dos programas, verá la misma página web. Si usted observa el código fuente que el explorador ve (en el explorador, escoja Ver⇨Código Fuente), verá el mismo código fuente para ambos programas.

Escribir enunciados PHP

La sección PHP que usted agrega a su archivo HTML consta de un arreglo de enunciados PHP. Cada enunciado PHP es una instrucción para que PHP haga algo. En el programa ¡Hola, Mundo! mostrado en la Lista 6-2, la sección PHP contiene sólo un enunciado PHP simple. El enunciado `echo` ordena a PHP a mostrar como output el texto entre comillas dobles.

Los enunciados PHP terminan con un punto y coma (;). PHP no nota los espacios en blanco ni los finales de líneas. Continúa leyendo un enunciado hasta topar con un punto y coma o la etiqueta PHP de cierre, sin importar cuántas líneas abarque el enunciado. Omitir el punto y coma es un error común, el cual da como resultado un mensaje de error que se ve así:

```
Parse error: expecting `','' or `';'' in /hello.php on line 6
```

Observe que el mensaje de error le indica el número de la línea donde se encontró con problemas. Esta información le ayuda a localizar el error en su programa. Este mensaje de error probablemente significa que se omitió el punto y coma al final de la línea 5.

Le recomiendo escribir sus programas PHP con un editor que enumere las líneas. Si su editor no le permite especificar a cuál línea usted desea ir, tendrá que contar las líneas manualmente desde la parte superior del archivo cada vez que reciba un mensaje de error. Hallará muchos editores que sirven para editar PHP en `phpeditors.linuxbackup.co.uk`.

A veces, un grupo de enunciados se combina para formar un *bloque*. Un bloque se encierra entre llaves ({ }).Un bloque de enunciados se ejecuta en conjunto. Un uso común para los bloques es en un *bloque condicional*, en el cual los enunciados se ejecutan sólo cuando ciertas condiciones son verdaderas. Por ejemplo, usted quizás quiera que su programa haga lo siguiente:

```
if (el cielo esta azul)
{
  poner correa a dragon;
  sacar a dragon a pasear por el parque;
}
```

Estos enunciados se encierran entre llaves para garantizar que se ejecuten como bloque. Si el cielo está azul, tanto `poner la correa a dragon` como `sacar a dragon a pasear por el parque` se ejecutarán. Si el cielo no está azul, ninguno de los enunciados se ejecutará (ni la correa, ni la caminata).

Los enunciados PHP que usan bloques, tales como los enunciados if (los cuales explico en el Capítulo 7), son *enunciados complejos*. PHP lee el enunciado complejo en su totalidad, sin detenerse en el primer punto y coma que encuentre. PHP sabe que probablemente se topará con uno o más bloques, y busca la llave de cierre del último bloque en los enunciados complejos. Observe que hay un punto y coma antes de la llave final. Este punto y coma es obligatorio, pero no hace falta un punto y coma después de la llave final.

Si usted quisiera, podría escribir toda la sección PHP en una sola línea larga, siempre y cuando separara los enunciados con puntos y comas y encerrara los bloques entre llaves. Sin embargo, un programa escrito así sería imposible de leer. Por ende, usted debería poner los enunciados en líneas separadas, excepto en aquellas ocasiones cuando haya enunciados realmente cortos.

Observe que los enunciados dentro de los bloques están sangrados. La sangría no es necesaria para PHP. Se usa estrictamente para facilitar la lectura. Usted debería usar sangrías en los enunciados de un bloque, para que los lectores del script puedan saber más fácilmente dónde empieza y dónde termina un bloque.

En general, a PHP no le importa si las palabras claves del enunciado están en mayúsculas o minúsculas. Echo, echo, ECHO y eCHo son todos iguales para PHP.

Mensajes de error y advertencias

PHP trata de ser de ayuda cuando surgen problemas. Proporciona mensajes de error y advertencias como sigue:

- **Mensaje de error:** Usted recibe este mensaje cuando el programa tiene un problema que no lo deja correr. El mensaje contiene tanta información como sea posible, para así ayudarle a identificar el problema. Un mensaje de error común es

```
Parse error: parse error in
    c:\catalog\test.php on line 6
```

  Con frecuencia, usted recibe este mensaje de error porque ha olvidado un punto y coma, un paréntesis o una llave.

- **Mensaje de advertencia:** Usted recibe este mensaje cuando el programa ve un problema que no es lo suficientemente serio como para evitar impedir que el programa corra. Los mensajes de advertencia no significan que el programa no puede ejecutarse; el programa continúa corriendo. Más bien, los mensajes de advertencia le indican que PHP cree que probablemente haya algo malo. Usted debería identificar el origen de la advertencia y luego decidir si necesita arreglarlo. Generalmente, es así.

- **Aviso:** Usted recibe un aviso cuando PHP ve una condición que podría ser un error o que podría estar perfectamente bien. Los avisos, como las advertencias, no causan que el script deje de correr. Los avisos tienen menor probabilidad de indicar problemas serios que las advertencias. Los avisos sólo le dicen que usted está haciendo algo inusual y es mejor revisar nuevamente lo que está haciendo para asegurarse de que realmente lo quiere hacer.

  Una razón común por la cual podría recibir un aviso es si está haciendo eco de variables que no existen. He aquí un ejemplo de lo que usted podría ver en ese caso:

```
Notice: Undefined variable: age
    in testing.php on line 9
```

Observe que todos los tipos de mensajes indican el nombre del archivo que está causando problemas y el número de la línea donde se encontró el problema.

PHP tiene varios tipos de mensajes de error y advertencia. El tipo que se envía depende del nivel de mensajes de error al cual se haya configurado PHP. Es importante ver todos los mensajes de error, pero tal vez usted no quiera ver todas las advertencias y avisos. A menudo el único problema con un aviso es el desagradable mensaje; el código hace exactamente lo que usted desea que haga. O bien, tal vez prefiera ver los avisos durante el desarrollo pero no una vez que la aplicación esté siendo usada por los clientes.

Para cambiar el nivel de mensajes de error, de manera que su sitio web muestre más o menos mensajes, usted debe ser el administrador PHP. Edite el archivo php.ini en su sistema. Contiene una sección que explica los niveles de mensajes de error y cómo configurarlos. Cambie la línea que establece su nivel de mensajes de error y luego guarde el archivo `php.ini` editado. ***Nota:*** Probablemente tendrá que reiniciar su servidor web antes de que los cambios en el archivo de configuración de PHP surtan efecto.

Si usted no es el administrador PHP de su sistema y no tiene acceso a `php.ini`, puede agregar un enunciado a cualquier programa que establezca el nivel de reporte de mensajes sólo para ese programa. Para fijar el nivel de mensajes de error, agregue el siguiente enunciado al inicio del programa:

```
error_reporting(OPTIONS);
```

OPTIONS es un código que le dice a PHP qué nivel de reporte de errores usar. Para ver todos los errores, use lo siguiente:

```
error_reporting(E_ALL);
```

Para ver todos los errores pero no los avisos, use lo siguiente:

```
error_reporting(E_ALL & ~E_NOTICE);
```

OPTIONS puede ser cualquiera de los códigos descritos en el archivo `php.ini`.

Usar variables PHP

Las variables son recipientes usados para guardar información. Una variable tiene un nombre y, en la variable, se almacena información. Por ejemplo, usted podría nombrar una variable `$edad` y almacenar el número 12 en ella. Después de almacenar información en una variable, se puede usar posteriormente en el programa. Uno de los usos más comunes para las variables es guardar la información que un usuario digita en un formulario.

Dar un nombre a una variable

Al poner nombres a las variables, recuerde las siguientes reglas:

- Todos los nombres de variables llevan el signo de dólar ($) delante de ellas. Esto le dice a PHP que es el nombre de una variable.
- Los nombres de variables pueden ser de cualquier longitud.
- Los nombres de variables pueden incluir letras, números y rayas únicamente.
- Los nombres de variables deben empezar con una letra o una raya. No pueden empezar con un número.
- Las letras mayúsculas y las minúsculas no son equivalentes. Por ejemplo, `$nombre` y `$Nombre` no son la misma variable. Si usted almacena información en `$nombre`, no puede tener acceso a esa información usando el nombre de variable `$Nombre`.

Al denominar variables, use nombres que indiquen claramente qué información está en la variable. Usar nombres de variables como `$var1`, `$var2`, `$A` o `$B` no contribuye con la claridad del programa. Aunque a PHP no le importa qué nombre usted escoja para la variable y no se confundirá, la gente que trate de seguir el programa tendrá muchos problemas recordando cuáles variables guardan cuál información. Nombres de variables como `$nombre`, `$edad` y `$Totaldelpedido` son más descriptivas y útiles.

Crear y asignar valores a las variables

Las variables pueden guardar números o cadenas de caracteres. Para almacenar información en variables, simplemente se usa un signo de igual (=). Por ejemplo, los cuatro siguientes enunciados PHP asignan información a variables:

```
$edad = 12;
$precio = 2.55;
$numero=-2;
$nombre = "Goliat Perez";
```

Observe que la cadena de caracteres está encerrada entre comillas, no así los números. Los detalles sobre cómo usar números y caracteres se describen más adelante en este capítulo. (Consulte: "Trabajar con números" y "Trabajar con cadenas de caracteres".)

Ahora, usted puede usar cualquiera de estos nombres de variables en un enunciado `echo` para ver el valor en esa variable. Por ejemplo, is usa el siguiente enunciado PHP en una sección PHP:

```
echo $edad;
```

el resultado es `12`. Si incluye la siguiente línea en un archivo HTML:

```
<p>Su edad es <?php echo $edad ?>.
```

el resultado en la página web es

```
Su edad es 12.
```

Cada vez que ponga información en una variable que no existía antes, usted crea esa variable. Por ejemplo, suponga que usa el siguiente enunciado PHP:

```
$nombre = "Jorge";
```

Si este enunciado representa la primera vez que usted menciona la variable `$nombre`, este enunciado crea la variable y la establece en `"Jorge"`. Si tiene una configuración previa del enunciado `$nombre` en `"Maria"`, este enunciado cambia el valor de `$nombre` a `"Jorge"`.

También puede eliminar información de una variable. Por ejemplo, el siguiente enunciado quita información de la variable `$edad`:

```
$edad = "";
```

La variable `$edad` existe pero no contiene ningún valor. Eso no quiere decir que `$edad` se establezca en 0 porque cero es un valor. Significa que `$edad` no almacena ninguna información.

Usted puede ir más allá y destruir la variable usando este enunciado:

```
unset($edad);
```

Una vez que se haya ejecutado este enunciado, la variable `$edad` dejará de existir.

Una variable guarda su información para todo el programa, no sólo para una sola sección PHP. Si una variable se establece en `"si"` al inicio de un archivo, seguirá guardando `"si"` al final de la página. Por ejemplo, suponga que su archivo tiene los siguientes enunciados:

```
<p>¡Hola, Mundo!
<?php
    $edad = 15;
    $nombre = "Jaime";
?>
<br>¡Hola otra vez, Mundo!<br>
<?php
    echo $nombre;
?>
```

El enunciado `echo` en la segunda sección PHP mostrará `Jaime`. La página web resultante de estos enunciados es

```
¡Hola, Mundo!
¡Hola otra vez, Mundo!
Jaime
```

Lidiar con avisos

Si usa un enunciado que incluya una variable que no exista, probablemente recibirá un aviso. Dependerá del nivel de mensajes de error al cual esté configurado PHP. Recuerde que los avisos no son lo mismo que los mensajes de error; su aparición no significa que el programa no puede correr: el programa continúa corriendo. Los avisos sólo le dicen que está haciendo algo fuera de lo común y debe revisarlo para asegurarse de que lo que está haciendo es realmente lo que quiere hacer. (Vea el cuadro: "Mensajes de error y advertencias".) Por ejemplo, suponga que usted usa los siguientes enunciados:

```
unset($edad);
echo $edad;
$edad2 = $edad;
```

Podría ver dos avisos: uno para el segundo enunciado y uno para el tercero. Los avisos se verán así:

```
Notice: Undefined variable: edad in testing.php on line 9
```

Suponga que usted definitivamente quiere usar estos enunciados. El programa funciona exactamente como desea que lo haga. Los únicos problemas son los molestos avisos. Puede evitar la aparición de avisos en un programa insertando el signo de arroba (@) en el punto donde el aviso se emitiría. Por ejemplo, puede evadir los avisos generados por los enunciados anteriores, si cambia los enunciados como sigue:

```
unset($edad);
echo @$edad;
$edad2 = @$edad;
```

Si usted es el administrador PHP, puede cambiar el nivel de mensajes de error para que los avisos no aparezcan. Para ver los detalles sobre cómo hacerlo, consulte el cuadro: "Mensajes de error y advertencias", en otra parte de este capítulo.

Usar constantes PHP

Las constantes PHP son muy parecidas a las variables. A las constantes se les da un nombre, y se almacena un valor en ellas. Sin embargo, las constantes son constantes; es decir, el programa no puede cambiarlas. Una vez que usted establece un valor para una constante, permanece igual. Si usó una constante para la edad y la fijó en 29, no se puede variar. ¿No sería maravilloso tener 29 años para siempre?

Las constantes se usan cuando se utiliza información en varios lugares en el programa, y no cambia durante el programa. Es útil establecer una constante para el valor al inicio del programa y usarla a través de todo el programa. Al crear una constante en lugar de una variable, usted se asegura de que no se cambiará accidentalmente. Al asignarle un nombre, usted sabrá instantáneamente de qué información se trata. Y, al establecer una constante una vez al inicio del programa (en vez de usar el valor a lo largo del programa), usted puede cambiar el valor en un lugar si necesita cambiarse, en lugar de perseguirlo por varios lugares en el programa para cambiarlo.

Por ejemplo, usted podría establecer una constante que sea el nombre de la compañía y una constante para la dirección de la compañía, y usarlas donde sea que las necesite. Luego, si la compañía se traslada, usted podría simplemente cambiar el valor en la dirección de la compañía al inicio del programa, y no haría falta hallar todos los lugares en su programa que hicieron eco de la dirección de la compañía para cambiarla.

Las constantes se establecen usando el enunciado `define`. El formato es

```
define("nombredeconstante","valordeconstante");
```

Por ejemplo, para fijar una constante con el nombre de la empresa, use el siguiente enunciado:

```
define("EMPRESA","Tienda de mascotas ABC");
```

Ahora, use la constante en su programa cada vez que necesite el nombre de su compañía:

```
echo Empresa;
```

Cuando se hace un eco de una constante, no puede encerrarla entre comillas. Si lo hace, hará eco del nombre de la constante, y no del valor. Se puede hacer un eco de una constante sin nada, como se muestra en el ejemplo anterior, o encerrándola entre paréntesis.

Los nombres de constantes pueden ser los mismos que se usan para las variables, aunque los nombres de constantes no empiezan con el signo de dólar ($). Por convención, las constantes reciben nombres escritos en mayúsculas, para poder ver fácilmente que son constantes. Sin embargo, a PHP realmente no le importa cómo se denomina la constante. Usted puede almacenar una cadena o un número en ella. El enunciado siguiente está perfectamente bien para PHP:

```
define ("EDAD",29);
```

Eso sí, no espere que la Madre Naturaleza le crea.

Trabajar con números

PHP le permite realizar operaciones aritméticas con números. Las operaciones aritméticas se indican mediante dos números y un operador aritmético. Por ejemplo, un operador es el signo más (+), de modo que usted puede indicar una operación matemática así:

```
1 + 2
```

También puede llevar a cabo operaciones matemáticas con variables que contengan números, como sigue:

```
$n1 = 1;
$n2 = 2;
$sum = $n1 + $n2;
```

La Tabla 6-1 muestra los operadores aritméticos que puede usar.

Suponga que usted definitivamente quiere usar estos enunciados. El programa funciona exactamente como desea que lo haga. Los únicos problemas son los molestos avisos. Puede evitar la aparición de avisos en un programa insertando el signo de arroba (@) en el punto donde el aviso se emitiría. Por ejemplo, puede evadir los avisos generados por los enunciados anteriores, si cambia los enunciados como sigue:

```
unset($edad);
echo @$edad;
$edad2 = @$edad;
```

Si usted es el administrador PHP, puede cambiar el nivel de mensajes de error para que los avisos no aparezcan. Para ver los detalles sobre cómo hacerlo, consulte el cuadro: "Mensajes de error y advertencias", en otra parte de este capítulo.

Usar constantes PHP

Las constantes PHP son muy parecidas a las variables. A las constantes se les da un nombre, y se almacena un valor en ellas. Sin embargo, las constantes son constantes; es decir, el programa no puede cambiarlas. Una vez que usted establece un valor para una constante, permanece igual. Si usó una constante para la edad y la fijó en 29, no se puede variar. ¿No sería maravilloso tener 29 años para siempre?

Las constantes se usan cuando se utiliza información en varios lugares en el programa, y no cambia durante el programa. Es útil establecer una constante para el valor al inicio del programa y usarla a través de todo el programa. Al crear una constante en lugar de una variable, usted se asegura de que no se cambiará accidentalmente. Al asignarle un nombre, usted sabrá instantáneamente de qué información se trata. Y, al establecer una constante una vez al inicio del programa (en vez de usar el valor a lo largo del programa), usted puede cambiar el valor en un lugar si necesita cambiarse, en lugar de perseguirlo por varios lugares en el programa para cambiarlo.

Por ejemplo, usted podría establecer una constante que sea el nombre de la compañía y una constante para la dirección de la compañía, y usarlas donde sea que las necesite. Luego, si la compañía se traslada, usted podría simplemente cambiar el valor en la dirección de la compañía al inicio del programa, y no haría falta hallar todos los lugares en su programa que hicieron eco de la dirección de la compañía para cambiarla.

Las constantes se establecen usando el enunciado `define`. El formato es

```
define("nombredeconstante","valordeconstante");
```

Por ejemplo, para fijar una constante con el nombre de la empresa, use el siguiente enunciado:

```
define("EMPRESA","Tienda de mascotas ABC");
```

Ahora, use la constante en su programa cada vez que necesite el nombre de su compañía:

```
echo Empresa;
```

Cuando se hace un eco de una constante, no puede encerrarla entre comillas. Si lo hace, hará eco del nombre de la constante, y no del valor. Se puede hacer un eco de una constante sin nada, como se muestra en el ejemplo anterior, o encerrándola entre paréntesis.

Los nombres de constantes pueden ser los mismos que se usan para las variables, aunque los nombres de constantes no empiezan con el signo de dólar ($). Por convención, las constantes reciben nombres escritos en mayúsculas, para poder ver fácilmente que son constantes. Sin embargo, a PHP realmente no le importa cómo se denomina la constante. Usted puede almacenar una cadena o un número en ella. El enunciado siguiente está perfectamente bien para PHP:

```
define ("EDAD",29);
```

Eso sí, no espere que la Madre Naturaleza le crea.

Trabajar con números

PHP le permite realizar operaciones aritméticas con números. Las operaciones aritméticas se indican mediante dos números y un operador aritmético. Por ejemplo, un operador es el signo más (+), de modo que usted puede indicar una operación matemática así:

```
1 + 2
```

También puede llevar a cabo operaciones matemáticas con variables que contengan números, como sigue:

```
$n1 = 1;
$n2 = 2;
$sum = $n1 + $n2;
```

La Tabla 6-1 muestra los operadores aritméticos que puede usar.

Tabla 6-1	Operadores aritméticos
Operador	***Descripción***
+	Suma dos números.
-	Resta el segundo número del primero.
*	Multiplica dos números.
/	Divide el primer número entre el segundo.
%	Encuentra el residuo cuando el primer número se divide entre el segundo número. Esto se llama *módulo*. Por ejemplo, en `$a = 13 % 4`, `$a` está establecido en `1`.

Se pueden hacer varias operaciones aritméticas de una sola vez. Por ejemplo, el siguiente enunciado realiza tres operaciones:

```
$resultado = 1 + 2 * 4 + 1;
```

El orden en que se realiza la operación aritmética es importante. Usted puede obtener resultados diferentes, dependiendo de cuál operación se realice primero. PHP multiplica y divide primero, y luego lleva a cabo las sumas y restas. Si todo lo demás es igual, PHP va de izquierda a derecha. En consecuencia, el enunciado anterior establece `$resultado` en *10*, en el siguiente orden:

```
$resultado = 1 + 2 * 4 + 1   (primero, hace la multiplicacion)
$resultado = 1 + 8 + 1       (luego, hace la primera suma a la izquierda)
$resultado = 9 + 1           (despues, hace la suma restante)
$resultado = 10
```

Usted puede cambiar el orden en el cual se lleva a cabo la operación usando paréntesis. La operación aritmética entre paréntesis se realizará primero. Por ejemplo, puede escribir el enunciado anterior con paréntesis, así:

```
$resultado = (1 + 2) * 4 + 1;
```

Este enunciado fija `$resultado en 13`, en el orden siguiente:

```
$resultado = (1 + 2) * 4 + 1  (primero, hace la operacion entre parentesis)
$resultado = 3 * 4 + 1        (luego, hace la multiplicacion)
$resultado = 12 + 1           (despues, hace la suma)
$resultado = 13
```

Como es mejor prevenir que lamentar, es recomendable usar paréntesis siempre que más de una respuesta sea posible.

A menudo, los números con los que trabaja son montos en dólares, tales como precios de productos. Usted quiere que los clientes vean los precios en el formato apropiado en las páginas web. En otras palabras, los montos en dólares siempre

deberían tener dos espacios decimales. Sin embargo, PHP almacena y despliega los números en el formato más eficiente. Si el número es 10.00, se despliega como 10. Para poner los números en el formato adecuado para dólares, puede usar `sprintf`. El enunciado siguiente formatea un número al monto de dólares:

```
$nuevonombrevariable = sprintf("%01.2f", $viejonombrevariable);
```

Este enunciado reformatea el número en *$viejonombrevariable* y lo almacena en el nuevo formato en *$nuevonombrevariable*. Por ejemplo, los siguientes enunciados despliegan el dinero en el formato correcto:

```
$price = 25;
$f_price = sprintf("%01.2f",$price);
echo "$f_price<br>";
```

Verá lo siguiente en la página web:

```
25.00
```

`sprintf` puede hacer otras cosas además de formatear los lugares de los decimales. Para más información sobre cómo usar `sprintf` para formatear valores, consulte el Capítulo 13.

Si desea separar los miles en su número mediante comas, puede usar: `number_format`. El siguiente enunciado crea un formato en dólares con comas:

```
$price = 25000;
$f_price = number_format($price,2);
echo "$f_price<br>";
```

Verá lo siguiente en la página web:

```
25,000.00
```

El `2` en el enunciado `number_format` establece el formato en dos espacios decimales. Usted puede usar cualquier número para obtener cualquier número de espacios decimales.

Trabajar con cadenas de caracteres

Una *cadena de caracteres* es un arreglo de caracteres. Los caracteres son letras, números y signos de puntuación. Cuando un número se usa como carácter, no es más que un carácter almacenado, al igual que una letra. No se puede usar en operaciones aritméticas. Por ejemplo, un número telefónico se almacena como una cadena de caracteres porque sólo necesita almacenarse: no hace falta sumarlo ni multiplicarlo.

Cuando usted almacena una cadena de caracteres en una variable, le indica a PHP dónde empieza y dónde termina la cadena usando comillas dobles o sencillas. Por ejemplo, los dos siguientes enunciados son iguales:

```
$string = "¡Hola, Mundo!";
$string = '¡Hola, Mundo!';
```

Suponga que desea almacenar una cadena como sigue:

```
$string = 'El restaurante Tom's es bueno';
echo $string;
```

Estos enunciados no funcionarán pues, cuando PHP ve la ' (comilla sencilla) después de Tom, piensa que este es el final de la cadena y despliega lo siguiente:

```
El restaurante Tom
```

Usted debe decirle a PHP que interprete la comilla sencilla (') como un apóstrofo y no como el final de una cadena. Puede hacerlo usando una barra inclinada (\) frente a la comilla sencilla. La barra inversa le indica a PHP que la comilla sencilla no tiene ningún significado especial; simplemente es un apóstrofo. Esto se llama *escapar* del carácter. Use los siguientes enunciados para desplegar toda la cadena:

```
$string = 'El restaurante Tom\'s es bueno';
echo $string;
```

Asimismo, cuando encierra una cadena entre comillas dobles, también debe usar la barra invertida delante de cualquier comilla doble en la cadena.

Cadenas entre comillas sencillas versus cadenas entre comillas dobles

Las cadenas de comillas sencillas y las que tienen comillas dobles se manejan en formas diferentes. Las cadenas entre comillas sencillas se almacenan literalmente, exceptuando \', que se almacena como un apóstrofo. En las cadenas entre comillas dobles, las variables y algunos caracteres especiales se evalúan antes de almacenar la cadena. Estas son las diferencias más importantes en el uso de comillas dobles o sencillas al escribir programas:

- **Manejar variables:** Si usted encierra una variable entre comillas dobles, PHP usa el valor de la variable. Sin embargo, si la encierra entre comillas sencillas, PHP usa el nombre literal de la variable. Por ejemplo, si usa los enunciados siguientes:

```
$edad = 12;
$resultado1 = "$edad";
$resultado2 = '$edad';
echo $resultado1;
echo "<br>";
echo $resultado2;
```

el output es

```
12
$edad
```

- **Iniciar una línea nueva:** Los caracteres especiales `\n` le dicen a PHP que empiece una nueva línea. Cuando usa comillas dobles, PHP empieza una línea nueva en `\n`; pero, con comillas sencillas, `\n` es una cadena literal. Por ejemplo, al usar los siguientes enunciados:

```
$cadena1 = "Cadena entre \ncomillas dobles";
$cadena2 = 'Cadena entre \ncomillas sencillas';
```

`cadena1` da el siguiente output

```
Cadena entre
comillas dobles
```

y `cadena2` da este output

```
Cadena entre \ncomillas sencillas
```

- **Insertar un tabulador:** Los caracteres especiales `\t` le dicen a PHP que inserte un tabulador. Cuando se usan comillas dobles, PHP inserta un tabulador en `\t` pero, con comilla sencillas, `\t` es una cadena literal. Por ejemplo, cuando se usan los siguientes enunciados:

```
$cadena1 = "Cadena entre \tcomillas dobles";
$cadena2 = 'Cadena entre \tcomillas sencillas';
```

`cadena1` da el siguiente output

```
Cadena entre    comillas dobles
```

y `cadena2` da este otro output

```
Cadena entre \tcomillas sencillas
```

Las comillas que encierran la cadena entera determinan el tratamiento de variables y caracteres especiales, incluso si hay otro conjunto de comillas dentro de la cadena. Por ejemplo, vea los enunciados siguientes:

```
$numero = 10;
$cadena1 = "Hay '$numero' personas en fila.";
$cadena2 = ' Hay '$numero' personas esperando.';
echo $cadena1,"<br>\n";
echo $cadena2;
```

El output es el siguiente:

```
Hay '10' personas en fila.
Hay "$numero" personas esperando.
```

Juntar cadenas

Usted puede juntar cadenas, mediante un proceso llamado *concatenación*, usando un punto (.). Por ejemplo, puede juntar cadenas con los siguientes enunciados:

```
$cadena1 = 'iHola';
$cadena2 = 'Mundo!';
$unacadena = $cadena1.$cadena2;
echo $unacadena;
```

El enunciado `echo` da el siguiente output

```
iHolaMundo!
```

Observe que no aparece ningún espacio entre `Hola y Mundo`. Esto es porque no se incluyeron espacios en ninguna de las dos cadenas que se unieron. Usted puede agregar un espacio entre las palabras usando el siguiente enunciado de concatenación en lugar del anterior:

```
$unacadena = $cadena1." ".$cadena2;
```

Puede usar.= para agregar caracteres a una cadena existente. Por ejemplo, puede usar los siguientes enunciados en lugar de los enunciados anteriores:

```
$unacadena= "iHola";
$unacadena.= " Mundo!";
echo $unacadena;
```

El enunciado `echo` dará el siguiente output:

```
iHola Mundo!
```

Usted también puede deshacer cadenas. Puede separarlas en un carácter dado o buscar una subcadena en una cadena. Puede usar funciones para realizar estas y otras operaciones en una cadena. Explico estas funciones en el Capítulo 7.

Trabajar con fechas y horas

Las fechas y horas pueden ser elementos importantes en una aplicación con base de datos para la Web. PHP tiene la capacidad de reconocer fechas y horas y manejarlas en forma diferente que las cadenas de caracteres simples. Las fe-

chas y horas se almacenan en el PC en un formato llamado *marca de tiempo*. No obstante, este no es un formato en el cuál ni usted ni yo quisiéramos ver la fecha. PHP convierte las fechas de su anotación en una marca de tiempo que el PC puede entender, y de la marca de tiempo a un formato que sea familiar a las personas. PHP maneja las fechas y horas usando funciones incorporadas.

El formato de marca de tiempo es una Marca de tiempo Unix, el cual es un número entero que representa el número de segundos desde el 1 de enero de 1970 00:00:00 GMT (Greenwich Mean Time) hasta la hora representada por la marca de tiempo. Este formato facilita el cálculo del tiempo entre dos fechas: sólo reste una marca de tiempo de la otra.

Formatear una fecha

La función que usará más a menudo es `date.date` convierte una fecha u hora del formato de marca de tiempo al formato que usted especifique. El formato general es

```
$mifecha = date("formato",$timestamp);
```

`$timestamp` es una variable con una marca de tiempo almacenada. Usted guardó anteriormente la marca de tiempo en la variable, usando una función PHP que describiré posteriormente en esta sección. Si `$timestamp` no se incluye, la hora actual se obtendrá del sistema operativo y se usará. Así, usted puede obtener la fecha de hoy con el siguiente enunciado:

```
$hoy = date("Y/d/m");
```

Si hoy es 10 de agosto, 2003, este enunciado devuelve

```
2003/10/08
```

El *formato* es una cadena que especifica el formato de la fecha que usted desea almacenar en la variable. Por ejemplo, el formato `"yy-m-d"` da como resultado 03-8-10, y `"M.d.yyyy"` da ago.10.2003. La Tabla 6-2 indica algunos de los símbolos que puede usar en la cadena formato. (Para una lista completa de símbolos, consulte la documentación en `www.php.net`.) Las partes de la fecha se pueden separar con guiones (-), puntos (.), diagonales (/) o espacios.

Tabla 6-2 Símbolos para formato de fechas

Símbolo	*Significado*	*Ejemplo*
M	Mes en texto, abreviado	ene
F	Mes en texto, sin abreviar	enero

(continúa)

Tabla 6-2 *(continuación)*

Símbolo	*Significado*	*Ejemplo*
m	Mes en números precedido por ceros	02, 12
n	Mes en números sin cero precedente	1, 12
d	Día del mes; dos dígitos precedidos por ceros	01, 14
j	Día del mes sin el cero precedente	3, 30
l	Día de la semana en texto, sin abreviar	viernes
D	Día de la semana en texto, abreviado	vie
w	Día de la semana en números	De 0 (domingo) a 6 (sábado)
Y	Año en cuatro dígitos	2002
y	Año en dos dígitos	02
g	Hora entre 0 y 12 sin ceros precedentes	2, 10
G	Hora entre 0 y 24 sin ceros precedentes	2, 15
h	Hora entre 0 y 12 precedida por ceros	01, 10
H	Hora entre 0 y 24 precedida por ceros	00, 23
i	Minutos	00, 59
s	Segundos	00, 59
a	am o pm en minúsculas	am, pm
A	AM o PM en mayúsculas	AM, PM

Almacenar una marca de tiempo en una variable

Puede asignar una marca de tiempo con la fecha y hora actuales a una variable con los siguientes enunciados:

```
$hoy = time();
```

Otra forma de almacenar una marca de tiempo actual es mediante el enunciado

```
$hoy = strtotime("today");
```

Puede almacenar marcas de tiempo específicas usando `strtotime` con varias palabras claves y abreviaturas que se parecen mucho al ingles. Por ejemplo, puede crear una marca de tiempo para enero 15, 2003, así:

```
$Fechaclave = strtotime("january 15 2003");
```

`strtotime` reconoce las siguientes palabras y abreviaturas:

- **Nombres de meses:** Los nombres de los doce meses y sus abreviaturas
- **Días de la semana:** Los siete días de la semana y algunas abreviaturas
- **Unidades de tiempo:** Año, mes, quincena, semana, día, hora, minuto, segundo, am, pm
- **Algunas palabras útiles en inglés:** ago, now, last, next, this, tomorrow, yesterday
- **Menos y más:** `+` or `-`
- **Todos los números**
- **Las zonas de tiempo:** Por ejemplo, `gmt` (Greenwich Mean Time), `pdt` (Pacific Daylight Time) y `akst` (Alaska Standard Time)

Puede combinar las palabras y abreviaturas de varias maneras. Todos los siguientes enunciados son válidos:

```
$Fechaclave = strtotime("tomorrow");    # mañana a esta hora
$Fechaclave = strtotime("now + 24 hours");
$Fechaclave = strtotime("last saturday");
$Fechaclave = strtotime("8pm + 3 days");
$Fechaclave = strtotime("2 weeks ago"); # hace dos semanas exactas
$Fechaclave = strtotime("next year gmt"); #1 de hoy en un año
$Fechaclave = strtotime("this 4am");    # 4 AM hoy
```

Si quisiera saber hace cuánto tiempo fue `$Fechaclave`, podría restarla de `$hoy`. Por ejemplo:

```
$tiempotranscurrido = $hoy - $Fechaclave;
```

Esto le da el número de segundos entre la fecha importante y hoy. O use el enunciado

```
$ tiempotranscurrido =(($hoy - $Fechaclave)/60)/60
```

para averiguar el número de horas desde esa fecha clave.

Usar fechas con MySQL

A menudo, usted desea almacenar una fecha en su base de datos MySQL. Por ejemplo, tal vez quiera almacenar la fecha en que un cliente hizo un pedido o la hora en que un miembro se conectó. MySQL también reconoce fechas y horas y las maneja en forma diferente a las cadenas de caracteres simples. No obstante, MySQL no las maneja igual que PHP. Para usar fechas y horas en su aplicación, usted necesita entender tanto cómo PHP maneja las fechas (cosa que describí en las secciones anteriores) y cómo las maneja MySQL.

Comento los tipos de datos `DATE` y `DATETIME` para MySQL en detalle en el Capítulo 3. A continuación, le presento un resumen:

- `DATE`: Las columnas de fechas en MySQL esperan que las fechas tengan el año primero, luego el mes y, por último, el día. El año puede ser `yyyy` o `yy`. El mes puede ser `mm` o `m`. El día puede ser dd o d. Las partes de la fecha se pueden separar con un guión (-), una diagonal (/), un punto (.) o un espacio.
- `DATETIME`: Las columnas de horas en MySQL esperan tanto la fecha como la hora. El fecha se formatea como lo describí en la viñeta anterior. La hora aparece después de la fecha, en el formato `hh:mm:ss`.

Las fechas y horas deben tener el formato correcto de MySQL para almacenarlas en su base de datos. Las funciones PHP pueden usarse para formatear. Por ejemplo, usted puede formatear la fecha de hoy en el formato MySQL con este enunciado:

```
$today = date("Y-m-d");
```

Puede formatear una fecha específica usando el enunciado

```
$importantDate = date("Y.m.d",strtotime("Jan 15 2003"));
```

Luego puede almacenar la fecha formateada en una base de datos con una consulta SQL como sigue:

```
UPDATE Miembro SET createDate="$today"
```

Comparar valores

En los programas a menudo se usan *enunciados condicionales*. Es decir, si algo es verdadero, su programa hace una cosa pero, si no lo es, su programa hace algo diferente. He aquí dos ejemplos de enunciados condicionales:

```
if el usuario es un niño
   muestre el catalogo de juguetes
if el usuario no es un niño
   muestre el catalogo de equipo electronico
```

Para saber cuáles condiciones existen, el programa debe hacer preguntas. Su programa luego realiza la tarea con base en las respuestas. Algunas preguntas (condiciones) que usted podría querer plantear (y las acciones que podría desear que se lleven a cabo) son

- ¿El cliente es un niño? Si es así, despliegue el catálogo de juguetes.
- ¿Cuál producto tiene más ventas? Despliegue el más popular primero.
- ¿Digitó el cliente la contraseña correcta? Si es así, despligue la página web exclusiva para miembros.
- ¿El cliente vive en Ohio? Si es así, despliegue el mapa de las tiendas en Ohio.

Para hacer una pregunta en un programa, usted crea un enunciado que compara valores. El programa prueba el enunciado y determina si el enunciado es falso o verdadero. Por ejemplo, puede formular las preguntas anteriores así

- El cliente es menor de 13 años. ¿Falso o verdadero? Si es verdadero, despliegue el catálogo de juguetes.
- Las ventas del producto 1 son más altas que las del producto 2. ¿Falso o verdadero? Si es verdadero, despliegue el Producto 1 primero; si es falso, despliegue el Producto 2 primero.
- La contraseña del cliente es *secreto*. ¿Falso o verdadero? Si es verdadero, muestre la página web exclusiva para miembros.
- El cliente vive en Ohio. ¿Falso o verdadero? Si es verdadero, despliegue un mapa con la ubicación de las tiendas en Ohio.

Las comparaciones pueden ser bastante simples. Por ejemplo, ¿es el primer valor más grande que el segundo valor? ¿O más pequeño? ¿O igual? Pero a veces hay que fijarse en las cadenas de caracteres para ver si tienen ciertas características en lugar de ver sus valores exactos. Por ejemplo, usted podría querer identificar cadenas que empiecen con S o cadenas que se parezcan a números telefónicos. Para este tipo de comparaciones, se compara una cadena con un patrón, cosa que describo en la sección: "Hacer coincidir cadenas de caracteres con patrones", más adelante en este capítulo.

Hacer comparaciones simples

Las comparaciones simples comparan un valor con otro. PHP ofrece varias formas de comparar valores. La Tabla 6-3 muestra las comparaciones disponibles.

Tabla 6-3	Valores comparados
Comparación	***Descripción***
==	¿Son los dos valores iguales?
>	¿Es el primer valor mayor que el segundo valor?
>=	¿Es el primer valor mayor que o igual al segundo valor?
<	¿Es el primer valor menor que el segundo valor?
<=	¿Es el primer valor menor que o igual al segundo valor?
!=	¿Son los dos iguales diferentes entre sí?
<>	¿Son los dos valores diferentes entre sí?

Usted puede comparar tanto números como cadenas. Las cadenas se comparan alfabéticamente, y todos los caracteres en mayúsculas van antes que los caracteres en minúsculas. Por ejemplo, *SS* viene antes que *Sa*. Los caracteres que son signos de puntuación también tienen un orden, y un carácter se puede considerar mayor que otro. Sin embargo, comparar una coma con un punto no tiene mucho valor práctico.

Las cadenas se coparan con base en su código ASCII (American Standard Code for Information Interchange). En el conjunto de caracteres ASCII, a cada carácter se le asigna un código ASCII que corresponde a un número decimal entre 0 y 127. Cuando se comparan cadenas, se comparan con base en este código. Por ejemplo, el número que representa la coma es 44. El punto corresponde al 46. Por lo tanto, si se compara un punto con una coma, el punto resulta más grande.

Las comparaciones a menudo se usan para ejecutar enunciados sólo bajo ciertas condiciones. Por ejemplo, en el ejemplo siguiente, el bloque de enunciados sólo se ejecuta cuando la comparación `$tiempo == "lluvia"` es cierta:

```
if ( $tiempo == "lluvia" )
{
    sacar sombrilla;
    cancelar el paseo;
}
```

PHP revisa la variable `$tiempo` para ver si el clima es igual a `"lluvia"`. Si es así, PHP ejecuta los dos enunciados. Si `$tiempo` no es igual a `"lluvia"`, PHP no ejecuta los dos enunciados.

El signo de comparación es dos signos de igual juntos (==). Uno de los errores más comunes es usar un sólo signo de igual para una comparación. Un sólo signo de igual pone el valor dentro de la variable. Entonces, un enunciado como if (`$tiempo = "lluvia"`) establecería `$tiempo` en `lluvia`, en lugar de verificar si el tiempo ya igualó a lluvia y, por lo tanto, sería siempre verdadero.

Por ejemplo, aquí hay una solución al problema de programación presentado al comienzo de esta sección. El problema es

```
si el usuario es un niño
   muestre el catalogo de juguetes
si el usuario no es un niño
   muestre el catalogo de equipo electronico
```

Para determinar si un cliente es un adulto, usted compara la edad del cliente con la edad en la cual se considera adulto a un cliente. Debe decidir a qué edad el cliente dejará de interesarse por los catálogos de juguetes y empezará a interesarse más en los catálogos de equipo electrónico. Suponga que decide que 13 parece ser la edad correcta. Entonces, usted pregunta si el cliente es menor que 13 años al comparar la edad del cliente con 13. Si la edad es menor que 13, muestra el catálogo de juguetes; si la edad es 13 o más de 13, muestra el catálogo de equipo electrónico. Estas comparaciones tendrían el siguiente formato:

```
$edad < 13      (¿es la edad del cliente menor que 13?)
$edad >= 13     (¿es la edad del cliente mayor que o igual a 13?)
```

Una forma de programar las acciones condicionales es usar los siguientes enunciados:

```
si ($edad < 13)
     $estatus = "menor";
si ($edad >= 13)
     $estatus = "adulto";
```

Estos enunciados le indican a PHP que compare la edad del cliente con 13. En el primer enunciado, si la edad del cliente es menor que 13, el estatus del cliente se establece en `"menor"`. En el segundo enunciado, si la edad del cliente es igual a o mayor que 13, el estatus del cliente se fija en `"adulto"`. Luego, usted muestra el catálogo de juguetes a los clientes cuyo estatus sea `menor` y muestra el catálogo de equipo electrónico a aquellos cuyo estatus sea `adulto`. Aunque puede escribir estos enunciados `si` más eficientemente, los enunciados mostrados aquí funcionarán bien. En el Capítulo 7 se presenta una descripción completa de los enunciados condicionales.

Hacer coincidir cadenas de caracteres con patrones

A veces, usted necesita comparar cadenas de caracteres para ver si concuerdan con ciertas características, y no si concuerdan con valores exactos. Por ejemplo, podría querer identificar cadenas que empiecen con S o cadenas que tengan números en ellas. Para este tipo de comparaciones, se comparan las cadenas con patrones. Estos patrones son *expresiones regulares*.

Usted probablemente haya usado algún tipo de coincidencia de patrones en el pasado. Por ejemplo, cuando usa un asterisco (*) como comodín al buscar archivos (`dir s*.doc o ls s*.txt`), usted está haciendo patrones coincidir. Así, `c*.txt` es un patrón. Cualquier cadena que empiece con `c` y que termine con la cadena.`txt`, con cualesquiera caracteres entre la `c` y.`txt`, concuerda con el patrón. Las cadenas `cow.txt`, `c3333.txt` y `c3c4.txt` todas coinciden con el patrón. Usar expresiones regulares es sólo una variación más complicada de usar comodines.

El uso más común para concordar patrones en las páginas web es verificar el input de un formulario. Si la información no tiene sentido, usted seguramente no querrá almacenarla en su base de datos. Por ejemplo, si el usuario digita un nombre en un formulario, usted puede chequear si parece un nombre real al comparar patrones. Ya sabe que un nombre consta principalmente de letras y espacios. Otros caracteres válidos podrían ser un guión (-) (como en el nombre *Smith-Kline*) y una comilla sencilla ('): por ejemplo, en O'Hara. Usted puede revisar el nombre al establecer un patrón que sea una cadena sólo con letras, espacios, guiones y comillas sencillas, y luego cotejarlo con el nombre en el patrón. Si el nombre no concuerda (es decir, si contiene caracteres que no están en el patrón, tal como números o un signo de pregunta (?)) no es un nombre real.

Los patrones consisten en caracteres literales y caracteres especiales. Los *caracteres literales* son caracteres normales, sin ningún otro significado especial. Una c es una c sin más significado que ser una de las 27 letras del alfabeto español. Los caracteres especiales tienen significados especiales en el patrón, tal como el asterisco (*) cuando se usa como comodín. La Tabla 6-4 muestra los caracteres especiales usados en los patrones.

Tabla 6-4 Caracteres especiales usados en patrones

Carácter	*Significado*	*Ejemplo*	*Concuerda*	*No concuerda*
^	Inicio de línea	`^c`	casa	mi casa
$	Final de línea	`c$`	tic	stick
.	Cualquier carácter sólo	`..`	Cualquier cadena que contenga por lo menos dos caracteres	a, I
?	Carácter precedente opcional	`lea?n`	lean, len	loan
( )	Agrupa caracteres literales en una cadena que debe concordar exactamente	`l(ea)n`	lean	len, ln

Carácter	*Significado*	*Ejemplo*	*Concuerda*	*No concuerda*
[ ]	Encierra un conjunto de caracteres literales opcionales	l[ea]n	len, lan	lean, ln
-	Representa todos los caracteres entre dos caracteres	m[a-c]n	man, mbn, mcn	mdn, mun, maan
+	Uno o más de los elementos anteriores	puerta [1-3] +	Puerta111, puerta131	puerta, puerta55
*	Cero o más de los elementos anteriores	puerta [1-3]*	puerta, puerta311	puerta4, puerta445
{,}	Los números de inicio y final en un rango de repeticiones	a{2,5}	aa, aaaaa	a, xx3
\	El siguiente carácter es literal	m*n	m*n	men, mean
(| |)	Un conjunto de cadenas alternas	(Tom|Tommy)	Tom, Tommy	Thomas, To

Los caracteres literales y especiales se combinan para crear patrones, los cuales a veces son patrones largos y complicados. Una cadena se compara con el patrón y, si concuerda, la comparación es verdadera. A continuación, hay algunos ejemplos de patrones con un desglose del patrón y muestras de cadenas que concuerdan y otros que no concuerdan:

- `^[A-Z].*` **— Cadenas que empiezan con una letra mayúscula**
  - `^[A-Z]` — Letra mayúscula al inicio de la cadena
  - `.*` — Una cadena de caracteres que tiene uno o más caracteres de longitud

  Cadenas que concuerdan:
  - Oigámoslo otra vez, Sam
  - Yo

  Cadenas que no concuerdan:
  - oigámoslo otra vez, Sam
  - yo

- `Querido (hijo|hermano)` – **Dos cadenas alternas**
  - `Querido` – Caracteres literales
  - `(hijo|hermano)` – Hijo o hermano

  Cadenas que concuerdan:
  - Querido hijo
  - Mi Querido hermano

  Cadenas que no concuerdan:
  - Querido Goliat
  - hijo
- `^[0-9]{5}(\-[0-9]{4})?$` – **Cualquier código postal**
  - `^[0-9]{5}` – Cualquier cadena de cinco números
  - `\-` – literal
  - `[0-9]{4}` – Una cadena de números de cuatro caracteres de longitud
  - `( )?` – Agrupa las dos últimas partes del patrón y las hace opcionales

  Cadenas que concuerdan:
  - 90001
  - 90002 — 4323

  Cadenas que no concuerdan:
  - 9001
  - 12 — 4321
- `^.+@.+\.com$` – **Cualquier cadena con @ incluida que termine en.com**
  - `^.+` – Cualquier cadena de uno o más caracteres al inicio
  - `@` – Una @ (signo de arroba) literal. @ no es un carácter especial
  - `.+` – Cualquier cadena de uno o más caracteres
  - `\.` – Un punto literal
  - `com$` – Una cadena `com` literal al final de la cadena.

  Cadenas que concuerdan:
  - maria@suempresa.com

  Cadenas que no concuerdan:
  - maria@suempresa.net
  - @maria.com

Puede comparar una cadena con un patrón usando `ereg`. El formato general es

```
ereg("patron",cadena);
```

- `Querido (hijo|hermano)` – **Dos cadenas alternas**
  - `Querido` – Caracteres literales
  - `(hijo|hermano)` – Hijo o hermano

  Cadenas que concuerdan:
  - Querido hijo
  - Mi Querido hermano

  Cadenas que no concuerdan:
  - Querido Goliat
  - hijo
- `^[0-9]{5}(\-[0-9]{4})?$` – **Cualquier código postal**
  - `^[0-9]{5}` – Cualquier cadena de cinco números
  - `\-` – literal
  - `[0-9]{4}` – Una cadena de números de cuatro caracteres de longitud
  - `( )?` – Agrupa las dos últimas partes del patrón y las hace opcionales

  Cadenas que concuerdan:
  - 90001
  - 90002 — 4323

  Cadenas que no concuerdan:
  - 9001
  - 12 — 4321
- `^.+@.+\.com$` – **Cualquier cadena con @ incluida que termine en.com**
  - `^.+` – Cualquier cadena de uno o más caracteres al inicio
  - `@` – Una @ (signo de arroba) literal. @ no es un carácter especial
  - `.+` – Cualquier cadena de uno o más caracteres
  - `\.` – Un punto literal
  - `com$` – Una cadena `com` literal al final de la cadena.

  Cadenas que concuerdan:
  - maria@suempresa.com

  Cadenas que no concuerdan:
  - maria@suempresa.net
  - @maria.com

Puede comparar una cadena con un patrón usando `ereg`. El formato general es

```
ereg("patron",cadena);
```

El patrón o la cadena pueden ser literales, como sigue:

```
ereg("[0-9]*","1234");
```

o se puede almacenar en variables, así:

```
ereg($patron,$cadena);
```

Para usar `ereg` para verificar el nombre que un usuario digitó en un formulario, compare el nombre con un patrón, como sigue:

```
ereg("^[A-Za-z' -]+$",$nombre)
```

El patrón en este enunciado hace lo siguiente:

- Usa ^ y $ para indicar el inicio y el final de la cadena. Eso significa que todos los caracteres en la cadena deben concordar con el patrón.
- Encierra todos los caracteres literales permitidos en la cadena entre [ ]. No se permite ningún otro carácter. Los caracteres permitidos son letras mayúsculas y minúsculas, un apóstrofo ('), un espacio en blanco y un guión (-).

  Usted puede especificar un rango de caracteres usando un guión dentro de los corchetes [ ]. Cuando hace eso, como en `A-Z` arriba, el guión no representa un carácter literal. Ya que usted quiere incluir un guión como carácter literal permitido en su cadena, debe agregar un guión que no esté entre otros dos caracteres. En este caso, el guión se incluye al final de la lista de caracteres literales.
- Sigue la lista de caracteres literales entre corchetes [ ] con un +. El signo de más significa que la cadena puede contener cualquier número de los caracteres dentro de [ ] pero debe contener al menos un carácter.

Unir comparaciones con and/or/xor

A veces una comparación es suficiente para verificar una condición pero, a menudo, usted debe plantear más de una pregunta. Por ejemplo, suponga que su compañía ofrece catálogos para diferentes productos en distintos idiomas. Usted debe saber cuál producto desea ver el cliente y en qué idioma necesita verlo. Este es el formato general para un arreglo de comparaciones:

```
comparacion and|or|xor comparacion and|or|xor comparacion and|or|xor...
```

Las comparaciones están conectadas por medio de una de las tres palabras siguientes:

- `and`: Ambas comparaciones son verdaderas.
- `or`: Una de las comparaciones o ambas comparaciones son verdaderas.
- `xor`: Una de las comparaciones es verdadera pero no ambas.

La Tabla 6-5 muestra algunos ejemplos de comparaciones múltiples.

Tabla 6-5 Comparaciones múltiples

Condición	*Es cierta si*
`$cliente == "Smith" or $cliente == "Jones"`	El cliente se apellida Smith o Jones.
`$cliente == "Smith" and $custState =="OR"`	El cliente se apellida Smith, y el cliente vive en Oregón.
`$cliente == "Smith" or $custState == "OR"`	El cliente se apellida Smith o el cliente vive en Oregón o ambas.
`$cliente == "Smith" xor $custState == "OR"`	El cliente se apellida Smith, o el cliente vive en Oregón — pero no ambas.
`$cliente!= "Smith" and $custAge < 13`	El cliente tiene cualquier apellido menos Smith y tiene menos de 13 años de edad.

Usted puede encadenar juntas tantas comparaciones como sea necesario. Las comparaciones que usan `and` se prueban primero, seguidas de las comparaciones que usan `xor` y, finalmente, se prueban las que usan `or`. Por ejemplo, la siguiente es una condición que incluye tres comparaciones:

```
$edad == 200 or $edad == 300 and $nombre == "Goliat"
```

Si el nombre del cliente es Goliat y tiene 300 años de edad, este enunciado es verdadero. El enunciado también es verdadero si el cliente tiene 200 años, independientemente de cuál sea su nombre. Esta condición no es verdadera si el cliente tiene 300 años, pero su nombre no es Goliat. Usted obtiene estos resultados porque el programa chequea la condición como sigue:

1. **Se compara and.** El programa revisa `$edad` para ver si es igual a `300`, y revisa `$nombre` para ver si es igual a `Goliat`. Si ambos concuerdan, la condición es verdadera, y el programa no necesita revisar `or`. Si sólo una o ninguna de las variables es igual a los valores designados, la prueba continúa.
2. **Se compara or.** El programa revisa `$edad` para ver si es igual a `200`. Si es así, la condición es verdadera. Si no lo es, la condición es falsa.

Usted puede cambiar el orden en el cual se hacen las comparaciones usando paréntesis. La palabra entre paréntesis se evalúa primero. Por ejemplo, usted puede reescribir el enunciado anterior con paréntesis como sigue:

```
( $edad == 200 or $edad == 300 ) and $nombre == "Goliat"
```

Los paréntesis cambian el orden en el cual se revisan las condiciones. Ahora, `or` se verifica primero. Esta condición es verdadera si el nombre del cliente

es Goliat y tiene 200 o 300 años. Usted obtiene estos resultados porque el programa chequea la condición como sigue:

1. **Se compara or.** El programa revisa `$edad` para ver si es igual a 200 o 300. Si es así, esta parte de la condición es verdadera. Sin embargo, la comparación en el otro lado de and también debe ser verdadera, de modo que la prueba continúa.
2. **Se compara and.** El programa revisa `$nombre` para ver si es igual a Goliat. Si es así, la condición es verdadera. Si no lo es, la condición es falsa.

Use los paréntesis libremente, incluso cuando cree que conoce el orden de las comparaciones. Usar paréntesis innecesarios no hace ningún daño, pero obtener resultados inesperados de las comparaciones sí.

Si está familiarizado con otros lenguajes, tal como C, quizás haya usado `||` (para `or`) y `&&` (para `and`) en lugar de las palabras. Los símbolos `||` y `&&` funcionan PHP también. El enunciado `$a < $b && $c > $b` es tan válido como el enunciado `$a < $b and $c > $b`. El signo `||` se revisa antes de la palabra `or`, y el signo `&&` se revisa antes de la palabra `and`.

Agregar comentarios a su programa

Los comentarios son notas que están insertadas en el programa mismo. Añadir a sus programas comentarios que describen su propósito y lo que hacen es esencial. Es importante para el factor lotería; o sea, si usted se gana la lotería y se retira a una vida lujosa en la costa francesa, alguien más tendrá que terminar la aplicación. La nueva persona necesita saber qué debe hacer su programa y cómo lo hace. De hecho, los comentarios lo benefician a usted también. Tal vez deba revisar el programa el próximo año, cuando los detalles estén enterrados profundamente en su mente, debajo de un montón de proyectos más recientes.

Use comentarios libremente. PHP los ignora; los comentarios son para los humanos. Puede insertar comentarios en su programa en cualquier parte, siempre y cuando le diga a PHP que son comentarios. El formato para los comentarios es

```
/* texto de comentario
mas texto de comentario */
```

Sus comentarios pueden ser tan largos o tan cortos como lo necesite. Cuando PHP ve el código que indica el inicio de un comentario (`/*`), ignora todo hasta ver el código que indica el final del comentario (`*/`).

El siguiente es un posible formato para los comentarios al inicio de cada programa:

```
/*  nombre:          catalog.php
    descripcion: Programa que muestra descripciones de pro-
                 ductos. Las descripciones se almacenan en una base
                 de datos. Las descripciones de los productos se
                 seleccionan de la base de datos con base en la ca-
                 tegoria que el usuario digita en un formulario.
    escrito por: Lola Designer
    creado:      2/1/02
    modificado:    3/15/02
*/
```

Usted debería usar comentarios a lo largo del programa para describir lo que el programa hace. Los comentarios son especialmente importantes cuando los enunciados del programa son complicados. Use comentarios tal como el siguiente con frecuencia:

```
/* Obtenga la informacion de la base de datos */
/* Verifique si el cliente tiene mas de 18 años de edad */
/* Agregue los cargos por envio al total del pedido */
```

PHP también tiene un formato para comentarios cortos. Usted puede especificar que una sola línea es un comentario usando el signo de libras (#) o dos diagonales (//) de la manera siguiente:

```
# Esta es la linea 1 del comentario
// Esta es la linea 2 del comentario
```

También puede usar # o // en el centro de una línea para señalar el inicio de un comentario. PHP ignorará todo desde el # o // hasta el final de la línea. Esto es útil para comentar un enunciado en particular, como en el ejemplo siguiente:

```
$promedio = $Totalpedido/$nArticulos // compute precio promedio
```

A veces, usted realmente quiere enfatizar un comentario. El siguiente formato hace un comentario más vistoso:

```
#########################################
##  Revise bien esta seccion##
#########################################
```

Los comentarios PHP no se incluyen en el código HTML que se envía al explorador del usuario. El usuario no ve estos comentarios.

Use los comentarios tan frecuentemente como sea necesario en el script para hacerlo más claro. Sin embargo, usar demasiados comentarios es un error. No comente cada línea o todo lo que hace en el script. Si su script está muy lleno de comentarios, los comentarios realmente importantes se perderán en la maraña. Sólo use comentarios para etiquetar las secciones y explicar códigos inusuales o complicados, y no los códigos que sean obvios.

Capítulo 7

Bloques de construcción PHP para programas

En este capítulo

- Hacer eco de output en páginas web
- Asignar valores a variables
- Incrementar variables
- Detener programas y escapar de ellos
- Crear y usar series
- Usar enunciados condicionales
- Construir y usar ciclos para enunciados repetidos
- Usar funciones

Los programas en PHP son una serie de enunciados en un archivo dotada de una extensión que le dice al servidor web que busque las secciones en PHP del archivo. (La extensión generalmente es `.php` o `.phtml`, aunque puede ser cualquier cosa que el servidor deba esperar, según como haya sido configurado.) PHP empieza al inicio del archivo y ejecuta cada enunciado, en orden, conforme los recibe. Los enunciados son los bloques de construcción de los programas en PHP.

Los bloques de construcción básicos son enunciados simples: un solo enunciado seguido de un punto y coma. Un programa simple consta de un arreglo de enunciados simples. Por ejemplo, el programa Hola Mundo descrito en el Capítulo 6 es un programa simple. Sin embargo, los programas que conforman una aplicación con base de datos para la Web no son muy sencillos. Son dinámicos e interactúan tanto con el usuario como con la base de datos. En consecuencia, los programas requieren de bloques de construcción más complejos.

Estas son algunas tareas de programación comunes que requieren de bloques de construcción complejos:

- **Almacenar juntos grupos de valores relacionados:** A menudo usted tiene información que se relaciona, tal como la descripción, la foto y el precio de un producto o una lista de clientes. Almacenar esta información

como un grupo al cual se pueda tener acceso bajo un solo nombre es eficiente y útil. Esta característica de PHP es un arreglo.

- **Configurar enunciados que se ejecuten sólo cuando se cumplan ciertas condiciones:** Los programas deben hacer esto frecuentemente. Por ejemplo, tal vez usted quiera mostrar un catálogo de juguetes a un niño y un catálogo de equipo electrónico a un adulto. Este tipo de enunciados es un enunciado condicional. Los enunciados condicionales PHP son el enunciado `if` y el enunciado `case`.
- **Configurar un bloque de enunciados que se repite:** A menudo, usted necesita repetir enunciados. Por ejemplo, quizás desee crear una lista de todos sus clientes. Para hacerlo, podría usar dos enunciados: uno que entresaque la fila de clientes de su base de datos y otro que almacene el nombre del cliente en una lista. Tendría que repetir estos dos enunciados para cada fila en la base de datos de clientes. La característica que le permite hacer esto es un ciclo. Tres tipos de ciclos son los ciclos `for`, `while` y `do..while`.
- **Escribir bloques de enunciados que se pueden usar muchas veces:** Muchas tareas se realizan en más de una parte de la aplicación. Por ejemplo, usted tal vez quiera recuperar información sobre los productos de la base de datos y desplegarla numerosas veces en una aplicación. Extraer y mostrar la información podría requerir de varios enunciados. Escribir un bloque de enunciados que despliegue la información del producto y usar ese bloque repetidamente, es mucho más eficiente que escribir los enunciados una y otra vez, cada vez que necesita mostrar la información sobre el producto. PHP le permite volver a usar los bloques de enunciados creando una función.

En este capítulo, aprenderá cómo usar los bloques de construcción de programas PHP. Describo los enunciados simples usados más frecuentemente y los enunciados complejos y las variables más útiles. Usted descubrirá cómo armar los bloques de construcción y para qué se usan. Luego, en el Capítulo 8, aprenderá cómo usar estos bloques de construcción para trasladar datos dentro y fuera de una base de datos.

Enunciados simples útiles

Un enunciado simple es un enunciado único seguido de un punto y coma (;). He aquí algunos enunciados simples que se usan en programas PHP:

- Enunciado `echo`: produce output que los exploradores manejan como HTML
- Enunciado de asignación: asigna valores a variables
- Enunciado de incremento: aumenta o disminuye números en las variables
- Enunciado `exit`: detiene la ejecución de su programa
- Llamar funciones: Usa bloques de enunciados almacenados en cualquier lugar de un programa

Comento estos enunciados simples y cuándo usarlos en las siguientes secciones.

Usar enunciados echo

Los enunciados `echo` se usan para producir output. El output de un enunciado `echo` se envía al explorador del usuario, el cual maneja el output como HTML (Lenguaje de Marcado de Hipertexto: HyperText Markup Language).

El formato general de un enunciado echo es

```
echo objetosalida,objetosalida,objetosalida...
```

donde aplican las reglas siguientes:

- El *objeto salida* puede ser un número, una cadena o una variable. Una cadena debe estar encerrada entre comillas. La diferencia entre las comillas dobles y las sencillas se explica en el Capítulo 6.
- Indique tantos *objeto salidas* como necesite.
- Separe cada *objeto salida* con una coma.

La Tabla 7-1 muestra algunas enunciados `echo` y su output. Para los propósitos de la tabla, suponga que `$cadena1` está establecida en `Hola` y `$cadena2`, en `Mundo!`

Tabla 7-1 Enunciados echo

Enunciado echo	*Output*
`echo "Hola";`	`Hola`
`echo 123;`	`123`
`echo " Hola ","Mundo!";`	`HolaMundo!`
`echo Hola Mundo!;`	No es válida; el resultado es un mensaje de error
`echo "Hola Mundo!";`	`Hola Mundo!`
`echo 'Hola Mundo!';`	`Hola Mundo!`
`echo $cadena1;`	`Hola`
`echo $cadena1,$cadena2;`	`HolaMundo!`
`echo "$cadena1 $cadena2";`	`Hola Mundo!`
`echo "Hola",$cadena2;`	`Hola Mundo!`
`echo "Hola"," ",$cadena2;`	`Hola Mundo!`
`echo '$cadena1',"$cadena2";`	`$cadena1World!`

Tabla 7-2 Etapas de la entrega de la página web

Enunciado echo	*Código fuente en HTML*	*Despliegue en la página web*
`echo "Hola Mundo!";`	`Hola Mundo!`	`Hola Mundo!`
`echo "Hola Mundo!";` `echo "Aqui estoy!";`	`Hola Mundo!` `Aqui estoy!`	`Hola Mundo!Aqui estoy!`
`echo "Hola Mundo!\n";` `echo "Aqui estoy!";`	`Hola Mundo!` `Aqui estoy`	`Hola Mundo! Aqui estoy!`
`echo "Hola Mundo! ";` `echo "Aqui estoy!";`	`Hola World! ` `Aqui estoy!"`	`Hola Mundo!` `Aqui estoy!`
`echo "Hola Mundo! \n";` `echo "Aqui estoy!";`	`Hola Mundo! ` `Aqui estoy!"`	`Hola Mundo!` `Aqui estoy!`

La Tabla 7-2 resume las diferencias entre las etapas en la creación de una página web con PHP. Para observar estas diferencias más de cerca, tome en cuenta los dos siguientes enunciados `echo`:

```
echo "Linea 1";
echo "Linea 2";
```

Si usted pone estas líneas en un programa, probablemente esperaría que la página web mostrara lo siguiente:

```
Linea 1
Linea 2
```

Sin embargo, este no es el output que obtendrá. La página web realmente se vería así:

```
Linea 1Linea 2
```

Si se fija en el código fuente para la página web, verá exactamente lo que se envía al explorador, que es esto:

```
Linea 1Linea 2
```

Observe que esta la línea que aparece como output y se envía al explorador contiene exactamente los caracteres a los cuales usted hizo eco: nada más, ni nada menos. Las cadenas de caracteres a las cuales usted hizo eco no contenían espacios, de modo que ningún espacio aparece entre las líneas.

Además, observe que las dos líneas aparecen en la misma línea. Si desea empezar una línea nueva, debe enviar una señal indicando el inicio de una nueva línea. Para señalar que una línea nueva inicia aquí en PHP, debe hacer eco del carácter especial `\n`. Cambie los enunciados `echo` a lo siguiente:

```
echo "linea 1\n";
echo "linea 2";
```

Ahora obtuvo lo que quería, ¿verdad? Bueno, en realidad, no. Ahora se ve lo siguiente en la página web:

```
linea 1 linea 2
```

Si se fija en el código fuente, verá esto:

```
linea 1
linea 2
```

Entonces, el carácter `\n` hizo lo suyo: empezó una línea nueva en el output. Sin embargo, el HTML muestra el resultado en la página web como una línea. Si usted desea que el HTML despliegue dos líneas, debe usar una etiqueta, tal como la etiqueta `<br>`. Entonces, cambie el carácter PHP especial al final de la línea a una etiquete HTML, así:

```
echo "linea 1<br>";
echo "linea 2";
```

Ahora verá lo que quiere en la página web:

```
linea 1
linea 2
```

Si se fija en el código fuente para este resultado, verá esto:

```
linea 1<br>linea 2
```

Use `\n` libremente. De lo contrario, su código fuente en HTML tendrá líneas realmente largas. Por ejemplo, si hace `echo` de un formulario largo, éste podría consistir en una sola línea larga en el código fuente, aunque se vea bien en la página web. Use `\n` para separar el código fuente en HTML en líneas razonables. Tomarse el tiempo adicional de agregar estas separaciones valdrá la pena si debe resolver algún problema en la página web que no luce como usted esperaba. Es mucho más fácil examinar el código fuente si no mide un kilómetro de largo.

Usar enunciados de asignación

Los enunciados de asignación son enunciados que asignan valores a las variables. El nombre de la variable se indica a la izquierda del signo de igual; el valor por asignar a la variable se detalla a la derecha del signo de igual. Este es el formato general:

```
$nombrevariable = valor;
```

El valor puede ser un valor único o una combinación de valores, incluyendo los valores en las variables. Una variable puede guardar números o caracteres pero

no ambos a la vez. Por lo tanto, un valor no puede ser una combinación de números y caracteres. Los siguientes son enunciados de asignación válidos:

```
$numero = 2;
$numero = 2+1;
$numero = (2 - 1) * (4 * 5) -17;
$numero2 = $numero + 3;
$cadena = "Hola Mundo";
$cadena2 = $cadena." Otra vez!";
```

Si combina números y cadenas en un valor, no obtendrá un mensaje de error; simplemente no obtendrá el output esperado. Por ejemplo, los siguientes enunciados combinan números y cadenas:

```
$numero = 2;
$cadena = "Hola";
$combinado = $numero + $cadena;
$combinado2 = $numero.$cadena;
echo $combinado;
echo <br>;
echo $combinado2;
```

El output de estos enunciados es

```
2          ($cadena es evaluado como 0)
2Hola      ($numero es evaluado como un caracter)
```

Usar enunciados de incremento

A menudo una variable se usa como `contador`. Por ejemplo, suponga que desea asegurarse de que todos vean el logotipo de su compañía; por eso, lo muestra tres veces. Usted establece una variable en 0. Cada vez que despliegue el logotipo, agrega 1 a la variable. Cuando e valor de la variable llegue a 3, usted sabrá que es momento de dejar de mostrar el logotipo. Los siguientes enunciados muestran el uso de un `contador`:

```
$contador=0;
$contador = $contador + 1;
echo $contador;
```

Estos enunciados darían como output 1. Puesto que los contadores se usan tan a menudo, PHP le proporciona atajos. Los siguientes enunciados tienen el mismo efecto que los enunciados anteriores:

```
$contador=0;
$contador++;
echo $contador;
```

Este enunciado echo también da como output 1 porque ++ suma 1 al valor actual del `$contador`. O bien, puede usar el siguiente enunciado:

```
$contador--;
```

Este enunciado resta 1 del valor actual del `$contador`.

A veces, usted podría querer hacer una operación aritmética diferente. Puede usar cualquiera de los atajos siguientes:

```
$contador+=2;
$contador-=3;
$contador*=2;
$contador/=3;
```

Estos enunciados suman 2 al `$contador`, restan 3 del `$contador`, multiplican el `$contador` por 2 y dividen el `$contador` entre 3, respectivamente.

Usar exit

En ocasiones, usted quiere que el programa deje de ejecutarse, que simplemente se detenga en algún punto en medio del programa. Por ejemplo, si el programa encuentra un error, a menudo usted prefiere que se detenga en lugar de continuar con más enunciados. El enunciado exit detiene el programa. No se ejecuta ningún otro enunciado después del enunciado `exit`. El formato de un enunciado `exit` es

```
exit("mensaje");
```

`mensaje` es el mensaje que se despide como output cuando el programa se despide. Por ejemplo, podría usar el enunciado

```
exit("El programa esta retirandose");
```

También puede parar el programa con el enunciado `die`, como sigue:

```
die("El programa esta muriendo");
```

El enunciado die es igual que el enunciado `exit`. `die` es solamente otro nombre para `exit`. A veces, es mucho más divertido decir die.

Usar llamados a funciones

Las funciones son bloques de enunciados que realizan ciertas tareas específicas. Imagine las funciones como mini-programas o subprogramas. El bloque de enunciados se almacena bajo un nombre de función, y usted puede ejecutar el bloque de enunciados en cualquier lugar que desee al llamar a la función por su nombre. (Puede consultar los detalles sobre cómo usar funciones en la sección "Usar funciones", más adelante en este capítulo.)

Usted puede llamar a una función indicando su nombre seguido de paréntesis, como sigue:

```
nombrefuncion();
```

Por ejemplo, podría tener una función que extraiga todos los nombres de los clientes que viven en cierto estado de la base de datos y despliegue los nombres en una lista en el formato `apellido, nombre`. Usted escribe los enunciados que llevan a cabo estas tareas y los almacena como una función bajo el nombre `conseguirnombres`. Luego, cuando usted llama a la función, debe especificar cuál estado. Puede usar el siguiente enunciado en cualquier lugar de su programa para obtener la lista de nombres de clientes que viven en ese estado en particular, el cual en este caso es California:

```
conseguir_nombres('CA');
```

El valor entre paréntesis se entrega a la función para que ésta sepa cuál estado usted está especificando. Esto es pasar el valor. Se puede pasar una lista de valores.

PHP brinda muchas funciones incorporadas. Por ejemplo, en el Capítulo 6, comento una función incorporada llamada `unset`. Usted puede anular una variable llamada `$testvar` usando este llamado de función:

```
unset($testvar);
```

Usar arreglos PHP

Los arreglos son variables complejas. Un arreglo almacena un grupo de valores bajo un único nombre de variable. Un arreglo es útil para almacenar valores relacionados. Por ejemplo, usted puede almacenar información sobre una camisa (tal como tamaño, color y precio) en un solo arreglo llamado `$infocamisa`. Usted puede fácilmente manejar, tener acceso a y modificar la información en un arreglo. Por ejemplo, PHP tiene varios métodos para clasificar un arreglo.

Las siguientes secciones le dan los detalles sobre los arreglos.

Cómo crear arreglos

La forma más simple de crear un arreglo es asignar un valor a una variable con corchetes ([ ]) al final de su nombre. Por ejemplo, suponiendo que usted no haya usado una referencia para `$mascotas` anteriormente en el programa, el enunciado siguiente crea un arreglo llamado `$mascotas`:

```
$mascotas[1] = "dragon";
```

En este punto, se ha creado un arreglo llamado `$mascotas`, la cual tiene sólo un valor: `dragon`. Luego, use los enunciados siguientes:

```
$mascotas[2] = "unicornio";
$mascotas[3] = "tigre";
```

Ahora el arreglo `$mascotas` contiene tres valores: `dragon`, `unicornio` y `tigre`.

Un arreglo se puede considerar como una lista de parejas de claves y valores. Para obtener un valor en particular, usted especifica la clave entre corchetes. En el arreglo anterior, las claves son números: 1, 2 y 3. Sin embargo, también puede usar palabras como claves. Por ejemplo, los enunciados siguientes crean un arreglo de capitales estatales:

```
$capitales['CA'] = "Sacramento";
$capitales['TX'] = "Austin";
$capitales['OR'] = "Salem";
```

Usted puede usar atajos en lugar de escribir enunciados de asignación separados para cada número. Un atajo usa los siguientes enunciados:

```
$mascotas[] = "dragon";
$mascotas[] = "unicornio";
$mascotas[] = "tigre";
```

Cuando usted crea un arreglo usando este atajo, a los valores se les asignan automáticamente claves que son números de serie, empezando con el número 0. Por ejemplo, el enunciado siguiente

```
echo "$mascotas[0]";
```

envía el siguiente resultado:

```
dragon
```

El primer valor en un arreglo con un índice numerado es 0, a menos que usted deliberadamente lo establezca en un número diferente. Un error común al trabajar con series es pensar que el primer número es 1, y no 0.

Un atajo mucho mejor aún es usar el enunciado siguiente:

```
$mascotas = array( "dragon","unicornio","tigre");
```

Este enunciado crea la misma serie que el atajo anterior. Asigna números como claves, empezando con 0. Usted puede usar un enunciado parecido para crear series con palabras como claves. Por ejemplo, el enunciado siguiente crea el arreglo de capitales estatales:

```
$capitales = array( "CA" => "Sacramento", "TX" => "Austin",
                    "OR" => "Salem" );
```

Cómo visualizar arreglos

Se puede hacer eco del valor de un arreglo así:

```
echo $capitales['TX'];
```

Si usted incluye el valor del arreglo en un enunciado `echo` más largo que esté encerrado entre comillas dobles, tal vez deba encerrar el nombre del valor del arreglo entre llaves, así:

```
echo "La capital de Texas es {$capitales['TX']}<br>";
```

Puede ver la estructura y los valores de cualquier serie usando el enunciado `print_r`. Para mostrar el arreglo `$capitales`, use el enunciado siguiente:

```
print_r($capitales);
```

Este enunciado `print_r` dará el siguiente output:

```
Array
(
  [CA] => Sacramento
  [TX] => Austin
  [OR] => Salem
)
```

Este output muestra la clave y los valores para cada elemento en el arreglo.

El resultado aparecerá en la página web con HTML, lo cual significa que se desplegará en una línea larga. Para ver el resultado en la web en el formato útil que describo aquí, envíe etiquetas HTML que indiquen al explorador que muestre el texto tal como lo recibe, sin cambiarlo, usando los enunciados siguientes:

```
echo "<pre>";
print_r($capitales);
echo "</pre>";
```

Cómo quitar valores de los arreglos

A veces, usted necesita eliminar completamente un valor de un arreglo. Por ejemplo, suponga que tiene la siguiente serie:

```
$mascotas = array( "dragon, "unicornio", "tigre",
                   "lora", "escorpion" );
```

Esta serie cuenta con cinco valores. Ahora usted decide que ya no desea vender escorpiones en su tienda de mascotas, por lo cual usa el enunciado siguiente para tratar de retirar `escorpion` del arreglo:

```
$mascotas[4] = "";
```

Aunque este enunciado establece `$mascotas[4]` en una cadena vacía, no lo remueve del arreglo. Usted todavía tiene un arreglo con cinco valores; uno de los valores es vacíol. Para eliminar completamen‘e un elemento del arreglo, necesita usar destruirlo, como sigue:

```
unset($mascotas[4]);
```

Ahora su arreglo sólo cuenta con cuatro valores.

Cómo ordenar arreglo

Una de las características más útiles de los arreglos es que PHP las puede clasificar. PHP almacena originalmente los elementos de un arreglo en el orden en que usted los crea. Si usted muestra el arreglo entero sin cambiar el orden, los elementos aparecerán en el orden en que fueron creados. A menudo, usted desea cambiar ese orden. Por ejemplo, tal vez desee mostrar el arreglo en orden alfabético por valor o por ciave.

PHP puede distribuir los arreglos de varias formas. Para distribuir un arreglo que tiene números como claves, use el enunciado `sort` como sigue:

```
sort($mascotas);
```

Este enunciado clasifica según los valores y les asigna claves nuevas que son los números apropiados. Los valores se clasifican por números primero, seguidos por las mayúsculas y, finalmente, las minúsculas. Por ejemplo, considere el arreglo `$mascotas`, creado en la sección anterior:

```
$mascotas[0] = "dragon";
$mascotas[1] = "unicornio";
$mascotas[2] = "tigre";
```

Después de usar el enunciado `sort` siguiente

```
sort($mascotas);
```

el arreglo se transforma en

```
$mascotas[0] = "dragon";
$mascotas[1] = "tigre";
$mascotas[2] = "unicornio";
```

Si usa `sort()` para ordenar un arreglo con palabras como claves, las claves cambiarán a números, y las palabras claves se perderán.

Para distribuir series que tengan palabras como claves, use el enunciado `asort` como sigue:

```
asort($capitales);
```

Este enunciado clasifica las capitales por valores, y retiene la clave original para cada valor en lugar de asignar una clave numérica. Por ejemplo, considere el arreglo de capitales estatales creada en la sección anterior:

```
$capitales['CA'] = "Sacramento";
$capitales['TX'] = "Austin";
$capitales['OR'] = "Salem";
```

Después del siguiente enunciado de clasificación

```
asort($capitales);
```

el arreglo se convierte en

```
$capitales['TX'] = "Austin";
$capitales['CA'] = "Sacramento";
$capitales['OR'] = "Salem";
```

Observe que las claves permanecieron con el valor cuando los elementos fueron reordenados. Ahora los elementos están en orden alfabético, y la clave estatal correcta todavía está con la capital estatal correspondiente. Si las claves hubieran sido números, los números estarían ahora en un orden diferente. Por ejemplo, si el arreglo original fuera

```
$capitales[1] = "Sacramento";
$capitales[2] = "Austin";
$capitales[3] = "Salem";
```

después de un enunciado *asort*, la nueva serie sería

```
$capitales[2] = Austin
$capitales[1] = Sacramento
$capitales[3] = Salem
```

Dudo que usted quiera usar `asort` en un arreglo con claves numéricas.

Hay varios otros enunciados `sort` que ordenan de otras formas. La Tabla 7-3 incluye todos los enunciados `sort` disponibles.

Tabla 7-3 Maneras en que se pueden ordenar las series

Enunciado sort	*Qué hace*
`sort(`*`$nombrearreglo`*`)`	Ordena según valor; asigna números nuevos como claves
`asort(`*`$nombrearreglo`*`)`	Ordena según valor; mantiene la misma clave
`rsort(`*`$nombrearreglo`*`)`	Ordena según valor en orden inverso; asigna números nuevos como claves
`arsort(`*`$nombrearreglo`*`)`	Ordena según valor en orden inverso; mantiene la misma clave
`ksort(`*`$nombrearreglo`*`)`	Ordena según clave
`krsort(`*`$nombrearreglo`*`)`	Ordena según clave en orden inverso
`usort(`*`$nombrearreglo, nombrefuncion`*`)`	Ordena según función (vea: "Usar funciones", más adelante en este capítulo)

Cómo obtener valores de un arreglo

Usted puede recuperar cualquier valor individual de un arreglo al acceder a él directamente. He aquí un ejemplo:

```
$CAcapital = $capitales['CA'];
echo $CAcapital ;
```

El output de estos enunciados es

```
Sacramento
```

Si usted usa un elemento que no existe en el arreglo en un enunciado, aparecerá un aviso. (Puede leer sobre los avisos en el Capítulo 6.) Por ejemplo, suponga que usa el siguiente enunciado:

```
$CAcapital = $capitales['CAx'];
```

Si el arreglo `$capitales` existe pero ningún elemento tiene la clave `CAx`, verá el siguiente aviso:

```
Notice: Undefined index: CAx in d:\testserie.php on line 9
```

Recuerde que un aviso no causa que el script se detenga. Los enunciados después del aviso continuarán ejecutándose. Pero, como no se ha puesto ningún valor a `$CAcapital`, los enunciados `echo` siguientes reproducirán un espacio en blanco. Puede evitar que el aviso aparezca usando el símbolo @:

```
@$CAcapital = $capitales['CAx'];
```

Usted puede obtener varios valores de un arreglo simultáneamente usando el enunciado `list` o todos los valores de un arreglo usando el enunciado `extract`.

El enunciado `list` obtiene los valores de un arreglo y los pone en variables. Los siguientes enunciados incluyen un enunciado `list`:

```
$infocamisa = arreglo ("grande", "azul", 12.00);
sort ($infocamisa);
list($primervalor,$segundovalor) = $infocamisa;
echo $primervalor,"<br>";
echo $segundovalor,"<br>";
```

La primera línea crea el arreglo `$infocamisa`. La segunda línea ordena el arreglo. La tercera establece dos variables llamadas `$primervalor` y `$segundovalor` y copia los dos primeros valores en `$infocamisa` en las dos variables nuevas, como si usted hubiera usado los dos enunciados

```
$primervalor=$infocamisa[0];
$segundovalor=$infocamisa[1];
```

El tercer valor en `$infocamisa` no se copia en una variable porque sólo hay dos variables en el enunciado `list`. El output de los enunciados `echo` es

```
azul
grande
```

Observe que el resultado está en orden alfabético, y no en el orden en el cual los valores fueron introducidos. Está en orden alfabético porque el arreglo se ordenó después de haberse creado.

Usted puede extraer todos los valores de un arreglo con palabras como claves usando `extract`. Cada valor se copia en una variable con un nombre que se corresponde con la clave. Por ejemplo, los siguientes enunciados obtienen toda la información de `$infocamisa` y hacen eco de ella:

```
extract($infocamisa);
echo "su talla es $talla; su color es $color; su precio es
        $precio";
```

El output de estos enunciados es

```
su talla es grande; su color es azul; su precio es 12.00;
```

Cómo moverse por un arreglo

A menudo usted querrá hacer algo a cada valor de un arreglo. Tal vez quiera hacer eco de cada valor, almacenar cada valor en la base de datos o sumarle seis a cada valor en el arreglo. En jerga técnica, moverse por todos y cada uno de los valores de un arreglo se conoce como iteración. A veces, también se le llama atravesar. Estas son dos maneras de moverse por un arreglo:

- Manualmente: Mueva el puntero de un valor del arreglo a otro
- Usar `foreach`: Muévase automáticamente por el arreglo, desde el inicio hasta el final, de valor en valor

Moverse manualmente por un arreglo

Usted puede moverse por un arreglo manualmente usando el puntero. Para hacerlo, piense en el arreglo como una lista. Imagine al puntero señalando un valor en la lista. El puntero permanece en el valor hasta que usted lo mueva. Después de moverlo, se queda ahí hasta que lo mueva de nuevo. Puede mover el puntero mediante las siguientes instrucciones:

- `current($nombrearreglo)`: Se refiere al valor que está actualmente debajo del puntero; no mueve el puntero
- `next($nombrearreglo)`: Mueve el puntero hasta el valor que está después del valor actual
- `previous($nombrearreglo)`: Mueve el puntero hasta el valor que está antes de la ubicación actual del puntero
- `end($nombrearreglo)`: Mueve el puntero hasta el último valor en el arreglo
- `reset($nombrearreglo)`: Mueve el puntero hasta el primer valor en el arreglo

Los siguientes enunciados lo mueven manualmente por un arreglo con las capitales estatales:

```
$valor = current ($capitales);
echo "$valor<br>";
$valor = next ($capitales);
echo "$valor<br>";
$valor = next ($capitales);
echo "$valor<br>";
```

A menos que usted haya movido el puntero anteriormente, éste se ubica en el primer elemento cuando usted empieza a moverse por el arreglo. Si piensa que el puntero del arreglo puede haber sido movido anteriormente en el script o si su output del arreglo parece empezar en alguna parte del centro, use el enunciado `reset` antes de empezar a moverse, como sigue:

```
reset($capitales);
```

Al usar este método para moverse por un arreglo, usted necesita un enunciado de asignación y un enunciado `echo` para cada valor en el arreglo: en este caso, para cada uno de los 50 estados de los Estados Unidos. El output es una lista de todas las capitales estatales.

Este método le da flexibilidad. Se puede mover por un arreglo de cualquier manera: no sólo de valor en valor. Puede moverse hacia atrás, ir directamente al final, saltar un valor de por medio usando dos enunciados next seguidos, o cualquier método que le sea útil. Sin embargo, si quiere moverse por todo el arreglo de principio a fin, de valor en valor, PHP brinda `foreach`, el cual hace exactamente lo que necesita de forma más eficiente. `foreach` se describe en la siguiente sección.

Usar foreach para moverse por un arreglo

`foreach` se mueve por el arreglo de valor en valor y ejecuta el bloque de enunciados usando cada valor en el arreglo. El formato general es

```
foreach( $nombrearreglo as $nombreclave => $nombrevalor  )
{
    bloque de enunciados;
}
```

Complete la información siguiente:

- *nombrearreglo:* El nombre del arreglo por la cual usted se está moviendo
- *nombreclave:* El nombre de la variable donde desea almacenar la clave. El nombreclave es opcional. Si no escribe `$nombreclave =>`, el valor se pondrá en `$nombrevalor`.
- *nombrevalor:* El nombre de la variable donde desea almacenar el valor

Por ejemplo, el siguiente enunciado `foreach` se mueve por el arreglo de muestra de las capitales estatales y repite una lista:

```
$capitales = array ( "CA" => "Sacramento", "TX" => "Austin",
                     "OR" => "Salem" );
ksort ($capitales);
foreach ( $capitales as $estado => $ciudad )
{
    echo "$ciudad, $estado<br>";
}
```

Los enunciados anteriores dan el siguiente output en la página web:

```
Sacramento, CA
Salem, OR
Austin, TX
```

Usted puede usar la siguiente línea en lugar de la línea foreach en los enunciados anteriores:

```
foreach ( $capitales as $ciudad )
```

Cuando usa este enunciado `foreach`, sólo la ciudad aparecerá en el output. Entonces, usted puede usar el siguiente enunciado `echo`:

```
echo "$ciudad<br>";
```

El output con estos cambios es

```
Sacramento
Salem
Austin
```

Cuando `foreach` empieza a moverse por un arreglo, mueve el puntero al principio del arreglo. Usted no necesita volver a establecer un arreglo antes de moverse por el con `foreach`.

Arreglos multidimensionales

En las secciones anteriores de este capítulo, describo arreglos que son una sola lista de parejas de claves y valores. Sin embargo, en algunas ocasiones, tal vez usted desee guardar valores con más de una clave. Por ejemplo, suponga que desea almacenar los precios de estos productos juntos en una variable:

- camisa, 20.00
- pantalones, 22.50
- manta, 25.00
- colcha, 50.00
- lámpara, 44.00
- alfombra, 75.00

Puede almacenar estos productos en un arreglo como sigue:

```
$preciosproductos ['camisa'] = 20.00;
$preciosproductos ['pantalones'] = 22.50;
$preciosproductos ['manta'] = 25.00;
$preciosproductos ['colcha'] = 50.00;
$preciosproductos ['lampara'] = 44.00;
$preciosproductos ['alfombra'] = 75.00;
```

Su programa puede fácilmente registrar esta serie cada vez que necesite saber el precio de un artículo. Pero suponga que tiene 3 000 productos. Su programa necesitaría rebuscar entre 3 000 productos para encontrar el que tenga camisa o alfombra como clave.

Observe que la lista de productos y precios incluye una gran variedad de productos, los cuales pueden clasificarse en grupos: ropa, ropa de cama y muebles. Si usted clasifica los productos, el programa sólo tendría que registrar una clasificación para encontrar el precio correcto. Clasificar los productos sería mucho más eficiente. Puede clasificar los productos al poner los precios en un arreglo multidimensional, como sigue:

```
$preciosproductos ['ropa']['camisa = 20.00;
$preciosproductos ['ropa']['pantalones'] = 22.50;
$preciosproductos ['ropa de cama']['manta'] = 25.00;
$preciosproductos ['ropa de cama']['colcha'] = 50.00;
$preciosproductos ['muebles']['lampara'] = 44.00;
$preciosproductos ['muebles']['alfombra'] = 75.00;
```

Este tipo de arreglo es un arreglo multidimensional, ya que es como un arreglo de series. La Figura 7-1 muestra la estructura de `$preciosproductos` como un arreglo de series. La figura muestra que `$preciosproductos` tiene tres parejas de claves y valores. Las claves son ropa, ropa de cama y muebles. El valor para cada clave es un arreglo con dos parejas de claves y valores. Por ejemplo, el valor para la clave ropa es un arreglo con dos parejas de claves y valores: camisa/20.00 y pantalones/22.50.

$preciosproductos	**clave**	**valor**	
		clave	*valor*
	ropa	camisa	20.00
		pantalones	22.50
	ropa de cama	manta	25.00
		colcha	50.00
	muebles	lampara	44.00
		alfombra	75.00

Figura 7-1: un arreglo de series.

`$preciosproductos` es una arreglo bidimensional. PHP también puede entender series multidimensionales de cuatro, cinco, seis o más niveles de profundidad. Sin embargo, a mí me empieza a doler la cabeza cuando trato de comprender un arreglo con más de tres niveles de profundidad. La posibilidad de confusión aumenta cuando el número de dimensiones aumenta.

Usted puede obtener los valores de un arreglo multidimensional usando los mismos procedimientos que se utilizan con un arreglo unidimensional. Por ejemplo, puede tener acceso a un valor directamente con esta enunciado:

```
$preciocamisa = $preciosproductos['ropa']['camisa'];
```

También puede repetir el valor:

```
echo $preciosproductos['ropa']['camisa'];
```

Sin embargo, si usted combina el valor entre comillas dobles, debe usar llaves para encerrar el nombre de la variable. El $ que inicia el nombre de la variable debe estar inmediatamente después de {, sin ningún espacio, como sigue:

```
echo "El precio de una camisa es \${$preciosproductos['ropa']['camisa']}";
```

Observe la diagonal inversa (\) frente al signo de dólar ($). La diagonal inversa le dice a PHP que $ es un signo de dólares literal y no el inicio del nombre de una variable. El output es

```
El precio de una camisa es $20
```

Usted puede moverse por un arreglo multidimensional usando enunciados `foreach` (descritos en la sección anterior). Necesita un enunciado `foreach` para cada serie. Un enunciado `foreach` está dentro de otro enunciado foreach. Poner enunciados dentro de otros enunciados se llama anidar.

Como un arreglo bidimensional, tal como `$preciosproductos`, contiene dos series, hacen falta dos enunciados `foreach` para moverse por ella. Los enunciados siguientes toman los valores del arreglo multidimensional y las presentan en una tabla en HTML:

```
echo "<table border=1>";
foreach( $preciosproductos as $categoria )
{
   foreach( $categoria as $producto => $precio )
   {
     $f_price = sprintf("%01.2f", $precio);
     echo "<tr><td>$producto:</td><td>\$$f_price</td></tr>";
   }
}
echo "</table>";
```

La Figura 7-2 muestra la página web producida por estos enunciados en PHP.

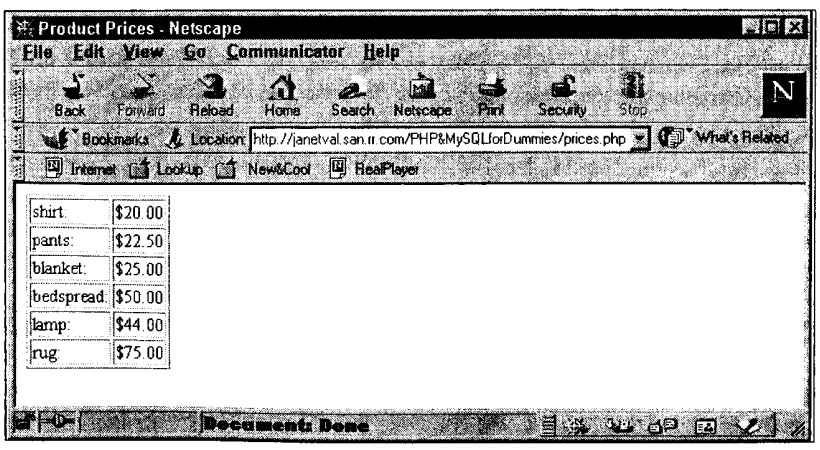


Figura 7-2: Salida en la página web para el arreglo multidimensional.

Así es como el programa interpreta estos enunciados:

1. Da como output la etiqueta `table`.
2. Entresaca la primera pareja de clave y valor en el arreglo `$preciosproductos` y almacena el valor en la variable `$categoria`. El valor es un arreglo.
3. Extrae la primera pareja de clave y valor en el arreglo `$categoria`. Almacena la clave en `$producto` y almacena el valor en `$precio`.
4. Da el formato correcto al valor en `$precio` para dinero.
5. `Hace echo` de una fila en la tabla para el producto y su precio.
6. Va a la siguiente pareja de clave y valor en el arreglo `$categoria`.
7. Formatea el precio y repite la siguiente fila de la tabla para el producto y su precio.
8. Como no hay más parejas de clave y valor en `$categoria`, el enunciado `foreach` interno termina.
9. Va a la siguiente pareja de clave y valor en el enunciado `foreach` externo. Pone el siguiente valor en `$categoria`, el cual es un arreglo.
10. Repite el procedimiento en los Pasos 2 — 9 hasta llegar a la última pareja de clave y valor en la última serie de `$categoria`. Termina el enunciado `foreach` interno. El enunciado `foreach` externo termina.
11. Da como output la etiqueta `/table` para terminar la tabla.

En otras palabras, el enunciado `foreach` externo empieza con la primera pareja de clave y valor en el arreglo. La clave es ropa, y el valor de esta pareja es un arreglo que se pone en la variable `$categoria`. El enunciado `foreach` interno luego se mueve por el arreglo en `$categoria`. Cuando llega a la última pareja de clave y valor en `$categoria`, acaba. El programa regresa entonces al ciclo externo, el cual continúa con la segunda pareja de clave y valor... y así sucesivamente hasta que el enunciado `foreach` externo llegue al final del arreglo.

Enunciados condicionales útiles

Un enunciado condicional ejecuta un bloque de enunciados sólo cuando se cumplen ciertas condiciones. Estos son dos tipos de enunciados condicionales útiles:

- enunciado `if`: Establece una condición y la prueba. Si la condición es verdadera, se ejecuta el bloque de enunciados.
- enunciado `switch`: Establece una lista de condiciones alternativas. Somete a prueba la condición para determinar si es verdadera y ejecuta el bloque de enunciados apropiado.

Describo estos enunciados en más detalle en las dos siguientes secciones.

Usar enunciados if

Un enunciado `if` pregunta si ciertas condiciones existen. Un bloque de enunciados se ejecutará dependiendo de cuáles condiciones se cumplan. El formato general de un enunciado condicional `if` es

```
if ( condicion ... )
{
   bloque de enunciados
}
elseif  ( condicion ... )
{
   bloque de enunciados
}
else
{
   bloque de enunciados
}
```

El enunciado `if` consta de tres secciones:

- `if`: Esta sección es obligatoria. Prueba la condición.
  - **Si la condición es verdadera:** El bloque de enunciados se ejecuta. Una vez que los enunciados se hayan ejecutado, el programa continúa con los siguientes enunciados, después del enunciado condicional; si el enunciado condicional contiene secciones `elseif` o `else`, el programa las pasa por alto.
  - **Si la condición no es verdadera:** El bloque de enunciados no se ejecuta. El programa pasa a la siguiente instrucción, la cual puede ser una instrucción `elseif`, una `else` o la siguiente instrucción después del enunciado condicional `if`.
- `elseif`: Esta sección es opcional. Prueba una condición. Usted puede usar más de una sección `elseif` si lo desea.
  - **Si la condición es verdadera:** El bloque de enunciados se ejecuta. Una vez ejecutado el bloque de enunciados, el programa continúa con el siguiente enunciado después del enunciado condicional; si el enunciado `if` contiene secciones `elseif` adicionales o una sección `else`, el programa las pasa por alto.
  - **Si la condición no es verdadera:** El bloque de enunciados no se ejecuta. El programa pasa a la siguiente instrucción, la cual puede ser `elseif`, `else` o la siguiente instrucción después del enunciado condicional `if`.
- `else`: Esta sección es opcional. Sólo se permite una sección `else`. Esta sección no prueba una condición; en cambio, ejecuta el bloque de enunciados. Si el programa ha entrado a esta sección, significa que la sección if y todas las secciones `elseif` no son verdaderas.

Cada sección del enunciado condicional `if` prueba una condición que consiste en una o más comparaciones. Una comparación hace una pregunta que puede ser falsa o verdadera. Algunas condiciones son

```
$a == 1;
$a < $b
$c != "Hello"
```

La primera comparación pregunta si `$a` es igual a 1; la segunda pregunta si `$a` es más pequeña que `$b`; la tercera comparación pregunta si `$c` no es igual a `"Hello"`. Usted puede usar dos o más comparaciones en una condición conectándolas con `and`, `or` o `xor`. Comento en detalle cómo comparar valores y usar más de una comparación en el Capítulo 6. El siguiente ejemplo usa las tres secciones del enunciado condicional if. Suponga que usted tiene versiones en alemán, francés, italiano e inglés del catálogo de sus productos. Desea que su programa muestre la versión correcta del idioma con base en dónde vive un cliente. Los enunciados siguientes fijan una variable a la versión correcta del catálogo (dependiendo del país donde vive el cliente) y establecen un mensaje en el idioma correcto. Entonces, usted podrá mostrar un mensaje en el idioma apropiado.

```
if ($pais == "Alemania" )
{
   $version = "aleman";
   $mensaje = " Sie sehen unseren Katalog auf Deutsch";
}
elseif ($pais == "Francia" )
{
   $version = "frances";
   $mensaje = " Vous verrez notre catalogue en francais";
}
elseif ($pais == "Italia" )
{
   $version = "italiano";
   $mensaje = " Vedrete il nostro catalogo in Italiano";
}
else
{
  $version = "ingles";
  $mensaje = "You will see our catalog in English";
}
echo "$mensaje<br>";
```

El enunciado condicional `if` procede como sigue:

1. **Compara la variable $pais con "Alemania".** Si son iguales, `$version` se establece en "alemán", `$mensaje` se fija en alemán y el programa salta hasta el enunciado echo. Si `$pais` no es igual a `Alemania`, `$version` y $mensaje no se establecen, y el programa pasa a la sección `elseif`.
2. **Compara la variable $pais con "Francia".** Si son iguales, `$version` y `$mensaje` se establecen, y el programa pasa al enunciado echo. Si `$pais` no es igual a Francia, `$version` y `$mensaje` no se fijan, y el programa pasa a la segunda sección `elseif`.

3. **Compara la variable $pais con "Italia".** Si son iguales, `$version` se fija en `"italiano"`, y el programa pasa al enunciado echo. Si `$pais` no es igual a Italia, `$version` y `$mensaje` no se establecen, y el programa pasa a la sección else.
4. `$version` **se establece en inglés.** y `$mensaje` se fija en inglés. El programa continúa con el enunciado `echo`.

Observe que sólo se hace `echo` del mensaje en este ejemplo. Sin embargo, la variable `$version` se almacena porque versión es información útil que se puede usar más adelante en el programa.

Cuando el bloque a ejecutarse por cualquier sección del enunciado condicional `if` contiene un solo enunciado, las llaves no son necesarias. Veamos: si el ejemplo anterior hubiese tenido un único enunciado en los bloques, como sigue:

```
if ($pais == "Francia")
{
   $version = "frances";
}
```

Podría escribirla como sigue:

```
if ($pais == "Francia" )
   $version = "frances";
```

Este atajo puede ahorrarle algo de digitación pero, cuando se usan varios enunciados `if`, puede ocasionar confusión.

Usted puede tener un enunciado condicional if dentro de otro enunciado condicional `if`. Poner un enunciado dentro de otro se llama anidar. Por ejemplo, suponga que necesita contactar a todos los clientes que viven en Chile. Planea enviarles un mensaje electrónico a los que tienen dirección electrónica y una carta a aquellos que no la tengan. Puede identificar los grupos de clientes usando las siguientes enunciados `if` anidados:

```
if ( $paiscliente == "Chile" )
{
   if ( $direccionemail != "" )
   {
     $metodocontacto = "carta";
   }
   else
   {

     $metodocontacto = "email";
   }
}
else
{
   $metodocontacto = "no se necesita";
}
```

Estos enunciados primero revisan si el cliente vive en Chile. Si el cliente vive allí, el programa busca una dirección electrónica. Si la dirección electrónica está en blanco, el método de contacto se fija en `carta`. Si la dirección electrónica no está en blanco, el método de contacto es `email`. Si el cliente no vive en Chile, la sección `else` establece el método de contacto para indicar que el cliente no será contactado del todo.

Usar enunciados switch

En la mayoría de las situaciones, los enunciados condicionales `if` funcionan mejor. No obstante, a veces usted tiene una lista de condiciones y desea ejecutar diferentes enunciados para cada una de las condiciones. Por ejemplo, suponga que su programa calcula el impuesto de ventas. ¿Cómo maneja las variaciones en el impuesto de ventas en diferentes lugares? El enunciado `switch` fue diseñado para este tipo de situaciones.

El enunciado `switch` prueba el valor de una variable y ejecuta el bloque de enunciados para el valor que concuerda con la variable. El formato general es

```
switch ( $nombrevariable )
{
  case valor :
     bloque de enunciados;
     break;
  case valor :
     bloque de enunciados;
     break;
  ...
  default:
     bloque de enunciados;
     break;
}
```

El enunciado `switch` prueba el valor de `$nombrevariable`. El programa luego pasa a la sección case para ese valor y ejecuta los enunciados hasta llegar al enunciado break o al final del enunciado `switch`. Si no hay una sección case para el valor de `$nombrevariable`, el programa ejecuta la sección `default`. Usted puede usar tantas secciones case como necesite. La sección `default` es opcional. Si usa una sección `default`, se acostumbra ponerla al final, aunque puede colocarse en cualquier parte.

Los siguientes enunciados establecen el impuesto de ventas para los diferentes estados de los Estados Unidos

```
switch ( $estadocliente )
{
  case "OR" :
     $tasaimpuestoventas = 0;
     break;
  case "CA" :
     $tasaimpuestoventas = 1.0;
     break;
  default:
     $impuestoventas = .5;
     break;
}
$impuestoventas = $costototalpedido * $tasaimpuestoventas;
```

En este caso, el impuesto de ventas para Oregón es 0, el de California es el 100 por ciento, y para los demás estados es el 50 por ciento. El enunciado `switch` mira el valor de `$estadocliente` y pasa a la sección que concuerde con el valor. Por ejemplo, si `$estadocliente` es `TX`, el programa ejecuta la sección default y establece `$tasaimpuestoventas` en.5. Después del enunciado `switch`, el programa computa `$impuestoventas` en.5 veces el costo del pedido.

Los enunciados `break` son esenciales en la sección case. Si una sección `case` no incluye un enunciado `break`, el programa no deja de ejecutarse al final de la sección `case`. El programa continúa ejecutando los enunciados después del final de la sección case, hasta la siguiente sección case, y continúa hasta llegar al final del enunciado `switch` (o tal vez un enunciado `break` en una sección `case` posterior).

La última sección case en un enunciado `switch` en realidad no requiere de un enunciado `break`. Éste puede omitirse. Sin embargo, es una buena idea incluirlo por razones de claridad.

Usar ciclos

Los ciclos, los cuales se usan con frecuencia en los programas, establecen un bloque de enunciados que se repite. A veces, el ciclo se repite un número específico de veces. Por ejemplo, un ciclo para hacer eco de todas las capitales estatales de los Estados Unidos necesita repetirse 50 veces. Algunas veces, el ciclo se repite hasta que cierta condición exista. Por ejemplo, un ciclo que despliega información del producto para todos los productos necesita repetirse hasta que haya mostrado todos los productos, sin importar cuántos productos haya. Estos son los tres tipos de ciclos:

- Ciclo `for` básico: Establece un contador; repite un bloque de enunciados hasta que el contador alcance un número específico.
- Ciclo `while`: Fija una condición; verifica la condición; y, si es verdadera, repite un bloque de enunciados.

- Ciclo `do..while`: Establece una condición; ejecuta un bloque de enunciados; verifica la condición; si la condición es verdadera, repite el bloque de enunciados.

Describo cada uno de estos ciclos en detalle en las secciones siguientes.

Usar ciclos for

Los ciclos for más básicos se basan en un contador. Usted fija los valores inicial y final para el contador, además de cómo el contador se incrementa. El formato general es

```
for (valorinicial;condicionfinal;incremento)
{
    bloque de enunciados;
}
```

Rellene los siguientes valores:

- *valorinicial:* Un enunciado que establece una variable para que sea su contador y la fija a su valor inicial. Por ejemplo, el enunciado $i=1; establece $i como la variable contador y la establece en igual a 1. Frecuentemente, la variable contador empieza con 0 ó 1. El valor inicial puede ser una combinación de números (2 + 2) o una variable.
- *condicionfinal:* Un enunciado que establece su valor final. Siempre y cuando este enunciado sea verdadero, el bloque de enunciados sigue repitiéndose. Cuando este enunciado no es verdadero, el ciclo termina. Por ejemplo, el enunciado $i<10; fija el valor final para el ciclo en 10. Cuando $i es igual a 10, el enunciado ya no es verdadero (porque $i ya no es menor que 10), y el ciclo deja de repetirse. El enunciado puede incluir variables, como `$i<$tamaño;`.
- *incremento:* Un enunciado que incrementa su contador. Por ejemplo, el enunciado $i++; suma 1 a su contador al final de cada bloque de enunciados. Usted puede usar otros enunciados de incremento, tales como `$I+=1;` o `$i--;`.

El ciclo básico for establece una variable (por ejemplo, una variable llamada $i,) que es un contador. Esta variable tiene un valor durante cada ciclo. La variable $i se puede usar en el bloque de enunciados que se repite. Por ejemplo, el siguiente ciclo simple despliega `¡Hola, Mundo!` tres veces:

```
for ($i=1;$i<=3;$i++)
{
 echo "$i. ¡Hola, Mundo!<br>";
}
```

Los enunciados en el bloque no necesitan tener sangría. A PHP no le importa si están sangrados o no. Sin embargo, usar sangría en los bloques le ayuda mucho a entender el programa.

El output de estos enunciados es

```
1. ¡Hola, Mundo!
2. ¡Hola, Mundo!
3. ¡Hola, Mundo!
```

Los ciclos `for` son particularmente útiles para moverse en ciclo por un arreglo. Suponga que usted tiene un arreglo de nombres de clientes y desea desplegarlos todos. Puede hacerlo fácilmente con un ciclo:

```
for ($i=0;$i<100;$i++)
{
  echo "$nombresclientes[$i]<br>";
}
```

El output despliega una página web con una lista de todos los nombres de clientes, uno en cada línea. En este caso, usted sabe que tiene cien nombres de clientes; pero imagine que no sabe cuántos clientes hay en esta lista. Le pregunta a PHP cuántos valores hay en el arreglo y usar ese valor en su ciclo `for`. Por ejemplo, puede usar los siguientes enunciados:

```
for ($i=0;$i<sizeof($nombresclientes);$i++)
{
   echo "$nombresclientes[$i]<br>";
}
```

Ciclos for avanzados

La estructura de un ciclo for es bastante flexible y le permite construir ciclos para casi cualquier propósito. Un ciclo for tiene este formato general:

```
for ( enunciados iniciales;
      enunciados condicionales;
      enunciados finales)
{
     bloque de enunciados;
}
```

Donde

- Los enunciados iniciales se ejecutan una vez al inicio del ciclo.
- Los enunciados condicionales se prueban para cada iteración del ciclo.

✔ Los enunciados finales se ejecutan una vez al final del ciclo.

Cada una de las secciones de enunciados se separa con un punto y coma (;). Cada sección puede contener tantos enunciados como sean necesarios, separados por comas. Cualquier sección puede estar vacía.

El siguiente ciclo tiene enunciados en las tres secciones:

```
for ($i=0,$j=1;$t<=4;$i++,$j++)
{
  $t = $i + $j;
  echo "$t<br>";
}
```

El output de estos enunciados es

```
1
3
5
```

El ciclo se ejecuta en el siguiente orden:

1. La sección inicial, la cual contiene dos enunciados, se ejecuta; $i se establece en 0 y $j se establece en 1.
2. La sección condicional que contiene un enunciado se evalúa. ¿Es $t menor que o igual a 4? Sí, entonces el enunciado es verdadero. El ciclo continúa ejecutándose.
3. Los enunciados en el bloque de enunciados se ejecutan. $t se convierte en igual a $i más $j, lo cual es 0 + 1, lo cual es igual a 1. Después, $t se repite para dar el output 1.
4. La sección final, la cual contiene dos enunciados, se ejecuta: $i++ y $j++. Se le suma uno a $i para que sea igual a 1, y 1se suma a $j para que sea igual a 2.
5. La sección condicional se evalúa. ¿Es $t menor que o igual a 4? Como $t es igual a 1 en este punto, el enunciado es verdadero. El ciclo continúa ejecutándose.
6. Los enunciados en el bloque de enunciados se ejecutan. $t se hace igual que $i más $j, que es 1 + 2, lo cual equivale a 3. Luego, $t se repite para dar el output 3.
7. La sección final, que contiene dos enunciados, se ejecuta: $i++ y $j++. Se suma uno a $i para que sea igual a 2, y 1 se suma a $j para que sea igual a 3.
8. La sección condicional se evalúa. ¿Es $t menor que o igual a 4? Como ahora $t es igual a 3, el enunciado es verdadero. El ciclo continúa ejecutándose.
9. Los enunciados en el bloque de enunciados se ejecutan. $t se hace igual a $i más $j, lo cual es 2 + 3, lo cual equivale a 5. Después, $t se repite para dar como output 5.
10. La sección final, la cual contiene dos enunciados, se ejecuta: $i++ y $j++. Se suma uno a $i para que sea igual a 3, y 1 se suma a $j para que se igual a 4.
11. La sección condicional se evalúa. ¿Es $t menor que o igual a 4? Como $t ahora equivale a 5, el enunciado no es verdadero. El ciclo no continúa ejecutándose. El ciclo termina, y el programa continúa hacia el siguiente enunciado después del final del ciclo.

Observe que el valor final es `sizeof($nombresclientes)`. Este enunciado averigua el número de valores en el arreglo y usa ese número. De este modo, su ciclo se repite exactamente el número de veces que corresponde con el número de valores en el arreglo.

El primer valor en un arreglo con un índice numérico es 0, a no ser que usted deliberadamente lo establezca en un número diferente. Un error común cuando se trabaja con series es pensar que el primer número es 1 en vez de 0.

Usar ciclos while

Un ciclo `while` continúa repitiéndose siempre y cuando ciertas condiciones sean verdaderas. El ciclo funciona como sigue:

1. Usted establece una condición.
2. La condición se prueba al inicio de cada ciclo.
3. Si la condición es verdadera, el ciclo se repite. Si la condición no es verdadera, el ciclo se detiene.

El formato general de un ciclo while es

```
while ( condicion )
{
   bloque de enunciados
}
```

Una condición es cualquier expresión que pueda ser falsa o verdadera. Las comparaciones, tales como las que se presentan a continuación, a menudo se usan como condiciones. (Para información detallada sobre cómo usar comparaciones, consulte el Capítulo 6).

```
$prueba <= 10
$prueba1 == $prueba2
$a == "si" y $b != "si"
$apellido != "Perez"
```

Siempre y cuando la condición sea verdadera, el ciclo se repetirá. Cuando la condición resulte falsa, el ciclo se detendrá. Los siguientes enunciados configuran un ciclo `while` que registra un arreglo en busca de un cliente apellidado `Perez`:

```
$clientes = serie( "Huang", "Perez", "Herrera" );
$variabledeprueba = "no";
$k = 0;
while ( $variabledeprueba  != "si" )
{
  if ($clientes[$k] == "Perez" )
  {
```

```
    $variabledeprueba  = "si";
    echo "Perez<br>";
  }
  else
  {
    echo "$clientes[$k], no Perez<br>";
  }
  $k++;
}
```

Estos enunciados muestran lo siguiente en una página web:

```
Huang, no Perez
Perez
```

El programa ejecuta los enunciados anteriores de esta manera:

1. Establece las variables antes de empezar el ciclo: `$clientes` (un arreglo con tres valores), `$variabledeprueba` (una variable de prueba establecida en "no") y `$k` (una variable contador establecida en 0).
2. Empieza el ciclo probando si `$variabledeprueba != "si"` es verdadera. Como `$variabledeprueba` se fijó en "no", el enunciado es verdadero, de modo que el ciclo continúa.
3. Prueba el enunciado `if`. ¿Es `$clientes[$k] == "Perez"` verdadero? En este punto, `$k` es 0, por lo cual el programa verifica `$clientes[0]`. Dado que `$clientes[0]` es "Huang", el enunciado no es verdadero. Los enunciados en el bloque if no se ejecutan, y el programa pasa al enunciado `else`.
4. Ejecuta el enunciado en el bloque `else`. El bloque `else` da como output la línea "Huang, no Perez". Esta es la primera línea del output.
5. Suma uno a $k, que ahora es igual a 1.
6. Llega al final del ciclo.
7. Va al inicio del ciclo.
8. Prueba la condición nuevamente. ¿Es `$variabledeprueba != "si"` verdadera? Como `$variabledeprueba` no ha cambiado y todavía está establecida en "no", es verdadera, de modo que el ciclo continúa.
9. Prueba la condición if. ¿Es `$clientes[$k] == "Perez"` verdadera? En este punto, `$k` es 1, por lo cual el programa revisa `$clientes[1]`. Como `$clientes[1]` es "Perez", el enunciado es verdadero. Entonces, el ciclo entra al bloque if.
10. Ejecuta los enunciados en el bloque if. Establece `$variabledeprueba` en "si". Da como output "Perez". Esta es la segunda línea del output.
11. Suma uno a $k, que ahora se convierte en 2.
12. Llega al final del ciclo.
13. Va al inicio del ciclo.

14. Prueba la condición de nuevo. ¿Es `$variabledeprueba != "si"` verdadera? Como `$variabledeprueba` ha cambiado y ahora está establecida en "sí", no es verdadera. El ciclo se detiene.

Es posible escribir un ciclo `while` que sea infinito; o sea, un ciclo que continúe para siempre. Usted puede, fácilmente y sin proponérsele, escribir un ciclo en el cual la condición siempre sea verdadera. Si la condición nunca se vuelve falsa, el ciclo nunca acaba. Para una descripción de los ciclos infinitos, consulte la sección "Ciclos infinitos", más adelante en este capítulo.

Usar ciclos do..while

Los ciclos `do..while` son muy parecidos a los ciclos while. Un ciclo `do..while` continúa repitiéndose siempre y cuando ciertas condiciones sean verdaderas. Usted establece una condición. La condición se prueba al final de cada ciclo. Si la condición es verdadera, el ciclo se repite. Cuando la condición no sea verdadera, el ciclo se detiene.

El formato general para un ciclo `do..while` es

```
do
{
   bloque de enunciados
} while ( condicion );
```

Los siguientes enunciados establecen un ciclo que busca al cliente apellidado `Perez`. Este programa hace lo mismo que el programa en la sección anterior, el cual usa un ciclo `while`:

```
$clientes = array( "Huang", "Perez", "Herrera" );
$variabledeprueba = "no";
$k = 0;
do
{
  if ($clientes[$k] == "Perez" )
  {
    $variabledeprueba = "si";
    echo "Perez<br>";
  }
  else
  {
    echo "$clientes[$k], no Perez<br>";
  }
  $k++;
} while ( $variabledeprueba != "si" );
```

El output de estos enunciados en un explorador es

```
Huang, no Perez
Perez
```

Este es el mismo output mostrado en el ejemplo del ciclo `while`. La diferencia entre un ciclo `while` y un ciclo `do..while` es el lugar donde se verifica la condición. En un ciclo `while`, ésta se verifica al inicio del ciclo. Por lo tanto, el ciclo nunca se ejecutará si la condición nunca es verdadera. En el ciclo `do..while`, la condición se revisa al final del ciclo. Por lo tanto, el ciclo siempre se ejecuta al menos una vez, incluso si la condición nunca es verdadera.

Por ejemplo, en el ciclo anterior que busca el apellido `Perez`, suponga que la condición original se establece en sí, en lugar de no, usando este enunciado:

```
$variabledeprueba = "si";
```

La condición sería falsa desde el inicio. Nunca sería verdadera. En un ciclo `while`, no habría ningún output. El bloque de enunciados nunca correría. Sin embargo, en un ciclo `do..while`, el bloque de enunciados se correría una vez antes de probar la condición. Así, el ciclo `while` no produciría ningún output, pero el ciclo `do..while` produciría el siguiente output:

```
Huang, no Perez
```

El ciclo `do..while` produce una línea de output antes de probar la condición. No produce la segunda línea del output porque la condición resulta falsa.

Ciclos infinitos

Usted puede fácilmente configurar los ciclos para que nunca se detengan. Los ciclos de este tipo son ciclos infinitos. Se repiten para siempre. Sin embargo, casi nunca se crean ciclos infinitos intencionalmente. Por lo general se trata de un error en la programación. Por ejemplo, un cambio ligero en el programa que establece un ciclo `while` puede convertirlo en un ciclo infinito.

Este es el programa mostrado en la sección "Usar ciclos while", la cual aparece anteriormente en este capítulo:

```
$clientes = array ( "Huang", "Perez", "Herrera" );
$variabledeprueba = "no";
$k = 0;
while ( $variabledeprueba != "si" )
{
  if ($clientes[$k] == "Perez" )
  {
```

```
    $variabledeprueba = "si";
    echo "Perez<br>";
  }
  else
  {
    echo "$clientes[$k], no Perez<br>";
  }
  $k++;
}
```

Este es el programa con un ligero cambio:

```
$clientes = array ( "Huang", "Perez", "Herrera" );
$variabledeprueba = "no";
while ( $variabledeprueba != "si" )
{
  $k = 0;
  if ($clientes[$k] == "Perez" )
  {
    $variabledeprueba = "si";
    echo "Perez<br>";
  }
  else
  {
    echo "$clientes[$k], no Perez<br>";
  }
  $k++;
}
```

El pequeño cambio es mover el enunciado `$k = 0;` desde afuera del ciclo hacia adentro del ciclo. Esta pequeña modificación lo convierte en un ciclo sin fin. El output de este programa cambiado es

```
Huang, no Perez
Huang, no Perez
Huang, no Perez
Huang, no Perez
...
```

Esto se repetirá para siempre. Cada vez que el ciclo corra, reconfigura `$k` a 0. Luego obtiene `$clientes[0]` y `hace echo` de él. Al final del ciclo, `$k` se incrementa a 1. No obstante, cuando el ciclo empieza de nuevo, `$k` se restablece nuevamente en 0. En consecuencia, sólo el primer valor en el arreglo, `Huang`, es leído. El ciclo nunca llega al apellido `Perez`, y `$variabledeprueba` nunca se establece en `"si"`. El ciclo es interminable.

No se avergüence si escribe un ciclo infinito. Le garantizo que el mejor de los expertos en programación del mundo ha escrito muchos ciclos infinitos. No es nada serio. Si usted está probando un programa y obtiene output en su página web que se repite infinitamente, se detendrá por sí solo en pocos instantes. El tiempo predeterminado es de 30 segundos, pero el período de pausa puede haber sido

cambiado por el administrador PHP. Usted también puede hacer clic en el botón Stop en su buscador para detener el despliegue en su buscador. Después, averigüe por qué el ciclo se está repitiendo indefinidamente y arréglelo.

Un error común que puede originar un ciclo infinito es usar un único signo de igual (=) cuando debería haber usado dos signos de igual (==). Un signo de igual único almacena un valor en una variable; el signo de igual doble prueba si dos valores son iguales. Si escribe la siguiente condición con un solo signo de igual:

```
while ($variabledeprueba = "si")
```

siempre será verdadera. La condición simplemente fija `$variabledeprueba igual a "si"`. Esto no es una pregunta que pueda ser falsa. Probablemente, lo que usted quiso escribir es esto:

```
while ($variabledeprueba == "si")
```

Esta es una pregunta que quiere saber si `$variabledeprueba` es igual a `"si"`, a la cual puede responderse como falso o verdadero.

Usted puede salvaguardar sus programas de este error en particular cambiando la condición a `"si" == $variabledeprueba`. Es menos lógico de leer, pero protege contra el problema del signo de igual único. Si usa un solo signo de igual en lugar de uno doble en esta condición, obtendrá un error, y el programa no correrá.

Otro error común es dejar por fuera el enunciado que incrementa el contador. Por ejemplo, en el programa ilustrado anteriormente en esta sección, si usted deja por fuera el enunciado `$k++;`, `$k` siempre será 0, y el resultado es un ciclo infinito.

Escapar de un ciclo

A veces usted desea que su programa escape de un ciclo. PHP proporciona dos enunciados para este propósito:

- `break`: Sale completamente de un ciclo y continúa con los enunciados del programa después del ciclo.
- `continue`: Salta al final del ciclo donde la condición se prueba. Si la condición resulta positiva, el programa continúa desde el principio del ciclo.

`break y continue` generalmente se usan en un enunciado condicional. `break`, en particular, se usa más a menudo en enunciados `switch`, como comenté anteriormente en el capítulo.

Los siguientes dos conjuntos de enunciados muestran la diferencia entre `continue y break`. Los primeros enunciados usan el enunciado `break`.

```
$contador = 0;
while ( $contador < 5 )
{
  $contador++;
  If ( $contador == 3 )
  {
      echo "break<br>";
      break;
  }
  echo "Final del ciclo while: contador=$contador<br>";
}
echo "Despues del ciclo break<p>";
```

Los enunciados siguientes usan el enunciado `continue`:

```
$contador = 0;
while ( $contador < 5 )
{
  $contador++;
  If ( $contador == 3 )
  {
      echo "continue<br>";
      continue;
  }
  echo "Final del ciclo while: contador=$contador<br>";
}
echo "Despues del ciclo continue<br>";
```

Estos enunciados construyen dos ciclos iguales, excepto que el primero usa break, y el segundo usa `continue`. El output de estos primeros enunciados que usan el enunciado `break` aparece en su explorador como sigue:

```
Final del ciclo while: contador=1
Final del ciclo while: contador=2
break
Despues del ciclo break
```

El output del segundo grupo de enunciados con el enunciado continue es el siguiente:

```
Final del ciclo while: contador=1
Final del ciclo while: contador=2
continue
Final del ciclo while: contador=4
Final del ciclo while: contador=5
Despues del ciclo continue
```

El primer ciclo termina con el enunciado `break`. El ciclo se detiene y pasa inmediatamente al enunciado después del ciclo. El segundo ciclo no termina con el enunciado `continue`. Simplemente impide la tercera repetición del ciclo y salta de regreso al principio del ciclo. Luego termina el ciclo, con la cuarta y quinta repeticiones, antes de continuar el enunciado después del ciclo.

Un uso para los enunciados `break` es asegurarse contra los ciclos infinitos. Los siguientes enunciados dentro de un ciclo pueden detenerlo en un punto razonable:

```
$prueba4intinito++;
if ($prueba4infinito > 100 )
{
   break;
}
```

Si usted está seguro de que su ciclo nunca debería repetirse más de 100 veces, estos enunciados impedirán que el ciclo se haga infinito. Use cualquier número que le parezca razonable para el ciclo que está construyendo.

Usar funciones

Las aplicaciones a menudo realizan la misma tarea en diferentes puntos en el programa o en diferentes programas. Por ejemplo, su aplicación podría mostrar el logotipo de la compañía en varias páginas web distintas o en diferentes partes del programa. Suponga que usa los siguientes enunciados para desplegar el logotipo de la compañía:

```
echo '<hr width="50" align="left">',"\n";
echo '<img src="/images/logo.jpg" width="50"
          height="50"><br>',"\n";
echo '<hr width="50" align="left"><br>',"\n";
```

Puede crear una función que contenga los enunciados anteriores y darle el nombre de `mostrar_logotipo`. Entonces, cada vez que el programa necesite mostrar el logotipo, usted sólo debe llamar a la función que contiene los enunciados con un simple llamado de función, como sigue:

```
mostrar_logotipo();
```

Observe los paréntesis después del nombre de la función. Estos son obligatorios en un llamado de función, pues le dicen a PHP que esta es una función.

Usar una función ofrece varias ventajas:

- **Menos digitación:** Basta con digitar los enunciados una sola vez: en la función. Para siempre después de eso, usted simplemente usa el llamado de función y nunca tendrá que digitar los enunciados de nuevo.
- **Lectura más fácil:** La línea `mostrar_logotipo()` es mucho más fácil de entender para una persona con un solo vistazo.
- **Menos errores:** Después de haber escrito su función y haberle arreglado todos sus problemas, funcionará correctamente cada vez que la use.

- **Más fácil de cambiar:** Si decide cambiar cómo debe realizarse la tarea, sólo hace falta cambiarla en un lugar. Basta con cambiar la función; no hay necesidad de buscar en cien lugares diferentes de su programa donde llevó a cabo la tarea y cambiar el código cien veces. Por ejemplo, suponga que cambió el nombre del archivo de gráficos donde guarda el logotipo de la compañía. Usted cambia el nombre del archivo en un lugar, la función, y operará correctamente en todos los demás lugares.

Puede crear una función poniendo el código en un bloque de funciones. El formato general es

```
function nombredelafuncion()
{
   bloque de enunciados;
   return;
}
```

Por ejemplo, usted crea la función para mostrar el logotipo de la compañía con los siguientes enunciados:

```
function mostrar_logotipo()
{
   echo '<hr width="50" align="left">',"\n";
   echo '<img src="/images/logo.jpg" width="50"
            height="50"><br>',"\n";
   echo '<hr width="50" align="left"><br>',"\n";
   return;
}
```

El enunciado `return` detiene la función y regresa al programa principal. El enunciado `return` al final de la función no es obligatorio, pero hace que la función sea más fácil de entender. El enunciado `return` a menudo se usa para un final condicional de una función.

Imagine que su función despliega un catálogo de equipo electrónico. Podría usar el siguiente enunciado al inicio de la función:

```
if ( $edad < 13 )
   return;
```

Si la edad del cliente es menos de 13 años, la función se detiene y el catálogo de equipo electrónico no se mostrará.

Usted puede poner funciones en cualquier parte del programa, pero la práctica común es poner todas las funciones juntas al inicio o al final del archivo del programa. Las funciones que planea usar en más de un programa pueden estar en un archivo separado. Cada programa tiene acceso a las funciones del archivo externo. Para más información sobre la organización de aplicaciones en archivos y cómo tener acceso a archivos separados, revise el Capítulo 10.

Observe que la función del ejemplo es bastante simple. No usa variables, y no comparte ninguna información con el programa principal. Sólo realiza una tarea independiente cuando se la llama. Usted puede usar variables en funciones y pasar información de una función al programa principal, siempre y cuando conozca las reglas y limitaciones. Las secciones restantes en este capítulo le explican cómo usar variables y pasar valores.

Usar variables en funciones

Usted puede crear y usar variables que sean locales a la función. O sea, puede crear y usar una variable dentro de su función. Sin embargo, la variable no estará disponible fuera de la función; no está disponible para el programa principal. Usted puede lograr que la variable sea disponible usando un enunciado especial llamado `global`, el cual hace que una variable sea disponible en cualquier lugar del programa. Por ejemplo, la función siguiente crea una variable:

```
function formato_nombre ()
{
   $primer_nombre = "Goliat";
   $apellido = "Perez";
   $nombre = $apellido.", ".$primer_nombre;
}
formato_nombre();
echo "$nombre";
```

Estos enunciados no producen ningún output. En el enunciado `echo`, `$nombre` no contiene ningún valor. La variable `$nombre` se creó dentro de la función, de modo que no existe fuera de ella.

Puede crear una variable dentro de una función que exista fuera de la función usando el enunciado `global`. Los enunciados siguientes contienen la misma función, con un enunciado `global` añadido:

```
function formato_nombre()
{
   $primer_nombre = "Goliat";
   $apellido = "Perez";
   global $nombre;
   $nombre = $apellido.", ".$primer_nombre;
}
formato_nombre();
echo "$nombre";
```

El programa ahora repite esto:

```
Perez, Goliat
```

El enunciado `global` permite que la variable esté disponible en cualquier ubicación del programa. Usted debe crear la variable global antes de poder

usarla. Si el enunciado `global` sigue al enunciado de asignación `$nombre`, el programa no produce ningún output.

Las mismas reglas aplican cuando usted está usando una variable que fue creada en el programa principal. No puede usar una variable que haya sido creada fuera de una función en la función, a menos que la variable sea global, como se muestra en los enunciados siguientes:

```
$primer_nombre = "Goliat";
$apellido = "Perez";
function formato_nombre()
{
   global $primer_nombre, $apellido;
   $nombre = $apellido.", ".$primer_nombre;
   echo "$nombre";
}
formato_nombre();
```

Si no usa el enunciado `global`, `$apellido` y `$primer_nombre` dentro de la función son variables diferentes, creadas cuando usted les da un nombre. No tienen valores. El programa no produciría ningún output sin el enunciado `global`.

Cómo pasar valores entre una función y el programa principal

Usted puede pasar valores hacia dentro de la función y recibir valores de la función. Por ejemplo, imagine que escribe una función para sumar el impuesto de ventas correcto a un pedido. La función tendría que saber el costo del pedido y en cuál estado vive el cliente. La función necesitaría enviar de vuelta el monto del impuesto de ventas.

Pasar valores a una función

Puede pasar valores a una función poniendo los valores entre paréntesis cuando llama a la función, como sigue:

```
nombredefuncion(valor,valor,...);
```

Por supuesto, las variables no pueden simplemente aparecer. La función debe estar esperándolas. El enunciado `function` incluye los nombres de las variables para los valores que está esperando, como sigue:

```
function nombredefuncion($nombrevariable1,$nombrevariable2,...)
{
    enunciados
    return;
}
```

Por ejemplo, la siguiente `function` calcula el impuesto de ventas:

```
function calcular_impuestodeventas($cantidad,$estadocliente)
{
  switch ( $estadocliente )
  {
    case "OR" :
      $tasaimpuestoventas = 0;
      break;
    case "CA" :
      $tasaimpuestoventas = 1.0;
      break;
    default:
      $tasaimpuestoventas =.5;
      break;
  }
  $tasaimpuestoventas = $cantidad * $tasaimpuestoventas;
  echo "$impuestoventas<br>";
}
$precio = 2000.00;
$estadocliente = "CA";
calcular_impuestodeventas($precio,$estadocliente);
```

La primera línea muestra que la función espera dos valores, como sigue:

```
function calcular_impuestodeventas($cantidad,$estadocliente)
```

La última línea es el llamado de función, el cual pasa dos valores a la función `calcular_impuestodeventas`, como los espera. El monto del pedido y el estado donde vive el cliente se pasan. El output de este programa es `2000` porque la tasa del impuesto de ventas para California es del 100 por ciento.

Usted puede pasar tantos valores como necesite. Los valores pueden ser variables o valores, incluyendo valores que se calculen. Los siguientes llamados de función son válidos:

```
calcular_impuestodeventas (2000,"CA");
calcular_impuestodeventas (2*1000,"");
calcular_impuestodeventas (2000,"C"."A");
```

Los valores se pueden pasar en un arreglo. La función recibe la variable como un arreglo. Por ejemplo, los siguientes enunciados pasan un arreglo:

```
$seriedenumeros = array( 100, 200);
sumarnumeros($seriedenumeros);
```

La función recibe todo el arreglo. Por ejemplo, suponga que la función empieza con el siguiente enunciado:

```
function sumarnumeros($numeros)
```

La variable $numeros es un arreglo. La funcion puede incluir enunciados tales como

```
return $numeros[0] + $numeros[1];
```

Los valores pasados se pasan según posición. Es decir, el primer valor en la lista que usted pasa se usa como el primer valor en la lista que la función espera, el segundo se usa para el segundo, y así sucesivamente. Si sus valores no están en el mismo orden, la función usará el valor equivocado al realizar la tarea. Por ejemplo, en calcular_impuestodeventas, podría llamar a calcular_impuestodeventas para que pase valores en el orden equivocado, como sigue:

```
calcular_impuestodeventas($estadocliente,$valordelpedido);
```

La función usa el estado como el valor del pedido, el cual establece en 0 porque el valor pasado es una cadena. Establece el estado al número en $valordelpedido, el cual no concordaría con ninguna de sus categorías. El output sería 0.

Si no envía suficientes valores, la función establece el valor faltante en una cadena vacía para una variable en cadena, o a 0 para un número. Si envía demasiados valores, la función ignora los valores adicionales.

Si pasa el número equivocado de valores a una función, podría recibir un mensaje de advertencia, como sigue, dependiendo del nivel de mensajes de error para el cual PHP esté configurado.

```
Warning: Missing argument 2 for compute_salestax() in /test7.php on line 5
```

Para más detalles sobre los mensajes de advertencia, consulte el Capítulo 6.

Puede fijar valores predeterminados para usarse cuando un valor no se pase. Los valores predeterminados se establecen cuando usted escribe la función asignando un valor predefinido para el valor o los valores que esté esperando, como sigue:

```
function sumar_2_numeros($num1=1,$num2=1)
{
   $total = $num1 + $num2;
   return $total;
}
```

Si uno de los valores no se pasa (o si no se pasa ninguno), la función usa los valores predefinidos asignados. Pero, si se pasa un valor, éste se usa en lugar del valor predeterminado. Por ejemplo, usted podría usar uno de los siguientes llamados:

```
sumar_2_numeros(2,2);
sumar_2_numeros(2);
sumar_2_numeros();
```

Los resultados son, en orden consecutivo:

```
$total = 4
$total = 3
$total = 2
```

Obtener un valor de una función

Cuando usted llama a una función, puede pasar valores como se describió anteriormente. La función también puede devolver un valor al programa que la llamó. Use el enunciado return para regresar un valor al programa que ha emitido el llamado. El programa puede almacenar el valor en una variable o usarlo directamente; por ejemplo, puede usarlo en un enunciado condicional. El enunciado `return` también devuelve el control al programa principal; es decir, detiene la función.

El formato general del enunciado `return` es

```
return valor;
```

Por ejemplo, en el programa de impuesto de la sección anterior, yo hice eco del impuesto de ventas usando los siguientes enunciados:

```
$impuestodeventas = $monto * $tasaimpuestodeventas;
echo "$impuestodeventas<br>";
```

Podría devolver el impuesto de ventas al programa principal, en lugar de hacer eco de él, usando el siguiente enunciado:

```
$impuestodeventas = $monto * $tasaimpuestodeventas;
return $impuestodeventas;
```

De hecho, yo podría usar un atajo y enviarlo de regreso al programa principal con un enunciado:

```
return $monto * $tasaimpuestodeventas;
```

El enunciado `return` envía el `impuestodeventas` de regreso al programa principal y termina la función. El programa principal puede usar el valor en cualquiera de las formas usuales. Los enunciados siguientes usan el llamado de función de manera válida:

```
$impuestodeventas = calcular_impuestodeventas($precio,$estadocliente);

$costototal = $precio + calcular_impuestodeventas($precio,$estadocliente);

if ( calcular_impuestodeventas($precio,$estadocliente) >
          100000.00 )

        $echo "Gracias, muchas, muchas, muchas gracias<br>";
```

```
foreach($pedidocliente as $monto)
{
   $total = $monto + calcular_impuestodeventas($monto,$estadocliente);
   echo "Su total es $total<br>";
}
```

Un enunciado `return` puede devolver sólo un valor. Sin embargo, el valor devuelto puede ser un arreglo, de modo que usted en realidad puede devolver muchos valores desde una función.

Puede usar enunciados `return` en un enunciado condicional para devolver diferentes valores para diferentes condiciones. Por ejemplo, la siguiente función devuelve una de dos cadenas diferentes:

```
function comparar_valores($valor1,$valor2)
{
  if($valor1 < $valor2)
  {
     return "menor que";
  }
  else
  {
     return "no menor que";
  }
}
```

Aunque la función contiene dos enunciados return, sólo una va a ejecutarse, dependiendo de los valores en `$valor1` y `$valor2`.

Usar funciones incorporadas

Las muchas funciones integradas de PHP son una de las razones por las cuales PHP es tan poderoso y útil para las páginas web. Las funciones incluidas en PHP son funciones normales. No son muy diferentes de las que usted mismo puede crear. Pero PHP ya ha hecho el trabajo por usted.

Describo algunas de las funciones incorporadas en este capítulo y capítulos anteriores. Por ejemplo, vea el Capítulo 6 para más información sobre las funciones `unset` y `number_format`. Algunas funciones muy útiles para interactuar con su base de datos MySQL se comentan en el Capítulo 8. Otras funciones útiles se incluyen en la Parte V de este libro. Y, por supuesto, todas las funciones se enumeran y describen en la documentación PHP, en el sitio web de PHP: `www.php.net/docs.php`.

Capítulo 8

Datos entran, datos salen

En este capítulo

- Conectarse a la base de datos
- Obtener información de la base de datos
- Usar formularios HTML con PHP
- Extraer datos de un formulario HTML
- Procesar la información que los usuarios digitan en formularios HTML
- Almacenar datos en la base de datos
- Usar funciones para insertar y extraer datos de la base de datos

PHP y MySQL funcionan muy bien juntos. Esta sociedad dinámica hace que PHP y MySQL sean tan atractivos para el desarrollo de aplicaciones con bases de datos para la Web. Ya sea que usted tenga una base de datos llena de información que quiere poner a la disposición de los usuarios (como un catálogo de productos) o una base de datos que está esperando para que los usuarios la rellenen (por ejemplo, una base de datos de membresía), PHP y MySQL trabajan juntos para implementar su aplicación.

Una de las características más fuertes de PHP es su habilidad para interactuar con bases de datos. PHP provee funciones que hacen de la comunicación con MySQL algo extremadamente simple. Se usan las funciones de PHP para enviar consultas SQL a la base de datos. No hace falta conocer los detalles de la comunicación con MySQL; PHP se encarga de eso. Usted nada más necesita saber sobre las consultas SQL y cómo usar las funciones de PHP.

En capítulos anteriores de este libro, describo las herramientas que se usan para construir una aplicación con base de datos para la Web. Usted averigua cómo construir consultas SQL en el Capítulo 4 y cómo construir y usar los bloques de construcción del lenguaje PHP en los Capítulos 6 y 7. En este capítulo, descubrirá cómo usar estas herramientas para las tareas específicas que una aplicación con una base de datos para la Web debe realizar.

Funciones de PHP/MySQL

Las funciones incorporadas de PHP se usan para interactuar con MySQL. Estas funciones se conectan al servidor MySQL, seleccionan la base de datos correcta, envían consultas SQL y llevan a cabo otras comunicaciones con la base de datos MySQL. Usted no necesita conocer los detalles sobre la interacción con la base de datos, ya que PHP se encarga de todos los detalles. Usted sólo debe saber cómo usar las funciones.

En este momento, PHP (empezando por la versión 5) cuenta con dos conjuntos de funciones para interactuar con MySQL. Un conjunto de funciones se usa para interactuar con la versión 4.0 de MySQL, o versiones anteriores. El otro conjunto de funciones se usa para interactuar con la versión 4.1 de MySQL y las posteriores.

A lo largo de este libro, mis ejemplos y programas muestran las funciones que trabajan con MySQL versión 4.0. Al momento de escribir este libro, la versión actual estable de MySQL es la versión 4.0. Además, las funciones de PHP que se usan con MySQL versión 4.1 se describen en la documentación del sitio web de PHP como experimentales, con una advertencia: *Use esta extensión bajo su propio riesgo.* Por el momento, la versión 4.1 de MySQL sólo está disponible como software alpha, definido en el manual de MySQL como sigue: *Esta publicación contiene una gran sección de código nuevo que no ha sido probabo al 100%.* Por lo tanto, estoy escribiendo este libro con base en el uso de MySQL 4.0 y de las funciones de PHP que trabajan con esta versión.

Las funciones de PHP que se usan con MySQL 4.0 tienen el siguiente formato general:

```
mysql_funcion(valor,valor,...);
```

La segunda parte del nombre de la función es específica a la función, generalmente una palabra que describe lo que hace la función. Además, la función requiere que se pasen uno o más valores, y que se especifiquen cosas tales como la conexión con la base de datos, la ubicación de los datos, etc. A continuación, dos de las funciones que se discuten en este capítulo:

```
mysql_connect($conectar);
mysql_query("enunciado SQL",$conectar);
```

Para cuando usted lea este libro, es probable que MySQL 4.1 sea la versión actual estable de MySQL. Si es así, puede usar MySQL 4.1 y las funciones que trabajan con dicha versión. Al momento de escribir este libro, las funciones discutidas en él son las mismas para MySQL 4.0 y MySQL 4.1 excepto por un cambio en el nombre. Los nombres de las funciones que se usan con MySQL 4.1 empiezan con `mysqli_` en vez de `mysql_`. Así, las funciones mostradas arriba tendrían los siguientes nombres si se usan con MySQL 4.1:

```
mysqli_connect($conectar);
mysqli_query($conectar,"enunciado SQL");
```

La funcionalidad y la sintaxis de las funciones son iguales o muy similares. Por ejemplo, observe la diferencia entre las dos siguientes funciones.

```
mysql_query($sql,$conectar);
mysqli_query($conectar,$sql);
```

Observe que el orden de los elementos pasados es distinto. Esto es así para varias funciones.

Esta discusión se refiere a las funciones mysqli tal y como son ahora. Las funciones `mysqli` pueden cambiar, por supuesto. Si una de las funciones `mysqli` no parece funcionar como debiera, revise el manual para hallar cualquier diferencia de uso posible.

La i añadida al nombre es por *improved (mejorado)*; este conjunto de funciones MySQL es proporcionado por la extensión mejorada de MySQL. En este momento, no se incluye soporte para `mysqli` en PHP en forma predeterminada. Debe activarse cuando se instala PHP. Sin embargo, para cuando MySQL 4.1 sea la versión estable, es probable que `mysqli` sea parte de PHP sin necesidad de activarlo específicamente. Para ver la condición actual de `mysqli`, revise la documentación en `www.php.net/manual/en/ref.mysqli.php`.

Hasta el momento de escribir esto, usted puede usar la siguiente opción de instalación para activar mysqli:

```
--with-mysqli=DIR
```

`DIR` es la ruta hacia el directorio donde se encuentra un programa llamado `mysql_config`, el cual fue instalado cuando se instaló MySQL 4.1.

Hacer una conexión

Antes de poder almacenar o extraer datos, usted debe conectarse a la base de datos. La base de datos podría estar en el mismo PC con sus programas PHP, o podría estar en un PC diferente. Usted no necesita saber los detalles sobre la conexión con la base de datos, porque PHP maneja todos los detalles. Todo lo que necesita saber es el nombre y la ubicación de la base de datos. Imagine la conexión a la base de datos como una conexión telefónica. No hace falta conocer cómo las palabras se trasladan de un teléfono a otro. Usted sólo debe saber es el código de área y el número telefónico. La compañía telefónica maneja los detalles.

Después de conectarse a la base de datos, usted envía consultas SQL a la base de datos MySQL usando una función de PHP diseñada específicamente para este propósito. Puede enviar tantas consultas como necesite. La conexión permanece abierta hasta que usted específicamente la cierre o el programa termine. Asimismo, en una conversación telefónica, la conexión permanece abierta hasta que usted la termine al colgar el teléfono.

Conectarse al servidor MySQL

El primer paso para comunicarse con su base de datos MySQL es conectarse al servidor MySQL. Para hacerlo, usted debe saber el nombre del PC donde se localiza la base de datos, el nombre de su cuenta MySQL y la contraseña para su cuenta MySQL. Para abrir la conexión, use la función `mysql_connect` como sigue:

```
$conexion=mysql_connect
("direccion","cuentamysql","clave")
    or die ("mensaje");
```

Complete la siguiente información:

- *direccion:* El nombre del PC donde MySQL está instalado: por ejemplo, `servidordb.miempresa.com`. Si la base de datos MySQL está en el mismo PC que su sitio web, puede usar `localhost` como el nombre del PC. Si esta información está en blanco (""), PHP asume que es `localhost`.
- *cuentamysql:* En nombre de su cuenta MySQL. (Discuto las cuentas MySQL en detalle en el Capítulo 5). Puede dejar esta información en blanco (""), lo cual significa que cualquier cuenta puede conectarse; pero, generalmente, esa es una mala idea por razones de seguridad.
- *clave:* La contraseña de su cuenta MySQL. Si su cuenta MySQL no requiere de una contraseña, no digite nada entre las comillas: "".
- *mensaje:* El mensaje que se envía al buscador si la conexión falla. La conexión falla si el PC o la red están caídos o si el servidor MySQL no está corriendo. También puede fallar si la información brindada no es correcta: por ejemplo, si hay un error de digitación en la contraseña.

  Sería bueno que usara un `mensaje` descriptivo durante el desarrollo, tal como `No se pudo conectar al servidor`, pero use un mensaje más general adecuado para los clientes una vez que la aplicación esté en uso, tal como `El catalogo de mascotas no esta disponible por el momento. Por favor intente de nuevo mas tarde.`

La `direccion` incluye un número de puerto necesario para la conexión. Casi siempre, el número del puerto es 3306. En algunas pocas ocasiones, el administrador de MySQL necesita configurar MySQL para conectarse a un puerto diferente. En esos casos, el número de puerto es obligatorio para la conexión. El número de puerto se especifica así `nombrehuesped:numeropuerto`. Por ejemplo, usted podría usar `localhost:8808`.

Con estos enunciados, `mysql_connect` trata de abrir una conexión con el PC nombrado, usando el nombre de la cuenta y la contraseña proporcionados. Si la conexión falla, el programa deja de correr en ese punto y envía el `mensaje` al explorador.

El siguiente enunciado se conecta al servidor MySQL en el PC local usando una cuenta MySQL llamada `catalogo` que no requiere de una contraseña:

Manejo de errores en MySQL

Las funciones `mysql`, tales como `mysql_connect` y `mysql_query`, se usan en el lenguaje PHP para interactuar con las base de datos MySQL. Si una de estas funciones no se ejecuta correctamente, se envía un mensaje de error MySQL con información sobre el problema. Sin embargo, este mensaje de error no será enviado al explorador, a menos que el programa lo mande deliberadamente. Estas son las tres formas comunes de llamar a las funciones `mysql`:

- **Llamar a la función sin manejo de errores.** La función es llamada sin ningún enunciado que brinde mensajes de error. Por ejemplo, la función `mysql_connect` se puede llamar como sigue:

  ```
  $conexion = mysql_connect($huesped,$usuario,$clave);
  ```

  Si este enunciado falla (por ejemplo, la cuenta no es válida), la conexión no se hace, pero los enunciados restantes en el programa continúan ejecutándose. En muchos casos, esto no es útil porque algunos de los enunciados en el resto del programa (tal como extraer o almacenar datos en la base de datos) podrían depender de contar con una conexión abierta.

- **Llamar a la función con un enunciado die.** La función es llamada con un enunciado `die` que envía un mensaje al buscador. Por ejemplo, la función `mysql_connect` se puede llamar así:

  ```
  $conexion = mysql_connect($huesped,$usuario,$clave)
      or die ("No se pudo conectar al servidor ");
  ```

  Si este enunciado falla, la conexión no se hace y el enunciado `die` se ejecuta. El enunciado `die` detiene el programa y envía el mensaje al explorador. Si la conexión no se puede establecer, ningún otro enunciado se ejecuta. Usted puede poner el mensaje que desee en el enunciado `die`.

- **Llamar a la función en un enunciado if.** La función es llamada usando un enunciado `if` que ejecuta un bloque de enunciados si la conexión falla. Por ejemplo, la función `mysql_connect` puede llamarse así:

  ```
  if (!$conexion = mysql_connect($huesped,$usuario,$clave))
  {
      $mensaje = mysql_error();
      echo "$mensaje<br>";
      die();
  }
  ```

  Si esta conexión falla, los enunciados en el bloque `if` se ejecutan. La función `mysql_error` devuelve el mensaje de error de MySQL y lo guarda en la variable `$mensaje`. Luego, se hace eco al mensaje de error. El enunciado `die` termina el programa, de modo que ningún otro enunciado se ejecuta. Note el ! (signo de admiración) en el enunciado `if`. ! significa "not". En otras palabras, el enunciado `if` es verdadero si el enunciado de asignación no es verdadero.

El manejo de errores que usted quiera incluir en su programa dependerá de lo que espera que suceda en el programa. Al desarrollar el programa, se espera que ocurran algunos errores. Por ende, durante el desarrollo, usted probablemente querrá un manejo de errores que sea más descriptivo, tal como el tercer método de la lista anterior. Por ejemplo, suponga que está usando una cuenta llamada `root` para tener acceso a su base de datos y que comete un error de digitación, como en los enunciados siguientes:

```
$huesped = "localhost";
$usuario = "raz";
$contraseña = "";
if (!$conexi%n = mysql_connect($huesped,$usuario,$contraseña))
      {
          $mensaje = mysql_error();
          echo "$mensaje<br>";
          die();
}
```

Como digitó "raz" en lugar de "raiz", vería un mensaje de error parecido al siguiente:

```
Acceso negado al usuario: 'raz@localhost' (Contraseña usada: NO)
```

Este mensaje de error tiene la información que usted necesita para averiguar cuál es el problema; muestra su nombre de cuenta con el error de digitación. Sin embargo, una vez que su programa esté corriendo y los clientes lo estén usando, usted probablemente no querrá que sus usuarios vean un mensaje de error técnico como el anterior. En cambio, usted seguramente preferirá usar el segundo método con un enunciado general en el mensaje `die`, tal como `El catalogo de mascotas no esta disponible por el momento. Por favor intente de nuevo mas tarde.`

```
$conexion = mysql_connect("localhost","catalogo","")
     or  die ("No se pudo conectar al servidor.");
```

Por razones de seguridad, es una buena idea almacenar la información de la conexión en variables y usar las variables en el enunciado de conexión, como sigue:

```
$huesped="localhost";
$usuario="catalogo";
$clave="";
$conexion = mysql_connect($huesped,$usuario,$clave)
     or  die ("No se pudo conectar al servidor.");
```

De hecho, para tener aún más seguridad, usted puede poner los enunciados de asignación para los datos sobre la conexión en un archivo separado, en un lugar oculto, de modo que el nombre de la cuenta y la contraseña no estén ni siquiera en el programa. Le explico cómo hacerlo en el Capítulo 10.

La variable `$conexion` contiene información que identifica la conexión. Usted puede tener más de una conexión abierta al mismo tiempo usando más de un nombre de variable. Una conexión permanece abierta hasta que usted la cierre o el programa termine. Una conexión se cierra así:

```
mysql_close($nombreconexion);
```

Por ejemplo, para cerrar la conexión del ejemplo anterior, use este enunciado:

```
mysql_close($conexion);
```

Seleccionar la base de datos correcta

Una vez establecida y abierta la conexión con el servidor MySQL, usted necesita decirle a MySQL con cuál base de datos desea interactuar. Use la función `mysql_select_db` como sigue:

```
$db =
mysql_select_db("nombrebasededatos",$nombreconexion)
     or die ("mensaje");
```

Rellene la siguiente información:

- *`nombrebasededatos`:* El nombre de la base de datos.
- *`nombreconexion`:* La variable que contiene los datos sobre la conexión. Si usted no introduce una conexión, PHP usa la última conexión que fue abierta.
- *`mensaje`:* El mensaje que se envía al explorador si la base de datos no se puede seleccionar. La selección podría fallar porque la base de datos no fue encontrada, lo cual generalmente es el resultado de un error de digitación en el nombre de la base de datos.

Por ejemplo, usted puede seleccionar la base de datos `Catalogodemascotas` con el siguiente enunciado:

```
$db = mysql_select_db("Catalogodemascotas",$conexion)
     or die ("No se pudo seleccionar la base de datos.");
```

Si `mysql_select_db` no puede seleccionar la base de datos, el programa se detiene en este punto y el mensaje `No se pudo seleccionar la base de datos` se envía al explorador.

Por razones de seguridad, es una buena idea almacenar el nombre de la base de datos en una variable y usar la variable en el enunciado de conexión, como sigue:

```
$basededatos = "Catalogodemascotas";
$db = mysql_select_db($basededatos,$conexion)
     or die ("No se pudo seleccionar la base de datos.");
```

De hecho, para mayor seguridad aún, usted puede poner el enunciado de asignación para el nombre de la base de datos en un archivo separado, en una ubicación escondida (tal como se sugirió para los enunciados de asignación para la información conexión), de modo que el nombre de la base de datos no esté ni siquiera en el programa. Le explico cómo hacerlo en el Capítulo 10.

La base de datos permanece seleccionada hasta que usted explícitamente seleccione una base de datos diferente. Para seleccionar una base de datos diferente, sólo use un nuevo enunciado de función `mysql_select_db`.

Enviar consultas SQL

Una vez abierta la conexión con el servidor MySQL, y cuando PHP sepa con cuál base de datos usted desea interactuar, usted envía su consulta SQL. La consulta es una solicitud al servidor MySQL de que almacene, actualice o recupere algunos datos. (Consulte el Capítulo 4 para más información sobre el lenguaje SQL y cómo construir consultas en SQL).

Para interactuar con la base de datos, ponga su consulta SQL en una variable y envíela al servidor MySQL usando la función `mysql_query`, como en el ejemplo siguiente:

```
$consulta = "SELECT * FROM Mascota";
$resultado = mysql_query($consulta)
    or   die ("No se pudo ejecutar la consulta.");
```

La consulta se ejecuta en la base de datos seleccionada para la última conexión que usted abrió. Si fuera necesario (si usted tuviera más de una conexión abierta, por ejemplo), puede enviar la consulta a un servidor de la base de datos específico así:

```
$resultado = mysql_query($consulta,$conexion)
    or    die ("No se pudo ejecutar la consulta.");
```

La variable `$resultado` guarda información sobre el resultado de la ejecución de la consulta. La información depende de si la consulta obtiene información de la base de datos o no:

- **Para consultas que no obtienen datos:** La variable `$resultado` contiene información sobre si la consulta se ejecutó satisfactoriamente o no. Si fue exitosa, `$resultado` se establece en `VERDADERO`; si no lo fue, `$resultado` se establece en `FALSO`. Algunas consultas que no devuelven datos son `INSERT` y `UPDATE`.
- **Para consultas que devuelven datos:** La variable `$resultado` contiene un identificador de resultado que identifica dónde están los datos obtenidos, y no los datos obtenidos en sí. Algunas consultas que sí devuelven datos son `SELECT` y `SHOW`.

El uso de comillas dobles y simples puede resultar algo confuso al asignar la cadena de consulta a `$consulta`. En realidad, las comillas se usan en dos niveles: las comillas necesarias para asignar la cadena a `$consulta` y las comillas que son parte de la consulta en lenguaje SQL. Las siguientes reglas le ayudarán a evitar problemas con las comillas:

- Use comillas dobles al comienzo y al final de la cadena.
- Use comillas simples antes y después de los nombres de variables.
- Use comillas simples antes y después de cualquier valor literal.

Los siguientes son ejemplos de cómo asignar cadenas de consulta:

```
$consulta = "SELECT nombre FROM Miembro";
$consulta = "SELECT nombre FROM Miembro WHERE apellido='Perez'";
$consulta = "UPDATE Miembro SET apellido='$apellido'";
```

La cadena de consulta misma no incluye un punto y coma (;). Por lo tanto, no ponga un punto y coma dentro de la última comilla. El único punto y coma está al final; este es el punto y coma de PHP que termina el enunciado.

Extraer información de una base de datos

Extraer información de una base de datos es una tarea común en las aplicaciones con bases de datos para la Web. Estos son dos usos comunes dados a la información de una base de datos:

- **Use la información para ejecutar enunciados condicionalmente.** Por ejemplo, usted podría obtener la ciudad de residencia del Directorio de miembros y enviar diferentes mensajes a los miembros que viven en ciudades distintas.
- **Despliegue la información en una página web.** Por ejemplo, tal vez usted quiera mostrar información sobre productos en su base de datos.

Para poder usar la información de una base de datos en un programa, usted necesita poner la información en variables. Después, puede usar las variables en enunciados condicionales, enunciados `echo` u otros enunciados. Extraer información de una base de datos es un proceso que consta de dos pasos:

1. Se construye una consulta `SELECT` y se envía a la base de datos. Cuando la consulta se ejecuta, los datos seleccionados se almacenan en una ubicación temporal.
2. Se mueven los datos de la ubicación temporal a las variables y se usan en el programa.

Enviar una consulta SELECT

La consulta `SELECT` se utiliza para obtener datos de una base de datos. Las consultas `SELECT` se escriben en lenguaje SQL. (Discuto la consulta `SELECT` en detalle en el Capítulo 4.)

Para extraer datos de la base de datos, construya la consulta `SELECT` que necesita, almacénela en una variable, y luego envíe la consulta a la base de datos. Los siguientes enunciados seleccionan toda la información de la tabla `Mascotas` en la base de datos `Catalogodemascotas`:

```
$consulta = "SELECT * FROM Mascota";
$resultado = mysql_query($consulta)
    or     die ("No se pudo ejecutar la consulta.");
```

La función mysql_query extrae los datos solicitados por la consulta SELECT y los almacena en una ubicación temporal. Imagine que estos datos se almacenan en una tabla, parecida a la tabla MySQL, con la información en filas y columnas.

La función devuelve un identificador de resultado que contiene la información necesaria para encontrar la ubicación temporal donde los datos están almacenados. En los enunciados anteriores, el identificador de resultado se pone en la variable $resultado. El siguiente paso después de ejecutar la función es mover los datos desde su ubicación temporal hacia variables que se puedan usar en el programa.

Si la función falla (por ejemplo, si la consulta es incorrecta) $resultado contiene FALSO.

Extraer y usar los datos

La función mysql_fetch_array se usa para extraer los datos de su ubicación temporal. La función extrae una fila de datos de la ubicación temporal. La tabla temporal de datos podría contener sólo una fila de datos o, más probablemente, su consulta select podría dar como resultado más de una fila de datos en la tabla temporal de datos. Si necesita tomar más de una fila de datos de la ubicación temporal, use la función mysql_fetch_array en un ciclo.

Extraer una fila de datos

Para mover los datos de su ubicación temporal y ponerlos en variables que puedan usarse en su programa, se utiliza la función PHP mysql_fetch_array. El formato general de la función mysql_fetch_array es

```
$fila = mysql_fetch_array($identificadorderesultado,tipodearreglo);
```

Este enunciado toma una fila de la tabla de datos en la ubicación temporal y la pone en una variable de arreglo llamada $fila. Complete la siguiente información:

- *identificadorderesultado:* La variable que señala hacia la ubicación temporal de los resultados.
- *tipodearreglo:* El tipo de serie en la cual se ponen los resultados. Puede ser uno de dos tipos de series o ambos tipos. Use uno de los valores siguientes:
  - MYSQL_NUM: Un arreglo con una pareja de clave y valor para cada columna en la fila, la cual usa números como claves.
  - MYSQL_ASSOC: Un arreglo con una pareja de clave y valor para cada columna en la fila, la cual los nombres de columnas como claves.
  - MYSQL_BOTH: Un arreglo con ambos tipos de claves. En otras palabras, el arreglo tiene dos parejas de clave y valor para cada columna: uno con un número como clave y uno con el nombre de la columna como clave. Si no se especifica el tipo de serie en el llamado de función, se asume que es MYSQL_BOTH.

La función `mysql_fetch_array` extrae una fila de datos de la ubicación temporal. En algunos casos, una fila es lo único que usted seleccionó. Por ejemplo, para verificar la contraseña digitada por un usuario, sólo necesita obtener la contraseña del usuario de la base de datos y compararla con la contraseña que el usuario digitó. Los siguientes enunciados revisan una contraseña:

```
$entradadelusuario = "secreta";   // contraseña digitada por
          el usuario en el formulario
$consulta = "SELECT contraseña FROM Miembro
                WHERE nombredeentrada='gsmith'";
$resultado = mysql_query($consulta)

     or    die ("No se pudo ejecutar la consulta.");
$fila = mysql_fetch_array($resultado,MYSQL_ASSOC);
if ( $entradadelusuario == $fila['clave'] )
{
    echo "Entrada aceptada<br>";
  enunciados que muestran las paginas web exclusiva para miembros
}
else
{
    echo "Clave invalida<br>";
   enunciados que permiten al usuario probar otra contraseña
}
```

Observe los siguientes puntos sobre los enunciados anteriores:

- La consulta `SELECT` solicita únicamente un campo (`contraseña`) de una fila (la fila para `gsmith`).
- La función `mysql_fetch_array` devuelve un arreglo llamado `$fila` con nombres de columnas como claves.
- El enunciado `if` compara la contraseña que el usuario digitó (`$entradadeusuario`) con la contraseña obtenida de la base de datos (`$fila['contraseña']`) para ver si son iguales, usando los dos signos de igual (`==`).
- Si la comparación es `verdadera`, las contraseñas concuerdan, y el bloque `if` (el cual muestra las páginas web exclusivas para miembros) se ejecuta.
- Si la comparación no es `verdadera`, el usuario no digitó una contraseña que concuerda con la contraseña almacenada en la base de datos, y el bloque `else` se ejecuta. El usuario ve un mensaje de error que le indica que la contraseña no es correcta y se le devuelve a la página web de registro.

PHP brinda un atajo que resulta conveniente para usar variables recuperadas con la función `mysql_fetch_array`. Usted puede usar la función `extract`, la cual divide el arreglo en variables que tienen el mismo nombre que la clave. Por ejemplo, puede usar la función `extract` para reescribir los enunciados anteriores para comprobar la contraseña. Así se hace:

```
$entradadeusuario = "secreta"; #clave que el usuario digito
          en el formulario HTML
$consulta = "SELECT contraseña FROM Miembro
                WHERE nombredeentrada='gsmith'";
$resultado = mysql_query($consulta)
    or    die ("No se pudo ejecutar la consulta.");
$fila = mysql_fetch_array($resultado,MYSQL_ASSOC);
extract($fila);
if ( $entradadeusuario == $clave )
{
    echo "entrada aceptada<br>";
  enunciados que muestran las paginas web exclusivas para miembros
}
else
{
    echo "clave invalida<br>";
   enunciados que permiten al usuario probar otra contraseña
}
```

Usar un ciclo para obtener todas las filas de datos

Si seleccionó más de una fila de datos, use un ciclo para extraer todas las filas de la ubicación temporal. Los enunciados del ciclo en el bloque de ciclos extraen una fila de datos y la procesan. El ciclo se repite hasta que todas las filas se hayan recuperado. Puede usar un ciclo `while` o un ciclo `for` para recuperar esta información. (Para más detalles sobre los ciclos `while` y `for`, consulte el Capítulo 7.)

La manera más común de procesar la información es usar un ciclo `while` como sigue:

```
while ( $fila = mysql_fetch_array($resultado))
{
   bloque de enunciados
}
```

Este ciclo se repite hasta haber recuperado la última fila. Si usted deseara simplemente hacer eco de todos los datos, por ejemplo, usaría un ciclo parecido al siguiente:

```
while ( $fila = mysql_fetch_array($resultado))
{
   extract($fila);
   echo "$Tipo_mascota: $Nombre_mascota<br>";
}
```

Ahora, fíjese en el ejemplo de cómo obtener información para la aplicación Catálogo de mascotas. Asuma que el Catálogo de mascotas tiene una tabla llamada `Mascota` con cuatro columnas: `Idmascota`, `Tipomadescota`, `Descripcionmascota` y `precio`. La Tabla 8-1 muestra un conjunto de datos de muestra para la tabla `Mascota`.

Tabla 8-1 Datos de muestra para la tabla Mascota

Nombremascota	*Tipomascota*	*Descripciónmascota*	*Precio*
Unicornio	Caballo	Cuerno en espiral centrado en la frente	10000
Pegaso	Caballo	Volador; alas brotan de su espalda	15000
Pony	Caballo	Muy pequeño; la mitad del tamaño de un caballo estándar	500
Dragón asiático	Dragón	Cuerpo serpentino	30000
Dragón medieval	Dragón	Cuerpo parecido al de una lagartija	30000
León	Gato	Grande; melenudo	2000
Grifo	Gato	Cuerpo de león; cabeza de águila; alas	25000

El programa `listadodemascota.php` en la Lista 8-1 selecciona todos los caballos de la tabla `Mascota` y despliega la información en una tabla HTML en la página web. La variable `$tipomascota` contiene información que un usuario digitó en un formulario.

Lista 8-1: Muestra elementos del Catálogo de mascotas

```
?php
/* Programa: listadodeMascotas.php
 * Descripcion: Muestra todas las mascotas en la categoria
          seleccionada.
 */
?>
<html>
<head><title>Catalogo de mascotas</title></head>
<body>
<?php
  $usuario="catalogo";
  $huesped="localhost";
  $clave="";
  $basededatos = "Catalogodemascotas";
  $conexion = mysql_connect($huesped,$usuario,$clave)
       or die ("No se pudo conectar al servidor");
  $db = mysql_select_db($basededatos,$conexion)
       or die ("No se pudo seleccionar la base de datos");
  $tipomascota = "caballo";  //caballo fue digitado en un
          formulario por el usuario
  $consulta = "SELECT * FROM Mascota WHERE
          Tipomascota='$tipomascota'";
  $resultado = mysql_query($consulta)
       or die ("No se pudo ejecutar la consulta.");

  /* Mostrar resultados en una tabla */
  $tipo_mascota = ucfirst($tipomascota)."s";
```

```
    echo "<tr><td colspan='3'><hr></td></tr>";
    while ($fila = mysql_fetch_array($resultado))
    {
        extract($fila);
        $f_precio = number_format($precio,2);
         echo "<tr>\n
              <td>$Nombremascota</td>\n
              <td>$Descripcionmascota</td>\n
              <td align='right'>\$$f_precio</td>\n
              </tr>\n";
        echo "<tr><td colspan='3'><hr></td></tr>\n";
    }
    echo "</tabla>\n";
?>
</body></html>
```

La Figura 8-1 muestra la página web desplegada por el programa de la Lista 8-1. La página web muestra los elementos `Mascota` para el `Tipodemascota` caballo; el despliegue está formateado como una tabla HTML.

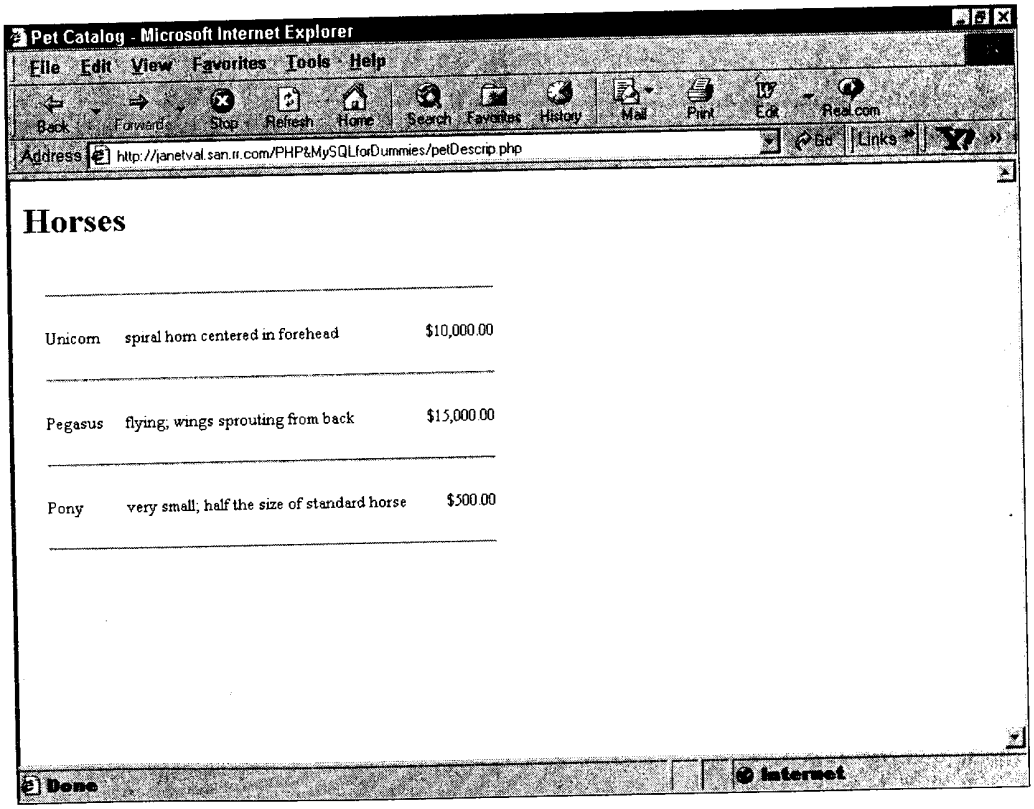


Figura 8-1: La página web resultante de `listado-demascotas.php`.

El programa de la Lista 8-1 usa un ciclo `while` para extraer todas las filas de la ubicación temporal. En algunos casos, quizás usted necesite usar un ciclo `for`. Por ejemplo, si debe usar un número en su ciclo, un ciclo `for` es más útil que uno `while`.

Para usar un ciclo `for`, necesita saber cuántas filas de datos se seleccionaron. Puede averiguar cuántas filas hay en el almacenaje temporal usando la función PHP `mysql_num_rows` como sigue:

```
$nfilas = mysql_num_rows($resultado);
```

La variable `$nfilas` contiene el número de filas almacenadas en la ubicación temporal. Usando este número, usted puede construir un ciclo `for` para obtener todas las filas, como sigue:

```
for ($i=0;$i<$nfilas;$i++)
{
   $fila = mysql_fetch_array($resultado))
   bloque de enunciados;
}
```

Por ejemplo, el programa en la Lista 8-1 muestra los elementos `Mascota` del tipo `caballo`. Suponga que usted desea numerar cada elemento. La Lista 8-2 muestra un programa, `listanumerada.php`, el cual despliega una lista numerada usando un ciclo `for`.

Lista 8-2: Muestra una lista numerada de elementos del Catálogo de mascotas

```
<?php
/* Programa: listanNumerada.php
 * Descripcion: Muestra una lista numerada de todas las
          mascotas en la categoria seleccionada.
 */
?>
<html>
<head><title>Catalogo de mascotas</title></head>
<body>
<?php
  $usuario="catalogo";
  $huesped="localhost";
  $clave="";
  $basededatos = "Catalogodemascotas";
  $conexion = mysql_connect($huesped,$usuario,$clave)
       or die ("No se pudo conectar al servidor ");
  $db = mysql_select_db($basededatos,$conexion)
       or die ("No se pudo seleccionar la base de datos");
  $tipomascota = "caballo";  //caballo fue digitado en un
          formulario por un usuario
  $consulta = "SELECT * FROM Mascota WHERE
          tipomascota='$tipomascota'";
  $resultado = mysql_query($consulta)
       or die ("No se pudo ejecutar la consulta.");
  $nfilas = mysql_num_rows($resultado);
```

```
/* Mostrar los resultados en una tabla */
echo "<h1>Caballos</h1>";
echo "<table cellspacing='15'>";
echo "<tr><td colspan='4'><hr></td></tr>";
for ($i=0;$i<$nfilas;$i++)
{
   $n = $i + 1;   #sumar 1 para que los numeros no empiecen
         en 0
   $fila = mysql_fetch_array($resultado);
   extract($fila);
   $f_precio = number_format($precio,2);
    echo "<tr>\n
         <td>$n.</td>\n
         <td>$Nombremascota</td>\n
         <td>$Descripcionmascota</td>\n
         <td align='right'>\$$f_precio</td>\n
         </tr>\n";
    echo "<tr><td colspan='4'><hr></td></tr>\n";
 }
 echo "</tabla>\n";
?>
</body></html>
```

La Figura 8-2 muestra la página web que resulta de usar el ciclo `for` en este programa. Observe que un número aparece antes de cada elemento `Mascota` en esta página web.

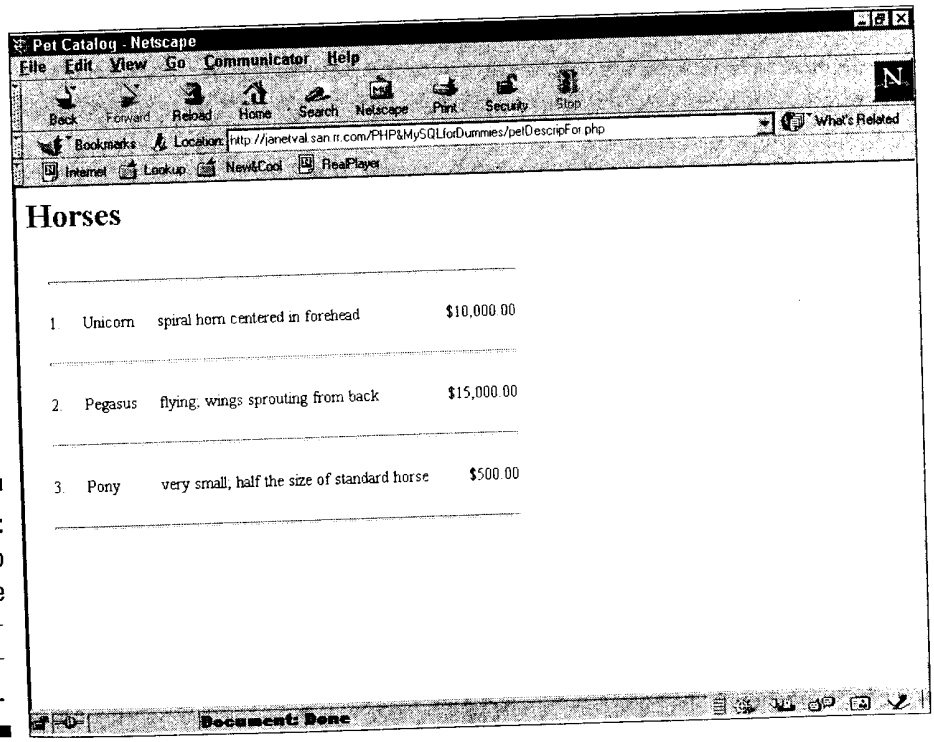


Figura 8-2: Página web resultante de `pet-Descrip-For.php`.

Usar funciones para extraer datos

En la mayoría de las aplicaciones, los datos se obtienen de la base de datos. A menudo, usted extrae datos en más de una ubicación de su programa o más de un programa en su aplicación. Las *funciones* (bloques de enunciados que realizan ciertas tareas específicas) están diseñadas para tales situaciones. (Le explico las funciones en detalle en el Capítulo 7.)

Una función para extraer datos de una base de datos puede ser realmente muy útil. Cada vez que el programa necesite extraer datos, usted llama a la función. Las funciones no sólo le ahorran mucha digitación, sino que también hacen que el programa sea más fácil de seguir.

Por ejemplo, considere un catálogo de productos, como el Catálogo de mascotas. Necesitará extraer información sobre un producto específico muchas veces. Puede escribir una función que extraiga los datos y luego usar dicha función cada vez que los necesite.

La lista 8-3 para el programa `obtienedatos.php` muestra cómo usar una función para extraer datos. La función en la Lista 8-3 obtendrá la información para cualquier mascota en el Catálogo de mascotas. La información sobre la mascota se pone en un arreglo, y el arreglo se devuelve al programa principal. El programa principal puede luego usar la información de cualquier forma que quiera. En este caso, hace eco de la información de la mascota en una página web.

Lista 8-3: Extrae datos de una base de datos usando una función

```
<?php
/* Programa: obtieneDatos.php
 * Descripcion: Extrae datos de una base de datos usando una
          funcion
 */
?>
<html>
<head><title>Catalogo de mascotas</title></head>
<body>
<?php
  $usuario="catalogo";
  $huesped="localhost";
  $clave="";
  $conexion = mysql_connect($huesped,$usuario,$clave)
       or die ("No se pudo conectar al servidor");

  $Infomascota = extraerInfomascota("Unicornio");
           //llamar funcion

  $f_precio = number_format($Infomascota['precio'],2);
  echo "<p><b>{$Infomascota['Nombremascota']}</b><br>\n
        Descripcion:
            {$Infomascota['Descripcionmascota']}<br>\n
        Precio: \${$Infomascota['precio']}\n"
```

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


<?php
function extraerInfomascota($Nombremascota)
{
  $db = mysql_select_db("Catalogodemascotas")
       or die ("No se pudo seleccionar la base de datos");
  $consulta = "SELECT * FROM Mascota WHERE
            Nombremascota='$Nombremascota'";
  $resultado = mysql_query($consulta)
       or die ("No se pudo ejecutar la consulta.");
  return mysql_fetch_array($resultado,MYSQL_ASSOC);
}
?>
```

La página web muestra

```
Unicornio
Descripcion: cuerno en espiral centrado en la frente
Precio: $10,000.00
```

Observe lo siguiente sobre el programa en la Lista 8-3:

- El programa es más fácil de leer con el llamado de función de lo que sería si todos los enunciados de la función estuvieran en el programa principal.
- Usted se puede conectar con el servidor MySQL una vez en el programa principal y llamar a la función muchas veces para extraer los datos. Si la conexión estuviera en la función en lugar de en el programa principal, se conectaría cada vez que usted llamara la función. Es más eficiente conectarse sólo una vez, si es posible.
- Si tiene sólo una conexión, `mysql_select_db` usará esa conexión. Si tiene más de una conexión, usted puede pasar la conexión y usarla en su llamado de función `mysql_select_db`. Si su aplicación sólo usa una base de datos, usted puede seleccionarla una vez en el programa principal en lugar de seleccionarla en la función.
- El llamado de función envía la cadena `"Unicornio"`. En la mayoría de los casos, el llamado de función usará el nombre de una variable.
- El programa crea la variable `$Infomascota` para recibir los datos de la función. `$Infomascota` es un arreglo porque la información almacenada en el es un arreglo.

La función anterior es muy simple: devuelve una fila de los resultados como arreglo. Pero las funciones pueden ser más complejas. La sección anterior proporciona un programa para extraer todas las mascotas de un tipo específico. El programa `obtienemascotas.php` en la Lista 8-4 usa una función para el mismo propósito. La función devuelve un arreglo multidimensional con los datos de mascota para todas las mascotas del tipo especificado.

Lista 8-4: Muestra una lista numerada de mascotas usando una función

```
<?php
/* Programa: obtieneMascotas.php
 * Descripcion: Muestra una lista numerada de elementos de
          una base de datos.
 */
?>
<html>
<head><title>Catalogo de mascotas</title></head>
<body>
<?php
  $usuario="catalogo";
  $huesped="localhost";
  $clave="";
  $conexion = mysql_connect($huesped,$usuario,$clave)
       or die ("No se pudo conectar al servidor");

  $Info_mascotas = extraertipodemascotas("caballo");
          //llamar funcion

  /* Mostrar los resultados en una tabla */
  echo "<h1>Caballos</h1>";
  echo "<table cellspacing='15'>";
  echo "<tr><td colspan='4'><hr></td></tr>";
  for ($i=1;$i<=pesode($Infomascotas);$i++)
  {
      $f_precio = number_format
           ($Infomascotas[$i]['precio'],2);
      echo "<tr>\n
           <td>$i.</td>\n
           <td>{$Infomascotas[$i]['Nombremascota']}</td>\n
           <td>{$nfomascota[$i]['Descripcionmascota']}</td>\n
           <td align='right'>\$$f_precio</td>\n
           </tr>\n";
     echo "<tr><td colspan='4'><hr></td></tr>\n";
  }
  echo "</tabla>\n";
?>
</body></html>

<?php
function extraertipodemascotas($Tipomascota)
{
  $db = mysql_select_db("Catalogodemascotas")
       or die ("No se pudo seleccionar la base de datos");
  $consulta = "SELECT * FROM Mascota WHERE
          Tipomascota='$Tipomascota'";
  $resultado = mysql_query($consulta)
       or die ("No se pudo ejecutar la consulta.");

  $j = 1;
  while ($fila=mysql_fetch_array($resultado,MYSQL_ASSOC))
  {
```

```
        foreach ($fila como $nombrecolumna => $valor)
        {
            $serie[$j][$nombrecolumna] = $valor;
        }
        $j++;
    }
    return $serie;
}
?>
```

El programa en la Lista 8-4 procede como sigue:

1. **Se conecta con el servidor MySQL en el programa principal.**
2. **Llama a la función** `extraertipodemascotas`**.** Pasa `"caballo"` como una cadena de caracteres y también establece `$Infomascotas` para que pueda recibir los datos devueltos por la función.
3. **La función selecciona la base de datos** `Catologodemascotas`**.**
4. **La función envía una consulta para extraer todas las filas con** `caballo` **en la columna** `Tipodemascota`**.** Los datos se almacenan en una tabla en una ubicación temporal. La variable `$resultado` identifica la ubicación de la tabla temporal.
5. **Establece un contador.** `$j` es un contador que se incrementa en cada ciclo. Empieza en 1 antes del ciclo.
6. **Inicia un ciclo** `while`**.** La función intenta extraer una fila de la tabla temporal de datos y resulta exitosa. Si no hubiera filas para extraer en la ubicación temporal, el ciclo `while` terminaría.
7. **Inicia un ciclo** `foreach`**.** El ciclo se mueve por la fila y procesa cada campo.
8. **Almacena los valores en el arreglo.** `$Infomascotas` es un arreglo multidimensional. Su primera clave es un número, el cual está establecido por el contador. Como esta es la primera vez que se atraviesa el ciclo `while`, el contador, `$j`, es ahora igual a `1`. Todos los campos en la fila se almacenan en `$Infomascotas` con el nombre de la columna como clave. (Explico las series multidimensionales en detalle en el Capítulo 7.)
9. **Incrementa el contador.** `$j` aumenta en 1.
10. **Llega al final del ciclo** `while`**.**
11. **Regresa al inicio del ciclo** `while`**.**
12. **Repite los Pasos 6 — 11 para cada fila en los resultados.**
13. **Devuelve** `$serie` **al programa principal.** `$serie` contiene todos los datos para todas las filas seleccionadas.
14. `$Infomascotas` **recibe datos de la función.** Todos los datos se pasan. La Figura 8-3 muestra la estructura de `$Infomascotas` después de que la función ha terminado de ejecutarse.

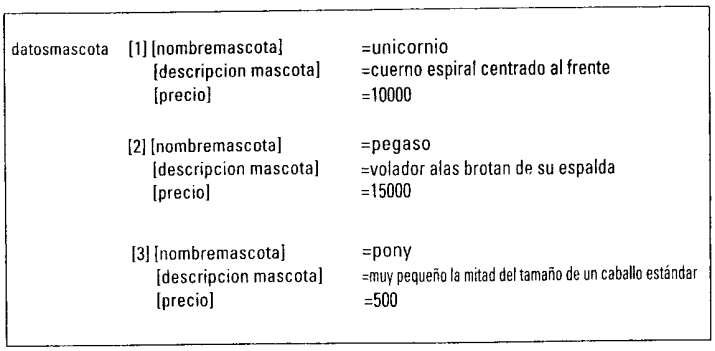


Figura 8-3: La estructura del arreglo multidimensional `$Infomascotas`.

15. **El programa principal envía las Descripciones de las mascotas al explorador en una tabla HTML.** Los datos apropiados se insertan desde el arreglo `$Infomascotas`.

La página web resultante del programa en la Lista 8-4 es idéntica a la página web mostrada en la Figura 8-2, la cual es producida por un programa que no usa una función. Las funciones no producen output diferente. Cualquier programa que se pueda escribir usando una función, también puede escribirse sin usar la función. Las funciones sólo facilitan la programación.

Obtener información del usuario

Muchas aplicaciones están diseñadas para formular preguntas que los usuarios responden digitando información. A veces, la información se almacena en una base de datos; otras, la información se usa en enunciados condicionales para entregar una página web individual. Algunas de las tareas más comunes de las aplicaciones que requieren que los usuarios respondan preguntas son

- **Hacer pedidos en línea:** Los clientes deben seleccionar productos y digitar la información sobre el envío y el pago.
- **Inscribirse:** Muchos sitios requieren que los usuarios proporcionen alguna información antes de recibir ciertos beneficios, tales como tener acceso a información especial o a software que se puede descargar.
- **Registrarse:** Muchos sitios restringen el acceso a sus páginas. Los usuarios deben digitar el nombre de una cuenta y una contraseña antes de poder ver las páginas web.
- **Ver información seleccionada:** Muchos sitios permiten a los usuarios especificar cuál información desean ver. Por ejemplo, un catálogo en línea podría permitir a los usuarios digitar el nombre del producto o seleccionar una categoría de productos que desean ver.

Las preguntas se formulan mediante formularios HTML. El usuario responde las preguntas digitando información en el formulario o seleccionando ele-

mentos de una lista. El usuario luego hace clic en un botón para enviar la información del formulario. Cuando el formulario es enviado, la información en él se pasa a un segundo programa separado, el cual procesa la información.

En las siguientes secciones, no le informo sobre el HTML requerido para mostrar un formulario; asumo que usted ya conoce HTML. (Si no lo conoce o necesita un refrescamiento, consulte *HTML 4 For Dummies, 4ta Edición*, por Ed Tittel y Natanya Pitts; Wiley Publishing, Inc.). Lo que sí le digo es cómo usar PHP para desplegar los formularios en HTML y procesar la información que los usuarios digitan en el formulario.

Usar formularios HTML

Los formularios HTML son muy importantes para los sitios web interactivos. Si usted no conoce los formularios HTML, deberá leer la sección de formularios en algún libro sobre HTML. Para desplegar un formulario usando PHP, puede hacer alguna de las siguientes operaciones:

- **Use enunciados** `echo` **para hacer eco del HTML para un formulario.** Por ejemplo:

```
<?php
  echo "<form action='procesaformulario.php'
              method='POST'>\n
     <input type='texto' nombre='nombre'>\n
     <input type='submit' valor='enviar nombre'>\n
     </form>\n";
?>
```

- **Use HTML simple fuera de las secciones en PHP.** Para un formulario estático simple, no hay razón para incluirlo en una sección en PHP. Por ejemplo:

```
<?php
  enunciados en la seccion PHP
?>
<form action="procesaformulario.php" method="POST">
<input type="texto" nombre="Nombre">
<input type="submit" valor="enviar nombre">
</form>
<?php
 enunciados en la seccion PHP
?>
```

Cualquiera de estos métodos produce el formulario que se muestra en la Figura 8-4.

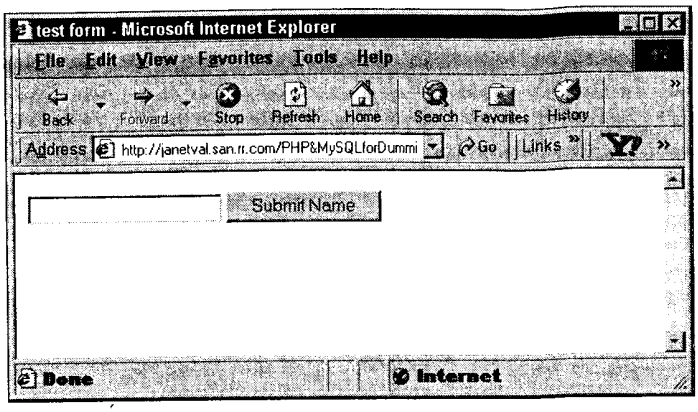


Figura 8-4: Un formulario producido por enunciados HTML.

Juan Cliente llena el formulario HTML. Hace clic en el botón Submit. Ahora usted tiene la información que deseaba: su nombre. Entonces, ¿dónde está? ¿Cómo puede tener acceso a él?

Usted obtiene la información en el formulario corriendo el programa que recibe la información del formulario. Cuando se hace clic en el botón Submit, PHP automáticamente corre el programa. El parámetro `action` en la etiqueta del formulario le dice a PHP cuál programa correr. Por ejemplo, en el programa anterior, el parámetro `action=procesaformulario.php` le dice a PHP que corra el programa *`procesaformulario`*`.php` cuando el usuario hace clic en el botón Submit. El programa `procesaformulario.php` puede mostrar, almacenar o usar de cualquier otro modo los datos en el formulario que recibe cuando el formulario es enviado.

Cuando el usuario hace clic en el botón Submit, el programa especificado en el atributo `action` corre, y los enunciados en dicho programa pueden extraer la información del formulario a partir de series PHP incorporadas y usar la información en enunciados PHP. La Tabla 8-2 muestra las series incorporadas que contienen la información del formulario.

Tabla 8-2 Series incorporadas con información de formularios

Serie	*Descripción*
`$ _POST`	Contiene elementos para todos los campos incluidos en un formulario si el formulario usa `method="POST"`.
`$HTTP_POST_VARS`	Igual que `$ _POST`.
`$ _GET`	Contiene todas las variables pasadas desde una página anterior como parte del URL. Esto incluye los campos pasados en un formulario usando `method="get"`.
`$HTTP_GET_VARS`	Igual que `$ _GET`.
`$_REQUEST`	Contiene todos los elementos de las series que están en `$_POST`, `$_GET` y `$_COOKIE`.

Cuando el formulario es enviado, el programa que corre puede obtener la información del formulario a partir del arreglo incorporado apropiado, como se muestra en la Tabla 8-2. En estos arreglos incorporados, cada índice de serie es el nombre del campo de input en el formulario. Por ejemplo, si el usuario digitó **Goliat Pérez** en el campo de input mostrado en la Figura 8-4 e hizo clic en el botón Submit, el programa `procesaformulario.php` corre y puede usar una variable de arreglo en el formato siguiente:

```
$_POST['nombre']
```

Observe que el nombre digitado en el formulario está disponible en el arreglo `$_POST` porque la etiqueta del formulario especificada `method='POST'`. Además, observe que la clave de el arreglo es el nombre dado en el campo en el formulario HTML con el atributo `nombre` asi: `nombre="fullname"`.

Un programa que muestra todos los campos en un formulario es un programa útil para probar el formulario. Usted puede ver cuáles valores se pasan desde el formulario para asegurarse de que su formulario esté formateado apropiadamente y envía los nombres de campos y los valores que usted esperaba. Todos los campos en un formulario del tipo `POST` son desplegados por el programa en la Lista 8-5, llamado `procesaformulario.php`. Cuando el formulario mostrado en la Figura 8-4 es enviado, se corre el siguiente programa.

Lista 8-5: Un srcipt que muestra todos los campos de un formulario

```
<?php
/*  Nombre del programa:  procesaformaulario.php
 *  Descripcion: El programa muestra toda la informacion que
 *          se paso de un formulario.
 */
  echo "<html>
        <head><title>Direccion del Cliente</title></head>
        <body>";
  foreach ($_POST as $campo => $valor)
  {
     echo "$campo = $valor<br>";
  }
?>
</body></html>
```

Si el usuario digitó el nombre *Goliat Pérez* en el formulario de la Figura 8-4, se despliega el siguiente output:

```
nombre = Goliat Perez
```

El resultado sólo muestra una línea porque solamente hay un campo en el formulario de la Figura 8-4.

El programa en la Lista 8-5 se escribe para procesar la información de cualquier formulario que use el método `POST`. Suponga que usted tiene un formu-

lario levemente más complicado, tal como el programa en la Lista 8-6, el cual despliega un formulario con varios campos.

Lista 8-6: Un programa que muestra un formulario de direcciones

```
<?php
/*  Nombre del Programa: muestraFormulario.php
 *  Descripcion: Los enunciados muestran un formulario que
          pide la direccion del cliente.
 */
  echo "<html>
        <head><title>Direccion del Cliente</title></head>
        <body>";
  $etiquetas = serie( "primernombre"=>"Primer Nombre:",
                      "segundonombre"=>"Segundo Nombre:",
                      "apellido"=>"Apellido:",
                      "calle"=>"Direccion de la calle:",
                      "ciudad"=>"Ciudad:",
                      "estado"=>"Estado:",
                      "codigopostal"=>"Codigo_postal:");
  echo "<p align='center'>
        <b>Por favor digite su direccion a
           continuacion.</b><hr>";
  echo "<form action='procesaformulario.php' method='POST'>
        <table width='95%' border='0' cellspacing='0'
              cellpadding='2'>\n";
  foreach($etiquetas como $campo=>$etiqueta)
  {
    echo "<tr>
          <td align='right'> <B>{$etiquetas[$campo]}
           </br></td>
          <td><input type='text' name='$campo' size='65'
             maxlength='65' ></td>
          </tr>";
  }
  echo "</tabla>
        <div align='center'><p><input type='submit'
           value='Enviar Direccion'> </p></div>
        </form>";
?>
</body></html>
```

Observe lo siguiente en cuanto a `muestraformulario.php`, tal y como aparece en la Lista 8-6:

- **Se crea un arreglo que contiene las etiquetas usadas en el formulario.** Las claves son los nombres de los campos.
- **El script `procesaformulario.php` lleva el nombre del script que corre cuando el formulario se envía.** La información en el formulario se envía a `procesaformulario.php`, el cual procesa la información.

- **El formulario se formatea con una tabla HTML.** Las tablas son una parte importante del HTML. Si usted no está familiarizado con las tablas HTML, consulte *HTML 4 For Dummies, 4ta Edición*, por Ed Tittel y Natanya Pitts (Wiley Publishing, Inc.).
- **El script hace un ciclo por el arreglo `$etiquetas` con un enunciado foreach.** El código HTML para una fila de la tabla es el output de cada ciclo. Los valores de serie apropiados se usan en el código HTML.

Por razones de seguridad, siempre incluya `maxlength` (define el número de caracteres que los usuarios pueden digitar en el campo) en su enunciado HTML. Limitar el número de caracteres ayuda a prevenir que personas malintencionadas digiten códigos maliciosos en los campos de su formulario. Si la información será almacenada en una base de datos, establezca `longitudmaxima` al mismo número del ancho de la columna en la tabla de la base de datos.

Cuando Goliat Pérez rellena el formulario mostrado en la Figura 8-5 (creado por el programa en la Lista 8-6) y lo envía, el programa `procesaformulario.php` corre y produce el siguiente output:

```
primernombre = Goliat
segundonombre =
apellido = Perez
calle = 1234 calle Alta
ciudad = Gran Ciudad
estado = TX
codigo postal = 88888
```

En `procesaformulario.php`, todos los elementos del arreglo incorporado `$_POST` se despliegan, ya que ambos formularios mostrados en esta sección usaron el método `POST`, tal como lo hace la mayoría de los formularios. Hay otras series incorporadas, además del arreglo `$_POST`, como se aprecia en la Tabla 8-2.

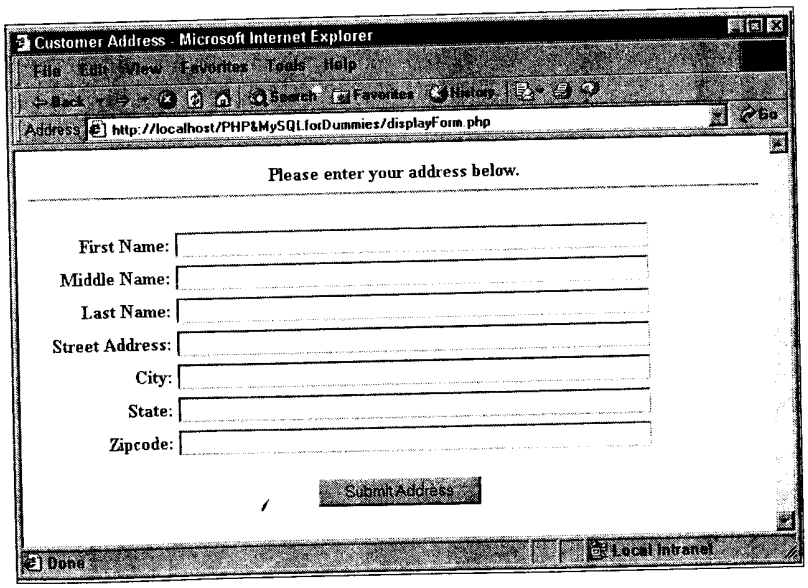


Figura 8-5: Formulario para introducir la dirección del cliente.

La misma información está disponible en dos conjuntos de arreglos. Use los arreglos más nuevos (cuyos nombres empiezan con _) porque pueden usarse en cualquier parte, incluso dentro de una función. (Le explico las funciones y el uso de variables dentro de las funciones en el Capítulo 7). Estas series, llamadas *superglobales o autoglobales*, fueron introducidas en PHP 4.1.0. Los arreglos más viejos, con nombres largos como `$HTTP_POST_VARS`, deben hacerse globales antes de poder usarse en una función, como explico en el Capítulo 7. Las series más viejas sólo deberían usarse cuando usted se ve forzado a usar una versión de PHP más vieja que PHP 4.1.0.

Hacer que los formularios sean dinámicos

PHP trae nuevas capacidades a los formularios HTML. Como usted puede usar variables en los formularios PHP, sus formularios ahora pueden ser dinámicos. Estas son las principales capacidades que PHP proporciona a los formularios:

- Usar variables para mostrar la información en los campos de texto de entrada
- Usar variables para crear listas dinámicas de las cuales los usuarios pueden hacer selecciones
- Usar variables para construir listas dinámicas de botones de opción
- Usar variables para construir listas dinámicas de casillas para marcar

Desplegar información dinámica en los campos del formulario

Cuando usted despliega un formulario en una página web, puede poner información en los campos en lugar de sólo mostrar un campo vacío. Por ejemplo, si la mayoría de sus clientes vive en Estados Unidos, podría automáticamente introducir **EEUU** en el campo del país cuando pide a los clientes su dirección.

Registrar arreglos largos

Una nueva configuración `php.ini`, introducida en PHP 5, le permite evitar que PHP cree los viejos y largos arreglos automáticamente. Es altamente improbable que usted necesite usarlas, a no ser que esté utilizando algunos scripts viejos que contengan las variables largas. La línea siguiente en `php.ini` controla esta configuración:

```
register_long_arrays = On
```

En este momento, esta configuración está activada (On) en forma predeterminada. A menos que usted corra los viejos scripts que necesitan los viejos arreglos, debería cambiar la configuración a desactivada (`Off`), para que PHP no haga este trabajo adicional.

Aunque la configuración actual predeterminada es `On`, eso podría cambiar. La configuración predeterminada podría cambiar a `Off` en una versión futura. Si está usando scripts viejos y obtiene errores en líneas que contienen las series largas, como `$HTTP_GET_VARS`, verifique la configuración de php.ini para las series largas. Podría estar desactivada (`Off`), y los arreglos largos que necesitan los scripts más viejos no se están siendo creadas del todo.

Si el cliente realmente vive en los Estados Unidos, usted le habrá ahorrado algo de digitación. Y si el cliente no vive en los Estados Unidos, simplemente puede reemplazar EEUU con el país apropiado. Además, si el programa automáticamente ingresa EEUU como el valor en el campo, usted sabe que esa información no contendrá errores.

Para mostrar un campo de texto que contiene información, usa el siguiente formato para los enunciados HTML del campo de input:

```
<input type="text" name="pais" value="EEUU">
```

Al usar PHP, usted puede utilizar una variable para desplegar esta información con cualquiera de los siguientes enunciados:

```
<input type="text" name="pais"
       value="<?php echo $pais ?>">
echo "<input type='text' name='pais' value'$pais'>";
```

El primer ejemplo crea un campo de input en una sección HTML, usando una sección PHP corta sólo para el valor. El segundo ejemplo crea un campo de entrada usando un enunciado echo dentro de una sección PHP. Si usted está usando un formulario largo con sólo una variable ocasional, usar el primer formato resulta más eficiente. Si su formulario usa muchas variables, es más eficiente usar el segundo formato.

Si tiene información de los usuarios almacenada en una base de datos, podría querer mostrar la información de la base de datos en los campos del formulario. Por ejemplo, podría mostrar la información al usuario, de modo que éste pueda hacer cualquier cambio necesario. O bien, usted podría mostrar la dirección de envío para el último pedido en línea del cliente, de manera que no haya necesidad de volver a digitar la dirección. La Lista 8-7 presenta el programa `muestradireccion.php`, el cual despliega un formulario con información de la base de datos. Este formulario es muy parecido al formulario mostrado en la Figura 8-5, excepto que este formulario tiene información en él (recuperada de la base de datos) y los campos en el formulario en la Figura 8-5 están vacíos.

Lista 8-7: Programa para mostrar un formulario HTML con información

```
<?php
/*  Nombre del programa: displayAddress
 *  Descripcion:  El script muestra un formulario con
                  informacion sobre direcciones obtenido de la ase
                  de datos.

 */
  echo "<html>
        <head><title>Direccion del Cliente</title></head>
        <body>";
  $etiquetas = serie( "primernombre"=>"Primer Nombre:",
                      "segundoNombre"=>"Apellido:",
                      "calle"=>"Direccion de la calle:",
                      "ciudad"=>"Ciudad:",
```

```
                    "estado"=>"Estado:",
                    "codigopostal"=>"Codigo_postal:");
$usuario="admin";
$huesped="localhost";
$clave="";
$basededatos = "DirectoriodeMiembros";
$nombreentrada = "gperez";     // nombre de registro del
    usuario

$conexion = mysql_connect($huesped,$usuario,$clave)
     or die ("No me pude conectar al servidor");
$db = mysql_select_db($basededatos,$conexion)
     or die ("No se pudo seleccionar la base de datos");
$consulta = "SELECT * FROM Miembro
                    WHERE nombreentrada='$nombreentrada'";
$resultado = mysql_query($consulta)
     or die ("No se pudo ejecutar la consulta.");
$fila = mysql_fetch_array($resultado);

echo "<p align='center'>
      <h1 align='center'>Direccion de
         $nombreentrada</h1>\n";
echo "<br><p align='center'>
      <font size='+1'><b>Por favor verifique la informacion
         a continuacion y cambie cualquier dato que esta
         incorrecto.</b></font>
      <hr>";
echo "<form action='procesaDireccion.php' method='POST'>
      <table width='95%' border='0' cellspacing='0'
            cellpadding='2'>\n";
foreach($etiquetas as $campo=>$etiqueta)
{
  echo "<tr>
        <td align='right'> <B>{$etiqueta[$campo]}
         </br></td>
        <td><input type='text' name='$campo'
             value='$fila[$campo]' size='65'
         maxlength='65'>
        </td>
        </tr>";
}
echo "</table>
      <div align='center'><p><input type='submit'
         value='Enviar Direccion'> </p></div>
      </form>";
?>
</body></html>
```

Observe lo siguiente en el programa en la Lista 8-7:

- **El enunciado del formulario transfiere la acción al programa** `procesadireccion.php`. Este programa procesa la información en el formulario y actualiza la base de datos con cualquier información que el usuario cambió. Este es un programa que usted mismo escribe. Verificar los datos en un formulario y guardar información en la base de datos se discuten más adelante en este ca-

pítulo, en las secciones "Revisar la información" y "Poner información en una base de datos".

- **Cada campo de input en el formulario recibe un nombre.** La información en el campo de input se almacena en una variable que tiene el mismo nombre que el campo de input.
- **El programa da a los nombres de campos en el formulario los mismos nombres de las columnas en la base de datos.** Esto simplifica el movimiento de la información entre la base de datos y el formulario; no requiere de la transferencia de información de una variable a otra.
- **Los valores de la base de datos se muestran en los campos del formulario con el parámetro valor en el enunciado del campo de input.** El parámetro `valor` muestra el valor apropiado del arreglo `$fila`, la cual contiene datos de la base de datos.

Por razones de seguridad, siempre incluya `maxlength` en su enunciado HTML. `maxlength` define el número de caracteres que un usuario puede digitar en el campo. Si la información será almacenada en una base de datos, establezca `maxlength` al mismo número del ancho de la columna en la tabla de la base de datos.

La Figura 8-6 muestra la página web resultante del programa en la Lista 8-7. La información en el formulario es la información que está almacenada en la base de datos.

Construir listas de selección

Un tipo de campo que usted puede usar en un formulario HTML es una lista de selección. En vez de digitar en un campo, sus usuarios seleccionan de una lista. Por ejemplo, en un catálogo de productos, usted podría proveer una lista de categorías entre las cuales los usuarios pueden seleccionar lo que desean ver. O bien, el formulario para las direcciones de los usuarios podría incluir una lista de países que los usuarios pueden seleccionar. O los usuarios podrían ingresar una fecha seleccionando el mes, día y año de una lista.

Use las listas de selección siempre que sea factible. Cuando el usuario selecciona un elemento de una lista, usted podrá estar seguro de que el elemento es correcto, sin errores ortográficos, caracteres extraños u otros problemas introducidos por los errores de digitación de los usuarios.

Una lista de selección en HTML para las categorías en el Catálogo de mascotas tiene el siguiente formato:

```
<form action="procesaformulario.php" method="POST">
<select name="Tipomascota">
  <value de la option="caballo">caballo
  < value de la option ="gato" selected>gato
  < value de la option ="dragon">dragon
</select>    
<input type="submit" value="Seleccione el tipo de mascota ">
</form>;
```

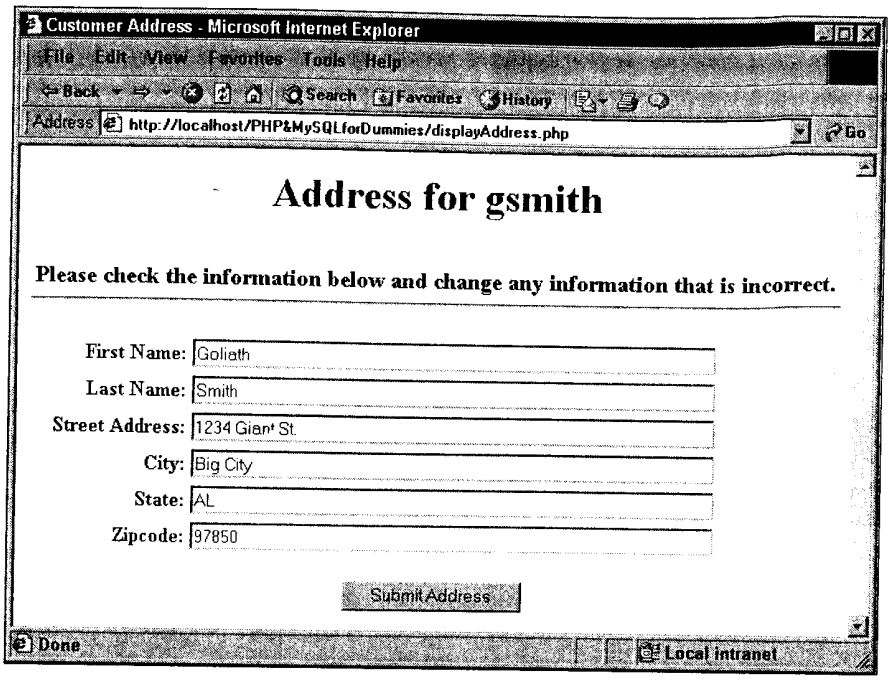


Figura 8-6: Formulario que muestra la dirección del usuario.

La Figura 8-7 muestra la lista de selección que estos enunciados HTML producen. Observe que *gato* es la opción escogida cuando el campo se muestra por primera vez. Usted determina esta selección predeterminada incluyendo `selected` en la etiqueta de opción.

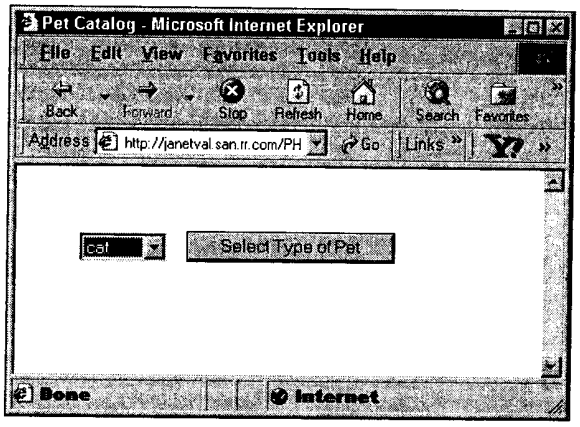


Figura 8-7: Un campo de selección para el Catálogo de mascotas

Cuando el usuario hace clic sobre la flecha en la casilla de lista desplegable de selección, toda la lista se despliega, como se aprecia en la Figura 8-8, y el usuario puede seleccionar cualquier elemento de la lista. Observe que gato está seleccionado hasta que el usuario seleccione un elemento diferente.

Al usar PHP, sus opciones pueden ser variables. Esta capacidad le permite construir listas de selección dinámicas. Por ejemplo, usted debe mantener la

lista estática de categorías de mascotas mostrada en el ejemplo anterior. Si añade una categoría nueva de mascota, debe agregar una etiqueta de opción manualmente. Sin embargo, con las variables PHP, usted puede construir la lista en forma dinámica a partir de las categorías en la base de datos. Cuando añada una nueva categoría en la base de datos, la nueva categoría se agregará automáticamente a su lista de selección sin necesidad de cambiar el programa PHP. La Lista 8-8 para el programa `construirselect.php` construye una lista de selección de categorías de mascotas a partir de la base de datos.

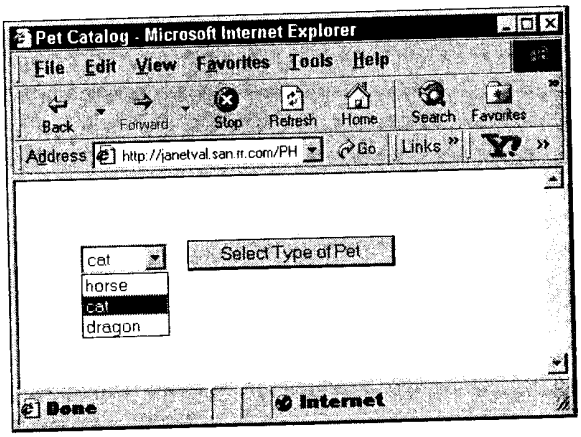


Figura 8-8: Un campo de selección para el Catálogo de mascotas con una lista desplegable.

Lista 8-8: Programa para construir una lista de selección

```
<?php
/*  Nombre del programa: construirSelect.php
 *  Descripcion:  El programa construye una lista de
          seleccion a partir de la base de datos.
 */
?>
<html>
<head><title>Tipos de mascotas</title></head>
<body>
<?php
  $usuario="catalogo";
  $huesped="localhost";
  $clave="";
  $basededatos = "CatalogodeMascotas";

  $conexion = mysql_connect($huesped,$usuario,$clave)
       or die ("No se pudo conectar al servidor ");
  $db = mysql_select_db($basededatos,$conexion)
       or die ("No se pudo seleccionar la base de datos");
  $consulta = "SELECT DISTINCT Tipomascota FROM Mascota ORDER
          BY Tipomascota";
  $resultado = mysql_query($consulta)
       or die ("No se pudo ejecutar la consulta.");
```

(continúa)

```
    /* crea un formulario que contiene la lista de seleccion */
    echo "<form action='procesaformulario.php' method='POST'>
          <select name='Tipomascota'>\n";

    while ($fila = mysql_fetch_array($resultado))
    {
       extract($fila);
       echo "<option value='$Tipomascota'>$Tipomascota\n";
    }
    echo "</select>\n";
    echo "<input type='submit' value='Seleccionar tipo de
             mascota'>
          </form>\n";
?>
</body></html>
```

Observe lo siguiente en el programa de la Lista 8-8:

- **Uso de `DISTINCT` en la consulta:** `DISTINCT` causa que la consulta extraiga cada tipo de mascota sólo una vez. Sin `DISTINCT`, la consulta devolvería cada tipo de mascota varias veces si apareciera varias veces en la base de datos.
- **Uso de `ORDER BY` en la consulta:** Los tipos de mascota se ordenan alfabéticamente.
- **Enunciados `echo` antes del ciclo:** Se hace `echo` de las etiquetas `form` y `select` antes de que el ciclo `while` empiece, porque se les hace `echo` sólo una vez.
- **Enunciados `echo` en el ciclo:** A las etiquetas `option` se les hace `echo` en el ciclo: una vez para cada tipo de mascota en la base de datos. Ningún elemento está marcado como seleccionado, por lo cual el primer elemento en la lista se selecciona automáticamente.
- **Enunciados `echo` después del ciclo:** A las etiquetas finales `form` y `select` se les hace `echo` después del ciclo porque se les hace `echo` sólo una vez.

La lista de selección producida por este programa es inicialmente la misma que la de la Figura 8-7, con *gato* seleccionado. Sin embargo, gato está seleccionado en este programa porque es el primer elemento en la lista, y no por estar seleccionado específicamente (como lo está en las etiquetas HTML que producen la Figura 8-7). La lista desplegable producida por este programa está en orden alfabético, como se aprecia en la Figura 8-9.

También puede usar variables PHP para establecer cuál opción está seleccionada cuando la casilla de selección aparece. Por ejemplo, suponga que usted desea que el usuario seleccione una fecha de las listas de selección mes, día y año. Usted cree que la mayoría de las personas seleccionará la fecha de hoy; por eso, quiere que la fecha de hoy esté seleccionada en forma predeterminada cuando la casilla se despliega. La Lista 8-9 muestra el programa `seleccionarfecha.php`, el cual despliega un formulario para seleccionar una fecha y selecciona la fecha de hoy automáticamente.

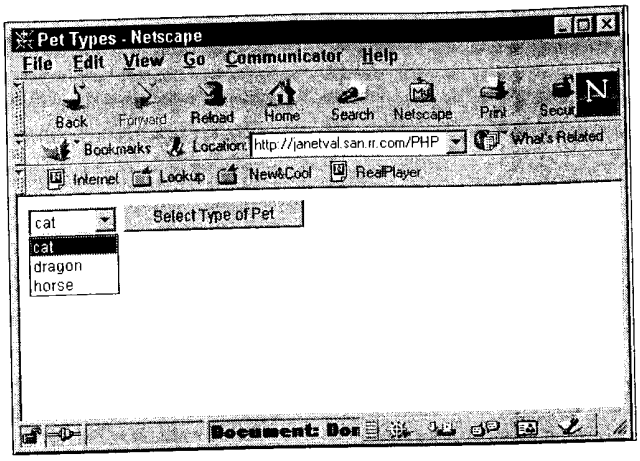


Figura 8-9: Un campo de selección para el Catálogo de mascotas producido por el programa `buildSelect.php`.

Lista 8-9: Programa para crear una lista de selección para la fecha

```
<?php
/*  Nombre del programa: seleccionarFecha.php
 *  Descripcion: El programa muestra una lista de seleccion
        que los clientes pueden usar para seleccionar una
        fecha.
 */
echo "<html>
      <head><title>Seleccione una fecha</title></head>
      <body>";

  /* crea una serie de meses */
  $nombremes = array(1=> "enero", "febrero", "marzo",
                         "abril", "mayo", "junio", "julio",
                         "agosto", "setiembre", "octubre",
                         "noviembre", "diciembre");

  $hoy = date();                          //almacena la fecha de
         hoy
  $f_hoy = date("M-D-A",$hoy);      //da formato a la fecha de
         hoy

  echo "<div align='center'>\n";

  /* muestra la fecha de hoy */
  echo "<p> <h3>Hoy es $f_hoy</h3><hr>\n";

  /* crea un formulario que contiene una lista de seleccion
         para la fecha */
  echo "<form action='procesaformulario.php'
         method='POST'>\n";

  /* construye una lista de seleccion para el mes */
  $hoymes = date("m",$hoy);  //extrae el mes en $hoy
  echo "<select name='fechames'>\n";
  for ($n=1;$n<=12;$n++)
```

Lista 8-9: (continuación)

```
{
   echo "<option value=$n";
   if ($hoymes == $n)
   {
     echo " checked";
   }
   echo "> $nombremes[$n]\n";
 }
 echo "</select>";

 /* construye la lista de seleccion para el dia */
 $hoydia= date("d",$hoy);    //extrae el dia de $hoy
 echo "<select name='fechadia'>\n";
 for ($n=1;$n<=31;$n++)
 {
   echo " <option value=$n";
   if ($diahoy == $n )
   {
     echo " checked";
   }
   echo "> $n\n";
 }
 echo "</select>\n";

 /* construye una lista de seleccion para el año */
 $annioinicio = fecha("Y", $hoy);  //obtenga el año de $hoy
 echo "<select name='fechaannio'>\n";
 for ($n=$annioinicio;$n<=$anniooinicio+3;$n++)
 {
   echo " <option value=$n";
   if ($anniooinicio == $n )
   {
     echo " selected";
   }
   echo "> $n\n";
 }
 echo "</select>\n";
 echo "</form>\n";
?>
</body></html>
```

La página web producida por el programa de la Lista 8-9 se muestra en la Figura 8-10. La fecha aparece sobre el formulario, de modo que pueda verse que la lista de selección muestra la fecha correcta. La lista de selección para el mes muestra los 12 meses al caer. La lista de selección para el día muestra 31 días al desplegarse. La lista de selección para el año muestra cuatro años.

El programa en la Lista 8-9 produce la página web de la Figura 8-10 siguiendo estos pasos:

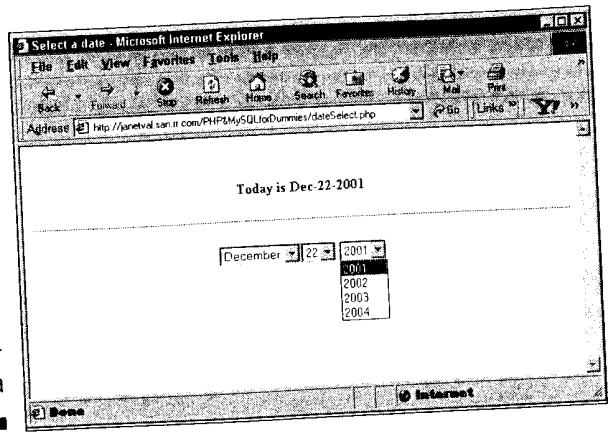


Figura 8-10: Un campo de selección para la fecha con la fecha de hoy seleccionada

1. **Crea un arreglo que contiene los nombres de los meses.** Las claves para el arreglo son los números. El primer mes, enero, empieza con la clave 1, de manera que las claves para el arreglo concuerden con los números de los meses.
2. **Crea variables que contienen la fecha actual.** `$hoy` contiene la fecha en un formato del sistema y se usa en el formulario. `$f_hoy` es una fecha formateada que se usa para mostrar la fecha en la página web.
3. **Muestra la fecha actual en la parte superior de la página web.**
4. **Construye el campo de selección para el mes.**
   i. Crea una variable que contiene el mes actual.
   ii. Hace eco de la etiqueta `select`, a la cual debería hacérsele eco una sola vez.
   iii. Empieza un ciclo `for` que se repite 12 veces.
   iv. Dentro del ciclo, hace eco de la etiqueta `opcion` usando el primer valor del arreglo `$nombremes`.
   v. Si el número del mes que se está procesando es igual al número del mes actual, añade la palabra `"selected"` a la etiqueta `option`.
   vi. Repite el ciclo 11 veces más.
   vii. Hace `echo` de la etiqueta `select` de cierre para el campo de selección, a la cual debería hacérsele eco una sola vez.
5. **Construye el campo de selección para el día.** Usa el procedimiento descrito en el Paso 4 para el mes. Sin embargo, sólo se usan números para esta lista de selección. El ciclo se repite 31 veces.
6. **Construye el campo de selección para el año.**
   i. Crea la variable `$añoinicio`, la cual contiene el año actual.
   ii. Hace eco de la etiqueta `select`, a la que se le debe hacer `echo` sólo una vez.
   iii. Inicia un ciclo `for`. El valor inicial para el ciclo es `$anioinicio`. El valor final para el ciclo es `$anioinicio+3`.

iv. Dentro del ciclo, hace `echo` de la etiqueta `option`, usando el valor inicial del ciclo `for`, que es el año actual.

v. Si el número de años que se está procesando es igual al número del mes actual, añade la palabra `"selected"` a la etiqueta de opción.

vi. Repite el ciclo hasta que el valor final sea igual a `$anioinicio+3`.

vii. Hace eco de la etiqueta de cierre `select` para el campo de selección, a la que se debe hace `echo` sólo una vez.

7. Hace eco de la etiqueta final para el formulario.

Crear listas de botones de opción

Tal vez usted quiera usar botones de opción en lugar de listas de selección. Por ejemplo, puede desplegar una lista de botones de opción para su Catálogo de mascotas, y pedir a los usuarios que seleccionen el botón para la categoría de mascota de su interés.

El formato para los botones de opción en los formularios es

```
<input type="radio" name="mascotas" value="Unicornio">
```

Puede construir una lista dinámica de botones de opción que represente todos los tipos de mascotas en su base de datos del mismo modo que se construye una lista de selección dinámica en la sección anterior. La Lista 8-10 muestra el programa `construyeradio.php`, el cual crea una lista de botones de opción con base en los tipos de mascotas.

Lista 8-10: Programa para construir una lista de botones de opción

```
<?php
/*  Nombre del programa: construyeRadio.php
 *  Descripcion: El programa muestra una lista de botones de
          opcion con la informacion de la base de datos.
 */
echo "<html>
      <head><title>Tipos de mascotas</title></head>
      <body>";
$usuario="catalogo";
  $huesped="localhost";
  $clave="";
  $basededatos = "CatalogodeMascotas";

  $conexion = mysql_connect($huesped,$usuario,$clave)
       or die ("No se pudo conectar al servidor");
  $db = mysql_select_db($basededatos,$conexion)
       or die ("No se pudo seleccionar la base de datos");
  $consulta = "SELECT DISTINCT Tipomascota FROM Mascota
                    ORDER BY Tipomascota";
  $resultado = mysql_query($consulta)
       or die ("No se pudo ejecutar la consulta.");
```

```
echo "<div style='margin-left: .5in'>
<p> 
<p><b>¿En que tipo de mascota esta interesado?</b>
<p>Por favor escoja un tipo de mascota de la lista
        siguiente:\n";


/* crea un formulario que contiene botones de opcion */
echo "<form action='procesaformulario.php'
        method='POST'>\n";

while ($fila = mysql_fetch_array($resultado))
{
    extract($fila);
    echo "<input type='radio' name='interest'
          value='$Tipomascota'>$Tipomascota\n";
    echo "<br>\n";
 }
echo "<p><input type='submit' value='Seleccione el tipo de
          mascota'>
       </form>\n";
?>
</div></body></html>
```

Este programa es muy parecido al de la Lista 8-9. La página web producida por este programa se muestra en la Figura 8-11.

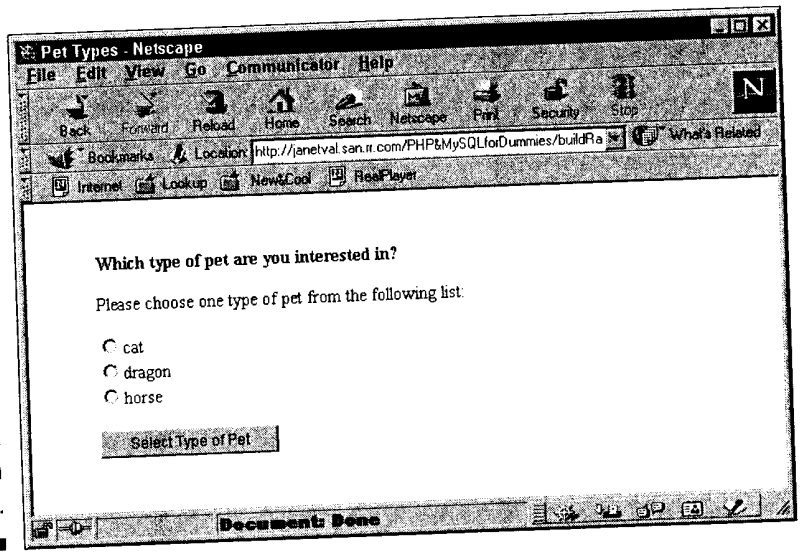


Figura 8-11: Lista de botones de opción producida por el programa en la Lista 8-10.

Construir listas de casillas para marcar

Tal vez usted prefiera usar casillas en su formulario. Las casillas para marcar son diferentes a las listas de selección y a los botones de opción, ya que permiten a los usuarios seleccionar más de una opción. Por ejemplo, si usted despliega una lista de categorías de mascotas usando casillas, un usuario puede marcar dos o tres o más categorías de mascotas. El programa `buildCheckbox.php` en la Lista 8-11 crea una lista de casillas para marcar.

Lista 8-11: Programa para construir una lista de casillas para marcar

```
<?php
/*  Nombre del programa: construyeCheckbox.php
 *  Descripcion: El programa muestra una lista de casillas
          para marcar con informacion de la base de datos.
 */
echo "<html>
      <head><title>Tipos de mascotas</title></head>
      <body>";
  $usuario="catalogo";
  $huesped="localhost";
  $clave="";
  $basededatos = "Catalogodemascotas";

  $conexion = mysql_connect($huesped,$usuario,$clave)
       or die ("No se pudo conectar al servidor");
  $db = mysql_select_db($basededatos,$conexion)
       or die ("No se pudo seleccionar la base de datos");
  $consulta = "SELECT DISTINCT Tipomascota FROM Mascota
                ORDER BY Tipomascota";
  $resultado = mysql_query($consulta)
       or die ("No se pudo ejecutar la consulta.");

  echo "<div style='margin-left: .5in'>
  <p> 
  <p><b>¿En que tipo de mascota esta interesado?</b>
  <p>Escoja tantos tipos de mascotas como quiera:\n";


  /* crea un formulario que contiene casillas para marcar */
  echo "<form action='procesaformulario.php'
          method='POST'>\n";

  while ($fila = mysql_fetch_array($resultado))
  {
     extract($fila);
     echo "<input type='checkbox'
           name='interest[$Tipo_mascota]'
           value='$Tipomascota'>$Tipomascota\n";
     echo "<br>\n";
  }
echo "<p><input type='submit' value='Seleccione el Tipo de
          Mascota'>
        </form>\n";
?>
</div></body></html>
```

Este programa es muy similar al programa en la Lista 8-10 que construye una lista de botones de opción. Sin embargo, observe que el campo de inout usa un arreglo `$interest` como el nombre del campo. Esto es porque más de una casilla puede ser seleccionada. Este programa creará un elemento en el arreglo con una pareja de clave y valor para cada casilla seleccionada. Por ejemplo, si el usuario selecciona tanto *caballo* como *dragón*, se crea la siguiente serie:

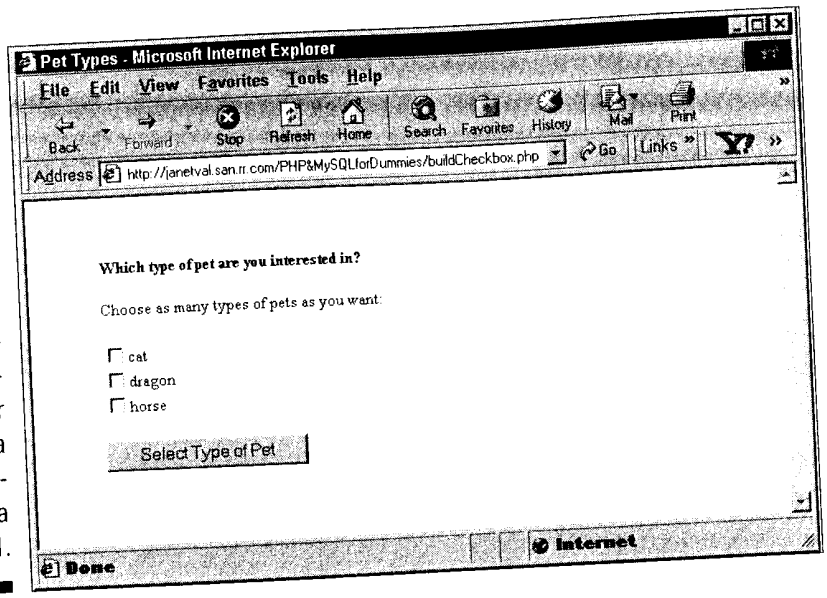


Figura 8-12: Una lista de casillas para marcar producida por el programa en la Lista 8-11.

```
$interest[caballo]=caballo
$interest[dragon]=dragon
```

El programa que procesa el formulario tiene las selecciones disponibles en el arreglo `POST`, como sigue:

```
$_POST['interest']['caballo']
$_POST['interest']['dragon']
```

La Figura 8-12 muestra la página web producida por `buildCheckbox.php`.

Usar la información del formulario

Como comenté anteriormente en esta sección, Juan Cliente rellena un formulario HTML, selecciona entre las listas y digita información en campos de texto. Hace clic en el botón Submit.

En la etiqueta `formulario`, usted le dice a PHP cuál programa correr cuando se hace clic en el botón Submit. Esto se hace al incluir action="`nombreprograma`" en la etiqueta `formulario`. Por ejemplo, en la mayoría de las listas de ejemplo en este capítulo, yo uso action="`procesaformulario.php`". Cuando el usuario hace clic en el botón Submit, el programa corre y recibe la información del formulario. EL manejo de la información del `formulario` es una de las mejores características de PHP. Usted no necesita preocuparse sobre los datos del formulario, puede simplemente obtenerlos de una de los arreglos incorporados y usarlos.

Los datos del formulario están disponibles en el programa de procesamiento en series, como se muestra en la Tabla 8-1. La calve para el elemento del

arreglo es el nombre del campo de input en el formulario. Por ejemplo, si hace eco del siguiente campo en su formulario

```
echo "<input type='text' name='primernombre'>";
```

el programa de procesamiento puede usar la variable `$_POST[primernombre]`, la cual contiene el texto que el usuario digitó en el campo. La información que el usuario selecciona de las listas desplegables de selección o de los botones de opción también está disponible para usar. Por ejemplo, si su formulario incluye la lista siguiente de botones de opción

```
echo "<input type='radio' name='interest' value='perro'>
          perro\n";
echo "<input type='radio' name='interest' value='gato'>gato\n";
```

usted puede tener acceso a la variable `$_POST[interest]`, la cual contiene ya sea perro o gato, dependiendo de lo que el usuario haya seleccionado.

Las casillas se manejan en forma algo diferente, ya que el usuario puede seleccionar más de una. Como se muestra en la Lista 8-11, los datos de una lista de casillas para marcar se pueden almacenar en un arreglo, de manera que todas las casillas estén disponibles. Por ejemplo, si su formulario incluye la lista siguiente de casillas para marcar

Post versus get

Para enviar información de un formulario, se usa uno de dos métodos. Los métodos pasan los datos del formulario de manera diferente y tienen ventajas y desventajas distintas.

- `Metodo get`: Los datos del formulario se pasan agregándolos al URL que llama al programa de procesamiento del formulario. Por ejemplo, el URL podría verse así:

  ```
  processform.php?apellido=
    Perez&pnombre=Goliat
  ```

  Las ventajas de este método son su simplicidad y velocidad. Las desventajas son que se pueden pasar menos datos y la información aparece en el explorador, lo cual puede representar un problema de seguridad en algunas situaciones.

- `Metodo post`: Los datos del formulario se pasan como un paquete en una comunicación separada con el programa de procesamiento. Las ventajas de este método son: se puede pasar información ilimitada y los datos están más seguros. Las desventajas son las operaciones auxiliares y menor velocidad.

Para programas CGI que no son de PHP, el programa que procesa el formulario debe encontrar la información y poner los datos en variables. En este caso, el método `get` es mucho más simple y fácil de usar. Muchos programadores usan el método `get` por esta razón. Sin embargo, PHP hace todo este trabajo por usted. Los métodos `get` y `post` son igualmente fáciles de usar en programas PHP. Por lo tanto, al usar PHP, casi siempre es mejor usar el método post, pues usted obtiene las ventajas del método post (paso de datos ilimitado, mejor seguridad) sin su principal desventaja (más difícil de usar).

```
echo "<input type='checkbox' name='interest[dog]'
          value='perro'>perro\n";
echo "<input type='checkbox' name='interest[cat]'
          value='gato'>gato\n";
```

usted puede tener acceso a los datos usando la variable multidimensional `$_POST[interest]`, la cual contiene lo siguiente:

```
$_POST[interest][perro] = perro
$_POST[interest][gato] = gato
```

En algunos casos, usted querrá tener acceso a todos los campos en el formulario. Tal vez quiera revisarlos todao para asegurarse de que el usuario no haya dejado ningún campo en blanco. Como se muestra en el programa `procesaformulario.php`, anteriormente en este capítulo (vea la Lista 8-5), puede usar `foreach` para moverse por el arreglo incorporado `$_POST` o `$_GET`. Muchos de los programas de muestra y los enunciados en este libro usan el método `post`. Las claves son los nombres de campos. Consulte el cuadro "Post versus get" para más información sobre los dos métodos.

Por ejemplo, suponga que su programa incluye los siguientes enunciados para desplegar un formulario:

```
echo "<form action='procesaformulario.php' method='POST'>\n";
echo "<input type='text' name='apellido' value='Pe-
          rez'><br>\n";
echo "<input type='radio' name='interest' value='perro'>
          perro\n";
echo "<input type='radio' name='interest' value='gato'>gato\n";
echo "<input type='hidden' name='hidvar' value='3'>\n";
echo "<br><input type='submit' value='Seleccione tipo de mascota'>
   </form>\n";
```

El programa `procesaformulario.php` contiene los siguientes enunciados que enumerarán todas las variables recibidas del formulario:

```
foreach ($_POST as $campo => $valor)
{
   echo "$campo, $valor<br>";
}
```

El resultado del ciclo `foreach` sería

```
apellido, Perez
interest, perro
hidvar, 3
```

El output muestra tres variables con estos tres valores por las siguientes razones:

- **El usuario no cambió el texto en el campo de texto.** El valor `"Perez"` que el programa mostró sigue siendo el texto en el campo de texto.

- **El usuario seleccionó el botón de opción para perro.** El usuario puede seleccionar sólo un botón de opción.
- **El programa pasó un campo escondido llamado hidvar.** El programa establece el valor para los campos escondidos. El usuario no puede afectar los campos escondidos.

Revisar la información

Juan Cliente completa un formulario HTML, selecciona de las listas y digita información en los campos de texto. Hace clic en el botón Submit. Ahora, usted tiene toda la información que deseaba. Bueno, tal vez. Juan puede haber digitado información que contiene algún error. O quizás haya digitado algo sin sentido. O incluso podría haber digitado información maliciosa que puede causarle problemas a usted o a otras personas que usan su sitio web. Antes de utilizar la información de Juan o almacenarla en su base de datos, es mejor revisarla, para cerciorarse de que sea la información solicitada. Revisar los datos es *validarlos*.

Validar los datos incluye lo siguiente:

- **Revisar en busca de campos vacíos:** Usted puede pedir a los usuarios que digiten información en un campo. Si el campo está en blanco, se le dice al usuario que la información es obligatoria, y el formulario reaparece para que el usuario pueda digitar la información faltante.
- **Verificar el formato de la información:** Usted puede revisar la información para ver si está en el formato correcto. Por ejemplo, *ab3&*xx* claramente no es un código postal válido.

Revisar en busca de campos vacíos

Cuando usted crea un formulario, puede decidir cuáles campos son obligatorios y cuáles son opcionales. Su decisión se implementa en el programa PHP. Usted revisa los campos obligatorios en busca de información. Si un campo obligatorio está vacío, envía un mensaje al usuario que indica que el campo es obligatorio, y luego vuelve a mostrar el formulario.

El formato general para revisar en busca de campos vacíos es

```
if ($apellido == "")
{
   echo "No digito su apellido. El apellido es obligato-
         rio.<br>\n";
    mostrar el formulario;
    exit();
}
echo " Bienvenido al club exclusivo para miembros.
      Puede seleccionar del menu a continuacion.<br>\n";
mostrar el menu;
```

Observe el enunciado `exit`. Los enunciados exit finalizan el programa. Sin el enunciado `exit`, el programa continuaría con los enunciados después del enunciado `if`. En otras palabras, sin el enunciado `exit`, el programa desplegaría el formulario y luego continuaría haciendo eco del enunciado de bienvenida, además del menú.

En muchos casos, usted quiere verificar todos los campos en el formulario. Puede hacerlo al moverse con un ciclo a través del arreglo `$ _POST`. Los siguientes enunciados registran el arreglo buscando campos vacíos:

```
foreach ($_POST as $valor)
{
   if ( $valor == "" )
   {
     echo "No ha completado todos los campos<br>\n";
     mostrar el formulario;
     exit();
   }
}
echo "Bienvenido";
```

Al volver a desplegar el formulario web, asegúrese de que contenga la información que el usuario ya digitó. Si los usuarios deben volver a digitar información, probablemente se frustren y se retiren de su sitio web.

En algunos casos, usted podría requerir que el usuario rellene la mayoría de los campos, pero no todos. Por ejemplo, podría solicitar un número de fax en el formulario o brindar un campo para el segundo nombre, pero realmente no quiere restringir la inscripción a su sitio web sólo a usuarios que tienen segundo nombre y fax. En este caso, puede hacer una excepción para los campos que no son obligatorios, como sigue:

```
foreach ($_POST as $campo => $valor)
{
   if ( $campo != "fax" and $campo != "segundonombre" )
   {
      if ( $valor == "" )
      {
        echo "No ha completado todos los campos<br>\n";
        mostrar el formulario;
        exit();
      }
   }
}
echo "Bienvenido";
```

Observe que el enunciado condicional `if` externo es verdadero sólo si el campo no es el campo fax y no es el campo segundo nombre. Para esos dos campos, el programa no llega al enunciado `if` interno, el cual revisa en busca de campos en blanco.

El algunos casos, tal vez usted quiera decir al usuario exactamente cuáles campos deben rellenarse. El programa `chequeodedatos.php` en la Lista 8-12 procesa un

formulario con cuatro campos: `primer_nombre`, `segundo_nombre`, `apellido` y `telefono`. Todos los campos son obligatorios, excepto `segundo_nombre`. En el ejemplo que aparece en la Figura 8-13, el usuario no digitó el primer nombre: El mensaje de error resultante cuando se procesa el formulario le dice al usuario cuál campo dejó vacío.

Lista 8-12: Programa que revisa en busca de campos en blanco

```
<?php
/*  Nombre del programa: chequeaDatos.php
 *  Descripcion: El programa revisa todos los campos del
          formulario para ver si hay campos en blanco.
 */
?>
<html>
<head><title>Campos vacios</title></head>
<body>
<?php
  /* fija una serie de etiquetas de campo */
  $arreglo_etiqueta = array ( "primer_nombre" => "Primer
          Nombre ",
                        "segundo_nombre " => "Segundo
          Nombre",
                        "apellido" => "Apellido",
                        "telefono" => "telefono");
  /* verifica cada campo excepto el segundo nombre por si hay
          campos en blanco */
  foreach ($_POST as $campo => $valor)
  {
    if ($campo != "segundo_nombre")
    {
      if ( $valor == "" )
      {
        $arreglo_Blanco[$campo] = "blanco";
      }
    }
  }  // terminar el ciclo foreach para $_POST
  /* si hay campos en blanco, mostrar el mensaje de error y
          el formulario */
  if (@sizeof($arreglo_Blanco) > 0) //si se encuentran campos
          en blanco
  {
    echo "<b>No completo uno o mas campos obligatorios.
            Debe digitar:</b><br>";
    /* mostrar la lista de informacion faltante */
    foreach($arreglo_Blanco sd $campo => $valor)
    {
      echo
          "   {$arreglo_etiqueta[$campo]}<br>
          ";
    } //terminar el ciclo foreach para blancos
    /* volver a mostrar el formulario */
    $primer_nombre=trim(strip_tags($_POST['primer_nombre']));
          $segundo_nombre=trim(strip_tags($_POST['segundo_no
          mbre']));
    $apellido=trim(strip_tags($_POST['apellido']));
```

```
$telefono=trim(strip_tags($_POST['telefono']));
echo "<p><hr>
  <form action='chequeaDatos.php' method='POST'>
  <center>
  <table width='95%' border='0' cellspacing='0'
        cellpadding='2'>
  <tr><td
       align='right'><b>{$arreglo_etiqueta['primer_nombre
       ']}:
          </br></td>
      <td><input type='text' name='primer_nombre'
       size='65'
              maxlength='65'
              value='{$primer_nombre}' ></td>
  </tr>
  <tr><td
       align='right'><b>{$arreglo_etiqueta['segundo_nombr
       e']}:
          </br></td>
      <td><input type='text' name='segundo_nombre'
       size='65'
              maxlength='65' value='$segundo_nombre' >
       </td>
   </tr>
   <tr><td
        align='right'><b>{$arreglo_etiqueta['apellido']}:
          </b></td>
       <td> <input type='text' name='apellido' size='65'
              maxlength='65' value='$apellido'> </td>
   </tr>
   <tr><td
        align='right'><b>{$arreglo_etiqueta['telefono']}:
          </b></td>
       <td> <input type='text' name='telefono' size='65'
              maxlength='65' value='$telefono'> </td>
   </tr>
   </tabla>
   <p><input type='submit'
              value='Enviar nombre y numero telefonico'>
   </form>
   </center>";
  exit();
 }
 echo "Bienvenido";
?>
</body></html>
```

Para revisar si hay vacíos, el programa hace lo siguiente:

1. **Establece un arreglo de etiquetas de campo.** Estas etiquetas se usan como etiquetas en el formulario, y también se usan para mostrar la lista de información faltante.
2. **Se mueve en ciclo por todas las variables pasadas desde el formulario, revisando si hay vacíos.** Las variables están en el arreglo `$ _POST`. Cualquier campo en blanco que se encuentre se agrega a un arreglo de campos en blanco `$arreglo_blanco`.

3. **Revisa si se encontraron campos vacíos.** Verifica el número de elementos en `$arreglo_blanco`.
4. **Si no se encontró ningún campo en blanco, salta al mensaje de bienvenida.**
5. **Si encontró uno o más campos en blanco:**
   i. Muestra un mensaje de error. Este mensaje explica al usuario que falta alguna información obligatoria.
   ii. Muestra una lista de información faltante. Hace un ciclo por `$serie_blanco` y muestra la(s) etiqueta(s).
   iii. Despliega el formulario. Como el formulario incluye nombres de variables en el atributo valor, la información que el usuario digitó anteriormente se recupera de `$_POST` y se despliega.
   iv. Sale. Se detiene después de desplegar el formulario. El usuario debe hacer clic en el botón Submit para continuar.

Recuerde. Los programas que procesan formularios usan la información del formulario. Si usted los corre por sí solos, no tienen ninguna información que se haya pasado del formulario y no correrán correctamente. Estos programas fueron hechos para correr cuando el usuario oprime el botón Submit para un formulario.

No se olvide del enunciado `exit`. Sin el enunciado `exit`, el programa continuaría y mostraría el mensaje de bienvenida después de mostrar el formulario.

La Figura 8-13 muestra la página web que resulta si el usuario no digitó su primer o segundo nombre. Observe que la lista de información faltante no incluye el segundo nombre, pues éste no es obligatorio. Además, observe que la información digitada originalmente por el usuario en el formulario todavía aparece en los campos del formulario.

Verificar el formato de la información

Cuando los usuarios deben digitar información en un formulario, usted puede esperar cierto número de errores de digitación. Puede detectar algunos de estos errores cuando el formulario es enviado, hacérselos saber al usuario y luego solicitarle que vuelva a digitar la información. Por ejemplo, si el usuario digita **8899776** en el campo del código postal, usted sabe que esto no es correcto para los Estados Unidos. Esta información es muy larga para ser un código postal y muy corta para ser el código postal +4.

Usted también debe protegerse de los usuarios maliciosos, usuarios que podrían querer dañar su sitio web o su base de datos, o robar información suya o de sus usuarios. No querrá que los usuarios digiten etiquetas HTML en los campos del formulario: algo que podría tener resultados inesperados cuando se envía a un explorador web. Una etiqueta particularmente peligrosa sería una etiqueta de script que permite a un usuario introducir un programa en el campo de un formulario.

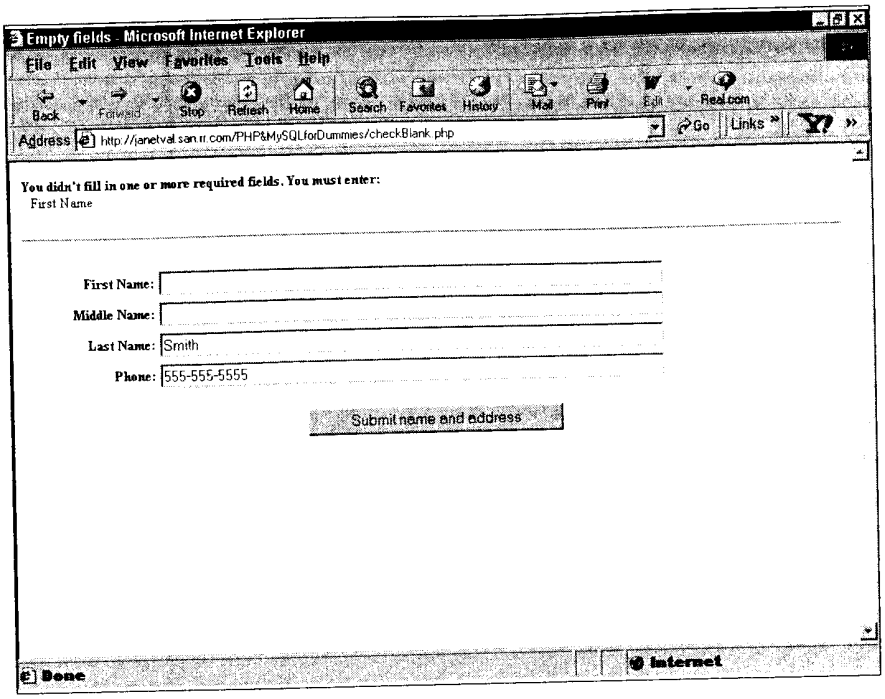


Figura 8-13: El resultado de procesar un formulario con información faltante.

Si usted revisa cada campo en busca de su formato esperado, puede pescar errores y protegerse de la mayoría del contenido malicioso. Sin embargo, verificar la información es un asunto delicado. Usted quiere hallar tantos datos incorrectos como sea posible, pero no desea bloquear ninguna información legítima. Por ejemplo, cuando revisa un número telefónico, podría limitarlo a números. El problema con esta verificación es que rechazaría números telefónicos legítimos en el formulario 555-5555 u (888) 555-5555. Por eso, usted también debe permitir guiones (-), paréntesis ( ) y espacios. Podría limitar el campo a una longitud de 14 caracteres, incluyendo paréntesis, espacios y guiones, pero esto rechazaría números del extranjero o números que incluyan una extensión. En resumidas cuentas: Usted debe pensar cuidadosamente cuál información desea aceptar o rechazar para cualquier campo.

Puede verificar la información en el campo usando *expresiones regulares*, las cuales son patrones. Usted compara la información en el campo con el patrón para ver si concuerda. Si no concuerda, la información en el campo es incorrecta, y el usuario debe digitarla de nuevo. (Consulte el Capítulo 6 para más sobre las expresiones regulares.)

En general, estos son los enunciados que se usan para revisar campos:

```
if ( !ereg("patron",$nombredevariable) )
{
    echo mensaje de error;
    volver a mostrar el formulario;
    exit();
}
echo "Bienvenido";
```

Observe que la condición en el enunciado `if` es negativa. O sea, el `!` (signo de admiración) significa `"not"`. Entonces, el enunciado if realmente dice: Si la variable no concuerda con el patrón, ejecute el bloque `if`.

Por ejemplo, suponga que usted desea revisar un campo de input que contiene el apellido del usuario. Puede esperar que los apellidos contengan letras y no números, y posiblemente caracteres como un apóstrofo y un guión (como en *O'Hara y Smith-Jones*, respectivamente) y también espacios (como en *Van Dyke*). Además, es difícil imaginar un nombre que tenga más de 50 caracteres. Así, usted puede usar los siguientes enunciados para chequear un apellido:

```
if ( !ereg("[A-Za-z' -]{1,50}",$apellido)
{
    echo mensaje de error;
    volver a mostrar el formulario;
    exit();
}
echo "Bienvenido";
```

Si quiere incluir un guión `(-)` como parte de un conjunto de caracteres permitidos encerrados entre corchetes `( [ ] )`, debe indicar el guión al inicio o al final de la lista. De otro modo, si lo pone entre dos caracteres, el programa lo interpretará como el rango entre los dos caracteres, como en A — Z.

En la sección anterior, usted aprendió cómo verificar cada campo del formulario para garantizar que no haya vacíos. Además de esta acción, probablemente también querrá verificar todos los campos que tienen datos para asegurarse de que los datos tengan un formato aceptable. Puede chequear el formato haciendo unos cuantos cambios simples al programa en la Lista 8-12. La Lista 8-13 muestra el programa modificado llamado `chequetodos.php`.

Lista 8-13: Programa que revisa todos los datos en los campos de un formulario

```
<?php
/*  Nombre del programa: chequeaTodos.php
 *  Descripcion: El programa revisa todos los campos del
           formulario para ver si hay campos en blanco o tienen
           el formato incorrecto.
 */
?>
<html>
<head><title>Campos vacios</title></head>
<body>
<?php
  /* establece una serie de etiquetas de campos */
  $arreglo_etiquetas= array ( "primer_nombre" => "Primer Nombre
             ",
                    "segundo_nombre" => "Segundo Nombre ",
                    "apellido " => "apellido",
                    "telefono" => "telefono");
  foreach ($_POST as $campo => $valor)
  {
```

```
    /* verificar cada campo excepto el segundo nombre por si hay
            campos en blanco */
    if ( $valor == "" )
    {
       if ($campo != "segundo_nombre")
       {
          $arreglo_blanco[$campo] = "blanco";
       }
    }
    elseif ($campo == "primer_nombre" o $campo ==
            "segundo_nombre"
                 o $campo == "apellido " )
    {
        if (!ereg("^[A-Za-z' -]{1,50}$",$_POST[$campo]) )
        {
            $mal_formato[$campo] = "malo";
        }
    }
    elseif ($campo == "telefono")
    {
      if(!ereg("^[0-9)( -]{7,20}(([xX]|(ext)|(ex))?[ -]?[0-
            9]{1,7})?$",$valor))
      {
          $mal_formato[$campo] = "malo";
      }
    }
}
/* si cualquier campo no esta bien, mostrar un mensaje de error
        y el formulario */
if (@sizeof($arreglo_blanco) > 0 or @sizeof($mal_formato) > 0)
{
  if (@sizeof($arreglo_blanco) > 0)
  {
      /* mostrar mensaje de informacion faltante */
      echo "<b>No completo uno o mas de los campos
          obligatorios. Debe digitar:</b><br>";
      /* mostrar la lista de la informacion faltante */
      foreach($arreglo_blanco como $campo => $valor)
      {
          echo
           "   {$arreglo_etiquetas[$campo]}<br>";
      }
  }
  if (@sizeof($mal_formato) > 0)
  {
      /* mostrar mensaje de informacion incorrecta */
      echo "<b>Uno o mas campos tienen informacion que parece
          incorrecta. Corrija el formato de:</b><br>";
      /* mostrar la lista de la informacion incorrecta */
      foreach($mal_formato as $campo => $valor)
      {
          echo
           "   {$arreglo_etiquetas[$campo]}<br>";
      }
  }
  /* volver a mostrar el formulario */
  $primer_nombre = $_POST['primer_nombre'];
  $segundo_nombre = $_POST['segundo_nombre'];
  $apellido = $_POST['apellido'];
  $telefono = $_POST['telefono'];
  echo "<p><hr>
    <form action='checkTodos.php' method='POST'>
```

```
      <center>
      <table width='95%' border='0' cellspacing='0'
            cellpadding='2'>
      <tr><td
            align='right'><B>{$arreglo_etiquetas['primer_nombre']
            }:</br></td>
        <td><input type='text' name='primer_nombre' size='65'
            maxlength='65'
                  value='$primer_nombre' > </td>
      </tr>
      <tr><td
            align='right'><B>{$arreglo_etiquetas['segundo_nombre'
            ]}:</br></td>
        <td><input type='text' name='segundo_nombre' size='65'
            maxlength='65'
                  value='$segundo_nombre' > </td>
      </tr>
      <tr><td
            align='right'><B>{$arreglo_etiquetas['apellido']}:</B
            ></td>
        <td> <input type='text' name='apellido' size='65'
            maxlength='65'
                  value='$apellido'> </td>
      </tr>
      <tr><td
            align='right'><B>{$arreglo_etiquetas['telefono']}:</B
            ></td>
        <td> <input type='text' name='telefono' size='65'
            maxlength='65'
                  value='$telefono'> </td>
      </tr>
      </table>
      <p><input type='submit' value='Enviar el nombre y el
            numero telefonico'>
      </form>
      </center>";
    exit();
  }
  echo "Bienvenido";
?>
</body></html>
```

Estas son las diferencias entre este programa y el programa de la Lista 8-12:

- **Este programa crea dos series para los datos problema.** Crea `$serie_blanco`, al igual que el programa anterior. Pero este programa también crea `$mal_formato` para los campos que contienen información con un formato no aceptable.
- **Este programa hace un ciclo por `$mal_formato` para crear una lista separada de datos problemáticos.** Si hay campos vacíos, crea un mensaje de error y enumera los campos con problemas, al igual que el programa anterior. Si hay campos con un formato inaceptable, este programa también crea un segundo mensaje de error y enumera los campos problemáticos.

La página web en la Figura 8-14 es el resultado obtenido cuando el usuario accidentalmente digita su primer nombre en el campo del segundo nombre y también digita un sinsentido como número telefónico. Observe que aparecen dos mensajes de error, los cuales muestran que el campo del primer nombre está en blanco y el campo del número telefónico contiene información incorrecta.

Dar a los usuarios la oportunidad de escoger, con múltiples botones de envío

Usted puede usar más de un botón Submit en un formulario. Por ejemplo, en un formulario para pedidos del cliente, podría usar un botón que diga *Enviar pedido* y otro que diga *Cancelar pedido*. Sin embargo, sólo puede enumerar un programa en la parte `action=nombreprograma` de su etiqueta `formulario`, lo cual significa que los dos botones corren el mismo programa. PHP resuelve este problema. Usando PHP, usted puede procesar el formulario de manera diferente, dependiendo de en cuál botón el usuario haga clic.

Los siguientes enunciados crean un formulario con dos botones de envío:

```
<form action="dosbotones.php" method="POST">
  <input type="texto" name="apellido" maxlenght="50"><br>
   <input type="submit" name="mostrar_boton"
          value="Mostrar direccion">
   <input type="submit" name="mostrar_boton"
          value="Mostrar numero telefonico">
</form>
```

Observe que los campos para los botones de envío tienen un nombre: `mostrar_boton`. Cada uno de los campos tiene un valor diferente. Cualquiera de los dos botones en que el usuario haga clic establece el valor para `$mostrar_boton`. El programa `dosbotones.php` en la Lista 8-14 procesa el formulario anterior.

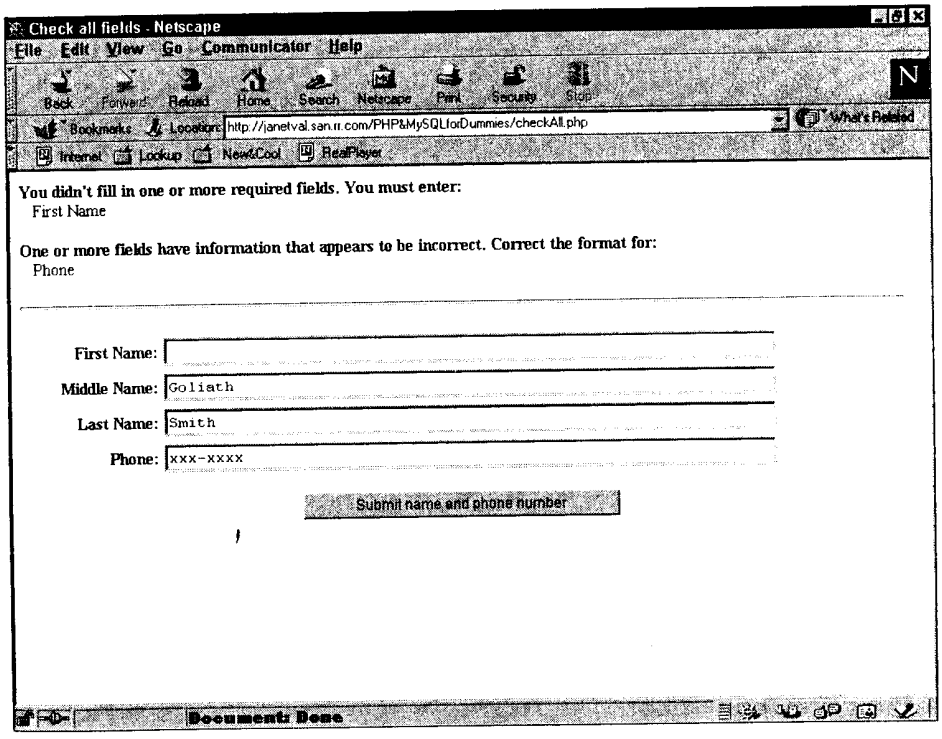


Figura 8-14: El resultado de procesar un formulario tanto con información faltante como con información incorrecta.

Lista 8-14: Programa que procesa dos botones de envío

```
<?php
/*  Nombre del programa: dosbotones.php
 *  Descripcion: El programa muestra diferente informacion
 *          dependiendo de cual boton de envio fue presionado.
 */
?>
<html>
<head><title>Direccion o Numero telefonico del
          cliente</title></head>
<body>
<?php
  $usuario="admin";
  $huesped="localhost";
  $clave="";
  $basededatos = "Directoriodemiembros";
  $conexion = mysql_connect($huesped,$usuario,$clave)
       or die ("No se pudo conectar al servidor");
  $db = mysql_select_db($basededatos,$conexion)
       or die ("No se pudo seleccionar la base de datos");
  if ($_POST['mostrar_boton'] == "Mostrar Direccion")
  {
     $consulta = "SELECT calle,ciudad,estado,codigopostal
           FROM Miembro
                WHERE apellido='$_POST[apellido]'";
     $resultado = mysql_query($consulta)
       or die ("No se pudo ejecutar la consulta.");
     $fila = mysql_fetch_array($resultado);
     extract($fila);
     echo "$calle<br>$ciudad, $estado  $codigo_postal<br>";
  }
  else
  {
     $consulta = "SELECT telefono FROM Miembro
                  WHERE apellido='$_POST[apellido]'";
  $resultado = mysql_query($consulta)
       or die ("No se pudo ejecutar la consulta.");
     $fila = mysql_fetch_array($resultado);
     echo "Telefono: {$fila['telefono']}<br>";
  }
?>
</body></html>
```

El programa ejecuta diferentes enunciados, dependiendo de cuál botón se ha presionado. Si el usuario hace clic en el botón de la dirección, el programa da como output la dirección que corresponde al nombre enviado en el formulario; si el usuario hace clic en el botón Mostrar número telefónico, el programa muestra el número de teléfono.

Insertar información en una base de datos

Su aplicación probablemente necesite almacenar datos en su base de datos. Por ejemplo, su base de datos podría almacenar información que un usuario digitó en un formulario para su uso; un Directorio de miembros es un buen ejemplo de esto. O bien, su base de datos podría almacenar datos temporalmente durante la aplicación. De cualquier modo, usted almacena datos enviando consultas SQL a MySQL. (Le explico las consultas SQL en detalles en el Capítulo 4.)

Preparar los datos

Debe preparar los datos antes de almacenarlos en la base de datos. Preparar los datos incluye lo siguiente:

- Poner los datos en variables
- Asegurarse de que los datos estén en el formato esperado por la base de datos
- Depurar los datos

Poner los datos en variables

Los datos se almacenan enviándolos a la base de datos en una consulta SQL. Usted almacena los datos en variables e incluye los nombres de las variables en la consulta. Este proceso es simple si se usa PHP. El usuario proporciona la mayoría de los datos que usted desea almacenar por medio de un formulario. Como comenté anteriormente en este capítulo, PHP almacena los datos en una variable con el nombre del campo del formulario, en forma invisible y automática, sin necesidad de que usted mismo los almacene. Usted nada más usa las variables que PHP proporciona. Ocasionalmente, querrá almacenar la información que usted mismo genera, tal como la fecha de hoy o el número del pedido de un cliente. Sólo debe guardar esta información en una variable, para poder incluirla en una consulta.

Usar el formato correcto

Al diseñar su base de datos, usted establece el tipo de datos para cada columna. Los datos que desea almacenar deben concordar con el tipo de datos de la columna en la cual los desea almacenar. Por ejemplo, si la columna espera datos del tipo de números enteros, los datos deben ser números. O si la columna espera datos que sean fechas, los datos enviados deben tener un formato que MySQL reconozca como una fecha. Si usted manda datos con el formato incorrecto, MySQL igualmente los almacenará, pero puede ser que no almacene el valor que usted espera. Estos son los detalles de cómo MySQL almacena datos para los tipos de datos usados más frecuentemente:

- `CHAR` o `VARCHAR`: Almacena cadenas. MySQL almacena prácticamente cualquier dato enviado a una columna de caracteres, incluyendo números o fechas, como cadenas. Cuando usted creó la columna, especificó su longi-

tud. Por ejemplo, si especificó `CHAR(20)`, sólo 20 caracteres se podrán almacenar. Si envía una cadena que tenga más de 20 caracteres, sólo los primeros 20 caracteres se almacenarán. Los caracteres restantes se eliminan.

Establezca `maxlength` para cualquier campo de input de texto en un formulario a la misma longitud del ancho de la columna en la base de datos donde se almacernarán los datos. De ese modo, el usuario no podrá digitar más caracteres de los que puede almacenar la base de datos.

- `INT o DECIMAL`: Almacena números. MySQL tratará de interpretar cualquier dato enviado a una columna de números como un número, ya sea que tenga sentido o no. Por ejemplo, podría interpretar una fecha como un número, y usted terminaría con un número como 2001.00. Si MySQL es incapaz de interpretar los datos enviados como un número, guardará 0 (cero) en la columna.
- `DATE`: Almacena fechas. MySQL espera fechas como números con el año primero, después el mes y, por último, el día. El año puede tener dos o cuatro dígitos (2001 ó 01). La fecha puede ser una cadena de números; o bien, se puede separar cada parte con un guión (-), un punto (.) o una diagonal (/). Algunos formatos válidos para las fechas son 20011203, 980103, 2001-3-2 y 2000.10.01. Si MySQL no puede interpretar los datos enviados como una fecha, almacena la fecha como 0000-00-00.
- `ENUM`: Almacena sólo los valores que usted permitió cuando creó la columna. Si envía datos que no son permitidos, MySQL almacena un 0.

En muchos casos, los datos se recopilan en un formulario y se almacenan en la base de datos tal y como están. Por ejemplo, los usuarios digitan sus nombres en un formulario, y el programa los almacena. Sin embargo, en algunos casos, los datos deben cambiarse antes de ser almacenados. Por ejemplo, si un usuario digita una fecha en un formulario en tres listas de selección separadas para mes, día y año (como describo en la sección "Construir listas de selección", anteriormente en este capítulo), los valores en los tres campos deben juntarse en una sola variable. Los siguientes enunciados juntan los campos:

```
$expDate = $_POST['expYear']."-";
$expDate.= $_POST['expMonth']."-";
$expDate.= $_POST['expDay'];
```

Otro caso en el cual usted podría querer cambiar los datos antes de almacenarlos es cuando está guardando números telefónicos. Los usuarios digitan los números de teléfono en una variedad de formatos, usando paréntesis, rayas, puntos o espacios. En vez de almacenar esta variedad de formatos en su base de datos, usted podría sólo guardar los números. Luego, al recuperar un número telefónico de la base de datos, puede formatearlo como quiera antes de mostrarlo. El siguiente enunciado retira caracteres de la cadena:

```
$telefono = ereg_replace("[ )(.-]","",$_POST['telefono']);
```

La función `ereg_replace` usa expresiones regulares para buscar un patrón. La primera cadena que se pasa es la expresión regular con la cual debe coincidir. Si cualquier parte de la cadena concuerda con el patrón, es reemplaza-

da por la segunda cadena. En este caso, la expresión regular es `[ )(.-]`, lo cual significa cualquiera de los caracteres entre los corchetes. La segunda cadena es "", la cual es una cadena sin nada. Por lo tanto, cualquier espacio, paréntesis, puntos o guiones en la cadena son reemplazados por nada.

Depurar los datos

La sección anterior "Obtener información del usuario," la cual describe el uso de los formularios HTML, discute cómo verificar los datos en los formularios. Los usuarios pueden digitar datos en un campo de texto, ya sea accidental o maliciosamente, los cuales pueden causar problemas para su aplicación, su base de datos o sus usuarios. Revisar los datos y aceptar sólo los caracteres esperados para la información solicitada puede evitar muchos problemas. Sin embargo, usted puede pasar algo por alto. Además, en algunos casos, la información que el usuario digita debe permitir básicamente cualquier cosa. Por ejemplo, usted normalmente no permitiría los caracteres < y > en un campo. No obstante, podría existir una situación en la cual el usuario necesite digitar estos caracteres, tal vez al digitar una fórmula o especificación técnica que los requiera.

PHP brinda dos funciones que pueden depurar los datos, hasta convertirlos en inofensivos:

- `strip_tags`: Esta función retira todo texto encerrado entre < y > de los datos. Busca un < de apertura y lo elimina junto con todo lo demás, hasta encontrar un > de cierre o llegar al final de la cadena. Usted puede incluir etiquetas específicas que desea permitir. Por ejemplo, el siguiente enunciado elimina todas las etiquetas de una cadena de caracteres excepto <b> e <i>:

```
$apellido = strip_tags($apellido,"<b><i>");
```

- `htmlspecialchars`: Esta función cambia algunos caracteres especiales que tienen significado para HTML a un formato HTML que permita mostrarlos sin ningún significado especial. Los cambios son

  - < se convierte en <
  - > b se convierte en >
  - & b se convierte en &

  De esta forma, los caracteres < y > se pueden mostrar en una página web sin ser interpretados por HTML como etiquetas. El siguiente enunciado cambia estos caracteres especiales:

```
$apellido = htmlspecialchars($apellido);
```

Si usted está seguro de que no desea permitir a sus usuarios digitar ningún carácter < o > en un campo de un formulario, use `strip_tags`. Sin embargo, si desea permitir los caracteres < o >, puede almacenarlos con seguridad después de que hayan sido procesados mediante `htmlspecialchars`.

Otra función que debería usar antes de almacenar los datos en su base de datos es `trim`. Los usuarios a menudo digitan espacios al inicio o al final de un campo

de texto sin querer. `Trim` elimina cualquier espacio al inicio o al final, para que no sean almacenados. Use el siguiente enunciado para eliminar estos espacios:

```
$apellido = trim($_POST['apellido']);
```

Agregar información nueva

Se usa la consulta `INSERT` (descrita en el Capítulo 4) para añadir información nueva a la base de datos. `INSERT` agrega una fila nueva a una tabla en una base de datos. El formato general es

```
$consulta = "INSERT INTO nombredetabla (col,col,col...)
                VALUES ('var','var','var'...)";
$resultado = mysql_query($consulta)
       or die ("No se pudo ejecutar la consulta.");
```

Por ejemplo, los enunciados para almacenar el nombre y número telefónico que un usuario digitó en un formulario son

```
$primer_nombre = "Goliat";        // del campo del formulario
$apellido = "Perez";            // del campo del formulario
$telefono = "555-555-5555";       // del campo del formulario
$consulta = "INSERT INTO Miembro (apellido,primer_nombre,telefono)
            VALUES ('$apellido','$primer_nombre','$telefono')";
$resultado = mysql_query($consulta)
       or die ("No se pudo ejecutar la consulta.");
```

La Lista 8-15 muestra un programa llamado `guardartelefono.php`, el cual almacena un nombre y número telefónico de un formulario.

Lista 8-15: Programa que almacena datos de un formulario

```
<?php
/*  Nombre del programa: guardaTelefono.php
 *  Descripcion: El programa revisa todos los campos del
          formulario en busca de campos vacios y formato
          incorrecto. Guarda los campos correctos en una base
          de datos.
 */
?>
<html>
<head><title>Numero telefonico del miembro</title></head>
<body>
<?php
  $primer_nombre = strip_tags(trim($_POST['primer_nombre']));
  $apellido = strip_tags(trim($_POST['apellido']));
  $telefono = strip_tags(trim($_POST['telefono']));
  $telefono = ereg_replace("[)( .-]","",$telefono);

  /* verificar informacion del formulario */
```

```
/* establecer serie de etiquetas de campos */
$arreglo_etiquetas = array( "primer_nombre" => "Primer
        nombre",
                        "apellido" => "Apellido",
                        "telefono" => "Telefono");
foreach ($_POST as $campo => $valor)
{
  /* revisar cada campo en busca campos en blanco */
  if ( $valor == "" )
  {
      $arreglo_blanco[$campo] = "blanco";
  }
  elseif ( ereg("(nombre)",$campo) )
  {
    if (!ereg("^[A-Za-z' -]{1,50}$",$_POST[$campo]) )
    {
          $mal_formato[$campo] = "malo";
    }
  }
  elseif ($campo == "telefono")
  {
    if(!ereg("^[0-9)( -]{7,20}(([xX]|(ext)|(ex))?[ -]?[0-
        9]{1,7})?$",$valor) )
    {
          $mal_formato[$campo] = "malo";
    }
  }
} // terminar foreach para $_POST
/* si algun campo no esta bien, mostrar mensaje de error y
        formulario */
if (@sizeof($arreglo_blanco) > 0 o @sizeof($mal_formato) >
        0)
{
  if (@sizeof($arreglo_blanco) > 0)
  {
      /* desplegar mensaje de informacion faltante */
      echo "<b>No completo uno o mas campos obligatorios.
              Debe digitar:</b><br>";
      /* mostrar la lista de informacion faltante */
      foreach($arreglo_blanco as $campo => $valor)
      {
         echo
         "   {$arreglo_etiquetas[$campo]}<br>
         ";
      }
  }
  if (@sizeof($mal_formato) > 0)
  {
      /* mostrar mensaje de informacion incorrecta */
      echo "<b>Uno o mas campos tienen informacion que
          parece ser incorrecta. Corrija el formato
          de:</b><br>";
      /* desplegar lista de informacion incorrecta */
      foreach($mal_formato as $campo => $valor)
      {
         echo
         "   {$arreglo_etiquetas[$campo]}<br>
         ";
      }
  }
  /* volver a mostrar formulario */
```

```
    echo "<p><hr>
      <form action='chequeaTodos.php' method='POST'>
      <center>
      <table width='95%' border='0' cellspacing='0'
           cellpadding='2'>
      <tr><td
           align='right'><B>{$arreglo_etiquetas['primer_nombre
           ']}:</br></td>
        <td><input type='text' name='primer_nombre' size='65'
           maxlength='65'
                    value='$primer_nombre' > </td>
      </tr>
      <tr><td
           align='right'><B>{$arreglo_etiquetas['apellido']}:<
           /B></td>
        <td> <input type='texto' name='apellido' size='65'
           maxlength='65'
                    value='$apellido'> </td>
      </tr>
      <tr><td
           align='right'><B>{$arreglo_etiquetas['telefono']}:<
           /B></td>
        <td> <input type='texto' name='telefono' size='65'
           maxlength='65'
                    value='$telefono'> </td>
      </tr>
      </table>
      <p><input type='submit' value='Enviar el nombre y numero
           telefonico'>
      </form>
      </center>";
    exit();
  }
  else   //si los datos estan bien
  {
    $usuario="admin";
    $huesped="localhost";
    $clave="";
    $basededatos = "Directoriodemiembros";
    $conexion = mysql_connect($huesped,$usuario,$clave)
       or die ("No se pudo conectar al servidor");
    $db = mysql_select_db($basededatos,$conexion)
       or die ("No se pudo seleccionar la base de datos");

    $consulta = "INSERT INTO Miembro
           (apellido,primer_nombre,telefono)
                    VALORES
           ('$apellido','$primer_nombre','$telefono')";
    $resultado = mysql_query($consulta)
       or die ("No se pudo ejecutar la consulta.");
    echo "Miembro nuevo agregado a la base de datos<br>";
  }
?>
</body></html>
```

Este programa amplía el programa en la Lista 8-14. Revisa los datos en el formulario en busca de campos vacíos o formatos incorrectos, y le pide al usuario que vuelva a digitar los datos cuando encuentra un problema. Si los datos están bien, el programa los recorta, los depura y los almacena en la base de datos.

El programa en la Lista 8-15 es un programa de demostración, el cual muestra cómo añadir una sola línea a una tabla de una base de datos. Si usted usa este programa, sólo puede agregar un cliente a la base de datos, pues el programa no crea ni inserta un nombre de registro único. Es necesario borrar el cliente que agregó antes de poder correr el programa nuevamente. Este no es un programa que se usaría para agregar clientes en una situación real. Tan sólo se trata de un programa de demostración muy sencillo para mostrar los principios que se necesitan. La Lista 12-3 en el Capítulo 12 es un ejemplo de un programa real para añadir clientes, el cual sería útil en una aplicación web verdadera.

Su aplicación podría necesita almaceanr datos en varios lugares diferentes. Una función que almacena datos de un formulario puede ser muy útil. La siguiente es una función que almacena todos los datos en un formulario:

```
funcion guardadatos($datosformulario,$nombretabla)
{
   if (!is_array($datosformulario))
   {
     return FALSE;
     exit();
   }
   foreach ($datosformulario como $campo => $valor)
   {
   $$datosformulario [$campo] = trim($$datosformulario [$campo]);
      $$datosformulario [$campo] = strip_tags($$datosformula-
           rio [$campo]);
      if ($campo == "telefono")
      {
          $$datosformulario [$campo] =
       ereg_replace("[)(.-]","",$ $datosformulario [$campo]);
      }
      $serie_campo[]=$campo;
      $serie_valor[]=$$datosformulario [$campo];
   }
   $campos=implode(",",$serie_campo);
   $valores=implode('","',$serie_valor);
   $consulta = "INSERT INTO $nombre_tabla ($campos)
                    VALORS (\"$valores\")";
   $resultado = mysql_query($consulta)
            or die ("No se pudo ejecutar la consulta.");
   return TRUE;
}
```

La función devuelve `TRUE` si termina de insertar los datos sin ningún error. Al principio, la función verifica si los datos pasados efectivamente son un arreglo. Si `$datosformulario` no es un arreglo, la función se detiene y devuelve `FALSE`.

Observe que esta función sólo trabaja si los nombres de los campos en el formulario son iguales a los nombres de las columnas en la tabla de la base de datos. Además, observe que esta función asume que usted ya está conectado al servidor MySQL y ha seleccionado la base de datos correcta. Al usar esta función, esta es la última parte del programa en la Lista 8-15:

```
else   //si los datos estan bien
  {
    $almacenado = guardadatos($_POST,"Miembro");
    echo "Nuevo miembro agregado a la base de datos<br>";
  }
?>
</body></html>
```

Observe cuánto más fácil de leer es este programa, con la mayoría de los enunciados en la función. Más aún, esta función trabaja para cualquier formulario, siempre y cuando los nombres de los campos en el formulario sean iguales a los de las columnas en la tabla de la base de datos. Si la función no es capaz de ejecutar la consulta, detiene la ejecución en ese punto e imprime el mensaje de error "No se pudo ejecutar la consulta". Si hay alguna circunstancia en la cual la consulta podría fallar, usted debe tomarla en cuenta.

Actualizar información existente

La información existente se actualiza con la consulta UPDATE, tal y como describo en el Capítulo 4. Actualizar significa cambiar datos en las columnas de filas que ya están en la base de datos: no añadir filas nuevas a la tabla de la base de datos. El formato general es

```
$consulta = "UPDATE nombretabla SET col=valor WHERE col=valor";
$resultado = mysql_query($consulta)
      or die ("No se pudo ejecutar la consulta.");
```

Por ejemplo, los enunciados para actualizar el número telefónico de Goliat Pérez son

```
$primer_nombre = "Goliat";        // del campo del formulario
$apellido = "Perez";            // del campo del formulario
$telefono = "555-555-5555";       // del campo del formulario
$consulta = "UPDATE Miembro SET telefono='$telefono'
                    WHERE apellido='$apellido'
                    AND primer_nombre='$primer_nombre'";
$resultado = mysql_query($consulta)
      or die ("No se pudo ejecutar la consulta.");
```

Si no usa una cláusula WHERE en una consulta UPDATE, el campo SET se establece para todas las filas. Kara vez es eso lo que usted desea hacer.

La Lista 8-16 muestra un programa llamado actualizantelefono.php, el cual almacena un nombre y un número telefónico de un formulario.

Lista 8-16: Programa que actualiza los datos

```
<?php
/*  Nombre del programa: actualizaTelefono.php
 *  Descripcion: El programa revisa el numero telefonico para
          ver si el formato es incorrecto. Actualiza el
          numero de telefono en la base de datos para el
          nombre especificado.
 */
?>
<html>
<head><title>Numero telefonico del miembro</title></head>
<body>
<?php
    $telefono = strip_tags(trim($_POST['telefono']));
    $telefono = ereg_replace("[)( .-]","",$telefono);
    $primer_nombre = $_POST['primer_nombre'];
    $apellido = $_POST['apellido'];

  /* verificar la informacion en el formulario */

  /* establecer una serie de etiquetas para los campos */
  $arreglo_etiquetas = array ( "primer_nombre" => "Primer
          nombre",
                        "apellido" => "Apellido",
                        "telefono" => "Telefono");
  foreach ($_POST as $campo => $valor)
  {
    /* revisar cada campo en busca de campos en blanco */
    if ( $valor == "" )
    {
        $arreglo_blanco[$campo] = "blanco";
    }
    elseif ( ereg("(nombre)",$campo) )
    {
        if (!ereg("^[A-Za-z' -]{1,50}$",$_POST[$campo]) )
        {
            $mal_formato[$campo] = "malo";
        }
    }
    elseif ($campo == "telefono")
    {
      if(!ereg("^[0-9)( -]{7,20}(([xX]|(ext)|(ex))?[ -]?[0-
          9]{1,7})?$",$valor))
      {
         $mal_formato[$campo] = "malo";
      }
    }
  }
  /* si algun campo no estaba bien, mostrar mensaje de error y
          formulario */
  if (@sizeof($arreglo_blanco) > 0 or @sizeof($mal_formato) >
          0)
  {
    if (@sizeof($arreglo_blanco) > 0)
    {
        /* desplegar mensaje de informacion faltante */
        echo "<b>No completo uno o mas de los campos
          obligatorios.
```

```
                Debe digitar:</b><br>";
        /* mostrar lista de informacion faltante */
        foreach($arreglo_blanco as $campo => $valor)
        {
         echo
            "   {$arreglo_etiquetas[$campo]}<br>
            ";
        }
    }
    if (@sizeof($mal_formato) > 0)
    {
        /* desplegar mensaje de informacion incorrecta */
        echo "<b>Uno o mas campos tienen informacion que
            parece incorrecta. Corrija el formato de:</b><br>";
        /* desplegar lista de informacion incorrecta */
        foreach($mal_formato as $campo => $valor)
        {
         echo
            "   {$arreglo_etiquetas[$campo]}<br>
            ";
        }
    }
    /* volver a mostrar el formulario */
    echo "<p><hr>
      <form action='chequeaTodos.php' method='POST'>
      <center>
      <table width='95%' border='0' cellspacing='0'
            cellpadding='2'>
      <tr><td align='right'>

            <b>{$arreglo_etiquetas['primer_nombre']}:</b></td>
          <td><input type='text' name='primer_nombre'
          size='65'
              maxlength='65' value='$primer_nombre' > </td>
      </tr>
      <tr><td align='right'>
              <b>{$arreglo_etiquetas['apellido']}:</b></td>
        <td> <input type='text' name='apellido' size='65'
                maxlength='65' value='$apellido'> </td>
      </tr>
      <tr><td align='right'>
             <b>{$arreglo_etiquetas['telefono']}:</b></td>
        <td> <input type='text' name='telefono' size='65'
              maxlength='65' value='$telefono'> </td>
      </tr>
      </table>
      <p><input type='submit'
              value='Enviar el nombre y numero telefonico'>
      </form>
      </center>";
    exit();
}
else   //si los datos estan bien
{
$usuario="admin";
$huesped="localhost";
$clave="";
$basededatos = "DirectoriodeMiembros";
$conexion = mysql_connect($huesped,$usuario,$clave)
```

```
        or die ("No se pudo conectar al servidor");
$db = mysql_select_db($basededatos,$conexion)
        or die ("No se pudo seleccionar la base de datos");

    $consulta = "UPDATE Miembro SET telefono='$telefono'
                    WHERE apellido='$apellido' AND
            primer_nombre='$primer_nombre'";
    $resultado = mysql_query($consulta)
        or die ("No se pudo ejecutar la
            consulta.".mysql_error());
    echo " El numero telefonico del miembro se ha
            actualizado<br>";

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

El programa de la Lista 8-16, el cual actualiza la base de datos, es casi idéntico al de la Lista 8-15, que agrega datos nuevos. Usar una consulta UPDATE en este programa (en lugar de la consulta INSERT usada para añadir datos nuevos) es la principal diferencia. Ambos programas verifican los datos y luego los depuran porque ambos programas almacenan datos en la base de datos.

Poner la información en archivos

A veces, usted desea recibir un archivo completo de información de un usuario, tal como currículos de usuarios para su sitio de búsqueda de empleos, o fotos para su sitio web de álbum fotográfico. O bien, imagine que está construyendo el catálogo a partir de información suministrada por el departamento de Ventas. Junto con el texto descriptivo sobre el producto, quiere que Ventas le proporcione una foto del producto. Usted puede ofrecer un formulario que Ventas puede usar para cargar un archivo de imagen.

Usar un formulario para subir el archivo

Puede mostrar un formulario que permite al usuario cargar un archivo usando un formulario HTML diseñado para dicho propósito. El formato general del formulario es el siguiente:

```
<form enctype="multipart/form-data"
        action="processfile.php" method="POST">
  <input type="hidden" name="MAX_FILE_SIZE" value="30000">
  <input type="file" name="archivo_usuario">
  <input type="submit" value="Subir archivo ">
</form>
```

Observe los puntos siguientes en relación con el formulario:

- **El atributo** enctype **se usa en la** etiqueta **form.** Usted debe fijar este atributo en `multipart/form-data` al subir un archivo, para garantizar que el archivo llegue correctamente.
- **Se incluye un campo oculto que envía un valor (en bytes) para** `MAX_FILE_SIZE`. Si el usuario trata de cargar un archivo que es más grande que este valor, no podrá hacerlo. Usted puede fijar este valor hasta un máximo de 2MB. Si necesita subir un archivo más grande, deberá cambiar la configuración predeterminada para `upload_max_filesize in php.ini` a un número mayor antes de enviar un valor superior a 2MB para `MAX_FILE_SIZE` en el campo oculto.
- **El campo de input que carga el archivo es del tipo** `file`. Observe que el campo tiene un nombre (`archivo_usuario`), al igual que los otros tipos de campos en un formulario. El nombre de archivo que el usuario digita en el formulario se envía al programa de procesamiento y está disponible en el arreglo incorporado llamado `FILES`. Le explico la estructura e información en `FILES` en la siguiente sección.

Cuando el usuario envía el formulario, el archivo se sube a una ubicación temporal. El script que procesa el formulario necesita copiar el archivo a otra ubicación, pues el archivo temporal se borra tan pronto como el script termina.

Procesar el archivo cargado

La información sobre el archive subido se almacena en el arreglo PHP incorporada llamada `$_FILES`. Hay un arreglo de información disponible para cada archivo que haya sido cargado, por lo cual `$_FILES` es un arreglo multidimensional. Como con cualquier otro formulario, usted puede extraer la información del arreglo usando el nombre del campo. La siguiente es el arreglo disponible de `$_FILES` para cada archivo que se carga.

```
$_FILES['nombrecampo']['nombre']
$_FILES['nombrecampo']['tipo']
$_FILES['nombrecampo']['nombre_temporal']
$_FILES['nombrecampo']['tamaño']
```

Por ejemplo, suponga que usa el siguiente campo para subir un archivo, como se mostró en la sección anterior:

```
<input type="file" nombre="archivo_usuario">
```

Si el usuario sube un archivo llamado `test.txt` usando el formulario, el arreglo resultante que puede usar el programa de procesamiento luce más o menos así:

```
$_FILES[archivo_usuario][nombre] = test.txt
$_FILES[archivo_usuario][tipo] = texto/simple
$_FILES[archivo_usuario][nombre_temporal] = D:\WINNT\php92C.tmp
$_FILES[archivo_usuario][tamaño] = 435
```

En esta serie, `nombre` es el nombre del archivo que se subió, `tipo` es el tipo de archivo, `nombre_temporal` es la ruta/nombre de archivo del archivo temporal y `435` es el tamaño. Observe que el nombre contiene sólo el nombre de archivo, pero `nombre_temporal` incluye la ruta del archivo además del nombre de archivo.

Si el archivo es demasiado grande para cargarse, el `nombre_temporal` en el arreglo se establece en `ninguno`, y el tamaño se establece en `0`. El programa de procesamiento debe mover el archivo subido desde la ubicación temporal hasta una ubicación permanente. El formato general del enunciado que mueve el archivo es el siguiente:

```
move_uploaded_file(ruta/nombrearchivotemporal,ruta/nombrear-
        chivopermanente);
```

ruta/nombrearchivotemporal está disponible en el elemento incorporado del arreglo `$_FILES['nombrecampo']['archivo_temporal']`. `ruta/nombrear chivopermanente` es la ruta hacia el archivo donde usted desea almacenar el archivo. El siguiente enunciado mueve el archivo subido en el campo de input, el cual tiene el nombre `archivo_usuario`, mostrado anteriormente en esta sección:

```
move_uploaded_file($_FILES['archivo_usuario']['nombre_temporal'],
  'c:\datos\new_file.txt');
```

El directorio de destino (en este caso, `c:\datos`) debe existir antes de poder mover el archivo hacia él. Este enunciado no crea el directorio de destino.

La seguridad es un asunto importante cuando se trata de cargar archivos. Permitir que extraños suban archivos a su PC es riesgoso; podría haber archivos maliciosos. Usted probablemente querrá revisar los archivos en busca de tantos factores como sea posible después de que haya sido cargados, usando enunciados condicionales para chequear las características del archivo, tales como el tipo y tamaño de archivo esperados. En algunos casos, para tener incluso más seguridad, sería una buena idea cambiar el nombre del archivo, de modo que los usuarios no sepan dónde están sus archivos o cómo se llaman.

Unir todo

Un ejemplo completo de un script se muestra en la Lista 8-17. Este programa despliega un formulario para que el usuario suba un archivo, guarda el archivo subido y luego muestra un mensaje después de que el archivo haya sido cargado exitosamente. Es decir, este programa despliega el formulario y lo procesa. El programa espera que el archivo subido sea un archivo de imagen, y lo prueba para asegurarse de que lo sea, pero cualquier tipo de archivo se puede subir. El código HTML para el formulario está en el archivo que se muestra en la Lista 8-18. Una página web que despliega el formulario aparece en la Figura 8/15.

Lista 8-17: Script que sube un archivo usando un formulario POST

```
<?php
 /* Nombre del programa: subirArchivo.php
  * Descripcion: Sube un archivo por medio de HTTP usando un
          formulario POST.
  */
  if(!isset($_POST['Subir']))                                    #5
  {
    include("form_upload.inc");
  } # endif
  else                                                           #9
  {
    if($_FILES['pix']['nombre_temporal'] == "ninguno")
          #11
    {
      echo "<b>El archivo no se subio exitosamente. Verifique
           el tamaño del archivo. El archivo debe ser menor
           que 500K.<br>";
      include("form_upload.inc");
      exit();
    }
    if(!ereg("image",$_FILES['pix']['type']))                    #16
    {
      echo "<b>El archivo no es una imagen. Por favor intente
           con otro archivo.</b><br>";
      include("form_upload.inc");
      exit();
    }
    else                                                         #23
    {
      $destino = 'c:\datos'."\\".$_FILES['pix']['nombre'];
      $archivo_temporal = $_FILES['pix']['nombre_temporal'];
      move_uploaded_file($archivo_temporal,$destino);
      echo "<p><b>El archivo se subio exitosamente:</b>
            {$_FILES['pix']['nombre']}
            ({$_FILES['pix']['tamannioo']})</p>";
    }
  }
?>
```

Añadí números al final de algunas de las líneas del script. El script se discute a continuación, con referencia a estos números:

5 Esta línea es un enunciado `if` que prueba si el formulario ha sido enviado. Si no, el formulario se muestra incluyendo el archivo que contiene el código del formulario. El archivo que se incluye se muestra en la Lista 8-18.

9 Esta línea empieza un bloque `else` que se ejecuta si el formulario se ha enviado. Este bloque incluye el resto del script y procesa el formulario enviado y sube el archivo.

11 Esta línea es un enunciado `if` que prueba si el archivo se subió exitosamente. Si no, un mensaje de error se despliega, y el formulario vuelve a aparecer.

16 Esta línea es un enunciado `if` que prueba si el archivo es una imagen. Si no, un mensaje de error se despliega, y el formulario vuelve a aparecer.

23 Esta línea inicia un bloque `else` que se ejecuta si el archivo ha sido subido exitosamente. El archivo se mueve a su destino permanente, y aparece un mensaje que indica que el archivo se ha subido.

La Lista 8-18 muestra el archivo que se incluye usado para mostrar el formulario para cargar.

Lista 8-18: Archivo incluido que muestra el formulario para subir un archivo

```
<!-  Nombre del Programa: form_upload.inc
    Descripcion: Muestra un formulario para subir un archivo -->
<html>
<head><title>Subir un archivo</title></head>
<body>
<ol><li>Digite el nombre de archivo de la foto del producto
          que desea subir o use el boton Buscar para navegar
          por el archivo de imagenes.</li>
    <li>Cuando la ruta hacia el archivo de la foto aparece en
          el campo de texto, haga clic en el boton Upload
          Picture.</li>
</ol>
<div align="center"><hr>
<form enctype="multipart/form-data"
        action="uploadFile.php" method="POST">
  <input type="hidden" nombre="MAX_FILE_SIZE" valor="500000">
  <input type="file" nombre="pix" tamaño="60">
  <p><input type="submit" nombre="Upload"
        valor="Upload Picture">
</form>
</body></html>
```

Observe que el archivo incluido no contiene código PHP, sólo código HTML.

El formulario que permite a los usuarios elegir un archivo para subir se muestra en la Figura 8-15. El formulario tiene un campo de texto para digitar el nombre del archivo y un botón Browse que permite al usuario navegar hasta el archivo y seleccionarlo.

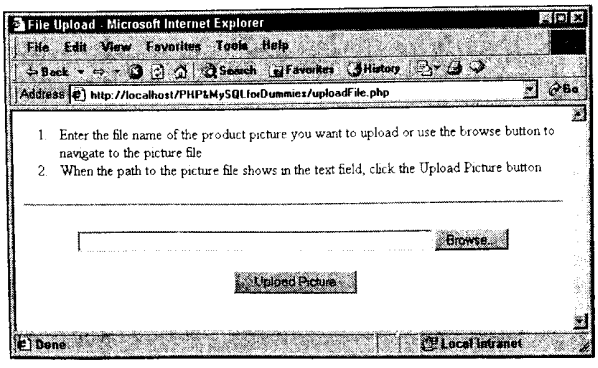


Figura 8-15: Formulario que permite a los usuarios subir un archivo de imagen.

Capítulo 9

Mover información de una página Web a otra

En este capítulo

- Mover a su usuario de una página a otra
- Mover información de una página a otra
- Agregar información a un URL
- Echar un vistazo a las cookies
- Usar campos ocultos en los formularios
- Descubrir sesiones PHP

La mayoría de los sitios web constan de más de una página web. Esto incluye páginas web estáticas que usted tal vez haya desarrollado en el pasado. Con las páginas web estáticas, los usuarios hacen clic en los vínculos para moverse de una página a la siguiente. Los usuarios hacen clic en un vínculo de una página web, y una nueva página aparece en su explorador. Cuando los usuarios se mueven de página en página de este modo, no se transfiere información de la primera página a la segunda. Cada página nueva que se envía al explorador del usuario es independiente de cualquier otra página que el usuario haya visto anteriormente. Con las páginas web dinámicas, usted quizás necesite transferir información de una página a otra. Si usted es un desarrollador de avanzado HTML, probablemente tenga algo de experiencia con métodos limitados para transferir información de una página a la siguiente usando formularios HTML y CGI (Common Gateway Interface) o cookies. Sin embargo, PHP es mucho más poderoso para pasar información de una página web a otra.

Mover a su usuario de una página a otra

Al usar sólo HTML, usted proporciona vínculos para que el visitante pueda ir de una página a otra en su sitio web. Cuando usa PHP, sin embargo, tiene tres opciones para mover a su usuario de una página a otra:

- **Vínculos:** Puede hacer eco de las etiquetas HTML que despliegan un vínculo. El formato general de un enunciado HTML que despliega un vínculo es

```
<a href="newpage.php">Texto que el usuarios ve como
        vinculo</a>
```

  Cuando los usuarios hacen clic sobre el vínculo, el programa `newpage.php` se envía a su buscador. Este método se usa extensivamente en las páginas web en HTML. Usted probablemente esté familiarizado con la creación de vínculos debido a su experiencia con HTML, pero si necesita un refrescamiento, puede averiguar más sobre los vínculos en cualquier libro sobre HTML, tal como HTML 4 For Dummies Quick Reference, 2ª Edición, por Deborah S. Ray y Eric J. Ray (Wiley Publishing, Inc.).

- **Botones para enviar formularios:** Puede usar un formulario HTML con uno o más botones de envío. Cuando el usuario hace clic en un botón de envío, el programa en la etiqueta `formulario` corre y envía una página web nueva al explorador del usuario. Usted puede crear un formulario sin campos (sólo con un botón de envío) pero el usuario debe hacer clic en el botón de envío para moverse a la página siguiente.

  Comento los formularios y los botones de envío en detalle en el Capítulo 8.

- **La función header:** Puede enviar un mensaje al servidor web que le dice que envíe una página nueva usando la función `header` de PHP. Cuando usa este método, puede mostrar una página nueva en el explorador del usuario sin que el usuario necesite hacer clic en un vínculo o un botón.

La función `header` puede usarse para enviar una página nueva al explorador del usuario. El programa usa un enunciado `header` y despliega la página web nueva sin necesidad de ninguna acción por parte del usuario. Cuando el enunciado header se ejecuta, la nueva página aparece. El formato de la función header que solicita una página nueva es

```
header("Location: URL");
```

El archivo localizado en el `URL` se envía al explorador del usuario. Cualquiera de los enunciados siguientes es un enunciado `header` válido:

```
header("Location: paginanueva.php");
header("Location:http://compañia.com/catalogo/catalogo.php");
```

Sin embargo, la función `header` tiene una limitación importante. El enunciado `header` sólo puede usarse antes de enviar cualquier otro output. No se puede mandar un mensaje solicitando una página nueva en medio de un programa después de haber hecho eco a algún output de la página web. Vea una discusión al respecto en el cuadro "Enunciados que deben ir antes del output".

A pesar de su limitación, la función `header` puede ser útil. Usted puede tener tantos enunciados PHP como desee antes de la función `header`, siempre y cuando no envíen output. Por lo tanto, los enunciados siguientes funcionarán:

URLs

Un URL (Uniform Resource Locator) es una dirección en la World Wide Web. Cada página web tiene su propio URL o dirección. El URL es utilizado por el servidor web para encontrar la página web y enviarla a un explorador.

El formato de un URL es

```
HTTP://nombredelservidor:numerodepuerto/ruta#meta?cadena=cadena
```

Este es un desglose de las partes que conforman el URL:

- `HTTP://nombredelservidor`: Este le indica al servidor que la dirección es un sitio web y le da el nombre del PC dondel sitio web se localiza. Otros tipos de transferencia se pueden especificar, tal como FTP (File Transfer Protocol), pero estos no se relacionan con el tema de este libro. Si se deja por fuera esta parte del URL, el servidor web asume que el PC es el mismo desde donde se está digitando el URL. Opciones válidas para esta parte son `HTTP://amazon.com` o `HTTP://localhost`. Nota: HTTP no debe estar necesariamente en mayúsculas.
- *:nymerodepuerto:* El servidor web intercambia información con Internet en un puerto particular del PC. La mayoría de las veces, el servidor web está configurado para comunicarse por medio del puerto 80. Si el número del puerto no se especifica, se asume que es el puerto 80. En algunas circunstancias poco comunes, un servidor web puede usar un número de puerto diferente, en cuyo caso el número del puerto debe especificarse. La razón más común para usar un número de puerto diferente es al montar un sitio web de prueba en otro puerto que está disponible sólo para los desarrolladores y probadores, pero no para los clientes. Cuando el sitio está listo para los clientes, se pone a la disposición en el puerto 80.
- *ruta:* Esta es la ruta hasta el archivo, la cual sigue las reglas de cualquier ruta. La raíz de la ruta es el directorio principal del sitio web. Si la ruta señala hacia un directorio, en lugar de hacia un archivo, el servidor web busca el nombre predeterminado del archivo, tal como `default.html` o `index.html`. La persona que administra el sitio web define el nombre del archivo predeterminado. La ruta `/catalogo/show.php` indica un directorio llamado `catalogo` que está en el directorio principal del sitio web y un archivo llamado `show.php`. La ruta `catalogo/show.php` indica un directorio llamado catálogo que está en el directorio actual.
- *#meta:* Una etiqueta HTML define una meta. Esta parte del URL muestra una página web en la ubicación donde está colocada la etiqueta objetivo. Por ejemplo, si la etiqueta `<a nombre="meta"></a>` está en medio del archivo en alguna parte, la página web aparecerá en la etiqueta, y no al inicio del archivo.
- *?cadena=cadena:* El signo de interrogación permite agregar información al final del URL. La información en formularios que usan el método get se pasa al final del URL en el formato `nombredecampo=valor`. Usted puede agregar información al final de un URL para pasarla a otra página. PHP automáticamente obtiene información del URL y la pone en series incorporadas. Usted puede pasar más de un par de `cadena=cadena` separando cada par con un signo (&): por ejemplo, `?estado=CA&ciudad=hogar`.

Enunciados que deben ir antes del output

Algunos enunciados PHP sólo pueden usarse antes de enviar cualquier output. Los enunciados `header` y `setcookie`, y las funciones `session` (todos ellos descritos en este capítulo), deben ir antes de enviar cualquier output. Si usted usa uno de estos enunciados después de enviar output, verá el siguiente mensaje:

```
Cannot add header information - headers already sent
```

El mensaje también proporcionará el nombre del archivo e indicará cuál línea envió el output anterior. O bien, tal vez usted no vea un mensaje del todo; la página nueva simplemente no aparecerá. (El que vea un mensaje de error o no dependerá del nivel de mensajes de error al que está configurado PHP; consulte el Capítulo 6 para más detalles). Los enunciados siguientes fallarán porque el mensaje `header` no es el primer output:

```
<html>
<head><title>comprobar encabezado</title></head>
<body>
<?php
  header("Location: http://empresa.com");
?>
</body>
</html>
```

Se envían tres líneas de código HTML antes del enunciado `header`. Los siguientes enunciados funcionarán, aunque no tienen mucho sentido:

```
<?php
   header("Location: http://empresa.com");
?>
<html>
<head><title>comprobar encabezado</title></head>
<body>
</body>
</html>
```

Las siguientes enunciados fallarán:

```
  <?php
   header("Location: http://empresa.com");
?>
<html>
<head><title>comprobar encabezado</title></head>
<body>
</body>
</html>
```

La razón por la cual estos enunciados fallan no es fácil de ver pero, si usted observa cuidadosamente, notará un espacio sencillo en blanco antes de la etiqueta PHP de apertura. Este espacio vacío es output para el explorador, aunque la página web resultante se vea vacía. Por lo tanto, el enunciado `header` falla porque hay output antes de él. Este es un error común, difícil de detectar.

```
<?php
 if ($edad_cliente < 13)
   {
  header("Location: Catalogodejuguetes.php");
   }
   else
   {
      header("Location: Catalogoequipoelectronico.php");
   }
?>
```

Estos enunciados corren un programa que despliega un catálogo de juguetes si la edad del cliente es menor que 13 años; por otro lado, corren un programa que muestra un catálogo de equipo electrónico, si la edad del cliente es 13 o más años.

Mover información de una página a otra

Las páginas HTML son independientes entre sí. Cuando un usuario hace clic en un enlace, el servidor web envía una página nueva al explorador del usuario, pero el servidor web no sabe nada sobre la página anterior. Para las páginas HTML estáticas, este proceso funciona bien. Sin embargo, muchas aplicaciones dinámicas necesitan pasar información de una página a otra. Por ejemplo, usted podría querer almacenar un nombre de usuario y referirse a esa persona por su nombre en otra página web.

Las aplicaciones web dinámicas a menudo consisten en muchas páginas y esperan que el usuario visualice varias páginas diferentes. El período que empieza cuando un usuario ve la primera página y finaliza cuando el usuario deja el sitio web es una sesión. A menudo usted desea que haya información disponible durante una sesión completa. Los siguientes son ejemplos de sesiones que necesitan compartir información entre páginas:

- **Restringir el acceso a un sitio web:** Suponga que su sitio web es restringido, y los usuarios se registran con una contraseña para tener acceso al sitio. Usted no quiere que los usuarios deban registrarse en cada página. Sólo quiere que lo hagan una vez y, después, tengan la posibilidad de ver todas las páginas que quieran. Usted desea que los usuarios traigan información consigo a cada página, la cual demuestre que se han registrado y están autorizados para visualizar la página. Usted quiere que los usuarios se registren y permanezcan registrados durante toda la sesión.
- **Proporcionar páginas web basadas en exploradores:** Como los exploradores interpretan algunas características HTML de maneras diferentes, usted quizás desee proporcionar diferentes versiones de sus páginas web para distintos exploradores. Debe verificar el buscador de su usuario cuando éste visualiza la primera página, y luego entregar todas las demás páginas con base en el tipo y la versión de explorador del usuario.

Con PHP, puede mover información de una página a otra usando cualquiera de estos métodos:

- **Agregar información al URL:** Usted puede añadir cierta información al final del URL de la página nueva, y PHP pondrá la información en series incorporadas que usted puede usar en la página nueva. Este método es más apropiado cuando necesita pasar sólo una pequeña cantidad de información.
- **Almacenar información mediante cookies:** Puede almacenar cookies (pequeñas cantidades de información que contienen parejas de `variable=valor`) en el PC del usuario. Una vez almacenada la cookie, puede obtenerse de cualquier página web. Sin embargo, los usuarios pueden rehusarse a aceptar las cookies. Por lo tanto, este método sólo funciona en ambientes en los cuales usted está seguro que el usuario tendrá las cookies activadas.
- **Pasar información usando formularios HTML:** Puede pasar información a un programa específico usando una etiqueta `formulario`. Cuando el usuario hace clic en el botón de envío, la información en el formulario se envía al siguiente programa. Este método es muy útil cuando usted necesita recopilar información de los usuarios.
- **Usar funciones de sesión PHP:** Empezando con PHP 4, hay funciones PHP disponibles que configuran una sesión de usuario y almacenan la información correspondiente en el servidor; se puede tener acceso a dicha información desde cualquier página web. Este método es más útil para sesiones en las cuales usted espera que los usuarios vean muchas páginas.

Agregar información al URL

Una manera simple de mover información de una página a otra es agregar información al URL. Ponga la información en el formato siguiente:

```
variable=valor
```

`variable` es el nombre de la variable, pero no use el signo de dólar ($) en ella. El `valor` es el valor por almacenarse en la variable. Puede agregar las parejas `variable=valor` en cualquier parte donde use un URL. El inicio de la información se señala con un signo de interrogación (?). Todos los enunciados a continuación son formas válidas de pasar información en el URL:

```
<form action="paginasiguiente.php?estado=CA" method="POST">

<a href="paginasiguiente.php?estado=CA">ir a la pagina si-
        guiente</a>

header("Location: paginasiguiente.php?estado=CA");
```

Puede agregar varias parejas `variable=valor`, si las separa con signos (&), así:

```
<form action="paginasiguiente.php?estado=CA&ciudad=hogar" method="POST">
```

He aquí dos razones por las cuales tal vez no quiera pasar información en el URL:

- **Seguridad:** El URL aparece en la línea de dirección del explorador, lo cual significa que la información añadida al URL también se ve. Si la información necesita ser segura, usted no querrá exhibirla tan públicamente. Por ejemplo, si está moviendo una contraseña de una página a la siguiente, probablemente no desea pasarla en el URL. Además, el URL puede ser marcado como favorito por el usuario. Puede haber razones por las cuales usted no quiera que sus usuarios guarden la información añadida al URL.
- **Longitud de la cadena:** Hay un límite sobre la longitud del URL. El límite difiere según el tipo y la versión del explorador, pero siempre hay un límite. Por lo tanto, si está pasando mucha información, quizás no haya espacio para ella en el URL.

Agregar información al URL es muy útil para una transferencia rápida y sencilla de datos. Por ejemplo, suponga que desea proporcionar una página web donde los usuarios puedan actualizar su número telefónico. Usted quiere que el formulario se comporte de la siguiente manera:

1. Cuando el usuario despliega el formulario por primera vez, el número telefónico de la base de datos aparece en el formulario, de modo que el usuario puede ver cuál es el número almacenado actualmente en la base de datos.
2. Cuando el usuario envía el formulario, el programa revisa el número telefónico para ver si el campo está en blanco o si está en un formato que no podría ser un número de teléfono.
3. Si el número telefónico resulta correcto, se almacena en la base de datos.
4. Si el número telefónico está en blanco o tiene datos incorrectos, el programa vuelve a enseñar el formulario. Sin embargo, esta vez usted no quiere mostrar los datos de la base de datos. Por el contrario, desea mostrar los datos incorrectos que el usuario digitó y envió en el campo del formulario.

El programa `mostrartelefono.php` en la Lista 9-1 muestra cómo usar el URL para determinar si esta es la primera vez que se muestra el formulario o una subsiguiente. El programa muestra el número telefónico para el nombre de registro del usuario y permite al usuario cambiarlo.

Lista 9-1: Programa que muestra el número telefónico en un formulario

```
<?php
/*  Nombre del programa: mostrarTel.php
 *  Descripcion: Muestra el numero telefonico recuperado de
          la base de datos y permite al usuario cambiarlo.
*/
?>
<html>
<head><title>Mostrar numero telefonico</title></head>
<body>
<?php
```

```
$huesped="localhost";
$usuario="admin";
$clave="";
$basededatos="Directoriodemiembros";
$nombre_entrada = "gperez";  // pasado de la pagina
        anterior
$conexion = mysql_connect($huesped,$usuario,$clave)
    or die ("No se pudo conectar al servidor ");
$db = mysql_select_db($basededatos,$conexion)
    or die ("No se pudo seleccionar la base de datos");

if (@$_GET['primero'] == "no")
{
  $telefono = $_POST['telefono'];
  if (!ereg("^[0-9)( -]{7,20}(([xX]|(ext)|(ex))?[ -]?[0-
        9]{1,7})?$",$telefono)
        or $telefono == "")
  {
      echo "<p align='center'>El numero telefonico parece
         no ser valido.<br>";
  }
  else
  {
      $consulta = "UPDATE Miembro SET telefono='$telefono'
                          WHERE
         nombrenetrada='$nombreentrada'";
      $output = mysql_query($consulta)
          or die ("No se pudo ejecutar la consulta.");
      echo "Numero telefonico ha sido actualizado.<br>";
      exit();
  }
}
else
{
  $consulta = "SELECT telefono FROM Miembro WHERE
        nombreentrada='$nombreentrada'";
  $output = mysql_query($consulta)
      or die ("No se pudo ejecutar la consulta.");
  $fila = mysql_fetch_array($output);
  extract($fila);
}

/* Mostrar telefono del usuario en formulario */
echo "<br><p align='center'>
     <font size='+1'><b>Por favor verifique el numero
         telefonico a continuacion y corrijalo si es
         necesario.</b></font>
     <hr>
     <form action='mostrarTel.php?first=no' method='POST'>
      <div align='center'>
      <table width='50%' border='0' cellspacing='0'
         cellpadding='2'>
      <tr><td align='right'><B>$nombrenetrada</br></td>
         <td align='center'><input type='text'
         name='telefono'
              size='20' maxlength='20' value='$telefono' >
         </td>
      </tr>
      <tr><td></td><td align='center'>
```

```
            <br><input type='submit' value='Enviar numero
            telefonico'></td>
          </tr>
          </table>
        </form>";
?>
</body></html>
```

Observe los siguientes puntos claves sobre este programa:

- **El mismo programa despliega y procesa el formulario.** El nombre de este programa es `mostrartelefono.php`. Observe que la etiqueta `formulario` incluye `action=mostrartelefono.php`, lo cual significa que, cuando el usuario hace clic en el botón de envío, el mismo programa corre nuevamente.
- **Se agrega información al URL.** La etiqueta `formulario` incluye `action=mostrartelefono.php?first=no`. Cuando el usuario hace clic en el botón de envío y `mostrartelefono.php` corre por segunda vez, la variable `$primero` se pasa con el valor "no".
- **El valor que se pasó para primero en el arreglo incorporado $_GET se revisa al inicio del programa.** Esto es para ver si esta es la primera vez que el programa corre.
- **Si $_GET[primero] es igual a "no", el número telefónico se revisa.** `$_GET[primero]` sólo es igual a no si el formulario está siendo enviado. `$_GET[primero]` no es igual a no si esta es la primera vez que se atraviesa por el programa.
  - Si el número telefónico está bien, se almacena en la base de datos y el programa termina.
  - Si el número de teléfono no está bien, aparece un mensaje de error.
- **Si $_GET[primero] no es igual a "no", el número de teléfono se recupera de la base de datos.** En otras palabras, si `$_GET[primero]` no es igual a no, se trata de la primera vez que el programa corre. El programa debería conseguir el número telefónico de la base de datos.
- **Si el programa alcanza los enunciados que despliegan el formulario, éste será mostrado.** Si esta no es la primera vez que se atraviesa por el programa y se envió un número telefónico correcto, el número se almacena en la base de datos, y el programa se detiene. Nunca llegará hasta los enunciados que despliegan el formulario. En todos los demás casos, el formulario se mostrará.

El formulario desplegado por el programa en la Lista 9-1 se muestra en la Figura 9-1: se ilustra la apariencia de la página web al aparecer por primera vez. Observe que el URL en el campo de dirección del explorador no tienen ninguna información añadida.

La Figura 9-2 muestra los resultados cuando un usuario digita un número telefónico sin sentido en el formulario de la Figura 9-1. Observe que el URL en el campo de dirección del explorador tiene ahora `?primero=no` agregado al final.

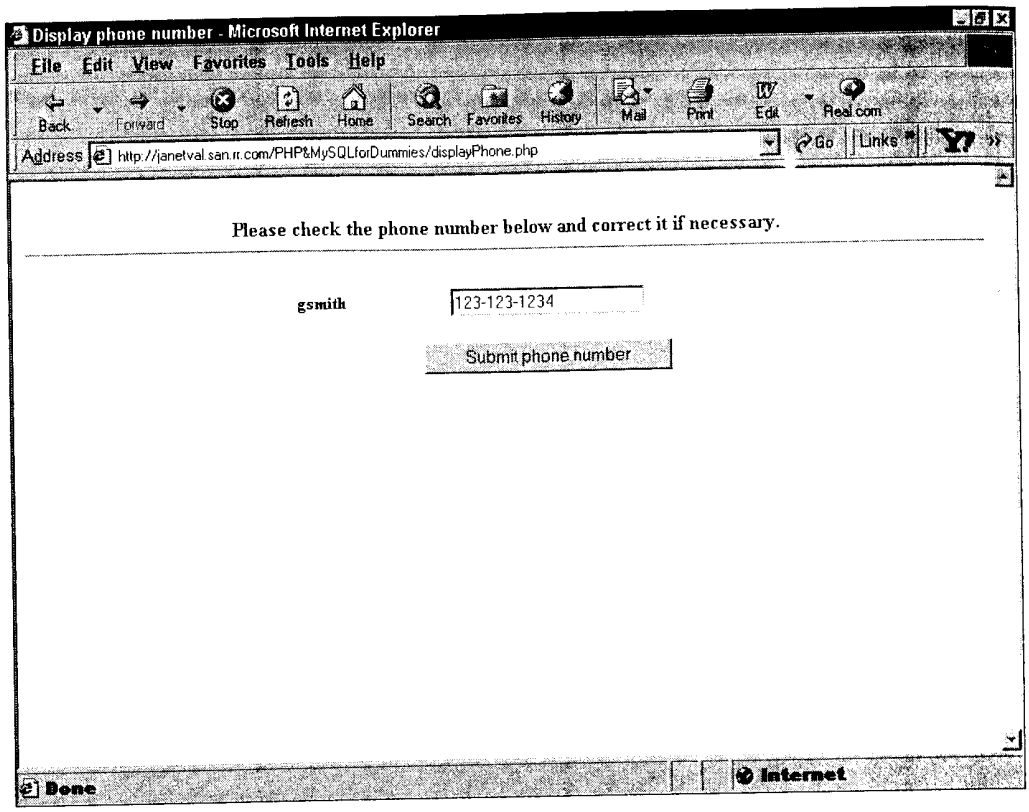


Figura 9-1: Formulario HTML para actualizar un número telefónico.

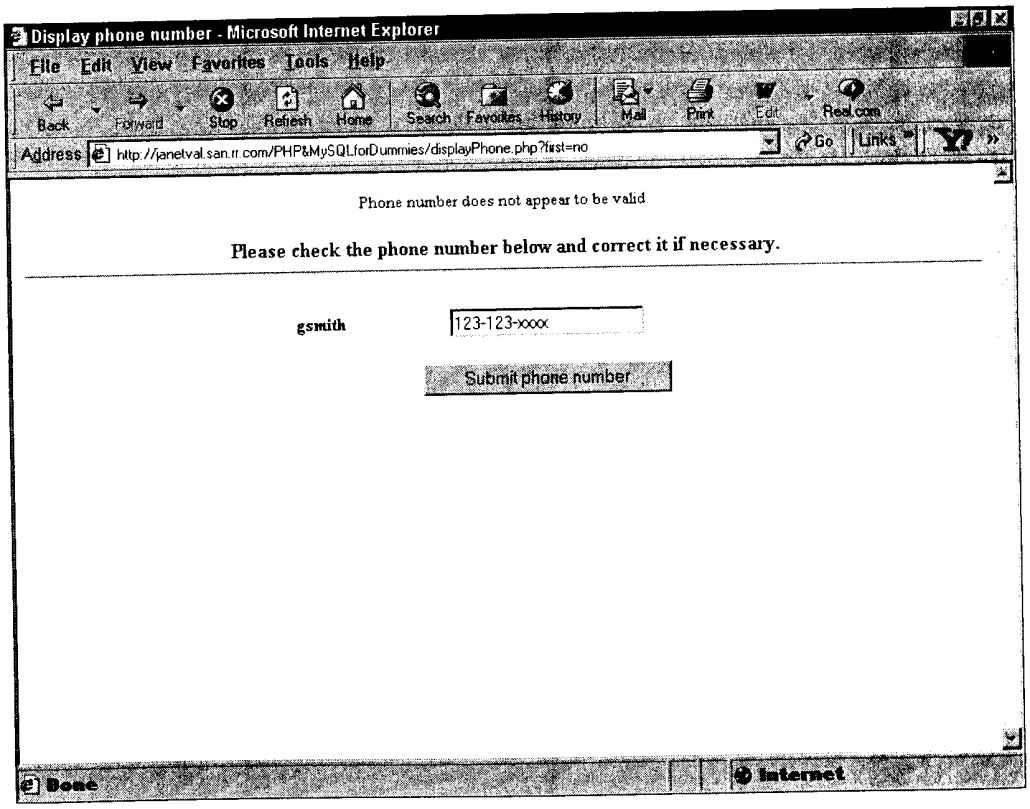


Figura 9-2: Formulario HTML cuando un usuario envía un número telefónico sin sentido.

Almacenar información mediante cookies

Usted puede almacenar información como cookies. Las cookies son pequeñas cantidades de información que contienen parejas `variable=valor`, parecidas a las parejas que pueden agregarse al URL. El explorador del usuario almacena las cookies en el PC del usuario. Su aplicación puede luego obtener la cookie de cualquier página web. Por qué se les llama cookies (galletas, en inglés) es uno de los grandes misterios de la vida. Tal vez sea porque, a primera vista, parecieran ser algo maravilloso; pero, al examinarlas más de cerca, usted se da cuenta de que no le hacen muy bien. Para algunas personas en algunas situaciones, las cookies no son útiles para nada.

A primera vista, las cookies parecen resolver todo el problema de mover datos de página en página. Sólo meta una cookie en el PC del usuario y consígala cuando la necesite. De hecho, la cookie se puede almacenar de modo tal que permanecerá ahí después de que el usuario deje su sitio y estará disponible cuando el usuario entre a su sitio web de nuevo, un mes después. ¡Problema resuelto! Bueno, no realmente. Las cookies no están bajo su control: están bajo el control del usuario. El usuario puede en cualquier momento borrarlas. De hecho, los usuarios pueden configurar sus exploradores para rehusarse a aceptarlas. Y muchos usuarios las rechazan o las borran rutinariamente. Muchos usuarios no se sienten cómodos con la idea de que un extraño almacene cosas en sus PCs, especialmente archivos que permanecen allí incluso después de dejar el sitio web del extraño. Es una actitud comprensible. Sin embargo, definitivamente limita la utilidad de las cookies. Si su aplicación depende de ellas y el usuario las tiene desactivadas, su aplicación no funcionará para él.

Las cookies fueron diseñadas originalmente para almacenar pequeñas cantidades de información por períodos cortos de tiempo. A menos que usted específicamente configure la cookie para permanecer durante un período más largo de tiempo, ésta desaparecerá cuando el usuario deje su sitio web. Aunque las cookies son útiles en algunas situaciones, usted probablemente no las necesitará en su aplicación con base de datos para la Web, por estas razones:

- **Los usuarios configurar sus exploradores para rechazar cookies.** A menos que usted sepa con certeza que todos sus usuarios tendrán las cookies activadas o pueda solicitarles que las activen (y espera que ellos respondan favorablemente), las cookies son un problema. Si su aplicación depende de ellas, no funcionará si están desactivadas.
- **PHP tiene características que funcionan mejor que las cookies.** Empezando con PHP 4, PHP incluye funciones que crean sesiones y almacenan información que está disponible durante toda la sesión. Esta característica de sesiones es más confiable y muchos más fácil de usar que las cookies para poner la información disponible en todas las páginas web en una sesión. Las sesiones no funcionan para almacenar información a largo plazo, pero las bases de datos MySQL pueden usarse con ese propósito.
- **Puede almacenar datos en su base de datos.** Su aplicación incluye una base de datos donde puede almacenar y recuperar datos, la cual gene-

ralmente es una mejor solución que una cookie. Los usuarios no pueden borrar los datos en su base de datos en forma inesperada. Como usted está usando una base de datos en esta aplicación, puede usarla para cualquier almacenamiento de datos que necesite, especialmente el almacenamiento a largo plazo. Las cookies son más útiles para aplicaciones que no hacen uso de bases de datos.

Las cookies se almacenan por medio de la función `setcookie`. El formato general es

```
setcookie("variable","valor");
```

`variable` es el nombre de la variable, pero no incluya el signo de dólar ($). Este enunciado almacena la información sólo hasta que el usuario salga de su sitio web. Por ejemplo, el siguiente enunciado

```
setcookie("estado","CA");
```

almacena `CA` en una variable cookie llamada estado. Después de configurar la cookie, la información estará disponible para sus otros programas PHP en el elemento de un arreglo incorporado como `$_COOKIE[estado]`. No necesita hacer nada para obtener la información de la cookie. PHP lo hace automáticamente. La cookie no estará disponible en el programa donde está configurada. El usuario debe ir a otra página o volver a mostrar la página actual antes de que la información de la cookie pueda usarse.

Las cookies también están disponibles en un arreglo llamado `$HTTP_COOKIE_VARS` con los nombres de variables como claves. Por ejemplo, el valor en la cookie configurada en el ejemplo anterior también se puede usar como `$HTTP_COOKIE_VARS['estado']`. Esta serie integrada debe usarse si está usando una versión de PHP anterior a PHP 4.1.

Si desea que la información almacenada en una cookie permanezca en un archivo en el PC del usuario después de que el usuario éste abandone su sitio web, configure su cookie con un tiempo de expiración, como sigue:

```
setcookie("variable","valor",tiempodeexpiracion);
```

El valor `tiempodeexpiracion` establece el tiempo cuando la cookie expirará. `tiempodeexpiracion` generalmente se establece usando la función `time` o `mktime` como sigue:

- `time`: Esta función devuelve el tiempo actual en un formato que el PC puede entender. Se usa la función time más un número de segundos para establecer el tiempo de expiración de la cookie, como se muestra en los enunciados siguientes:

```
setcookie("estado","CA",time()+3600); //expira en 1 hora
setcookie("Nombre",$Nombre,time()+(3*86400))// expira en
        3 dias
```

- `mktime`: Esta función devuelve una fecha y hora en un formato que el PC puede comprender. Usted debe proporcionar la fecha y hora deseadas en el siguiente orden: hora, minuto, segundo, mes, día y año. Si cualquier valor no se incluye, se usa el valor actual. La función `mktime` se utiliza para configurar el tiempo de expiración de una cookie, como se muestra en los enunciados siguientes:

```
setcookie("estado","CA",mktime(3,0,0,4,1,2003));
    //expira a las 3:00 AM el 1 de abril, 2003.
setcookie("estado","CA",mktime(12,0,0,,,));
    //expira al mediodia de hoy
```

Puede remover una cookie al establecer su valor en nada. Cualquiera de los enunciados siguientes retira la cookie:

```
setcookie("nombre");
setcookie("nombre","");
```

Empero, la función `setcookie` tiene una limitación importante. Sólo puede usarse antes de enviar cualquier output. No se puede establecer una cookie a la mitad de un programa, después de haber hecho eco de algún output hacia la página web. Consulte el cuadro "Enunciados que deben ir antes del output", anteriormente en este capítulo.

Pasar información con formularios HTML

El modo más común de pasar información de una página a otra es a través de formularios HTML. Un formulario HTML se despliega con un botón de envío. Cuando el usuario hace clic en el botón, la información en los campos del formulario se pasa al programa incluido en la etiqueta del formulario. El formato general es

```
<form action="procesaformulario.php" method="POST">
    etiquetas para uno o mas campos
    <input type="submit" valor="cadena">
</form>
```

El uso más común de un formulario es recopilar información de los usuarios (el cual comento en detalle en el Capítulo 8). Sin embargo, también se pueden usar formularios para pasar otros tipos de información usando campos ocultos, los cuales se agregan al formulario y se mandan con la información que el usuario digitó. De hecho, usted puede crear un formulario que tenga únicamente campos ocultos. Siempre necesitará un botón de envío, y la página nueva no se desplegará hasta que el usuario no haga clic en él, pero no hace falta incluir ningún campo para que el usuario complete.

Por ejemplo, los enunciados siguientes pasan el color de fondo preferido por el usuario a la próxima página cuando el usuario hace clic en un botón llamado Siguiente página:

```
<?php
   $color = "azul"; //pasada por medio de un formulario
   echo "<form action='paginasiguiente.php' method='POST'>
         <input type='oculto' nombre='color' valor='$color'>
         <input type='submit' valor='Siguiente pagina '>
         </form>\n";
?>
```

La página web muestra un botón de envío etiquetado Siguiente página, pero no le pide al usuario ninguna información. Cuando el usuario hace clic en el botón, `paginasiguiente.php` corre y puede usar el elemento del arreglo `$_POST[color]`, la cual contiene `"azul"`.

Usar sesiones PHP

Una sesión es el tiempo que un usuario pasa en su sitio web. Los usuarios pueden visualizar muchas páginas web durante el tiempo comprendido entre su ingreso al sitio y su abandono de él. A menudo usted quiere que la información persiga al usuario por el sitio, de modo que esté disponible en cada página. PHP, empezando con la versión 4.0, proporciona un modo para hacerlo.

Cómo funcionan las sesiones PHP

PHP le permite configurar una sesión en una página web y guardar variables como variables de sesión. Luego, usted abre la sesión en cualquier otra página y las variables de sesión están disponibles para su uso en series incorporadas `$_SESSION`. Para lograrlo, PHP hace lo siguiente:

1. **Asigna un número de identificación a la sesión:** El número es realmente largo y no tiene mucho sentido; es exclusivo para el usuario y nadie podría adivinarlo nunca. La identificación de sesión se almacena en una variable del sistema PHP llamada `PHPSESSID`.
2. **Almacena las variables de sesión en un archivo en el servidor:** El nombre del archivo es el número de identificación de la sesión. El archivo se almacena en `\tmp` en Unix/Linux; en Windows, se almacena en un directorio llamado `sessiondata`, bajo el directorio donde está instalado PHP.

   Si usted es el administrador de PHP, puede cambiar la ubicación donde se almacenan los archivos de sesión al editar el archivo de configuración php.ini. Encuentre la configuración para `session.save_path` y cambie la ruta a la ubicación donde desea almacenar los archivos.
3. **Pasa el número de identificación de la sesión a cada página:** Si el usuario tiene las cookies activadas, PHP pasa la identificación de sesión usando cookies. Si el usuario tiene las cookies desactivadas, PHP pasa la identificación de sesión en el URL para los vínculos o en una variable oculta para formularios que usan el método `post`.

4. **Obtiene las variables del archivo de la sesión para cada nueva página de sesión:** Cuando un usuario abre una página nueva que es parte de la sesión, PHP consigue las variables del archivo usando el número de identificación de sesión que fue pasado de la página vieja y lo pone en el arreglo incorporado `$_SESSION`. Usted puede usar los elementos del arreglo con el nombre de la variable como clave, y tienen el valor que usted asignó en la página anterior.

Las sesione no funciona a menos que `track_vars` esté activado. A partir de PHP 4.0.3, `track-vars` siempre está activado. Para versiones anteriores a 4.0.3, la opción `-enable-track-vars` debería usarse al instalar PHP.

Si los usuarios tienen las cookies desactivadas, las sesiones no funcionarán a no ser que `trans-sid` esté activado. Aprenda cómo activar y desactivar `trans-sid` más adelante, en la sección "Usar variables de sesión PHP".

Abrir sesiones

Usted debería abrir una sesión en cada página web. La sesión se abre con la función `session_start`, como sigue:

```
session_start();
```

La función primero revisa si hay un número existente de identificación de sesión. Si encuentra uno, configura el arreglo `$_SESSION`. Si no encuentra ninguno, empieza una sesión nueva al crear un nuevo número de identificación de sesión.

Puesto que las sesiones usan cookies si el usuario las tiene activadas, `session_start` está sujeto a la misma limitación que las cookies. O sea, se debe llamar a la función `session_start` antes de enviar cualquier output. Puede leer los detalles completos en el cuadro "Enunciados que deben ir antes del output", anteriormente en este capítulo.

Usted puede decirle a PHP que cada página en su sitio debería empezar automáticamente con `session_start`. Para hacer esto, edite el archivo de configuración `php.ini`. Si usted es el administrador de PHP, puede editar este archivo; de lo contrario, pida al administrador que lo haga. Busque la variable `session.auto_start` y establezca su valor en 1. Tal vez deba reiniciar el servidor web antes de que esto surta efecto. Con `auto_start` activado, no necesita agregar `session_start` al principio de cada página.

Usar variables de sesión PHP

Cuando desea guardar una variable como una variable de sesión (es decir, una que esté disponible en otras páginas web que el usuario podría visitar), guárdela en el arreglo `$_SESSION`, así:

```
$_SESSION['nombredevariable'] = valor;
```

El valor de la variable estará entonces disponible en el arreglo $_SESSION en otras páginas web. Por ejemplo, usted puede almacenar el estado donde vive el usuario usando el siguiente enunciado:

```
$_SESSION['estado'] = "CA";
```

Luego puede utilizar $_SESSION['estado'] en cualquier otra página web, y tendrá el valor CA.

Los dos siguientes programas muestran cómo usar sesiones para pasar información de una página a la próxima. El primer programa, testsession1.php en la Lista 9-2, muestra la primera página donde empieza la sesión. La Lista 9-3 muestra el programa testsession2.php para la segunda página de una sesión.

Lista 9-2: Empezar una sesión

```
<?php
  session_start();
?>
<html>
<head><title>Prueba de pagina Sesiones 1</title></head>
<body>
<?php
  $_SESSION['s'] = "prueba";
  echo "Esta es una prueba de la caracteristica sesiones.
       <form action='testsesion2.php' method='POST'>
       <input type='hidden' name='form_var'
              valor='prueba'>
       <input type='submit' value='ir a la proxima pagina '>
       </form>";
?>
</body></html>
```

Observe que dos variables están configuradas en este programa para que sean pasadas a la segunda página. Se crea la variable de sesión session_var. Además, se despliega un formulario con una variable oculta form_var, la cual también se pasa a la segunda página cuando se oprime el botón de envío. Ambas variables están establecidas en "prueba".

Lista 9-3: Segunda página de una sesión

```
<?php
  session_start();
?>
<html>
<head><title>Prueba de pagina Sesiones 2</title></head>
<body>
<?php
  echo "session_var = {$_SESSION['s']}<br>\n";
  echo "form_var = {$_POST['form_var']}<br>\n";
?>
</body></html>
```

Dirija su buscador a testsession1.php y luego haga clic en el botón de envío que dice Ir a la siguiente página. Entonces, verá el siguiente output de `sessionTest2.php`:

```
session_var = prueba
form_var = prueba
```

Como las sesiones funcionan de maneras diferentes para los usuarios con las cookies activadas y aquellos que las tienen desactivadas, usted debería probar los dos programas en ambas condiciones. Para desactivar las cookies en su explorador, se cambian las configuraciones para las opciones o preferencias.

Para desactivar las cookies en Internet Explorer, siga estos pasos:

1. **Escoja Herramientas⇨Opciones de Internet.**

   Se abre el recuadro de diálogo Opciones de Internet.

2. **Haga clic en la pestaña Seguridad en IE 5.5 o la pestaña Privacidad en IE 6.**
3. **Haga clic en el icono de Internet para resaltarlo.**
4. **Haga clic en el botón Nivel Personalizado.**

   Aparece el recuadro de diálogo Configuración de Suguridad.

5. **Desplácese hasta la sección Cookies y seleccione Deshabilitar para cada una de las configuraciones de cookie.**
6. **Haga clic en Aceptar.**

Para desactivar las cookies en Netscape Navigator, siga estos pasos:

1. **Escoja Editar⇨Referencias.**
2. **Resalte Avanzadas.**
3. **Marque Deshabilitar Cookies.**
4. **Haga clic en Aceptar.**

Si el output de `testsession2` muestra un valor en blanco para `$session_var` cuando desactiva las cookies en su buscador, probablemente sea porque `trans-sid` no está activado. Puede activarlo en su archivo `php.ini`. Encuentre la siguiente línea:

```
session.use_trans_sid = 0
```

Cambie el 0 a 1 para activar `trans-sid`. Si no puede arreglar este problema, todavía puede usar sesiones, pero debe pasar el número de identificación de la sesión en sus enunciados de programación, ya que PHP no lo pasará automáticamente cuando las cookies están desactivadas. Para más detalles sobre cómo usar sesiones cuando `trans-sid` no está activado, consulte la siguiente sección.

Para PHP 4.1.2 o anterior, `trans-sid` no está disponible a menos que haya sido activado usando la opción `--enable-trans-sid` cuando PHP se compiló.

Sesiones sin cookies

Muchos usuarios desactivan las cookies en sus exploradores. PHP revisa el explorador del usuario para ver si se permiten las cookies y se comparta de acuerdo con esto. Si el buscador del usuario permite cookies, PHP hace lo siguiente:

- Establece la variable `$PHPSESSID` igual al número de identificación de la sesión.
- Usa cookies para mover `$PHPSESSID` de una página a otra.

Si el explorador del usuario no acepta las cookies, PHP hace lo siguiente:

- **Establece una constante llamada SID.** La constante contiene una pareja `variable=valor` que luce como `PHPSESSID=`*`unalargacadenadenumeros`*
- **Podría mover el número de identificación de sesión de una página a otra, y podría no hacerlo,** dependiendo de si trans-sid está activado. Si está activado, PHP pasa el número de identificación de la sesión; si está desactivado, PHP no pasa el número.

Activar trans-sid tiene ventajas y desventajas. La ventaja es que las sesiones funcionan sin problemas incluso cuando los usuarios desactivan las cookies. También es mucho más fácil programar sesiones con trans-sid activado. La desventaja es que el número de identificación de sesión a menudo se pasa en el URL. En algunas situaciones, este número no debería mostrarse en la dirección del explorador. Además, cuando el número está en el URL, el usuario lo puede marcar como favorito. Luego, si el usuario regresa a su sitio usando la marca de favorito con el número de identificación de sesión en ella, el nuevo número de identificación de sesión de la visita actual se puede confundir con el número viejo de la visita anterior, y probablemente cause problemas.

Sesiones con trans-sid activado

Cuando `trans-sid` está activado y el usuario tiene las cookies desactivadas, PHP automáticamente envía el número de identificación de sesión en el URL o como un campo oculto en un formulario. Si el usuario se mueve a la página siguiente usando un vínculo, una función header o un formulario con el método get, el número se agrega al URL. Si el usuario se mueve a la página siguiente usando un formulario con el método post, el número se pasa en un campo oculto. PHP reconoce `$PHPSESSID` como el número de identificación de la sesión y maneja la sesión sin ninguna programación especial de su parte.

El número de identificación de la sesión sólo se añade a los URLs de las páginas en su sitio web. Si el URL de la página siguiente incluye el nombre de un servidor, PHP asume que el URL está en otro sitio web y no añade el número de identificación de la sesión. Por ejemplo, si su enunciado de vínculo es

```
<a href="paginanueva.php">
```

PHP añadirá el número de identificación. Sin embargo, si su enunciado es

```
<a href="HTTP://www.empresa.com/paginanueva.php">
```

PHP no lo agregará.

Sesiones con trans-sid desactivado

Cuando `trans-sid` no está activado, PHP no envía el número de identificación de la sesión a la siguiente página cuando los usuarios tienen las cookies desactivadas. Más bien, usted mismo debe enviar el número de identificación.

Afortunadamente, PHP proporciona una constante que usted puede usar para mandar la identificación de sesión usted mismo. Una constante es una variable con información que no puede modificarse. (Describo las constantes en el Capítulo 6). La constante que PHP proporciona se llama SID y contiene una pareja `variable=valor` que usted puede agregar al URL, como sigue:

```
<a href="paginasiguiente.php?<?php echo SID?>" > pagina si-
        guiente </a>
```

Este enunciado de vínculo agrega un signo de interrogación (?) y la constante `SID al URL. SID` contiene el número de identificación de sesión, formateado como `variable=valor`. El output de `echo SID` luce más o menos así:

```
PHPSESSID=877c22163d8df9deb342c7333cfe38a7
```

Por lo tanto, el URL que se envía es

```
<a href="paginasiguiente.php?PHPSESSID=877c22163d8df9deb342c7333cfe38a7>
  pagina siguiente </a>
```

Por alguna de varias razones (las cuales comento en la sección "Agregar información al URL", anteriormente en este capítulo), tal vez usted no quiera que el número de identificación de la sesión aparezca en el URL mostrado por el explorador. Para impedirlo, puede mandar el número de identificación en un campo oculto de un formulario usando el método post. Primero, consiga el número de identificación de la sesión; luego envíelo en un campo oculto. Los enunciados para hacerlo son

```
<?php
  $PHPSESSID = session_id();
  echo "<form action='paginasiguiente.php' method='POST'>
        <input type='oculto' nombre='PHPSESSID'
               valor='$PHPSESSID'>
        <input type='submit' valor='Pagina siguiente'>
        </form>";
?>
```

Estos enunciados hacen lo siguiente:

1. Almacenan el número de identificación de la sesión en una variable llamada `$PHPSESSID`. Usan la función `session_id`, la cual proporciona el número de identificación de la sesión actual.
2. Mandan `$PHPSESSID` en un campo oculto de un formulario.

En la página nueva, PHP automáticamente usará `$PHPSESSID` para conseguir cualquier variable de sesión, sin ninguna programación especial de su parte.

Hacer privadas las sesiones

Las funciones de sesión de PHP son ideales para sitios web que son restringidos y requieren que los usuarios se registren con un nombre de ingreso y una contraseña. Esos sitios web indudablemente tienen muchas páginas, y usted no querrá que el usuario deba conectarse en cada página. Las sesiones PHP pueden determinar si el usuario se ha registrado o no, y rechazar el acceso a los usuarios que no estén registrados. Puede usar sesiones PHP para hacer lo siguiente:

1. Mostrar a los usuarios una página de registro.
2. Si un usuario se registra satisfactoriamente, establece y almacena una variable de sesión.
3. Cada vez que el usuario vaya a una página nueva, revisa la variable de sesión para ver si el usuario se ha registrado.
4. Si el usuario ya se registró, muestra la página.
5. Si el usuario no se ha registrado, muestra la página de registro.

Para verificar si un usuario se ha registrado, agregue los siguientes enunciados en la parte superior de cada página:

```
<?php
  session_start()
  if ( @$_SESSION['login'] != "si" )
  {
    header("Location: paginaderegistro.php");
    exit();
  }
?>
```

En estos enunciados, `$_SESSION[login]` es una variable de sesión que se establece en "sí" cuando el usuario se registra. Los enunciados verifican si `$_SESSION[login]` es igual a `"si"`. Si no lo es, el usuario no está registrado, y se le envía a la página de registro. Si `$_SESSION[login]` es igual a "sí", el programa sigue adelante con el resto de los enunciados en la página web.

Cerrar sesiones PHP

Para sesiones restringidas en las cuales los usuarios deben registrarse, a menudo usted quiere que los usuarios se desconecten cuando hayan terminado. Para cerrar una sesión, use el enunciado siguiente:

```
session_destroy();
```

Este enunciado se deshace de toda la información sobre variables de sesión almacenada en el archivo de sesión. PHP ya no pasa el número de identificación de la sesión a la siguiente página. No obstante, el enunciado no afecta a las variables que están configuradas en la página actual: siguen teniendo los mismos valores. Si desea eliminar las variables de la página actual (además de impedir que pasen a la página siguiente) suprímalas con este enunciado:

```
unset($_SESSION );
```

Parte IV
Aplicaciones

La 5a Ola Por Rich Tennant

"La nueva tecnología realmente me ha ayudado a organizarme. Guardo los informes de mis proyectos debajo del PC, los presupuestos debajo de mi PC portátil, y los memorandos debajo de mi localizador".

En esta parte. . .

En esta parte, usted aprende a tomar el plan y la información para iniciar de la Parte I, la información sobre MySQL de la Parte II y la información sobre PHP de la Parte III, y a juntarlo todo en una sola aplicación con base de datos para la Web. Los Capítulos 11 y 12 le presentan dos aplicaciones de muestra, incluyendo sus bases de datos y todos sus programas PHP.

Capítulo 10

Unirlo todo

En este capítulo

- Organizar toda su aplicación
- Organizar programas individuales
- Hacer que su aplicación sea segura
- Documentar su aplicación

Los capítulos anteriores de este libro le brindan las herramientas necesarias para construir su aplicación con base de datos para la Web. En la Parte I, usted aprendió cómo funcionan PHP y MySQL y cómo tener acceso a ellos. Además, descubrió qué debe hacerse para construir su aplicación y en qué orden hacerlo. En la Parte II, averiguó cómo construir y usar una base de datos MySQL. En la Parte III, descubrió cuáles características tiene PHP y cómo usarlas. Además, esta parte también explicó cómo mostrar información en una página web, recopilar información de los usuarios y almacenar la información en una base de datos. Ahora, está listo para unirlo todo.

En este capítulo, le muestro cómo juntar todas las piezas para formar una aplicación completa. Para hacerlo, usted deberá

- Organizar la aplicación
- Asegurarse de que la aplicación es segura
- Documentar la aplicación

Aquí le describo cada uno de los pasos en detalle.

Organizar la aplicación

Organizar la aplicación es para beneficio suyo. En lo que respecta a PHP, la aplicación podría tener 8 millones de enunciados PHP en una sola línea de un archivo del PC. A PHP no le importan las líneas, las sangrías ni los archivos. Sin embargo, los humanos escriben y dan mantenimiento a los programas para la aplicación, y los humanos sí necesitan organización. Las aplicaciones requieren de dos niveles de organización:

- **El nivel de la aplicación:** La mayoría de las aplicaciones necesita más de un programa para tener funcionalidad completa. Usted debe dividir las funciones de la aplicación en un conjunto organizado de programas.
- **El nivel del programa:** La mayoría de los programas realizan más de una tarea específica. Es necesario dividir las tareas del programa en secciones dentro del programa.

Organizarse en el nivel de la aplicación

En general, las aplicaciones con bases de datos para la Web consisten en un programa por página web. Por ejemplo, tal vez usted tenga un programa que proporcione un formulario para recopilar información y un programa para almacenar la información en una base de datos, el cual le dice al usuario que los datos han sido almacenados.

Otro fundamento de la organización es un programa por tarea principal. Por ejemplo, usted podría tener un programa para presentar el formulario y un programa que almacene los datos en una base de datos. Para las aplicaciones web, la mayoría de las tareas importantes conllevan enviar una página web. Recopilar los datos de los usuarios requiere de una página web para el formulario HTML (HyperText Markup Language ó Lenguaje de Marcado de Hipertexto, por sus siglas en inglés); proporcionar información sobre los productos a los clientes requiere de páginas web; y cuando usted almacena datos en una base de datos, generalmente desea enviar una página de confirmación al usuario indicando que los datos fueron almacenados.

Un programa por página web o un programa por tarea principal no es una regla, sino más bien una guía. La única regla relacionada con la organización es que ésta debe ser clara y fácil de entender. Y esto es totalmente subjetivo. Aun así, la organización de una aplicación como el Catálogo de mascotas no necesita ser demasiado complicada. Suponga que el diseño del Catálogo de mascotas exige que la primera página enumere todos los tipos de mascotas (tales como gatos, perros y pájaros) entre los cuales el usuario puede seleccionar. Entonces, cuando el usuario haya seleccionado un tipo, todas las mascotas en el catálogo para ese tipo se muestran en la siguiente página web. Una organización razonable sería tener dos programas: uno para mostrar la página con los tipos de mascotas y otro para mostrar las mascotas según el tipo de mascota escogido.

Estas son algunas sugerencias adicionales para organizar sus programas:

- **Escoja nombres muy descriptivos para los programas en su aplicación.** Los nombres de los programas son parte de la documentación que hace que su aplicación se pueda entender. Por ejemplo, nombres útiles para los programas del Catálogo de mascotas serían `MostrarTiposdemascotas.php` y `MostrarMascotas.php`. Se acostumbra, aunque no es requisito, empezar los nombres de programas con una letra mayúscula. El uso de mayúsculas o minúsculas no es importante para los nombres de los programas en PCs con Windows, pero sí es muy importante para PCs con

Unix/Linux. Ponga atención a las letras mayúsculas y minúsculas de modo que sus programas puedan correr en cualquier PC si fuese necesario.

- **Ponga los archivos de los programas en subdirectorios con nombres significativos.** Por ejemplo, ponga todos los archivos de imágenes en un directorio llamado `imagenes`. Si sólo tiene tres archivos, podría bastarle un solo directorio, pero buscar entre docenas de archivos un archivo específico puede hacerle perder mucho tiempo.

Organizarse en el nivel del programa

Un programa individual bien organizado es muy importante por las siguientes razones:

- **Es más fácil de escribir.** Cuanto mejor organizado esté su programa, más fácil será para usted leerlo y comprenderlo. Puede ver lo que el programa hace y arreglar cualquier problema con mayor rapidez.
- **Es más fácil para otros entenderlo.** Tal vez otros necesiten comprender su programa. Después de cobrar esa enorme herencia y partir hacia la Isla del Mar del Sur que compró, alguien más tendrá que darle mantenimiento a su aplicación.
- **Es más fácil para usted darle mantenimiento.** Sin importar cuán a fondo usted la pruebe, su aplicación probablemente tendrá uno o dos problemas. Cuanto mejor organizado sea su programa, más fácil será para usted encontrar y reparar los problemas, especialmente seis meses después.
- **Es más fácil de cambiar.** Tarde o temprano, usted o alguien más deberá cambiar el programa. Las necesidades de los usuarios podrían cambiar. Las necesidades de su negocio podrían cambiar. La tecnología podría cambiar. La capa de ozono podría cambiar. Por una razón u otra, el programa necesitará modificarse. Averiguar qué hace el programa y cómo lo hace, para poder cambiarlo, resulta mucho más fácil si está bien organizado. Le garantizo que usted no recordará los detalles; sólo deberá poder entender el programa.

Las siguientes reglas producirán programas bien organizados. No quisiera llamarlas reglas, porque puede haber razones en cualquier ambiente específico para romper una o más de las reglas, pero le recomiendo pensar cuidadosamente antes de romper cualquiera de las reglas siguientes:

- **Divida los enunciados en secciones para cada tarea específica.** Empiece cada sección con un comentario que describa lo que hace la sección. Separe las secciones entre sí agregando líneas en blanco. Por ejemplo, para el Catálogo de mascotas, el primer programa podría tener tres secciones para tres tareas:

  1. **Hacer eco del texto introductorio, tal como el encabezado de la página y los enunciados.** El comentario antes de la sección podría ser `/* texto de apertura */`. Si el programa `hace echo` a una gran cantidad de texto y gráficos complicados, usted podría dividirlo en más de una sección, tal como `/* titulo y logotipo */` y `/* enunciados */`.

2. **Extraer una lista de tipos de mascotas de la base de datos.** Si esta sección es larga y complicada, puede dividirla en secciones más pequeñas, tales como 1) conectarse con la base de datos; 2) ejecutar la consulta `SELECT`; y 3) poner los datos en variables.

3. **Crear un formulario que muestre una lista de selección de los tipos de mascotas.** Los formularios a menudo son largos y complicados. Puede ser útil tener una sección para cada parte del formulario.

- **Use sangrías.** Use sangría en los bloques de enunciados PHP. Por ejemplo, sangre los bloques `if` y `while` como lo he hecho yo en el código de muestra para este libro. Si hay bloques anidados dentro de otros bloques, use una sangría adicional para el bloque anidado. Es mucho más fácil ver dónde empiezan y terminan los bloques cuando están sangrados, lo cual, a su vez, facilita la comprensión de lo que hace el programa. Poner sangrías en los enunciados HTML también puede resultar útil. Por ejemplo, si usted pone sangrías en las líneas entre las etiquetas de apertura y cierre para un formulario o entre la etiqueta `<table>` y la etiqueta `</table>`, puede ver más fácilmente qué hacen los enunciados.

- **Use comentarios libremente.** Definitivamente, añada comentarios al inicio para explicar lo que hace el programa. Y agregue comentarios para cada sección. Además, comente cualquier enunciado que usted considere que no es obvio, o enunciados en los cuales cree haber hecho algo de un modo poco común. Si le tomó un rato averiguar cómo hacer algo, es probable que amerite un comentario. No olvide los comentarios cortos al final de las líneas; a veces, sólo una palabra o dos son de gran ayuda.

- **Use enunciados simples.** A veces los programadores se dejan llevar por la idea de usar códigos concisos, con lo cual sacrifican la legibilidad. Anidar seis llamados de función uno dentro de otro puede ahorrar algunas líneas y teclazos, pero también hará que el programa sea más difícil de leer.

- **Repita los bloques de enunciados.** Si se da cuenta que está digitando las mismas diez líneas de enunciados PHP en varios lugares del programa, puede mover ese bloque de enunciados hacia otro campo y llamarlo cuando lo necesite. Una línea en su programa que diga `extreaerDatos()` es mucho más fácil de leer que las diez líneas que extraen los datos. Es más, si necesita cambiar algo en dichas líneas, puede hacerlo en un archivo externo en vez de tener que encontrarlo y cambiarlo en una docena de lugares diferentes en su programa. Hay dos maneras de repetir enunciados: las funciones y los enunciados `include`. El Capítulo 7 explica cómo escribir y usar funciones. Las dos secciones siguientes explican el uso de funciones y enunciados `include` en la organización de su programa.

- **Use constantes.** Si su programa usa el mismo valor muchas veces, tal como el impuesto de ventas de su estado, puede definir una constante al principio del programa para crear una constante llamada `CA_SALES_TAX` que es.97 y usarla cuando la necesite. Definir una constante que le brinde un nombre al número ayuda a cualquiera que lea el programa a entender cuál es el número; además, si alguna vez necesita cambiarlo, sólo deberá cambiarlo en un lugar. Las constantes se describen en detalle en el Capítulo 6.

Usar enunciados include

PHP le permite poner enunciados en un archivo *externo* (o sea, un archivo separado de su programa) e insertar el archivo donde quiera en el programa, usando el enunciado `include`. Los archivos `include` son muy útiles para almacenar un bloque de enunciados que se repite. Se agrega un enunciado `include` donde quiera usar los enunciados en lugar de añadir todo el bloque de enunciados en varios lugares diferentes. Este enunciado hace que sus programas sean mucho más cortos y fáciles de leer. El formato de un enunciado `include` es

```
include("nombredearchivo");
```

El archivo puede tener cualquier nombre. A mí me gusta usar la extensión `.inc`. Los enunciados en el archivo se incluyen tal y como están, en el punto donde se usa el enunciado `include`. Los enunciados se incluyen como HTML, no PHP. Por lo tanto, si desea usar enunciados PHP en su archivo `include`, debe incluir etiquetas PHP en el archivo `include`. De lo contrario, todos los enunciados en el archivo `include` se verán como HTML y serán enviados como output en la página web será tal y como estén.

Estas son algunas maneras de usar archivos `include` para organizar sus programas:

- **Ponga todo o la mayoría de su HTML en archivos `include`.** Por ejemplo, si su programa envía un formulario al explorador, ponga el HTML para el formulario en un archivo externo. Cuando necesite enviar el formulario, use un enunciado `include`. Poner el HTML en un archivo `include` es una buena idea si el formulario se muestra varias veces. Es incluso una buena idea si se muestra sólo una vez, pues hace que su programa sea mucho más fácil de leer. Los programas en los Capítulos 11 y 12 ponen el código HTML para los formularios en archivos separados e y haga `include` a los archivos cuando los formularios se despliegan.
- **Almacene la información necesaria para tener acceso a la base de datos en un archivo separado de su programa.** Almacene los nombres de las variables en el archivo como sigue:

  ```
  <?php
  $huesped="localhost";
  $usuario="raiz";
  $contraseña="";
  ?>
  ```

  Observe que este archivo necesita etiquetas `php` porque el enunciado `include` inserta el archivo como HTML. Incluya este archivo en la parte superior de cada programa que necesite conectarse a la base de datos. Si cualquier información (tal como la contraseña) cambia, simplemente cambie la contraseña en el archivo `include`. No necesita buscar entre todos los archivos del programa para cambiar la contraseña. Para un poco más de seguridad, es una buena idea usar un nombre de archivo engañoso, y no uno obvio como `claves_secretas.inc`.

- **Ponga sus funciones en archivos** `include`. Usted no necesita tener los enunciados para las funciones en el programa; puede ponerlos en un archivo `include`. Si cuenta con muchas funciones, organice las funciones que se relacionan en varios archivos `include`, tal como `funciones_dedatos.inc` y `funciones_delformulario.inc`. Use enunciados `include` al principio de sus programas, y enumere las funciones que se usan en el programa.

- **Almacene los enunciados que todos los archivos en su sitio web tienen en común.** La mayoría de los sitios web tiene muchas páginas web con muchos elementos en común. Por ejemplo, todas las páginas web empiezan con etiquetas `<html>`, `<head>` y `<body>`. Si usted almacena los enunciados comunes en un archivo `include`, puede incluirlos en cada página web, lo cual garantizará que todas sus páginas se vean iguales. Por ejemplo, usted podría tener los siguientes enunciados en un archivo `include`:

```
<html>
<head><title><?php echo $title ?></title></head>
<body topmargin="0">
<p align="center"><img src="logo.gif" width="100"
        height="200">
<hr color="rojo">
```

  Si incluye este archivo al inicio de cada programa en su sitio web, se ahorrará mucha digitación, y sabrá que todas las páginas concuerdan. Además, si desea modificar algo de la apariencia de todas sus páginas, sólo deberá cambiarlo en un lugar: el archivo `include`.

Puede usar un enunciado similar, como sigue:

```
include_once("nombredearchivo");
```

Este enunciado evita que los archivos include con variables parecidas se sobrescriban unos a otros. Use `include_once` cuando incluya sus funciones.

Puede usar el nombre de una variable para el nombre de archivo, así:

```
include("$nombredearchivo");
```

Por ejemplo, usted quizás desee mostrar mensajes diferentes en días diferentes. Puede almacenar estos mensajes en archivos con el nombre del día en que el mensaje debería desplegarse. Por ejemplo, podría tener un archivo llamado `dom.inc` con el siguiente contenido:

```
<p>No se preocupe. Duerma hasta tarde. No hay trabajo hoy.</p>
```

Y archivos parecidos para todos los días de la semana. Los enunciados siguientes se pueden usar para mostrar el mensaje correcto para el día actual:

```
$hoy = fecha("D");
include("$hoy".".inc");
```

Después del primer enunciado, `$hoy` contiene el día de la semana abreviado. El enunciado `fecha` se comenta en el Capítulo 6. El segundo enunciado incluye el archivo correcto, el cual usa el día almacenado en `$hoy`. Si `$hoy` contiene `dom`, el enunciado incluye un archivo llamado `dom.inc`.

Proteger sus archivos `include` es importante. La mejor manera de hacerlo es almacenar los archivos `include` en un directorio fuera de su espacio web, de modo que n los visitantes a su sitio web no puedan tener acceso a ellos.

Puede configurar un directorio `include` donde PHP busque cualquier archivo especificado en un enunciado `include`. Si usted es el administrador PHP, puede establecer un directorio `include` en el archivo `php.ini` (el archivo de configuración PHP en el directorio de su sistema, tal como describo en el Apéndice B). Encuentre la configuración para `include_path` y cámbiela a la ruta de su directorio preferido. Si hay un punto y coma al inicio de la línea, antes de `include_path`, quítelo. Los siguientes son ejemplos de configuraciones `include_path` en el archivo `php.ini`:

```
include_path=".;d:\include";          # para Windows
include_path=".:/usuario/local/include";  # para Unix/Linux/Mac
```

Estos dos enunciados especifican dos directorios donde PHP busca archivos `include`. El primer directorio es `dot` (es decir, el directorio actual), seguido de la ruta del segundo directorio. Usted puede especificar tantos directorios include como desee, y PHP buscará en ellos el archivo `include` en el orden en que se enumeran. Las rutas de los directorios se separan con un punto y como para Windows y dos puntos para Unix/Linux.

Si no tiene acceso a `php.ini`, puede configurar la ruta en cada script individual usando el siguiente enunciado:

```
ini_set("include_path","d:\hidden");
```

Este enunciado establece la ruta (`include_path`) hacia el directorio especificado sólo mientras el programa está corriendo. No establece el directorio para todo el sitio web.

Para tener acceso a un archivo desde un directorio `include`, simplemente use el nombre del archivo, como sigue. No hace falta usar el nombre completo de la ruta.

```
include("cleves_secretas.inc");
```

Si su archivo `include` no está en un directorio `include`, tal vez usted deba usar el nombre completo de la ruta en el enunciado `include`. Si el archivo está en el mismo directorio que el programa, basta con usar el nombre de archivo. No obstante, si el archivo se localiza en otro directorio, tal como un subdirectorio del directorio en el que está el programa o un directorio oculto fuera del espacio web, será necesario usar el nombre completo de la ruta al archivo, como sigue:

```
include("d:/hidden/claves_secretas.inc");
```

Usar funciones

Use las funciones con frecuencia para organizar sus programas. (En el Capítulo 7, le explico cómo crear y usar funciones en detalle). Las funciones son útiles cuando su programa necesita realizar la misma tarea en ubicaciones repetidas de un programa o en diferentes programas en la aplicación. Una vez escrita la función que lleva a cabo la tarea, y cuando usted sepa que funciona, puede usarla en cualquier parte en que la necesite.

Busque oportunidades para usar funciones. Su programa es mucho más fácil de leer y entender con una línea como esta:

```
extraerDatosdemiembros();
```

que con las 20 líneas de enunciados que en efecto extraen los datos.

De hecho, después de escribir programas PHP durante un tiempo, tendrá un arreglo de funciones que ha escrito para diversos programas. A menudo el programa que está escribiendo puede usar una función escrita para otra aplicación dos trabajos atrás. Por ejemplo, yo frecuentemente necesito usar una lista de los estados de EEUU. En lugar de incluir una lista con todos los 50 estados cada vez que la necesito, tengo una función llamada `extraerNombresdeestados ()` la cual devuelve un arreglo que guarda los nombres de los 50 estados en orden alfabético y una función llamada `extraerCodigosdeestados()` la cual devuelve un arreglo con los códigos de los estados que constan de dos letras, en el mismo orden. Estas funciones son útiles muy a menudo.

Dé a sus funciones nombre muy descriptivos. Los llamados de función en su programa deberían indicar exactamente qué hace la función. Está bien que sea largo. Usted no querrá ver una línea en su programa que diga

```
funcion1();
```

Esta línea no es muy informativa. Incluso una línea como la siguiente es menos informativa de lo que podría ser:

```
extraerDatos();
```

Es mejor ver una línea como esta:

```
extraerTodoslosnombresdemiembros ();
```

Mantenerlo en privado

Debe proteger su aplicación con base de datos para la Web. Alguna gente podría tener diseños atroces en su sitio web para propósitos tales como

- **Robar cosas:** Esperan encontrar un archivo por ahí, lleno de números válidos de tarjetas de crédito o la fórmula secreta de la eterna juventud.
- **Destrozar su sito web:** Algunas personas creen que es divertido. Algunas personas lo hacen para demostrar que pueden hacerlo.
- **Hacer daño a sus usuarios:** Una persona maliciosa puede agregar cosas a su sitio web que dañan a las personas que visitan su sitio, o que les roban cosas.

Este no es un libro sobre seguridad. La seguridad es un tema largo y complejo, y yo no soy experta en seguridad. Sin embargo, quiero orientar su atención hacia algunos aspectos y hacer algunas sugerencias que podrían ayudarle. Las siguientes medidas aumentarán la seguridad de su sitio web, pero si su sitio maneja información secreta y realmente importante, lea algunos libros sobre seguridad y hable con expertos:

- **Asegure la seguridad del PC que hospeda su sitio web.** Esto probablemente no sea su responsabilidad, pero sería bueno que hable con las personas responsables para discutir sus preocupaciones sobre seguridad. Se sentirá mejor si sabe que alguien se preocupa por la seguridad.
- **No deje que el servidor web muestre los nombres de archivos.** Los usuarios no necesitan saber los nombres de los archivos en su sitio web.
- **Oculte las cosas.** Almacene su información de modo que nadie pueda tener acceso a ella fácilmente desde la Web.
- **No confíe en la información de los usuarios.** Siempre depure cualquier información que no haya generado usted mismo.
- **Use un servidor web seguro.** Esto requiere de trabajo adicional, pero es importante si usted cuenta con información ultra secreta.

Asegure la seguridad del PC

Primero, el PC mismo debe estar seguro. El administrador del sistema del PC es responsable de mantener a los visitantes no autorizados y a los vándalos fuera del sistema. Las medidas de seguridad incluyen cosas tales como barreras de protección (firewalls), encriptación, ocultamiento de contraseñas, detectores de escaneado, etc. En la mayoría de los casos, usted no es el administrador del sistema. Si es así, debe investigar seriamente los temas relacionados con la seguridad. Si está usando una compañía de hospedaje web, sería bueno que discuta la seguridad con los encargados para sentirse más seguro de que está usando suficientes medidas de seguridad.

No deje que el servidor web muestre los nombres de archivos

Tal vez habrá notado que a veces usted recibe una lista de nombres de archivo cuando apunta a un URL. Si apunta a un directorio (y no a un archivo específico) y el directorio no contiene un archivo con el nombre de archivo predeterminado (tal como `index.html`), el servidor web podría desplegar una lista de archivos entre los cuales seleccionar. Usted probablemente no querrá que su servidor web haga esto; su sitio no será seguro si un visitante puede ojear cualquier archivo en él. En otros sitios web, quizás haya visto un mensaje de error que dice

```
Forbidden
You don't have permission to access /secretdirectory on this server.
```

En estos sitios, el servidor web está configurado de manera que no despliegue una lista de nombres de archivos cuando el URL apunta hacia un directorio. En su lugar, muestra este mensaje de error. Esto es más seguro que enumerar los nombres de los archivos. Si el nombre de archivo es enviado desde su sitio web, debe cambiarse una configuración para el servidor web. Si usted no es el administrador de su servidor web, solicite un cambio. Si usted sí lo es, depende de usted cambiar este comportamiento. Por ejemplo, en Apache, este comportamiento es controlado mediante una opción llamada *Indexes*, la cual se puede activar y desactivar en el archivo `httpd.conf` como sigue:

```
Options Indexes          // la activa
Options -Indexes         // la desactiva
```

Consulte la documentación para su servidor web para permitir o prohibir mostrar la lista del directorio en el explorador web del usuario.

Oculte las cosas

Mantenga la información tan privada como le sea posible. Por supuesto, las páginas web que desea que los visitantes vean deben almacenarse en un directorio de espacio web público. Pero no todo necesita almacenarse ahí. Por ejemplo, puede almacenar archivos include en un lugar totalmente diferente, en un espacio del PC al cual no se pueda tener acceso desde la Web. Su base de datos ciertamente no se almacena en su espacio web, pero podría estar incluso más segura si estuviera en un PC totalmente diferente.

Otra forma de ocultar cosas es darles nombres engañosos. Por ejemplo, el archivo include que contiene las variables de la base de datos no debería llamarse `claves.inc`. Un mejor nombre sería `Recetadesopadepollotio-Juan.inc`. Sé que esta sugerencia viola otras secciones del libro donde promuevo el uso de nombres de archivo descriptivos, pero este es un caso especial. Las personas maliciosas a veces hacen cosas obvias como digitar `www.susitio.com/claves.html` en su explorador para ver qué sucede.

No confíe en la información de los usuarios

Los usuarios malintencionados pueden usar los formularios en sus páginas web para enviar texto peligroso a su sitio web. Por lo tanto, nunca almacene información de los formularios directamente en su base de datos sin revisarla primero. Verifique la información que recibe en busca de formatos razonables y caracteres peligrosos. En particular, no debe aceptar etiquetas HTML, tales como etiquetas `<script>`, de los formularios. Al usar etiquetas script, un usuario podría digitar un script real, quizás uno malintencionado. Si usted acepta el campo del formulario sin revisarlo y lo almacena en su base de datos, podría tener cualquier cantidad de problemas, particularmente si el script almacenado fue enviado en una página web a un visitante de su sitio web. Para más sobre cómo chequear los datos de los formularios, vea el Capítulo 8.

Use un servidor web seguro

La comunicación entre su sitio web y sus visitantes no es totalmente segura. Cuando los archivos en su sitio web se envían al buscador del usuario, alguien en Internet entre usted y el usuario puede leer los contenidos de estos archivos conforme pasan. Para la mayoría de los sitios web, esto no representa un problema; sin embargo, si su sitio recopila o envía números de tarjetas de crédito u otra información secreta, use un servidor web seguro para proteger estos datos.

Los servidores web seguros usan Security Sockets Layer (SSL) para proteger la comunicación enviada a los exploradores y recibida de ellos. Esto es parecido a las llamadas telefónicas codificadas de las cuales oye hablar en las películas de espías. La información se *codifica* (se traduce a cadenas codificadas) antes de ser enviada por la Web. El software que la recibe la descodifica hasta dar con su contenido original. Además, su sitio web usa un certificado que verifica su identidad. Usar un servidor web seguro implica trabajo adicional, pero es necesario para algunas aplicaciones.

Puede saber cuándo se está comunicando usando SSL. El URL empieza con *HTTPS*, en lugar de *HTTP*.

La información sobre los servidores web seguros es específica al servidor web que usted está usando. Para averiguar más sobre cómo usar SSL, vea el sitio web del servidor web que está usando. Por ejemplo, si está usando Apache, revise dos proyectos de fuente abierta que implementan SSL para Apache en `www.modssl.org` y `www.apache-ssl.org`. También hay software comercial disponible, el cual proporciona un servidor seguro con base en el servidor web de Apache. Si está usando Microsoft Internet Information Server (IIS), busque SSL en el sitio web de Microsoft en `www.microsoft.com`.

Completar su documentación

Esto es lo último que diré aquí. Documentar su aplicación con base de datos para la Web es esencial. Se empieza con un plan que describe lo que supuestamente debe hacer la aplicación. Con base en su plan, usted crea un diseño para la base de datos. Mantenga el plan y el diseño al día. A menudo, conforme el proyecto avanza, se hacen cambios. Asegúrese de que su documentación cambie para concordar con las nuevas decisiones.

Mientras diseña sus programas, asocie las tareas en el plan de la aplicación con los programas que planea escribir. Enumere los programas y lo que hará cada uno. Si los programas son complicados, tal vez quiera incluir una breve descripción de cómo el programa realizará sus tareas. Si este es un esfuerzo de equipo, indique quién es responsable de cada programa. Al completar su aplicación, debería contar con los siguientes documentos:

- **Plan de la aplicación:** Describe lo que supuestamente hace la aplicación, e indica las tareas que llevará a cabo.
- **Diseño de la base de datos:** Describe las tablas y los campos que están en la base de datos.
- **Diseño de los programas:** Describe cómo los programas harán las tareas en el plan de la aplicación.
- **Comentarios del programa:** Describen los detalles sobre cómo funciona el programa individual

Imagine que han pasado cinco años y usted está a punto de reescribir su aplicación. ¿Qué deberá saber sobre la aplicación para poder cambiarla? Cerciórese de incluir toda la información que necesita saber en su documentación.

Capítulo 11

Construir un catálogo en línea

En este capítulo

- Diseñar un catálogo en línea
- Construir la base de datos para el Catálogo de Mascotas
- Diseñar las páginas web para el Catálogo de Mascotas
- Escribir los programas para el Catálogo de Mascotas

Los catálogos en línea están en todas partes en la Web. Todo negocio que tiene productos a la venta puede usar un catálogo en línea. Algunos negocios usan catálogos en línea para vender sus productos en línea, y algunos los usan para mostrar la calidad y el valor de sus productos al mundo. Muchos clientes esperan que los negocios estén en línea y proporcionen información sobre sus productos. Los clientes a menudo empiezan su búsqueda de un producto en línea: investigan acerca de su disponibilidad y costo por medio de la Web.

En este capítulo, aprenderá cómo construir un catálogo en línea. No escogí un catálogo de mascotas por ninguna razón en particular, excepto que sonaba más divertido que un catálogo sobre calcetines o bombillas. Y ver las fotos para un catálogo de mascotas fue mucho más divertido que ver fotos de calcetines. Introduje por primera vez el ejemplo del catálogo de mascotas en el Capítulo 3, y lo uso para muchos de los ejemplos a lo largo de este libro.

En general, todos los catálogos hacen lo mismo: brindar información sobre los productos a los clientes potenciales. El propósito general del catálogo es facilitar al máximo a los clientes la tarea de ver la información sobre los productos. Además, usted querrá que los productos se vean tan atractivos como sea posible, de modo que los clientes quieran comprarlos.

Diseñar la aplicación

El primer paso en el diseño es decidir qué debería hacer la aplicación. El propósito obvio del catálogo de mascotas es mostrar a los clientes potenciales información sobre las mascotas. Una tienda de mascotas podría también querer mostrar información sobre productos para mascotas, tales como alimentos

para mascotas, jaulas, peceras y juguetes. . . pero usted decide no incluir dichos artículos en su catálogo. El propósito de su aplicación de catálogo en línea es simplemente mostrar mascotas.

Para el cliente, desplegar la información es la única función del catálogo. Sin embargo, desde su perspectiva, usted también necesita dar mantenimiento al catálogo; es decir, necesita agregarle artículos. Por eso, debe incluir la tarea de añadir artículos al catálogo como parte de la aplicación del catálogo. Entonces, la aplicación tiene dos funciones distintas:

- Mostrar las mascotas a los clientes
- Agregar mascotas al catálogo

Mostrar mascotas a los clientes

El propósito básico de su catálogo en línea es permitir a los clientes ver las mascotas. Los clientes no pueden comprar las mascotas en línea, por supuesto. Enviar las mascotas por correo no es factible. Pero un catálogo puede exhibir las mascotas de un modo que motive a los clientes a correr a la tienda a comprarlos.

Si su catálogo sólo cuenta con tres mascotas, puede ser muy simple: una página que muestre las tres mascotas. Sin embargo, la mayoría de los catálogos tiene mucho más artículos. Generalmente, un catálogo abre con una lista de los tipos de productos (en este caso, mascotas) que hay disponibles, tales como gatos, perros, caballos y dragones. Los clientes seleccionan el tipo de mascota que deseen ver, y el catálogo muestra las mascotas individuales de ese tipo. Por ejemplo, si el cliente elige perro, entonces el catálogo mostraría pastores escoceses, perros perdigueros y lobos. Algunos tipos de productos podrían tener más niveles de categorías antes de que se vean los productos individuales. Por ejemplo, los muebles podrían tener tres niveles en lugar de dos. El nivel superior podría ser la habitación, como cocina, dormitorio, etc. El segundo nivel podría ser tipo, como sillas, mesas, etc. El tercer nivel sería el de productos individuales.

El propósito de un catálogo es motivar a aquellos que lo ven a hacer una compra inmediatamente. Para el catálogo de mascotas, las fotos son un factor crucial para motivar a los clientes a hacer una compra. Las fotos de las mascotas hacen a las personas exclamar ooooh y aaaah y decir: "¿No es hermoooosooooo?" Esto genera ventas. El propósito principal de su catálogo de mascotas es mostrar fotos de mascotas. Además, el catálogo también debería mostrar descripciones y precios.

Para mostrar las mascotas a los clientes, el Catálogo de Mascotas hará lo siguiente:

1. Mostrar una lista de los tipos de mascotas y permitir a los clientes seleccionar un tipo.
2. Mostrar información sobre las mascotas que concuerdan con el tipo seleccionado. La información incluye la descripción, el precio y una foto de la mascota.

Agregar mascotas al catálogo

Puede agregar artículos a su catálogo de varias maneras. Sin embargo, la tarea de añadir un artículo al catálogo es mucho más fácil si usa una aplicación diseñada para agregar sus productos específicos. En muchos casos, usted no es la persona que añadirá productos a su catálogo. Una razón para agregar una funcionalidad de mantenimiento a la aplicación de catálogo es para que alguien más pueda encargarse de esas aburridas tareas de mantenimiento. Cuanto más fácil sea dar mantenimiento al catálogo, menor probabilidad habrá de que contenga errores.

Una aplicación para agregar una mascota a su catálogo debe hacer lo siguiente:

1. **Pedir al usuario que escoja un tipo de mascota.** Una lista de selección de los posibles tipos de mascotas eliminaría muchos errores, tal como ortografías diferentes (perro y perros) y errores ortográficos. La aplicación también debe permitir al usuario añadir nuevas categorías cuando lo necesite.
2. **Pedir al usuario que introduzca el nombre de la mascota,** tal como perdiguero o tiburón. Una lista de selección de nombres ayudaría a prevenir errores. La aplicación también debe permitir al usuario agregar nombres nuevos cuando lo necesite.
3. **Pedir al usuario que introduzca la información sobre la mascota para la mascota nueva.** La aplicación debe especificar claramente cuál información se necesita.
4. **Almacenar la información en el catálogo.**

La aplicación de entrada del catálogo puede revisar los datos en busca de errores e introducir los datos en las ubicaciones correctas. La persona que digita una nueva mascota no necesita conocer el funcionamiento interno del catálogo.

Construir la base de datos

El catálogo mismo es una base de datos. No tiene que ser una base de datos; es posible almacenar un catálogo como un arreglode archivos HTML (Lenguaje de Marcado de Hipertexto, por sus siglas en inglés), los cuales contienen la información sobre el producto en etiquetas HTML y despliegan el archivo adecuado cuando el cliente hace clic en un vínculo. Sin embargo, a mí se me retuercen los ojos al pensar en darle mantenimiento a un catálogo de ese tipo. Imagine el tedio de agregar y remover artículos del catálogo manualmente... o encontrar la ubicación correcta para cada artículo buscando entre muchos archivos. Horrible. Por estas razones, poner su Catálogo de Mascotas en una base de datos es mejor.

La base de datos `Catalogodemascotas` contiene toda la información sobre las mascotas. Usa tres tablas:

- Tabla Mascota

- Tabla Tipodemascota
- Tabla Color

El primer paso para construir el catálogo de mascotas es construir la base de datos. Es casi imposible escribir programas sin una base de datos funcional con la cual probarlos. Primero se diseña la base de datos; luego se construye; luego se añaden los datos (o por lo menos algunos datos de muestra para usar mientras se desarrollan los programas).

Se han hecho algunos cambios al diseño de la base de datos en el Capítulo 3 para el Catálogo de mascotas. El desarrollo y las pruebas a menudo dan como resultado cambios. Tal vez usted se percate de que no tomó en cuenta algunos factores en su diseño o que ciertos elementos de su diseño no funcionan con datos del mundo real o son difíciles de programar. Es perfectamente normal que el diseño evolucione mientras usted trabaja en su aplicación. Sólo asegúrese de cambiar su documentación cuando su diseño cambie.

Construir la tabla Mascota

En su diseño del Catálogo de Mascotas, la tabla principal es la tabla Mascota, la cual contiene la información sobre las mascotas individuales que usted vende. La siguiente consulta SQL crea la tabla `Mascota`:

```
CREATE TABLE Mascota (
  IDmascota           INT(5)        NOT NULL AUTO_INCREMENT,
  Nombremascota       CHAR(25)      NOT NULL,
  Tipomascota         CHAR(15)      NOT NULL DEFAULT "Misc",
  Descripcionmascota  VARCHAR(255),
  precio              DECIMAL(9,2),
  pix                 CHAR(15)       NOT NULL DEFAULT "na.gif",
  PRIMARY KEY (IDmascota) );
```

Cada fila de la tabla Mascota representa una mascota. Las columnas son las siguientes:

- `IDmascota`: Un número secuencial para la mascota. En otro catálogo, este podría ser el número de un producto, un número de serie o un número usado para pedir el producto. La consulta `CREATE` define la columna `IDmascota` como sigue:
  - `INT(5)`: Se espera que los datos en el campo sean números enteros. La base de datos no aceptará una cadena de caracteres en este campo.
  - `PRIMARY KEY(IDmascota)`: Esta es la clave primaria, el campo que debe ser único. MySQL no permitirá dos filas con la misma `IDmascota`.
  - `NOT NULL`: Esta definición significa que este campo no puede estar vacío. Debe tener un valor. La clave primaria siempre debe establecerse en `NOT NULL`.

- AUTO-INCREMENT: Esta definición significa que el campo se rellenará automáticamente con un número secuencial si no se provee un número específico. Por ejemplo, si se añade una fila con el número 98 como IDmascota, la fila siguiente se agregará con 99 como IDmascota, a no ser que se especifique un número diferente. Esta es una manera útil de especificar una columna con un numero exclusivo, tal como un número de producto o de pedido. Usted siempre puede sobrescribir el número secuencial automático con uno propio; si usted no proporciona un número, se almacenará un número secuencial.

- Nombremascota: El nombre de la mascota, tal como león, pastor escocés o unicornio. La consulta CREATE define la columna Nombremascota así:
  - CHAR(25): Se espera que los datos en este campo sean una cadena de caracteres de 25 caracteres de longitud. El campo siempre ocupará hasta 25 caracteres de almacenamiento, con algún relleno si la cadena real almacenada tiene menos de 25 caracteres.
  - NOT NULL: Esta definición significa que este campo no puede estar vacío. Debe tener un valor. Después de todo, no tendría mucho sentido tener una mascota sin nombre en el catálogo.
  - No default value: Si usted trata de agregar una fila nueva a la tabla Mascota sin un Nombremascota, no será añadida. No tiene sentido tener un nombre predeterminado para una mascota.

- Tipomascota: El tipo de mascota, tal como perro o pez. La consulta CREATE define la columna Tipomascota así:
  - CHAR(15): Se espera que los datos en este campo sean una cadena de caracteres de 15 caracteres de longitud. El campo siempre guardará 15 caracteres, con algún relleno si la cadena real almacenada tiene menos de 15 caracteres.
  - NOT NULL: Esta definición significa que este campo no puede estar vacío. Debe tener un valor. La aplicación del catálogo en línea mostrará las categorías primero y luego las mascotas dentro de una categoría, de modo que una mascota sin categoría nunca se mostraría en la página web.
  - DEFAULT "Misc": El valor "Misc" se almacena si usted no proporciona un valor para Tipomascota. Esto garantiza que siempre se almacene un valor para Tipomascota.

- Descripcionmascota: Una descripción de la mascota. La consulta CREATE define la Descripcionmascota, como sigue:
  - VARCHAR(255): Este tipo de datos define el campo como una cadena de caracteres variable que puede constar de hasta 255 caracteres de longitud. El campo se almacena en su longitud real.

- precio: El precio de la mascota. La consulta CREATE define el precio de la siguiente manera:
  - DECIMAL(9,2): Este tipo de datos define el campo como un número con decimales que puede tener hasta nueve dígitos y dos es-

pacios decimales. Si usted almacena un número entero en este campo, aparecerá con dos decimales, tal como 9.00 o 2568.00.

- `pix`: El nombre de archivo de la foto de la mascota. Las fotos en un sitio web se almacenan en archivos gráficos con nombres como `perro.jpg`, `dragon.gif` o `gato.png`. Este campo almacena el nombre de archivo para la foto que usted desea mostrar para esta mascota. La consulta `CREATE` define `pix` como sigue:
  - `CHAR(15)`: Se espera que los datos en este campo sean una cadena de caracteres de 15 caracteres de longitud. Para algunas aplicaciones, los archivos de fotos podrían estar en otros directorios o en otro sitio web que requieren de un campo más largo, pero para esta aplicación, todas las fotos están en un directorio en el sitio web y tienen nombres cortos. El campo siempre ocupará 15 caracteres de almacenamiento, con relleno si la cadena real almacenada tiene menos de 15 caracteres.
  - `NOT NULL`: Esta definición significa que este campo no puede estar vacío. Debe tener un valor. Se necesita una fotografía para la mascota. Cuando un sitio web trata de mostrar una foto que no se puede encontrar, despliega un feo mensaje de error en la ventana del explorador, donde iría la imagen. Usted no querrá que su catálogo haga eso, por lo que su base de datos debería exigir un valor. En este caso, usted define un valor predeterminado para que siempre se coloque un valor en este campo.
  - `DEFAULT "na.gif"`: El valor `"na.gif"` se almacena si usted no proporciona un valor para `pix`. Así, siempre un valor se almacena en pix. El archivo `na.gif` podría ser una imagen que diga algo así: `"foto no disponible"`.

Observe los siguientes puntos sobre el diseño de esta tabla de base de datos:

- **Algunos campos son CHAR y otros son VARCHAR.** Los campos `CHAR` son más rápidos, mientras que los campos `VARCHAR` son más eficientes. Su decisión dependerá de lo que sea más importante para su aplicación en su ambiente: espacio en disco o velocidad.

  En general, los campos más cortos deberían ser `CHAR` porque los campos más cortos no desperdician mucho espacio. Por ejemplo, si su `CHAR` es de 5 caracteres, la mayor cantidad de espacio que podría desperdiciar es 4. Sin embargo, si su `CHAR` es 200, podría desperdiciar 199. Por lo tanto, para campos cortos, use `CHAR` para velocidad y muy poco desperdicio de espacio.

- **El campo IDmascota significa cosas diferentes para diferentes mascotas.** El campo `IDmascota` asigna un número exclusivo a cada mascota. Sin embargo, un número exclusivo no es necesariamente significativo en todos los casos. Por ejemplo, resulta significativo para un gatito individual, pero no para un pez dorado individual.

  En realidad, existen dos tipos de mascotas. Uno es la mascota única, tal como un perrito o un gatito. Después de todo, el cliente compra un perro

específico, no sólo un perro genérico. El cliente necesita ver la foto del animal. Por otro lado, algunas mascotas no son especialmente únicas, tales como los peces dorados o un periquito. Cuando los clientes compran un pez dorado, ven una pecera llena de peces y señalan uno. La única característica que realmente distingue a un pez dorado es su color. El cliente no necesita ver una foto del pez verdadero en su catálogo, sólo una foto de un pez dorado genérico, tal vez una que muestre los colores posibles.

En su catálogo, usted tiene ambos tipos de mascotas. El catálogo podría contener varias mascotas con el nombre gato pero con diferentes IDmascota. La foto mostraría la mascota individual. El catálogo también contiene mascotas que no son individuales, sino que representan mascotas genéricas, como los peces dorados. En este caso, sólo hay una entrada con el nombre pezdorado, con un único `IDmascota`.

He usado ambos tipos de mascotas en este catálogo para demostrar los diferentes tipos de productos que usted podría querer incluir en un catálogo. El catálogo de elementos únicos podría incluir productos tales como obras de arte o placas únicas de colección. Cuando el elemento único se vende, se elimina del catálogo. La mayoría de los productos son más genéricos, tales como ropa o automóviles. Aunque una foto muestre una camisa en particular, hay muchas camisas idénticas disponibles. Usted puede vender la camisa muchas veces sin necesidad de retirarla del catálogo.

Construir la tabla Tipodemascota

Usted asigna a cada mascota un tipo, como perro o dragón. La primera página web del catálogo enumera los tipos entre los cuales el cliente selecciona. También resultaría útil mostrar una descripción de cada tipo. Usted no querrá poner la descripción del tipo en la tabla principal `Mascota`, pues la descripción sería la misma para todas las mascotas en la misma categoría. Repetir información en una tabla va contra el buen diseño de una base de datos.

La base de datos `Catalogodemascotas` incluye una tabla llamada `Tipomascota` que guarda las descripciones del tipo. La siguiente consulta SQL crea la tabla `Tipomascota`:

```
CREATE TABLE Tipodemascota (
 Tipomascota          CHAR(15)      NOT NULL,
  Descripciontipo VARCHAR(255),
  PRIMARY KEY(Tipomascota)  );
```

Cada fila de esta tabla representa un tipo de mascota. Estas son las columnas:

- `Tipomascota`: El nombre del tipo. Observe que la columna `Tipomascota` se define igual en la tabla `Mascota` (la cual describo en la sección anterior) y en esta tabla. Eso hace posible unir las tablas y facilita la tarea de hacer coincidir filas en las tablas. Sin embargo, `Tipomascota` es la clave primaria en esta tabla pero no en la tabla `Mascota`. La consulta `CREATE` define la columna `Tipomascota` como sigue:

- CHAR(15): Se espera que los datos en este campo sean una cadena de caracteres de 15 caracteres de longitud.
- PRIMARY KEY(Tipomascota): Esta definición establece la columna Tipomascota como la clave primaria. Este es el campo que debe ser único. MySQL no permitirá que se introduzcan dos filas con el mismo Tipomascota.
- NOT NULL: Esta definición significa que este campo no puede estar vacío. Debe tener un valor. La clave primaria siempre debe establecerse en NOT NULL.

Descripciontipo: Una descripción del tipo de mascota. La consulta CREATE define Descripciontipo del siguiente modo:

- VARCHAR(255): Se espera que la cadena en este campo sea una cadena de caracteres variable que pueda constar de hasta 255 caracteres de longitud. El campo se almacena en su longitud real.

Construir la tabla Color

Al explicar la construcción de la tabla Mascota (consulte: "Construir la tabla Mascota", anteriormente en este capítulo), yo me refiero a los diferentes tipos de mascotas: mascotas que son únicas (como perritos y potrillos) y mascotas que no son únicas (como peces dorados y tortugas). Para las mascotas únicas, el cliente necesita ver una foto de la mascota. Para las mascotas que no son únicas, el cliente sólo necesita ver una foto genérica.

En algunos casos, las mascotas genéricas vienen en una variedad de colores, tales como periquitos azules y periquitos verdes. Tal vez usted quiera mostrar dos fotos para los periquitos: una de un periquito azul y una de uno verde. Sin embargo, como la mayoría de las mascotas no son del tipo de mascota genérica, usted no querrá agregar una columna de color a su tabla principal Mascota, pues quedaría en blanco en la mayor parte de las filas. En cambio, usted crea una tabla separada que contiene sólo mascotas que vienen en más de un color. Luego, cuando la aplicación del catálogo muestra las mascotas, puede revisar la tabla Color para ver si hay más de un color disponible; y, si es así, puede mostrar las fotos de la tabla Color.

La tabla Color apunta a fotos de mascotas cuando las mascotas vienen en diferentes colores, de modo que el catálogo pueda mostrar fotos de todos los colores disponibles. La siguiente consulta SQL crea la tabla Color:

```
CREATE TABLE Color (
  Nombremascota       CHAR(25)       NOT NULL,
  Colormascota        CHAR(15)       NOT NULL,
  pix                 CHAR(15)      NOT NULL DEFAULT "na.gif",
  PRIMARY KEY(Nombremascota,Colormascota)  );
```

Cada fila representa un tipo de mascota. Las columnas son como sigue:

- `Nombremascota`: El nombre de la mascota, tal como león, pastor escocés o dragón barbudo de China. Observe que la columna Nombremascota se define igual en la tabla `Mascota` y en esta. Esto hace posible la unión de las tablas y facilita la tarea de hacer coincidir las filas en las tablas. Sin embargo, `Nombremascota` es la clave primaria en esta tabla pero no en la tabla `Mascota`. La consulta `CREATE` define `Nombremascota` de la siguiente manera:
  - `CHAR(25)`: Se espera que los datos en este campo sean una cadena de caracteres de 25 caracteres de longitud.
  - `PRIMARY KEY(Nombremascota,Colormascota)`: La clave primaria debe ser única. Para esta tabla, dos columnas juntas son la clave primaria: esta columna y la columna `Colormascota`. MySQL no permitirá que se digiten dos filas con la misma información de `Nombremascota y Colormascota`.
  - `NOT NULL`: Esta definición significa que este campo no puede estar vacío. Debe tener un valor. La clave primaria siempre debe definirse como `NOT NULL`.
- `Colormascota`: El color de la mascota, tal como anaranjado o morado. La consulta `CREATE` define `Colormascota` de la manera siguiente:
  - `CHAR(15)`: Este tipo de datos define el campo como una cadena de caracteres de 15 caracteres de longitud.
  - `PRIMARY KEY(Nombremascota,Colormascota)`: La clave primaria debe ser única. Para esta tabla, dos columnas juntas son la clave primaria: esta columna y la columna `Nombremascota`. MySQL no permitirá que dos filas tengan la misma información de `Nombremascota y Colormascota`.
  - `NOT NULL`: Esta definición significa que este campo no puede estar vacío. Debe tener un valor. La clave primaria siempre debe definirse como `NOT NULL`.
- `pix`: El nombre del archivo que contiene la foto de la mascota. La consulta `CREATE` define pix como sigue:
  - `CHAR(15)`: Este tipo de datos define el campo como una cadena de caracteres con 15 caracteres de longitud.
  - `NOT NULL`: Esta definición significa que este campo no puede estar vacío. Debe tener un valor. Usted necesita una foto para la mascota. Cuando un sitio web trata de mostrar una foto que no se puede encontrar, despliega un feo mensaje de error en la ventana del explorador, donde iría la imagen. Usted no querrá que su catálogo haga eso, de modo que su base de datos debería exigir un valor. En este caso, la consulta `CREATE` define un valor predeterminado, para que se coloque siempre un valor en este campo.
  - `DEFAULT "na.gif"`: El valor `"na.gif"` se almacena si usted no brinda un valor para pix. De esta manera, siempre se almacena un valor para pix. El archivo `na.gif` podría contener una imagen que diga algo así `foto no disponible`.

Agregar datos a la base de datos

Puede agregar los datos a la base de datos de muchas maneras. Puede usar consultas SQL para añadir mascotas a la base de datos, o puede usar la aplicación que describo en este capítulo. Mi preferencia personal durante el desarrollo es agregar unos cuantos artículos de muestra al catálogo leyendo los datos de un archivo. Luego, cada vez que mis datos se vuelven totalmente extraños durante el desarrollo (como resultado de errores de programación o mi raro sentido del humor), puedo recrear la tabla de datos en un momento. Sólo haga DROP a la tabla, recréela con una consulta SQL y vuelva a leer los datos de muestra.

Por ejemplo, el archivo de datos para la tabla Mascota podría verse así:

```
<TAB>Pequines<TAB>Perro<TAB>Pequeño, lindo, energico. Buen
        sistema de alarma.<TAB>100.00<TAB>peke.jpg
<TAB>Gato casero<TAB>Gato<TAB>Gato amarillo y blanco. Extre-
        madamente jugueton. <TAB>20.00<TAB>catyellow.jpg
<TAB>Gato casero<TAB>Gato<TAB>Gato negro. Refinado, brillante.
        Le gustan los niños. <TAB>20.00<TAB>catblack.jpg
<TAB>Dragon barbado de China<TAB>Lagartija<TAB>Crece hasta
        los 2 pies de largo. Fascinante de mirar. Le gusta
        que lo abracen.<TAB>100.00<TAB>lizard.jpg
<TAB>Labrador<TAB>Perro<TAB>Perro negro. Cobrador grande e
        inteligente. A menudo seleccionado como perro guia
        por los ciegos.<TAB>100.00<TAB>lab.jpg
<TAB>Pez dorado<TAB>Pez<TAB>Variedad de colores. Barato. Fa-
        cil de cuidar. Bueno como primera mascota para ni-
        ños pequeños.<TAB>2.00<TAB>goldfish.jpg
<TAB>Tiburon<TAB>Pez<TAB>Elegante. Poderoso. Tratese con cui-
        dado.<TAB>200.00<TAB>shark.jpg
<TAB>Dragon asiatico<TAB>Dragon<TAB>Largo y serpentino. A me-
        nudo dorado o rojo.<TAB>10000.00<TAB>dragona.jpg
<TAB>Unicornio<TAB>Caballo<TAB>Hermoso corcel blanco con cuerno
        en espiral en la frente.<TAB>20000.00<TAB> unicorn.jpg
```

Estas son las reglas para los archivos de datos:

- Las etiquetas <TAB> representan tabuladores reales, del tipo que usted crea oprimiendo la tecla Tab.
- Cada línea representa una mascota y debe introducirse sin oprimir la tecla Enter o Return. Las líneas en el ejemplo anterior se muestran acomodadas en más de una línea, para que usted pueda ver la línea completa. Sin embargo, en el archivo real, las líneas de datos están acomodadas una por renglón.
- Aparece un tabulador al inicio de cada línea porque el primer campo no se introduce. El primer campo es IDmascota, el cual se introduce automáticamente; usted no necesita digitarlo. No obstante, sí necesita usar un tabulador para que MySQL sepa que hay un campo vacío al inicio.

Después, puede usar una consulta SQL para incluir el archivo de datos en la tabla Mascota:

```
LOAD DATA LOCAL INFILE "mascotas" INTO TABLE Mascota;
```

Cada vez que la tabla de datos se ponga rara, puede recrearla y leer los datos nuevamente.

La consulta `LOAD DATA LOCAL` podría no estar disponible en su versión de MySQL. Esta consulta debe activarse antes de poder usarla. Si no está activada, verá un error que dice `The used command is not allowed with this MySQL version`. Comento `LOAD DATA LOCAL` en detalle en el Capítulo 4.

Diseñar la apariencia

Una vez que usted sepa lo que hará la aplicación y cuál información contiene la base de datos, puede diseñar la apariencia y el aspecto de la aplicación. Esto incluye lo que el usuario ve y cómo el usuario interactúa con la aplicación. Su diseño debería ser atractivo y fácil de usar. Puede planear este diseño en papel, indicando lo que el usuario ve, tal vez con bocetos o descripciones escritas. En su diseño, incluya los componentes de interacción del usuario, como botones o vínculos, y describa sus acciones. Debería incluir cada página de la aplicación en el diseño. Si tiene suerte, contará con un diseñador gráfico que podrá desarrollar hermosas páginas web para usted. Si usted es como yo, hará lo mejor que pueda con una cantidad limitad de conocimiento sobre gráficos.

El Catálogo de Mascotas tiene dos versiones: una para el catálogo que el cliente ve y otra, menos sofisticada, para la parte de la aplicación que usa usted o quien añada las mascotas al catálogo.

Mostrar mascotas a los clientes

La aplicación incluye tres páginas que los clientes ven:

- **La página inicial de la tienda:** Esta es la primera página que ven los clientes. Indica el nombre del negocio y el propósito del sitio web.
- **La página con los tipos de mascotas:** Esta página incluye todos los tipos de mascotas y permite a los clientes seleccionar cuál tipo de mascota desean ver.
- **La página de mascotas:** Esta página muestra todas las mascotas del tipo seleccionado.

Página inicial de la tienda

La página inicial de la tienda es la página introductoria a la Tienda de Mascotas. Como la mayoría de las personas ya sabe lo que es una tienda de mascotas, esta página no necesita brindar muchas explicaciones. La Figura 11-1 muestra la página de inicio de la tienda. La única acción disponible para el cliente en esta página es un vínculo en el cual el cliente puede hacer clic para ver el Catálogo de Mascotas.

Página del tipo de mascotas

La página del tipo de mascotas enumera todos los tipos de mascotas en el catálogo. Cada tipo de mascota aparece con su descripción. La Figura 11-2 muestra la página con el tipo de mascotas. Aparecen botones de opción junto a cada tipo de mascota para que los clientes puedan seleccionar el tipo de mascota que desean ver.

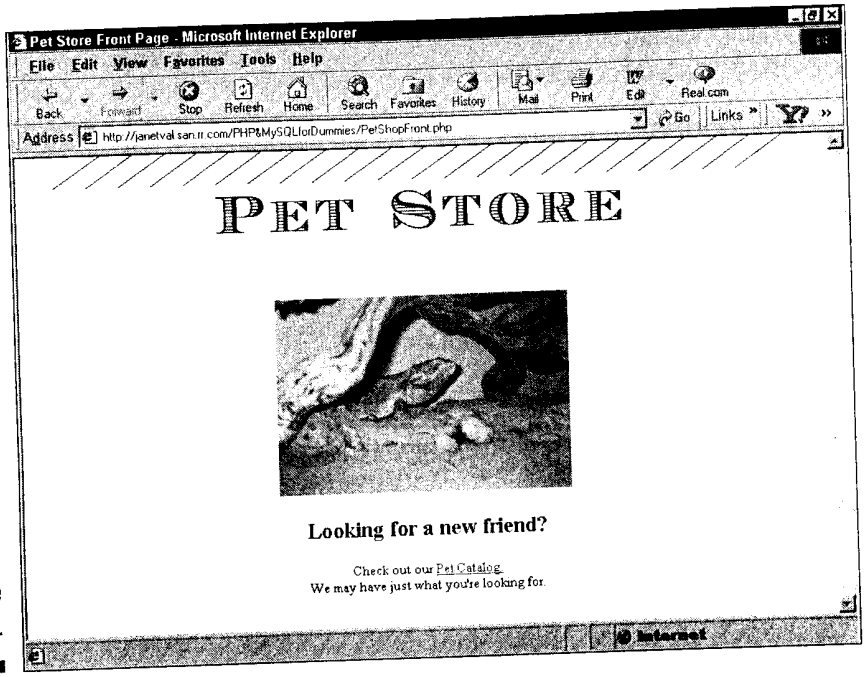


Figura 11-1: La página de apertura del sitio web de la Tienda de Mascotas.

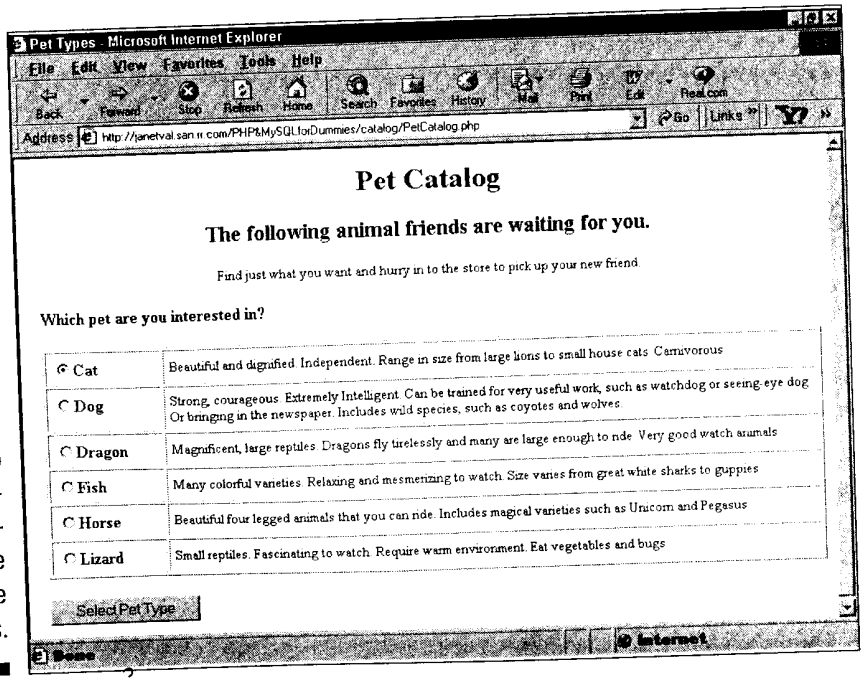


Figura 11-2: La página con el tipo de mascotas en el sitio web de la Tienda de Mascotas.

Página de mascotas

La página con las mascotas incluye todas las mascotas del tipo seleccionado. Cada mascota aparee con su `IDmascota`, descripción, precio y foto. La página de mascotas aparece en un formato diferente, dependiendo de la información en la base de datos del catálogo. Las Figuras 11-3, 11-4 y 11-5 mues'ran posibles páginas de mascotas. La Figura 11-3 muestra una página que incluye tres tipos diferentes de perros del catálogo.

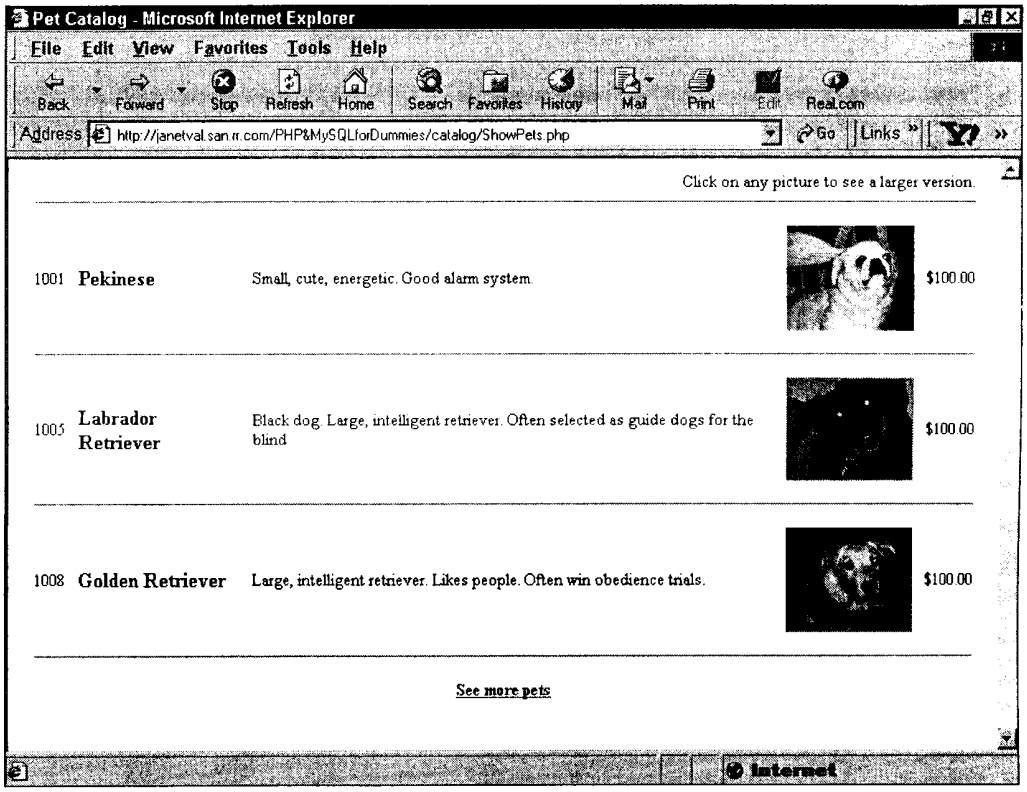


Figura 11-3: Esta página de mascotas muestra tres perros distintos.

La Figura 11-4 muestra que más de una mascota puede tener el mismo nombre de mascota. Observe que los gatos caseros tienen números diferentes de `IDmascota`.

La Figura 11-5 muestra el output cuando las mascotas se encuentran en la tabla `Color`, la cual muestra que hay más de un color disponible.

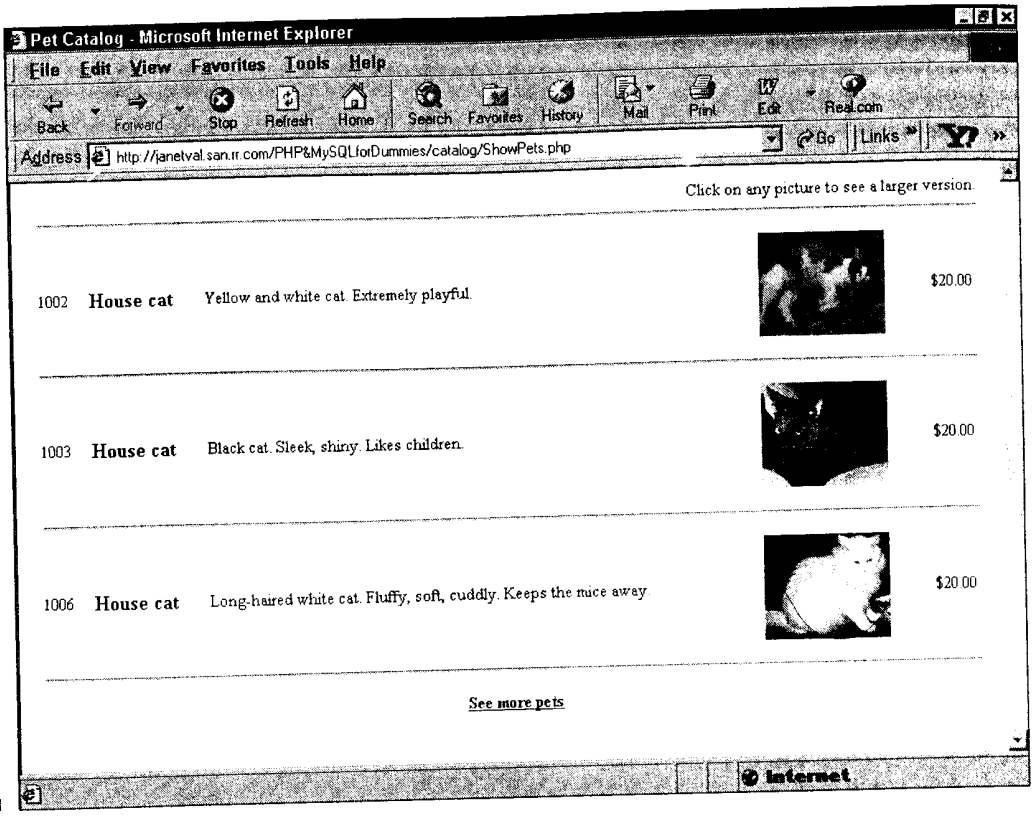


Figura 11-4: Esta página de mascotas muestra tres gatos con el mismo nombre de mascota.

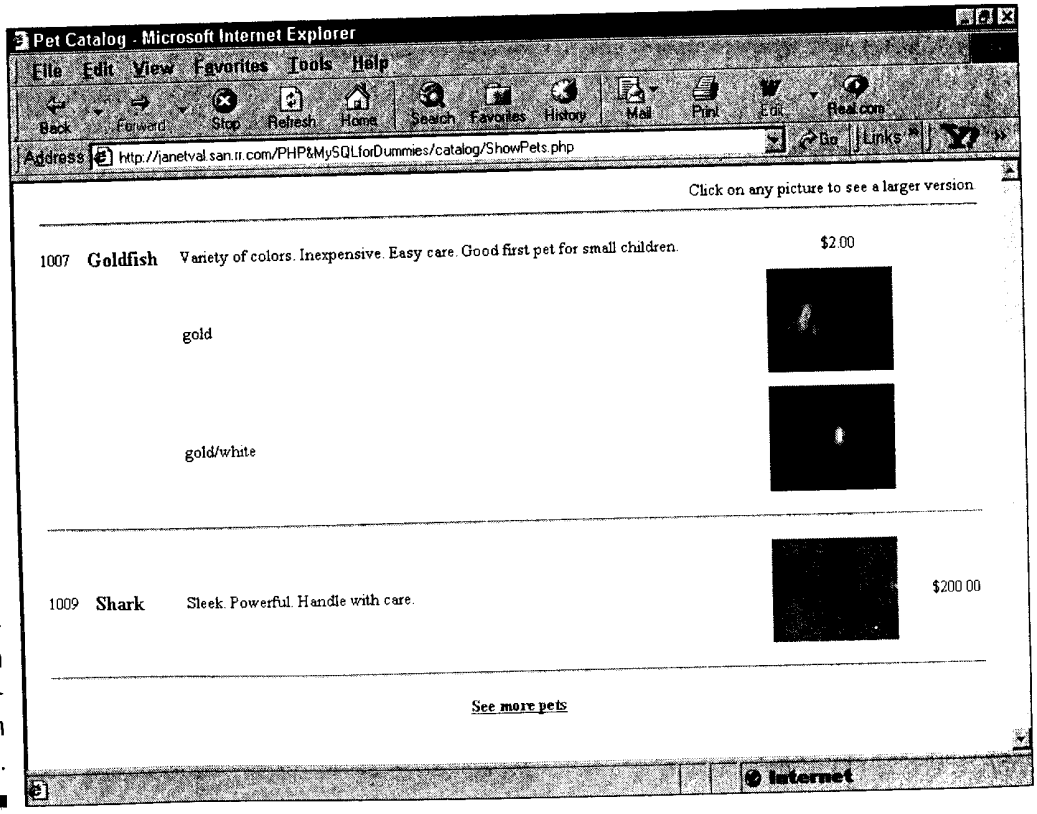


Figura 11-5: Esta página de mascotas muestra peces disponibles en dos colores.

En todas estas páginas, hay una línea en la parte superior que dice `Haga clic en cualquier foto para ver una version mas grande`. Si el cliente hace clic en la foto, aparecerá una versión más grande de la foto.

Agregar mascotas al catálogo

La aplicación incluye tres páginas que los clientes no ven; estas son las páginas que se usan para agregar mascotas al Catálogo de Mascotas. Las tres páginas funcionan en orden secuencial para añadir una sola mascota:

1. **Página para obtener el tipo de mascota.** La persona que agrega una mascota al catálogo selecciona el botón de opción para el tipo de mascota. El usuario también puede digitar un tipo nuevo de mascota.
2. **Página para obtener la información de la mascota.** El usuario selecciona el botón de opción para la mascota que se está añadiendo y rellena la descripción, el precio y el nombre de archivo de la foto. El usuario también puede digitar un nuevo nombre de mascota.
3. **Página de retroalimentación.** Se despliega una página con la información de la mascota que se añadió al catálogo.

Página para obtener el tipo de mascota

La primera página obtiene el tipo de mascota para la mascota que necesita ser agregada al catálogo. La Figura 11-6 muestra la página para obtener el tipo de mascota. Observe que todos los tipos de mascotas incluidos actualmente en el catálogo se enumeran, y se proporciona una sección donde el usuario puede introducir un nuevo tipo de mascota si es necesario.

Página para obtener la información de las mascotas

La Figura 11-7 muestra la segunda página. Esta página permite al usuario digitar la información sobre la mascota que va en el catálogo. Esta página enumera todos los nombres de mascotas en el catálogo para el tipo de mascota seleccionado, de modo que el usuario pueda seleccionar una. También brinda una sección donde el usuario puede digitar un nuevo nombre de mascota, si es necesario.

Página de retroalimentación

Cuando el usuario envía la información sobre la mascota, la información se añade a la base de datos `Catalogodemascotas`. La Figura 11-8 muestra una página que verifica la información añadida a la base de datos. El usuario puede hacer clic en un vínculo para regresar a la primera página y añadir otra mascota.

Página para obtener información faltante

La aplicación revisa los datos para cerciorarse de que el usuario haya digitado la información requerida y le pide al usuario cualquier información que no se haya digitado. Por ejemplo, si el usuario selecciona Categoría Nueva en la primera página, debe digitar el nombre y la descripción de una categoría. Si el usuario no digita el nombre o la descripción, se despliega una página que indica el problema y solicita la información. La Figura 11-9 muestra la página que los usuarios ven si se olvidan de digitar el nombre y la descripción de la categoría.

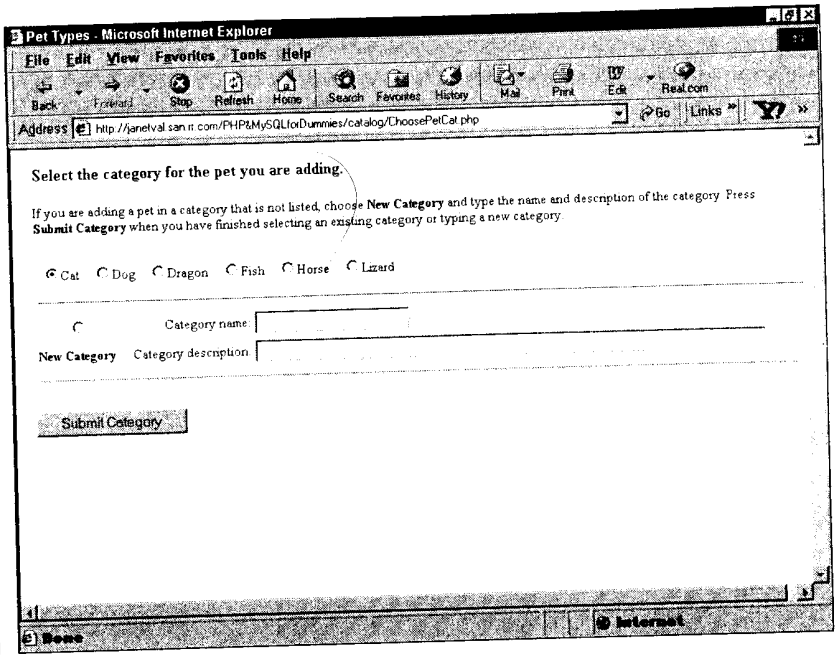


Figura 11-6: La primera página para agregar una mascota al catálogo.

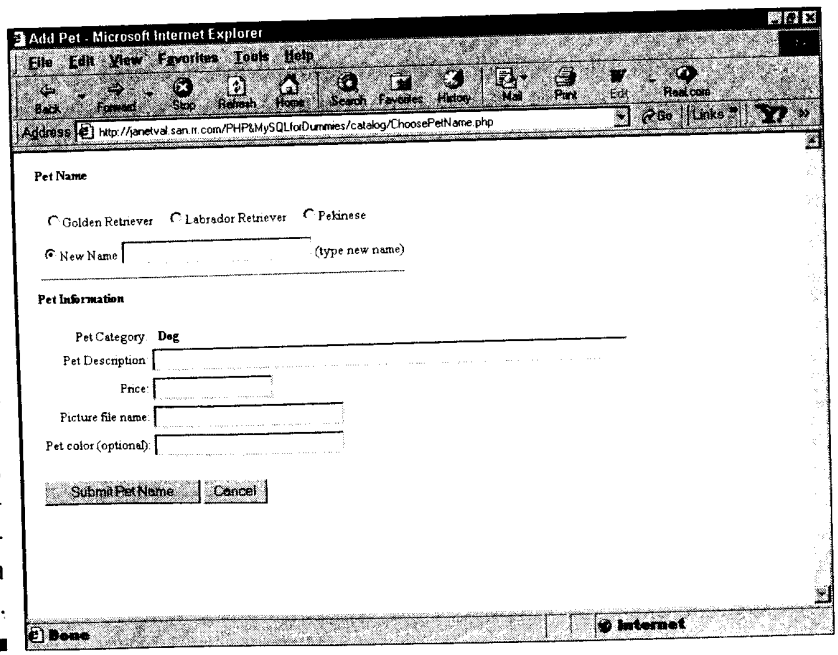


Figura 11-7: La segunda página solicita el nombre de la mascota.

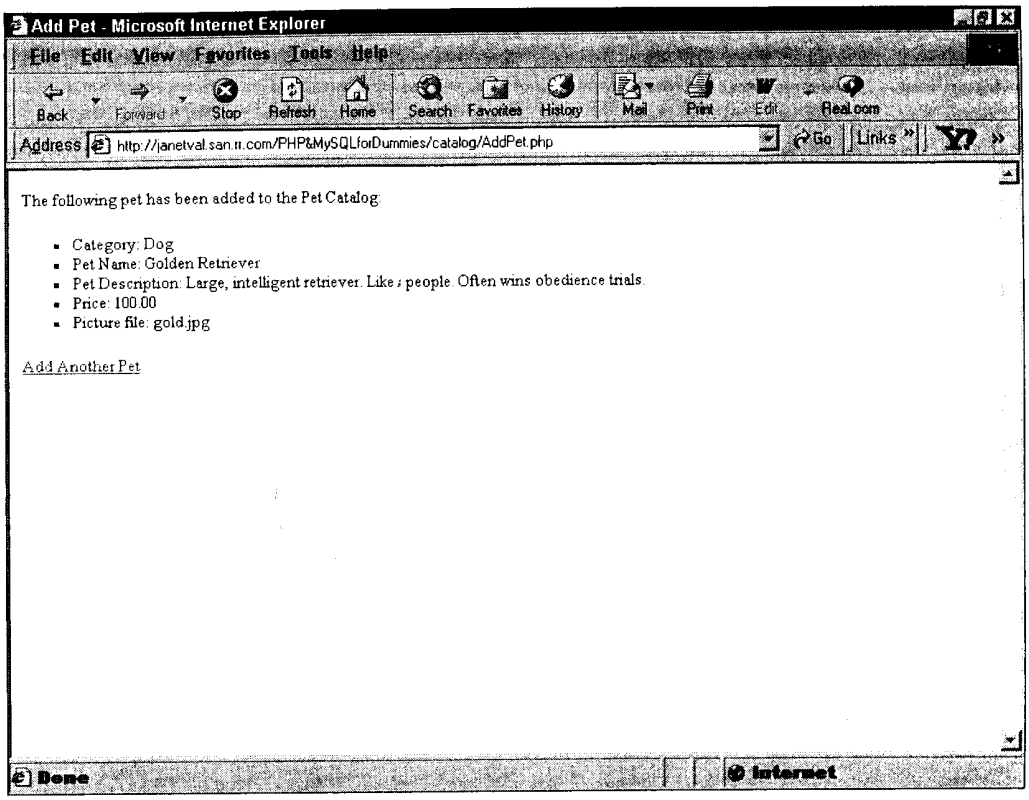


Figura 11-8: La última página brinda retroalimentación.

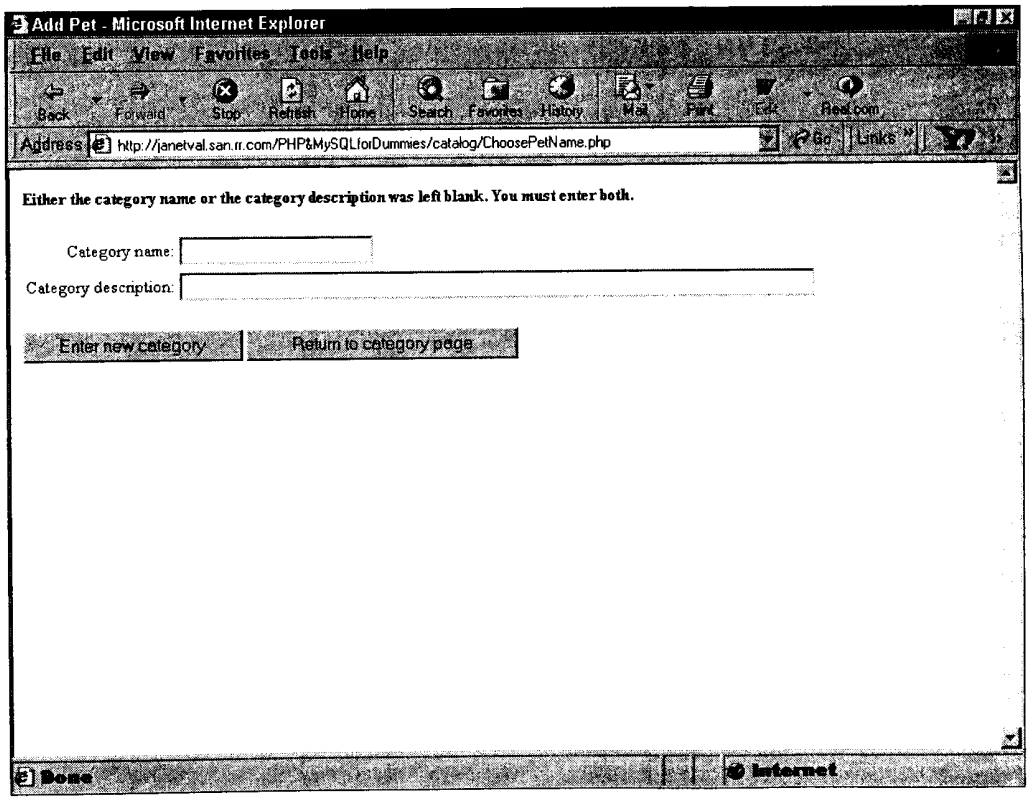


Figura 11-9: Esta página solicita una nueva categoría y su descripción.

Escribir los programas

Una vez que usted sepa cómo lucirán las páginas y qué harán, puede escribir los programas. En general, se escribe un programa para cada página, aunque a veces tiene más sentido separar los programas en más de un archivo o combinar programas en una página. (Para conocer los detalles sobre cómo organizar las aplicaciones, consulte el Capítulo 10.)

Como le explico en el Capítulo 10, conserve la información necesaria para conectarse a la base de datos en un archivo separado e incluya ese archivo en todos los programas que necesiten tener acceso a la base de datos. El archivo debería almacenarse en una ubicación segura y bajo un nombre engañoso por razones de seguridad. Para esta aplicación, la siguiente información se almacena en un archivo llamado `misc.inc`:

```
<?php
  $usuario="catalogo";
  $huesped="localhost";
  $clave="";
  $basededatos="Catalogodemascotas";
?>
```

La aplicación del Catálogo de Mascotas tiene dos conjuntos independientes de programas: un conjunto para mostrar el catálogo de mascotas a los clientes y un conjunto para insertar mascotas nuevas en el catálogo.

Mostrar mascotas a los clientes

La aplicación que muestra el Catálogo de Mascotas a los clientes tiene tres tareas básicas:

- Mostrar la página inicial de la tienda. Proporcionar un vínculo al catálogo.
- Mostrar una página donde los usuarios seleccionan el tipo de mascota.
- Mostrar una página con mascotas del tipo de mascota seleccionado.

Mostrar la página inicial de la tienda

La página inicial de la tienda no necesita ningún enunciado PHP. Simplemente despliega una página web con un vínculo. Los enunciados HTML son suficientes para hacerlo. La Lista 11-1 muestra el archivo HTML que describe la página inicial de la tienda.

Lista 11-1: Archivo HTML para la página inicial de la tienda

```
<?php
  /* Programa: tiendaMascotas.php
   * Descripcion: Despliega la pagina de apertura para la
          Tienda de Mascotas.
   */
?>
<html>
<head><title>Pagina inicial de la Tienda de
          Mascotas</title></head>
<body topmargin="0" leftmargin="0" marginheight="0"
      marginwidth="0">
<table width="100%" height="100%" border="0"
       cellspacing="0" cellpadding="0">
  <tr>
    <td align="center" valign="top" height="30">
      <img src="images/awning-top.gif" alt="awning">
    </td>
  </tr>
  <tr>
    <td align="center" valign="top">
      <img src="images/Name.gif" alt="Tienda de Mascotas">
      <p style="margin-top: 40pt">
      <img src="images/lizard-front.jpg" alt="foto animal"
        height="186" width="280">
      <p><h2>¿Busca un amigo?</h2>
      <p>Consulte nuestro <a href="catalogo.php">Catalogo de
          mascotas.</a>
        <br> Seguramente tenemos justo lo que usted necesita.
    </td>
  </tr>
</table>
</body></html>
```

Observe que el vínculo es un programa PHP llamado `Catalogo.php`. Cuando el cliente hace clic en el vínculo, el programa del Catálogo de Mascotas (`Catalogo.php`) se inicia.

Mostrar tipos de mascotas

La página del tipo de mascota (consulte la Figura 11-2) muestra al cliente una lista de todos los tipos de mascotas actualmente disponibles en el catálogo. La Lista 11-2 muestra el programa que produce la página web del tipo de mascotas.

Lista 11-2: Programa que despliega los tipos de mascotas

```
<?php
  /* Programa: Catalogo.php
   * Descripcion: Despliega una lista de categorias de mascotas
                  de la tabla Tipomascota. Incluye las descripciones.
                  El usuario marca un boton de opcion.
   */
?>
<html>
<head><title>Tipos de mascotas</title></head>
<body>
<?php
  include("misc.inc");                                              #11

  $conexion = mysql_connect($huesped,$usuario,$clave)               #13
       or die ("No se pudo conectar al servidor ");
  $db = mysql_select_db($basededatos,$conexion)                     #15
       or die ("No se pudo seleccionar la base de datos ");

  /* Selecciona todas las categorias de la tabla  Tipomascota */
  $consulta = "SELECT * FROM Tipomascota ORDER BY Tipomascota";
             #19
  $resultado = mysql_query($consulta)
       or die ("No se pudo ejecutar la consulta.");
            #21

  /* Despliegue el texto antes del formulario */
  echo "<div style='margin-left: .1in'>
  <h1 align='center'>Catalogo de mascotas</h1>
  <h2 align='center'>Los siguientes animales amigos le
            esperan.</h2>
   <p align='center'>Encuentre justo lo que desea y apresurese a
            venir a la tienda a recoger a su nuevo amigo.
  <p><h3>¿En cual mascota esta interesado?</h3>\n";

  /* Crear formulario que contenga una lista de seleccion */
  echo "<form action='mostrar.php' method='post'>\n";     #34
  echo "<table cellpadding='5' border='1'>";
  $contador=1;                                                      #36
  while ($fila = mysql_fetch_array($resultado))                     #37
  {
     extract($fila);                                                #39
     echo "<tr><td valign='top' width='15%'>\n";
     echo "<input type='radio' nombre='interes'
                  valor='$Tipomascota'\n";                          #42
     if ( $contador == 1 )                                          #43
     {
         echo "revisado";
     }
     echo "><font size='+1'><b>$Tipomascota</b></font>";  #47
     echo "</td>
           <td>$Descripciontipo</td>";                              #49
     echo "</tr>";
     $contador++;                                                   #51
  }
  echo "</table>";
  echo "<p><input type='submit' valor='Elija un tipo de mascota'>
        </form>\n";
?>                                                                  #55
</div>
</body></html>
```

El programa en la Lista 11-2 tiene números al final de algunas líneas. Estos números son una referencia para que yo me pueda referir a partes particulares del programa. Los números en la siguiente lista corresponden a los números en las líneas del listado anterior. He aquí una breve explicación de lo que estas líneas hacen en el programa:

11 El enunciado `include` extrae un archivo que contiene la información necesaria para conectarse a la base de datos. Lo llamo `misc.inc` porque me parece más seguro que `claves.inc`.

13 Se conecta al servidor MySQL.

15 Selecciona la base de datos `Catalogodemascotas`.

19 Una consulta que selecciona toda la información de la tabla `Tipomascota` y la pone en orden alfabético con base en el tipo de mascota.

21 Ejecuta la consulta en la línea 19.

34 Etiqueta de apertura para el formulario que guardará todos los tipos de mascotas. La meta de la acción es `muestraMascotas.php`: el programa que muestra las mascotas del tipo seleccionado.

36 Crea un contador con un valor inicial 1. El contador llevará un registro de cuántos tipos de mascotas se encuentran en la base de datos.

37 Empieza un ciclo `while`, el cual obtendrá las filas que contienen los tipos y la descripción de las mascotas que fueron seleccionadas de la base de datos en las líneas 19 y 20. El ciclo se ejecutará una vez para cada tipo de mascota que fue recuperado.

39 Separa la fila en dos variables: `$Tipomascota` y `$Descripcionmascota`.

41/42 Hace echo a una etiqueta del campo del formulario para un botón de opción. El valor es el valor en `$Tipomascota`. Este enunciado se ejecuta una vez en cada ciclo, y crea un botón de opción para cada tipo de mascota. Este enunciado `hace echo` sólo a parte de la etiqueta del campo del formulario.

43 Empieza un bloque if que se ejecuta sólo en el primer ciclo. `Hace echo` a la palabra `"revisado"` como parte del campo del formulario. Esto garantiza que uno de los botones de opción ha sido seleccionado en el formulario, de modo que el formulario no pueda enviarse sin haberse seleccionado un botón, lo cual daría como resultado desagradables mensajes de error o advertencias. El contador se estableció exclusivamente para este propósito.

Aunque agregar `"revisado"` a cada botón de opción funciona en algunos buscadores, confunde a otros. Sin embargo, la programación adicional requerida para agregar `"revisado"` a sólo un botón de opción puede evitar problemas potenciales.

47 Hace echo a la parte restante de la etiqueta del campo del formulario para el botón de opción: la parte que cierra la etiqueta.

49 Hace `echo` a la descripción de la mascota en una segunda celda en la fila de la tabla.

51 Suma 1 al contador para llevar un registro del número de veces que se ha ejecutado el ciclo.

54 Añade el botón de envío al formulario.

55 Cierra el formulario.

Cuando el usuario selecciona un botón de opción y luego hace clic en el botón de envío, el próximo programa (llamado `mostrar.php` en la etiqueta del formulario) corre, y muestra las mascotas para el tipo de mascota seleccionado.

Mostrar las mascotas

La página de mascotas (consulte las Figuras 11-3, 11-4 y 11-5) muestra al cliente una lista de todas las mascotas del tipo seleccionado actualmente disponibles en el catálogo. La Lista 11-3 muestra el programa que produce la página web de las macotas.

Lista 11-3: Programa que muestra una lista de mascotas

```
<?php
  /* Programa: mostrar.php
   * Descripcion: Muestra todas las mascotas en una
            categoria. La categoria se pasa en una variable
            desde un formulario. La informacion para cada
            mascota se despliega en una sola linea, a menos
            que la mascota venga en mas de un color. Si la
            mascota viene en colores, se muestra una sola
            linea sin una foto y una linea para cada color,
            con fotos, aparece despues de la linea. Se
            despliegan fotos chicas, las cuales son vinculos a
            fotos mas grandes.
   */
?>
<html>
<head><title>Catalogo de mascotas</title></head>
<body topmargin="0" marginheight="0">
<?php
  include("misc.inc");

  $conexion = mysql_connect($huesped,$usuario,$clave)
       or die ("No se pudo conectar al servidor");
  $db = mysql_select_db($basededatos,$conexion)
       or die ("No se pudo seleccionar la base de datos");

  /* Selecciona las mascotas de un tipo dado */
  $consulta = "SELECT * FROM Mascota
               WHERE Tipomascota=\"{$_POST['interes']}\"";
            #27
  $resultado = mysql_query($consulta)
       or die ("No se pudo ejecutar la consulta.");

  /* Desplegar resultados en una tabla */
  echo "<table cellspacing='10' border='0' cellpadding='0'
```

```
                width='100%'>";
    echo "<tr><td colspan='5' align='right'>
                Haga clic en cualquier foto para ver una
            version mas grande.<br><hr>
             </td></tr>\n";
    while ($fila = mysql_fetch_array($resultado,MYSQL_ASSOC))
            #38
    {
      $f_precio = number_format($fila['precio'],2);

      /* revisar si la mascota viene en colores */
      $consulta = "SELECT * FROM Color
                      WHERE
            Nombremascota='{$fila['Nombremascota']}'";    #44
      $resultado2 = mysql_query($consulta) or
            die(mysql_error()); #45
      $ncolors = mysql_num_rows($resultado2);
            #46

      /* desplegar una fila para cada mascota */
      echo "<tr>\n";
      echo "<td>{$fila['IDmascota']}</td>\n";
      echo "<td><font size='+1'>
                <b>{$fila['Nombremascota']}</b></font></td>\n";
      echo "<td>{$fila['Descripcionmascota']}</td>\n";
      /* mostrar foto si la mascota no viene en colores */
      if ( $ncolors <= 1 )                                  #55
      {
         echo "<td><a href='../images/{$fila['pix']}'
                            border='0'>
            <img src='../images/{$fila['pix']}' border='0'
              width='100' height='80'></a></td>\n";
      }
      echo "<td align='center'>\$$f_precio</td>\n
            </tr>\n";
      /* desplegar fila para cada color si la mascota viene en
            colores */
      if ($ncolors > 1 )                                    #65
      {
         while($fila2 =
            mysql_fetch_array($resultado2,MYSQL_ASSOC))
         {
           echo "<tr><td colspan=2> </td>
                    <td>{$fila2['Colormascota']}</td>
                    <td><a href='../images/{$fila2['pix']}'
                              border='0'>
              <img src='../images/{$fila2['pix']}' border='0'
                        width='100' height='80'></a></td>\n";
         }
      }
         echo "<tr><td colspan='5'><hr></td></tr>\n";
    }
    echo "</table>\n";
    echo "<div align='center'>
          <a href='catalogo.php'>
              <b>Ver mas mascotas</b></a></div>";
?>
</body></html>
```

Los siguientes números corresponden a los números de las líneas mostrados como comentarios al final de las líneas en la Lista 11-3. Yo documento sólo algunas de las líneas en este programa en el siguiente listado. Muchas de las tareas en la lista anterior también están en la mayoría de los programas de esta aplicación, tal como conectarse a la base de datos, crear formularios y ejecutar consultas. Como documento estas tareas comunes en la Lista 11-2, no las repito aquí. He aquí una breve explicación de lo que algunas de las otras líneas hacen en el programa:

26/27 Esta consulta selecciona todas las mascotas en el catálogo que concuerdan con el tipo escogido, el cual se pasó en un formulario desde la página anterior.

38 Establece un ciclo `while` que corre una vez para cada mascota seleccionada. El ciclo crea una línea de información para cada mascota encontrada.

43 Las líneas 43 a 46 verifican si hay alguna entrada en la tabla `Color` para la mascota. Observe que los resultados de la consulta se ponen en `$resultado2`. No se podían poner en `$resultado` porque este nombre de variable ya está en uso. `$ncolors` almacena el número de filas encontradas en la tabla `Color` para la mascota. Cada nombre de mascota se revisa en busca de colores cuando se procesa en el ciclo.

55 Inicia un bloque `if` que se ejecuta sólo si encontró cero o una fila para la mascota en la tabla `Color`. El bloque `if` muestra la foto de la mascota. Si el programa encontró más de un color para la mascota en la tabla `Color`, la mascota está disponible en más de un color y la foto no debería mostrarse aquí. En su lugar, una foto para cada color se mostrará en líneas posteriores. Consulte las Figuras 11-3 y 11-4, donde verá páginas de mascotas que despliegan las fotos y la información en una sola fila, tal como se hace en este bloque `if`.

65 Inicia un bloque `if` que se ejecuta si más de un color se encontrara para la mascota. El bloque `if` hace `echo` a una fila para cada color hallado en la tabla `Color`.

67 Establece un ciclo `while` dentro del bloque if, el cual corre una vez para cada color hallado en la tabla `Color`. El ciclo muestra una línea que incluye una foto para cada color. Vea en la Figura 11-5 una página de mascotas que despliega líneas separadas con fotos para cada color.

La página cuenta con un vínculo hacia más mascotas en su parte inferior. Este vínculo apunta al programa anterior que muestra los tipos de mascotas.

Agregar mascotas al catálogo

La aplicación que agrega una mascota nueva al catálogo debería llevar a cabo las siguientes tareas:

1. **Crear un formulario que solicite una categoría de mascota.** La persona que añade la mascota puede escoger uno de los tipos ya existentes o crear uno nuevo. Para crear un tipo nuevo, el usuario debe digitar el nombre de una categoría y la descripción.

2. **Si se crea un tipo nuevo, verificar que el nombre y la descripción se hayan digitado.**
3. **Crear un formulario que pide la información de la mascota: nombre, descripción, precio, nombre de archivo de la foto y color.** La persona que agrega la mascota puede escoger uno de los nombres de mascotas existentes en la categoría seleccionada o crear uno nuevo. Para crear un nombre nuevo de mascota, el usuario debe digitar el nombre de una mascota.
4. **Si se selecciona nuevo para el nombre de mascota, verificar que el nombre se haya digitado.**
5. **Almacenar la nueva mascota en la base de datos** `Catalogodemascotas`.
6. **Enviar una página de retroalimentación que muestre qué información se acaba de añadir al catálogo.**

Las tareas se realizan en tres programas:

- `Escogercategoria.php`: Crea el formulario de los tipos de mascotas (tarea 1)
- `Escogernombre.php`: Verifica los datos en la categoría de la mascota y crea el formulario con la información de la mascota (tareas 2 y 3)
- `Agregar.php`: Revisa el campo del nombre de la mascota, almacena la mascota nueva en la base de datos del catálogo y proporciona la retroalimentación (tareas 4, 5 y 6)

Escribir Tipomascota

El primer programa produce una página web con un formulario HTML donde la persona que agrega una mascota puede seleccionar un tipo de mascota para la mascota. Para facilitar la lectura y manutención del programa, como comento en el Capítulo 10, conserve algunos de los enunciados HTML usados por el programa en un archivo separado que pueda traer al programa usando un enunciado `include`. La Lista 11-4 presenta `escogercategorai.php`.

Lista 11-4: Programa que permite al usuario seleccionar un tipo de mascota

```
<?php
  /* Programa: escogerCategoria.php
   * Descripcion: Permite a los usuarios seleccionar un tipo
          de mascota. Todos los tipos de mascotas existentes
          de la tabla Tipomascota se muestran. Se
          proporciona una seccion para introducir un nuevo
          tipo de mascota. Las selecciones se presentan como
          botones de opcion, con campos de texto para el
          nombre y la descripcion de la nueva categoria.
   */
?>
<html>
<head><title>Tipos de mascotas</title></head>
<body>
<?php
```

```
include("misc.inc");
$conexion = mysql_connect($huesped,$usuario,$clave)
     or die ("No se pudo conectar al servidor");
$db = mysql_select_db($basededatos,$conexion)
     or die ("No se pudo seleccionar la base de datos");

/* extrae los tipos de la tabla Tipomascota en orden
          alfabetico */
$consulta="SELECT Tipo_mascota FROM Tipodemascota ORDER BY
          Tipodemascota";  #23
$resultado = mysql_query($consulta)
     or die ("No se pudo ejecutar la consulta.");

/* Desplegar texto antes de formulario */
echo "<div style='margin-left: .1in'>
<p><h3>Seleccione la categoria para la mascota que esta
          agregando.</h3>
        Si esta agregando una mascota en una categoria que
          no aparece, escoja <b>New Category</b> y digite el
          nombre y la descripcion de la categoria. Oprima
          <b>Submit Category</b> cuando haya terminado de
          seleccionar una categoria existente o de digitar
          una categoria nueva.\n";

/* Crear un formulario con una lista de seleccion */
echo "<form action='escogerNombre.php' method='POST'>\n";
echo "<table cellpadding='5' border='0'>\n";
echo "<tr>";
$contador=0;
          #40
while ($fila = mysql_fetch_array($resultado))
          #41
{
    extract($fila);
    echo "<td>
     <input type='radio' nombre='categoria'
          valor='$Tipomascota'";
            if ($contador == 0)
          #46
            {
                echo "revisado";
            }
     echo ">$Tipomascota </td>\n";
          #50
     $contador++;
          #51
}
echo "</tr></table>\n";

include("tablaNuevaCategoria.inc");
          #55

echo "<p><input type='submit' valor='Enviar categoria'>\n";
echo "</form>\n";
?>
</div>
</body></html>
```

Los números siguientes corresponden a los números de las líneas mostrados como comentarios al final de las líneas en la Lista 11-4. Sólo algunas de las líneas se documentan en la lista siguiente. Muchas de las tareas en el listado, tales como conectarse a la base de datos, crear formularios y ejecutar consultas, se encuentran en la mayoría de los programas en esta aplicación; consulte en la Lista 11-2 la explicación correspondiente. La siguiente lista proporciona una breve explicación de lo que las líneas siguientes hacen en el programa:

23 Consulta que selecciona todos los tipos de mascotas de la tabla `Tipomascota` y los distribuye en orden alfabético.

40 Crea un contador con un valor inicial de 0. El contador llevará un registro de cuántos tipos de mascotas se encuentran en la base de datos.

41 Inicia un ciclo `while` que se ejecuta una vez para cada tipo de mascota. El ciclo crea una lista de botones de opción para los tipos de mascotas, con un botón seleccionado. Estos son los detalles del ciclo `while`:

45 `Hace echo` a la etiqueta del campo del formulario para el botón de opción con el valor igual a `$Tipomascota`. Este enunciado se ejecuta una vez en cada ciclo, y crea un botón de opción para cada tipo de mascota. Este enunciado `hace echo` sólo de la primera parte de la etiqueta del campo del formulario.

46 Bloque `if` que se ejecuta sólo en el primer ciclo. `Hace echo` a la palabra `"revisado"` como parte del campo del formulario. Esto garantiza que uno de los botones de opción se seleccionará al desplegarse, de modo que el formulario no se pueda enviar sin haber seleccionado un botón, lo cual daría como resultado desagradables mensajes de error o advertencias. El contador se estableció exclusivamente con este propósito.

Aunque añadir `"revisado"` a cada botón de opción funciona en algunos exploradores, causa problemas en otros. La programación adicional requerida para agregar `"revisado"` a sólo un botón de opción puede evitar problemas.

50 Hace `echo` a la parte restante de la etiqueta del campo del formulario para el botón de opción, la parte que cierra la etiqueta.

51 Suma 1 al contador para llevar el registro del número de veces que el ciclo se ha ejecutado. Esta es la última línea del ciclo `while`.

55 Crea una tabla que pide el nombre y la descripción de la nueva mascota. El HTML para la tabla se incluye desde otro campo llamado `tablaNuevaCategoria.inc`. Como comento en el Capítulo 10, el HTML (especialmente aquel que describe un formulario) a menudo se guarda en un archivo separado para facilitar la lectura del programa principal y para que el formulario sea más fácil de modificar cuando sea necesario. Este archivo se muestra en la Lista 11-5.

Lista 11-5: Archivo que contiene el formulario con el tipo nuevo

```
<?php
  /* Programa:  tablaNuevaCategoria.inc
   * Descripcion: Codigo HTML que muestra una tabla para
          agregar una categoria nueva
   */
?>
<table width='100%'>
<tr><td colspan=3><hr></td></tr>
<tr>
<td align='center'>
<input type='radio' nombre='categoria' valor='nuevo'> 
</td>
<td align='right'>Nombre de la categoria:</td>
<td><input type='text' nombre='newCat' size='20'
maxlength='20'></td>
</tr>
<tr><td align='center'><b>Categoria nueva</b></td>
<td align='right'>Descripcion de la categoria:</td>
<td><input type='text' nombre='newDesc' size='70%'
maxlength='255'>
</td>
</tr>
<tr><td colspan=3><hr></td></tr>
</table>
```

Este archivo está casi totalmente en HTML, excepto por una sección en PHP al inicio que guarda el encabezado como comentarios. En realidad, yo podría haberlo hecho con comentarios HTML, pero me gusta más el estilo de los comentarios PHP.

Escribir Nombremascota

Este segundo programa acepta los datos del formulario en el primer programa. Revisa la información y solicita información faltante. Una vez que la información sobre el tipo de mascota se recibe correctamente, el programa crea un formulario donde el usuario puede seleccionar un nombre de mascota para la nueva mascota que se está añadiendo al catálogo y digitar la información para la mascota. Este programa, como el programa anterior, trae los formularios y las tablas HTML de archivos separados usando enunciados `include`. También llama a una función que está en un archivo `include`. Este programa trae dos archivos. La Lista 11-6 muestra `ChoosePetName.php`.

Lista 11-6: Programa que pide al usuario el nombre de la mascota

```
<?php
  /* Programa: escogerNombre.php
   * Descripcion: Permite al usuario digitar informacion para la
          mascota. Primero, el programa verifica la categoria
          nueva y la introduce en la tabla Tipomascota si es
          nueva. Luego, todas las mascotas en la categoria
          seleccionada se presentan con botones de opcion. El
```

```
            usuario puede digitar un nombre nuevo. Se
            proporcionan campos para introducir la descripcion,
            el precio y el nombre de archivo de la foto.
   */
?>
<?php
  if (@$_POST['newbutton'] == "Volver a la pagina de categorias "
      or @$_POST['newbutton'] == "Cancelar")                    #15
  {
    header("Location: escogerCategoria.php");
  }
?>
<html>
<head><title>Agregar mascota</title></head>
<body>
<?php
  include("misc.inc");
  include("functions.inc");

  $conexion = mysql_connect($huesped,$usuario,$clave)
       or die ("No se pudo conectar al servidor");
  $db = mysql_select_db($basededatos,$conexion)
       or die ("No se pudo seleccionar la base de datos");

  $categoria = $_POST['categoria'];
 /* Si se selecciona una categoria de mascotas nueva, revise si
            se rellenaron campos de texto. Si no se rellenaron,
            despliegueIos nuevamente para que el usuario digite
            el nombre y la descripcion de la categoria. Cuando
            los campos estan llenos, almacene la categoria nueva
            en la tabla Tipomascota.*/
  if ($_POST['categoria'] == "nueva")                           #38
  {
    if ($_POST['newCat'] == ""
         or $_POST['newDesc'] == "")                            #41
    {
       include("formNuevaCategoria.inc");
            #43
       exit();                                                  #44
     }
     /* agregue tipo nuevo de mascota a la tabla Tipomascota */
     else                                                       #47
     {
       addNewType($_POST['newCat'],$_POST['newDesc']);          #49
       $categoria = $_POST['newCat'];
     }
  }                                                             #52

  /* Selecciona nombres de mascotas de la tabla con la categoria
            dada. Si el usuario digito una categoria nueva, se
            busca. */
  $consulta = "SELECT DISTINCT Nombremascota FROM Mascota
            WHERE Tipodemascota='$categoria' ORDER BY
            Nombremascota"; #57
  $resultado = mysql_query($consulta)
```

```
                    or die ("No se pudo ejecutar la consulta");
$nrow = mysql_num_rows($resultado);                              #60

/* crea un formulario */
echo "<div style='margin-left: .1in'>";
echo "<form action='agregar.php' method='post'>\n";
echo "<p><b>Nombre de la mascota</b></p>\n";
if ($nrow < 1)                                                   #66
{
  echo "<hr><b>No hay nombres de mascotas actualmente en la
        base de datos para la categoria
        $categoria</b><hr>\n";
}
else                                                             #71
{
  echo "<table cellpadding='5' border='0'>";
  echo "<tr>";
  while ($fila = mysql_fetch_array($resultado))                  #75
  {
     extract($fila);
     echo "<td>";
     echo " <input type='radio' name='Nombremascota'
                value='$Nombremascota'";
     echo ">$Nombremascota</td>\n";
  }
  echo "</tr></table>";
}
include ("tabla_nuevoNombre.inc");                               #85

$Descripcionmascota=" ";$precio = "";$pix = "";$Colormascota =
        "";
include("formInfoNombre.inc");                                   #88

echo "<input type='hidden' name='categoria'
           value='$categoria'>\n";
echo "<p><input type='submit' value='Enviar nombre de la
        mascota'>
      <input type='submit' name='newbutton' value='Cancelar'>
      </form>\n";
?>
</div>
</body></html>
```

Los números siguientes corresponden a los números de las líneas mostradas como comentarios al final de las líneas en la Lista 11-6. Sólo algunas de las líneas se documentan en el listado siguiente porque muchas de las tareas en la lista se encuentran en la mayoría de los programas en esta aplicación. Las tareas comunes se documentan para la Lista 11-2 y se explican en otras partes del libro, por lo cual no las repetiré aquí. He aquí una breve explicación de lo que hacen las siguientes líneas en el programa:

15 Verifica si el usuario hizo clic en el botón de envío llamado Cancelar o en Regresar a la página de las categorías. Si es así, regresa a la primera página.

38 Inicia un bloque if que se ejecuta sólo si el usuario seleccionó el botón de opción para Categoría Nueva en el formulario del programa anterior. Este bloque revisa si el nombre y la descripción de la categoría nueva se completaron. Si el usuario olvidó digitarlos, se le pide el nombre y la descripción del tipo de mascota otra vez. Una vez rellenados el nombre y la descripción, el programa llama a una función que agrega la nueva categoría a la tabla `Tipomascota`. Las líneas siguientes describen este bloque `if` más detalladamente:

41 Inicia un bloque `if` que se ejecuta sólo si el nombre de la categoría y/o la descripción de la categoría están en blanco. Como el bloque `if` está dentro del bloque `if` para una categoría nueva, este bloque se ejecuta sólo si el usuario seleccionó Categoría Nueva para el tipo de mascota pero no completó el nombre de la categoría nueva además de su descripción.

43 Crea un formulario que pide el nombre y la descripción de la categoría. El HTML para el formulario se incluye desde un archivo. Este se ejecuta sólo cuando el enunciado `if` en la línea 38 es verdadero; es decir, si la categoría es nueva y el nombre de la categoría y/o su descripción están en blanco.

44 Detiene el programa después de mostrar el formulario en la línea 43. El programa no puede proceder hasta que el nombre de la categoría y su descripción se hayan digitado. Este bloque se repetirá hasta que el nombre y la descripción de la categoría se completen.

47 Empieza un bloque `else` que se ejecuta sólo si tanto el nombre como la descripción de la categoría están completos. Como este bloque está dentro del bloque `if` para el botón de opción nuevo, este bloque se ejecuta cuando el usuario seleccionó `nueva` y completó el nombre y la descripción de la nueva categoría.

49 Llama a una función que añade la categoría nueva a la tabla `Tipomascota`.

50 Hasta este punto, la categoría todavía es `"nueva"`. Esta línea establece `$categoria` al nuevo nombre de la categoría.

52 Esta línea termina el bloque `if`. Si el usuario seleccionó uno de los tipos de mascota existentes, los enunciados entre la línea 38 y esta línea no se ejecutaron.

56 Consulta que selecciona uno de cada uno de los nombres de mascota para el tipo de mascota escogido y los ordena alfabéticamente.

60 Verifica si se encontraron nombres de mascotas para el tipo de mascota elegido.

66 Comienza un bloque `if` que se ejecuta sólo si no se encontró ninguna mascota para el tipo de mascota. El bloque `hace echo` a un mensaje para el usuario, el cual especifica que no se hallaron mascotas para ese tipo de mascota.

71 Inicia un bloque `else` que se ejecuta si efectivamente se encontraron mascotas para el tipo de mascota. El bloque else crea una lista de botones de opción para los nombres de mascotas encontrados. La lista se crea con un ciclo `while` (que empieza en la línea 75) de la misma manera que la lista de categorías fue creada, tal como se explica en la Lista 11-4.

85/88 Las líneas 85 y 88 crean tablas que piden el nombre y la información para la nueva mascota, las cuales traen el HTML de archivos separados usando enunciados `include`.

Este programa trae tres archivos que contienen HTML usando enunciados `include`. Las Listas 11-7, 11-8 y 11-9 muestran los tres archivos que se incluyen: `formnuevacat.inc`, `tablanuevanombre.inc`, y `tablainfo.inc`.

Lista 11-7: Código HTML que crea un formulario para un nuevo tipo de mascota

```
<?php
/* Programa:formNuevaCategoria.inc
* Descripcion: Muestra un formulario para recopilar el nombre
          y la descripcion de la categoria.
*/
?>
<b>Ya sea el nombre de la categoria o la descripcion quedaron
          vacios. Debe digitar ambos.</b>
<form action="escogerNombre.php" method="POST">
 <table>
<tr>
<td align="right">Nombre de la categoria:</td>
<td><input type="text" name="newCat"
value="<?php echo $_POST['newCat'] ?>"
size="20" maxlength="20">
</td></tr>
<tr>
<td align="right">Descripcion de la categoria:</td>
<td><input type="text" name="newDesc"
value="<?php echo $_POST['newDesc'] ?>"
size="70%" maxlength="255">
</td></tr>
</table>
<input type="hidden" name="categoria" value="nueva">
<p><input type="submit" name="newbutton"
value="Digite la categoria nueva ">
<input type="submit" name="newbutton"
value="Regresar a la p·gina de las categorias ">
</form>
```

Este programa está casi completamente en código HTML. Observe los siguientes aspectos sobre este formulario:

- **Este formulario se crea sólo cuando el usuario selecciona el botón de opción para Categoría Nueva en la página web para el tipo de masco-**

ta, pero no digita el nombre o la descripción de la mascota. Este formulario se muestra para dar al usuario una segunda oportunidad de digitar el nombre o la descripción.

- **La mayoría del archivo está en HTML, con sólo dos pequeñas secciones en PHP que hacen eco a los valores de los dos campos.**
- **El formulario vuelve al programa que lo generó para su procesamiento.** Se procesa del mismo modo que el formulario enviado desde la primera página. Los nombres de los campos son iguales y se revisan otra vez para ver cuáles están en blanco.
- **Un campo oculto se incluye, el cual envía $categoría con un valor de "nuevo".** Si este formulario no enviara `$categoria`, el programa que lo procesa (el mismo programa que lo generó) no sabría que el tipo de mascota era nuevo y no ejecutaría el bloque if que debería ejecutarse cuando se selecciona una categoría nueva.

Lista 11-8: Archivo HTML que crea una tabla para un nombre nuevo

```
<?php
/* Programa: tablaNuevoNombre.inc
* Descripcion: Muestra la tabla para digitar el nombre de una
            nueva mascota
*/
?>
<table border="0">
<tr><td>
<input type="radio" name="Nombremascota"
value="nuevo" checked>Nombre nuevo</td>
      <td><input type="text" name="Nombrenuevo" size="25"
                maxlength="25"> (digite el nombre nuevo)</td>
</tr>
<tr><td colspan=2><hr></td></tr>
</table>
```

Este archivo está totalmente en HTML, sin nada de PHP. Despliega la sección de la página web del nombre de la mascota donde el usuario puede digitar un nuevo nombre de mascota.

Lista 11-9: HTML que crea una tabla para la información de las mascotas

```
<?php
/* Programa: formInfoNombre.inc
* Descripcion: Muestra la tabla para recolectar informacion
            sobre la mascota
*/
?>
<b>Informacion de la mascota </b><br>
<p><table>
<tr><td align="right">Categoria de mascota:</td>
<td><b>  <?php echo $categoria ?></b></td>
</tr>
```

```
  <tr><td align="right">Descripcion de la mascota:</td>
    <td><input type="text" name="Descripcionmascota"
            value="<?php echo $Descripcionmascota ?>"
size="65" maxlength="255">
</td></tr>
  <tr><td align="right">Precio:</td>
<td><input type="text" name="precio"
            value="<?php echo $precio ?>" size="15"
           maxlength="15">
</td></tr>
  <tr><td align="right">Nombre de archivo de la foto:</td>
    <td><input type="text" name="pix"
            value="<?php echo $pix ?>" size="25"
           maxlength="25">
</td></tr>
<tr><td align="right">Color de la mascota(opcional):</td>
    <td><input type="text" name="Colormascota"
            value="<?php echo $Colormascota ?>" size="25"
           maxlength="25">
</td></tr>
 </table>
```

Este archivo incluye pequeñas secciones en PHP para los valores de las variables. Por lo demás, está en HTML.

Además de HTML para tablas y formatos, el programa `escogernombre.php` en la Lista 11-6 llama a una función. La función se almacena en un archivo llamado `funciones.inc` y se incluye al inicio del programa. La Lista 11-10 muestra la función.

Lista 11-10: Función AgregarTiponuevo()

```
<?php
 /* Funcion: AgregarTiponuevo
  * Descripcion: Agrega un tipo nuevo de mascota y su
          decripci%n a la tabla Tipomascota. Verifica
          primero si realmente es un nuevo tipo de mascota y
          no lo agrega a la tabla si ya est· ahi.
  */
 funci%n AgregarTiponuevo($Tipomascota,$Descripciontipo)
{
/* Prepara los datos */
  $Tipomascota = ucfirst(strip_tags(trim($Tipomascota)));
  $Descripciontipo =
       ucfirst(strip_tags(trim($Descripciontipo)));
  /* Verifique si la nueva categoria ya est· en la tabla
          Tipomascota.
    Si no lo esta, a√·dala a la tabla. */
  $consulta = "SELECT Tipomascota FROM Tipomascota
WHERE Tipomascota='$Tipomascota'";
```

```
  $resultado = mysql_query($consulta) or
          die ("No se pudo ejecutar la consulta ");

  $ntype = mysql_num_rows($resultado); //
if ($ntype < 1)  // si el nuevo tipo no est· en la tabla
{
     $consulta = "INSERT INTO Tipomascota
          (Tipomascota,Descripciontipo)
               VALUES ('$Tipomascota','$Descripciontipo')";
$resultado = mysql_query($consulta)
               or DIE ("No se pudo ejecutar la consulta
          ");
}
  return;
}

?>
```

La función primero depura los datos. Luego revisa si el tipo de mascota ya está en la tabla `Tipomascota`. Si no lo está, la función lo añade a la tabla.

Escribir Agregarmascota

Este último programa acepta los datos del formulario en el segundo programa. Si se seleccionó nuevo para el nombre de la mascota, lo revisa para constatar que se digitó un nombre nuevo y lo pide nuevamente si se dejó en blanco. Una vez rellenado el nombre de la mascota, el programa almacena la información sobre la mascota de la página anterior. Observe que no verifica la otra información porque ésta es opcional. Este programa, al igual que en el programa anterior, inserta algunos formularios y tablas HTML de dos archivos separados usando el enunciado `include`. La Lista 11-11 presenta `agregar.php`.

Lista 11-11: Programa que agrega una mascota nueva al catálogo

```
<?php
  /* Programa: Agregar.php
   * Descripcion: Agrega una mascota nueva a la base de
          datos. Envia una pantalla de confirmacion al
          usuario.
   */

  if (@$_POST['newbutton'] == "Cancelar")                    #
          7
  {
     header("Location: escogerCategoria.php");
  }
  $Nombremascota = $_POST['Nombremascota'];
  $Nombrenuevo = $_POST['Nombrenuevo'];
  $precio = $_POST['precio'];
  $pix = $_POST['pix'];
  $Colormascota = $_POST['Colormascota'];
  $categoria = $_POST['categoria'];
  $Descripcionmascota = $_POST['Descripcionmascota'];

  if ($Nombremascota == "nuevo")
          #18
```

```
  {
    if ($Nombrenuevo == "")
          #20
    {
       include("formulario_Nombrenuevo.inc");
       exit();
    }
    else                                                    #25
    {
      $Nombremascota=trim($Nombrenuevo);

          $Nombremascota=ucfirst(strtolower(strip_tags($Nomb
          remascota)));
    }
  }
  if ($pix == "")                                           #31
      $pix = "na.gif";
?>
<html>
<head><title>Agregar mascota</title></head>
<body>
<?php
  include("misc.inc");                                      #38
  $conexion = mysql_connect($huesped,$usuario,$clave)
       or die ("No se pudo conectar al servidor");
  $db = mysql_select_db($basededatos,$conexion)
       or die ("No se pudo seleccionar la base de datos");
  /* Depura los datos */
  $Descripcionmascota =
          strip_tags(trim($Descripcionmascota));
  $precio = strip_tags(trim($precio));
  $pix = strip_tags(trim($pix));
  $Colormascota = strip_tags(trim($Colormascota));

  $consulta = "INSERT INTO Mascota
      (Nombremascota,Tipomascota,Descripcionmascota,
          precio,pix) VALUES
      ('$Nombremascota','$categoria','
          $Descripcionmascota','$precio',
       '$pix')";
  $resultado = mysql_query($consulta)
       or die ("No se pudo ejecutar la consulta.");
  $IDmascota = mysql_insert_id();
          #55

  echo "La siguiente mascota se ha agregado al
          Catalogo de mascotas:<br>
       <ul>
        <li>Categoria: $categoria
        <li>Nombre de la mascota: $Nombremascota
        <li>Descripcion de la mascota: $Descripcionmascota
        <li>Precio: $precio
        <li>Archivo de la foto: $pix \n";

  if ($Colormascota != "")
          #66
  {
    if ($Nombremascota == "Pez dorado" o $Nombremascota ==
          "Periquito")
```

```
    {
        $consulta = "SELECT Nombremascota FROM Color
                     WHERE Nombremascota='$Nombremascota'
                     AND Colormascota='$Colormascota'";
        $resultado = mysql_query($consulta)
                or die ("No se pudo ejecutar la consulta.");
        $num = mysql_num_rows($resultado);
        if ($num < 1)
        {
          $consulta = "INSERT INTO Color
            (Nombremascota,Colormascota,pix)
                    VALUES
            ('$Nombremascota','$Colormascota','$pix')";
          $resultado = mysql_query($consulta)
                or die ("No se pudo ejecutar la consulta.");
          echo "<li>Color: $Colormascota\n";
        }
    }
  }                                                          #85
  echo "</ul>";
  echo "<a href='escogerCategoria.php'>Agregar otra
          mascota</a>\n";
?>
</body></html>
```

Observe los números de líneas mostrados como comentarios al final de las líneas en la Lista 11-11. Los números en el siguiente listado corresponden a los números de las líneas en la lista. Documento sólo algunas de las líneas en el listado siguiente porque muchas de las tareas más comunes, como conectarse a la base de datos, se han documentado en programas anteriores en este capítulo.

7 Verifica si el usuario hizo clic en el botón Cancelar. Si es así, regresa a la primera página.

18 Inicia un bloque `if` que se ejecuta sólo si el usuario seleccionó `nuevo` para el nombre de la mascota. Si el nombre nuevo está en blanco, muestra un formulario que pide el nuevo nombre de la mascota (línea 20) repetidamente hasta que el usuario lo digite. Una vez completado el nuevo nombre, `$Nombremascota` se establece en el nombre nuevo (línea 25).

31 Si el nombre de archivo de la foto no se digitó, se establece en la foto predeterminada.

38 Las líneas 38 — 55 añaden la nueva mascota a la base de datos. Los datos son depurados antes de ser añadidos.

57 Inicia el eco de la página de retroalimentación.

66 Comienza un bloque `if` que se ejecuta sólo si el color se rellenó. El color sólo se almacena para periquitos y peces. La tabla `Color` se revisa para ver si el nombre y el color ya están ahí. Si no lo están, se añaden a la tabla `Color`. El bloque if termina en la línea 85.

Este programa trae un archivo HTML que crea el formulario para pedir al usuario el nombre de la mascota si el usuario olvidó digitarlo. La Lista 11-12 muestra el archivo que se incluye: `formulario_Nombrenuevo.inc`.

Lista 11-12: HTML que pide al usuario el nombre nuevo de la mascota

```
<?php
  /* Programa: formulario_Nombrenuevo.inc
   * Descripcion: Muestra el formulario para obtener un
           nombre de mascota.
   */
?>
<b>Debe digitar un nombre de mascota.</b>
<form action="agregar.php" method="post">
 <table><tr>
   <td align="right">Nombre de la mascota:</td>
   <td><input type="text" name="Nombrenuevo"
       value="<?php echo $Nombrenuevo ?>"
         size="25" maxlength="25">
   </td></tr>
 </table>
 <input type="hidden" name="categoria"
        value="<?php echo $categoria ?>">
 <input type="hidden" name="Nombremascota"
        value="<?php echo $Nombremascota ?>">
 <input type="hidden" name="Descripcionmascota"
        value="<?php echo $Descripcionmascota ?>">
 <input type="hidden" name="precio"
        value="<?php echo $precio ?>">
 <input type="hidden" name="pix" value="<?php echo $pix ?>">
 <input type="hidden" name="Colormascota"
        value="<?php echo $Color_mascota ?>">
 <p><input type="submit" name="newbutton"
        value="Digite un nombre de mascota nuevo ">
 <input type="submit" name="newbutton" value="Cancelar">
</form>
```

Este archivo crea el formulario que se despliega si el usuario olvida digitar el nombre de mascota nuevo. Es muy parecido al programa en la Lista 11-7 que se muestra cuando un usuario olvida digitar una categoría nueva. Observe que has dos campos ocultos se usan para pasar los valores para `$categoria` y `$Nombremascota`. Cuando el formulario se completa, los valores para estas dos variables se necesitan para almacenar la información de la mascota.

Al final, este programa brinda un vínculo a la primera página, para que el usuario pueda añadir otra mascota nueva al catálogo si así lo desea.

Capítulo 12

Construir un sitio Web exclusivo para miembros

En este capítulo

- Diseñar el sitio web exclusivo para miembros
- Construir la base de datos para el directorio de miembros
- Diseñar las páginas web para la sección Exclusivo para Miembros
- Escribir los programas para registrarse en la sección Exclusivo para Miembros

Muchos sitios web exigen que los usuarios se registren. A veces los usuarios no pueden ver ninguna página web sin haber digitado una contraseña, mientras que otras veces sólo una parte de la página web requiere del registro. Estas son algunas de las razones por las cuales usted podría exigir a los usuarios que se registren:

- **La información es secreta.** No querrá que nadie excepto un pequeño grupo de personas autorizadas vea la información. O quizás sólo sus propios empleados deberían ver la información.
- **La información o el servicio está a la venta.** La información o el servicio que su sitio web proporciona es su producto, y usted desea cobrar a las personas por él. Por ejemplo, tal vez posea un secreto sobre los datos de una encuesta por el cual los investigadores estén dispuestos a pagar. Por ejemplo, el Club de Automóviles AAA ofrece alguna información gratuitamente, pero hace falta ser miembro para ver la calificación de los hoteles.
- **Puede brindar un mejor servicio.** Si sabe quiénes son sus clientes o tiene alguna información sobre ellos, puede hacer que su interacción con el sitio web sea más fácil. Por ejemplo, si usted tiene una cuenta con Barnes and Noble.com o Gap y se registra en su sitio, ellos usan su dirección de envío almacenada, y usted nunca tendrá la necesidad de digitarla nuevamente.
- **Puede averiguar más sobre sus clientes.** Al departamento de mercadeo le gustaría saber quiénes están viendo su sitio web. Una lista de clientes con direcciones y números telefónicos e incluso sus gustos y preferencias resulta un instrumento útil. Si su sitio web ofrece algunas características atractivas, los clientes podrían estar dispuestos a brindarle

alguna información para poder tener acceso a su sitio. Por ejemplo, una persona podría estar dispuesta a responder algunas preguntas para poder descargar software gratuito o jugar un excelente juego en línea.

Por lo general, un registro requiere que el usuario digite una identificación de usuario y una contraseña. A menudo, los usuarios pueden crear sus propias cuentas en el sitio web, escogiendo su propia identificación de usuario y contraseña. A veces los usuarios pueden dar mantenimiento a sus cuentas (por ejemplo, cambiar su contraseña o número telefónico) en línea.

En el Capítulo 11, aprendió cómo construir un catálogo en línea para el sitio web de su Tienda de Mascotas. Ahora, usted desea agregar una sección a su sitio web que sea exclusiva para miembros. Planea ofrecer descuentos especiales, un boletín, una base de datos con información sobre las mascotas y más en dicha sección. Espera que los clientes consideren la sección suficientemente valiosa como para proporcionarle gustosamente sus direcciones y números telefónicos para obtener una cuenta de membresía que les permita gozar de los servicios de la sección restringida. En este capítulo, usted construye una sección de registro para la Tienda de Mascotas.

Diseñar la aplicación

El primer paso en el diseño es decidir lo que la aplicación debería hacer. Su función básica es recopilar información sobre los clientes y almacenarla en una base de datos. Ofrece a los clientes acceso a información y servicios valiosos para motivarlos a brindarle la información para la base de datos. Como no hay secretos estatales ni números de tarjetas de crédito en riesgo, usted debería facilitar al máximo la creación de y el acceso a las cuentas de los clientes.

La aplicación que proporciona acceso a la sección Exclusivo para Miembros de su Tienda de Mascotas debería hacer lo siguiente:

- **Brindar un medio para que los clientes monten sus propias cuentas con identificaciones de membresía y contraseñas.** Esto incluye recoger la información requerida del cliente para convertirse en miembro.
- **Proporcionar una página donde los clientes digiten su identificación de membresía y contraseña y luego verificar si son válidas.** Si es así, el cliente entra a la sección Exclusivo para Miembros. Si no, el cliente puede probar otro registro.
- **Mostrar las páginas en la sección Exclusivo para Miembros a cualquiera que se registre.**
- **Rehusarse a mostrar las páginas en la sección Exclusivo para Miembros a quien no esté registrado.**
- **Dar seguimiento a los registros de miembros.** Usted quiere saber quién entra al sitio y con cuánta frecuencia.

Construir la base de datos

La base de datos es el corazón y el propósito de esta aplicación. Guarda la información sobre los clientes que es la meta de la sección Exclusivo para Miembros. También guarda la identificación de membresía y la contraseña, de modo que el usuario pueda entrar a la sección Exclusivo para Miembros.

La base de datos de la aplicación exclusiva para miembros contiene dos tablas:

- Tabla Miembros
- Tabla Registro

El primer paso al construir la aplicación de registro es construir la base de datos. Es prácticamente imposible escribir programas sin tener una base de datos funcional para probarlos. Primero diseñe su base de datos; luego constrúyala; después, agregue algunos datos de muestra para usarlos en el desarrollo de los programas.

Se han hecho algunos cambios al diseño de la base de datos que desarrollé en el Capítulo 3 para la sección restringida Exclusivo para Miembros del sitio web de la Tienda de Mascotas. El desarrollo y las pruebas a menudo ocasionan cambios. Tal vez usted se percate de que no tomó algunos factores en cuenta en su diseño, o que ciertos elementos de su diseño no funcionan con datos del mundo real o son difíciles de programar. Es perfectamente normal que el diseño evolucione mientras usted trabaja en su aplicación. Sólo asegúrese de cambiar su documentación cuando su diseño cambie.

Construir la tabla Miembros

En su diseño para la aplicación de registro, la tabla principal es la tabla `Miembros`, la cual guarda toda la información introducida por los clientes, incluyendo su información personal (nombre, dirección, número telefónico, etc.) y la identificación de membresía y la contraseña. La siguiente consulta SQL crea la tabla `Miembros`:

```
CREATE TABLE Miembros (
 Nombreentrada        VARCHAR(20)              NOT NULL,
  fechacreacion       DATE                  NOT NULL,
  clave               VARCHAR(255)          NOT NULL,
  apellido            VARCHAR(50),
  nombre              VARCHAR(40),
  calle               VARCHAR(50),
  ciudad              VARCHAR(50),
  estado              CHAR(2),
  codigo postal       CHAR(10),
  email               VARCHAR(50),
  telefono            CHAR(15),
  fax                 CHAR(15),
PRIMARY KEY(Nombreregistro) );
```

Cada fila representa un miembro. Las columnas son

- `Nombreregistro`: identificación de membresía que el miembro usará al registrarse. El cliente escoge y digita el nombre de registro. La consulta `CREATE` define el `Nombreregistro` como sigue:
  - `CHAR(20)`: Este tipo de datos define el campo como una cadena de caracteres de 20 caracteres de longitud. El campo siempre ocupará hasta 20 caracteres de almacenamiento, con relleno si la cadena efectivamente almacenada cuenta con menos de 20 caracteres. Si una cadena con más de 20 caracteres se almacena, los caracteres después del 20 se eliminarán.
  - `PRIMARY KEY(Nombreregistro)`: La clave primaria identifica la fila y debe ser única. MySQL no permitirá dos filas con el mismo `Nombreregistro`.
  - `NOT NULL`: Esta definición significa que el campo no puede estar vacío. Debe tener un valor. La clave primaria siempre debe establecerse en `NOT NULL`.
- `Fechacreacion`: La fecha en que la fila se agregó a la base de datos; o sea, la fecha cuando el cliente creó la cuenta. La consulta define la `Fechacreacion` como
  - `DATE`: Esta es una cadena que se trata como una fecha. Las fechas aparecen en el formato AAAA-MM-DD. Pueden digitarse en ese formato o alguno similar, tal como AA/M/D o AAAAMMDD.
  - `NOT NULL`: Esta definición significa que este campo no puede estar vacío. Debe tener un valor. Puesto que el programa, y no el usuario, crea la fecha y la almacenan, nunca estará en blanco.
- `clave`: Una contraseña que el miembro usará al registrarse. El cliente escoge y digita la contraseña. La consulta `CREATE` define la contraseña de la manera siguiente:
  - `VARCHAR(255)`: Este enunciado define el campo como una cadena de caracteres variable, compuesta por hasta 255 caracteres de longitud. El campo se almacena en su longitud real. No se espera que la `clave` conste de 255 caracteres. De hecho, se espera que sea bastante corta. Sin embargo, usted piensa usar la función contraseña de MySQL para codificarla en lugar de almacenarla a plena vista. Una vez codificada, la cadena será más larga, de manera que usted está dejando suficiente espacio para la cadena más larga.
  - `NOT NULL`: Este enunciado significa que este campo no puede estar vacío. Debe tener un valor. Usted no permitirá una contraseña vacía en esta aplicación.
- `Apellido`: El apellido del cliente, tal y como el cliente lo digitó. La consulta `CREATE` define el campo como
  - `VARCHAR(50)`: Este tipo de datos define el campo como una cadena de caracteres variable que puede tener hasta 50 caracteres de longitud. El campo se almacena en su tamaño real.

- `Nombre`: El primer nombre del cliente, tal y como el cliente lo digitó. La consulta `CREATE` define el campo así
  - `VARCHAR(40)`: Este tipo de datos define el campo como una cadena de caracteres variable que puede tener hasta 40 caracteres de longitud. El campo se almacena en su tamaño real.
- `Calle`: La dirección de la calle donde vive el cliente, tal y como éste la digitó. La consulta `CREATE` define el campo como
  - `VARCHAR(50)`: Este tipo de datos define el campo como una cadena de caracteres variable que puede tener hasta 50 caracteres de longitud. El campo se almacena en su tamaño real.
- `Ciudad`: La ciudad en la dirección del cliente, tal y como éste la digitó. La consulta `CREATE` define el campo así
  - `VARCHAR(50)`: Este tipo de datos define el campo como una cadena de caracteres variable que puede tener hasta 50 caracteres de longitud. El campo se almacena en su tamaño real.
- `Estado`: El estado en la dirección del cliente. La cadena es el código de dos letras correspondiente al estado. El cliente selecciona los datos de una lista desplegable que contiene todos los estados. La consulta `CREATE` define el campo así
  - `CHAR(2)`: Este tipo de datos define el campo como una cadena de caracteres de dos caracteres de longitud. El campo siempre ocupará dos caracteres de almacenamiento, con relleno si la cadena efectivamente almacenada es menos de dos caracteres.
- `Codigo postal`: El código postal que el cliente digita. La consulta `CREATE` define el campo como
  - `CHAR(10)`: Este tipo de datos definen el campo como una cadena de caracteres de diez caracteres de longitud. El campo siempre ocupará 10 caracteres de almacenamiento, con relleno si la cadena efectivamente almacenada consta de menos de diez caracteres. El campo es lo suficientemente largo para guardar un código posta +4, tal como 12345 — 1234.
- `email`: La dirección electrónica que el cliente digita. La consulta `CREATE` define el campo como
  - `VARCHAR(50)`: Este tipo de datos define el campo como una cadena de caracteres variable que puede tener hasta 50 caracteres de longitud. El campo se almacena en su tamaño real.
- `Telefono`: El número telefónico que el cliente digita. La consulta `CREATE` define el campo así
  - `CHAR(15)`: Este tipo de datos define el campo como una cadena de caracteres de 15 caracteres de longitud. El campo siempre ocupará 15 caracteres de almacenamiento, con relleno si la cadena efectivamente almacenada tiene menos de 15 caracteres.

- Fax: El número de fax que digita el cliente. La consulta CREATE define el campo así
  - CHAR(15): Este tipo de datos define el campo como una cadena de caracteres de 15 caracteres de longitud. El campo siempre ocupará 15 caracteres de almacenamiento, con relleno si la cadena efectivamente almacenada tiene menos de 15 caracteres.

Observe que algunos campos son CHAR y otros son VARCHAR. Los campos CHAR son más rápidos, mientras que los campos VARCHAR son más eficientes en el uso del espacio en disco. Su decisión dependerá de lo que sea más importante para usted para su aplicación en su ambiente: la velocidad o el espacio en disco.

En general, los campos más cortos deberían ser CHAR, pues éstos no desperdician mucho espacio. Por ejemplo, si su CHAR es de 5 caracteres, la mayor cantidad de espacio que podría desperdiciar es 4. Sin embargo, si su CHAR es 200, podría desperdiciar 199 caracteres. Por lo tanto, para campos cortos, use CHAR para tener velocidad y muy poco espacio desperdiciado.

Construir la tabla Entrada

La tabla Registro da seguimiento a los registros de miembros al tomar nota de la fecha y hora cada vez que un miembro se registra. Como cada miembro tiene múltiples registros, estos datos requieren de su propia tabla. La consulta CREATE que construye la tabla Entrada es

```
CREATE TABLE Entrada (
  Nombreentrada       VARCHAR(20)   NOT NULL,
  Tiempoentrada       DATETIME      NOT NULL,
PRIMARY KEY(nombreentrada,Tiempoentrada) );
```

La tabla Entrada tiene sólo dos columnas, como sigue:

- Nombreentrada: La identificación de membresía que el cliente usa para registrarse. El nombreentrada es la conexión entre la tabla Miembros (la cual describo en la sección anterior) y esta tabla. Observe que la columna Nombreregistro se define igual en la tabla Miembros y en esta. Esto posibilita unir las tablas, y facilita el hacer concordar las filas en las tablas. La consulta CREATE define el Nombreentrada de la siguiente manera:
  - CHAR(20): Este tipo de datos define el campo como una cadena de caracteres de 20 caracteres de longitud. El campo siempre ocupará 20 caracteres de almacenaje, con relleno si la cadena efectivamente almacenada consta de menos de 20 caracteres. Si se almacena una cadena más larga, los caracteres después del 20 se eliminan.
  - PRIMARY KEY(Nombreentrada,Horaregistro): La clave primaria identifica la fila y debe ser única. Para esta tabla, dos columnas juntas

son la clave primaria. MySQL no permitirá que se introduzcan dos filas con el mismo `Nombreentrada` y la misma `Tiempoentrada`.

- `NOT NULL`: Esta definición significa que el campo no puede estar vacío. Debe tener un valor. La clave primaria siempre se establece en `NOT NULL`

- `Tiempoentrada`: La fecha y hora cuando el miembro se registró. Este campo usa tanto la fecha como la hora porque debe ser único. Es muy poco probable que dos usuarios entren en el mismo segundo al sitio web de la Tienda de Mascotas. Sin embargo, en algunos sitios web muy ocupados, podría suceder. En dichos sitios, usted tal vez necesite crear un número secuencial de registro para que sea la clave primaria única para el sitio. La consulta `CREATE` define `Horaregistro` del modo siguiente:

  - `DATETIME`: Esta es una cadena que se trata como una fecha y hora. La cadena se despliega en el formato AAAA-MM-DD HH:MM:SS.

  - `PRIMARY KEY(Nombreentrada,Tiempoentrada)`: La clave primaria identifica la fila y debe ser única. Para esta tabla, dos columnas juntas son la clave primaria. MySQL no permitirá dos filas con el mismo `Nombreentrada` y la misma `Tiempoentrada`.

  - `NOT NULL`: Esta definición significa que el campo no puede estar vacío. Debe tener un valor. La clave primaria debe siempre establecerse en `NOT NULL`.

Agregar datos a la base de datos

Esta base de datos tiene el objetivo de guardar los datos introducidos por los clientes, no por usted. Estará vacía cuando la aplicación se ponga por primera vez a disposición de los clientes, y hasta que los clientes añadan datos. Sin embargo, para probar los programas mientras usted los escribe, necesita tener al menos un par de miembros en la base de datos. Necesita un par de identificaciones de membresía y contraseñas para probar el programa de registro. Puede agregar un par de miembros falsos con el propósito de hacer las pruebas (usando la consulta `SQL INSERT`) y retirarlos cuando esté listo para abrir al público su aplicación exclusiva para miembros.

Diseñar la apariencia

Una vez que sepa lo que la aplicación hará y cuál información usted desea obtener de la base de datos, puede diseñar la apariencia y el aspecto. La apariencia incluye lo que el usuario ve y cómo el usuario interactúa con la aplicación. Su diseño debería ser atractivo y fácil de usar. Puede crear este diseño en papel, e indicar lo que el usuario ve, incluso con bocetos o descripciones escritas. También debería incluir en su diseño los componentes de interacción del usuario, tales como botones o vínculos, y describir sus acciones. Incluya todas las páginas de la aplicación en el diseño.

La aplicación exclusiva para miembros de la Tienda de Mascotas tiene tres páginas que son parte de los procedimientos de registro. Además, la aplicación incluye todas las páginas que son parte de la sección Exclusivo para Miembros, tales como la página que muestra los descuentos especiales y las páginas que presentan discusiones sobre el cuido de mascotas. En este capítulo, sólo constuiremos las páginas que son parte del procedimiento de registro. No se construyen las páginas que son parte de la sección Exclusivo para Miembros, aunque sí comento lo que necesita incluirse en ellas para protegerlas de los ojos de visitantes no registrados.

La aplicación de registro incluye tres páginas, más el grupo de páginas que compone la sección Exclusivo para Miembros, como sigue:

- **La página inicial de la tienda:** La primera página que ven los clientes. Indica el nombre del negocio y el propósito del sitio web. Le presento una página inicial de una tienda en el Capítulo 11 y, en este capítulo, usted la modifica para brindar acceso a la sección Exclusivo para Miembros.
- **Página de registro:** Permite a los clientes ya sea registrarse o crear una cuenta de membresía nueva. Muestra un formulario que el cliente deberá completar para obtener una cuenta nueva.
- **Página de bienvenida a los miembros nuevos:** Da la bienvenida a los usuarios nuevos y los llama por su nombre, lo cual les deja saber que su cuenta ha sido creada. Les brinda cualquier información que necesiten saber. Presenta un botón para que los usuarios puedan continuar hacia la sección Exclusivo para Miembros o para que regresen a la página principal.
- **Sección Exclusivo para Miembros:** Grupo de páginas web que encierra el contenido de la sección Exclusivo para Miembros.

Página inicial de la tienda

La página inicial de la tienda es la página introductoria a la Tienda de Mascotas. Como la mayoría de las personas ya saben lo que es una tienda de mascotas, esta página no necesita proveer muchas explicaciones. La Figura 12-1 muestra la página inicial de la tienda. Hay dos acciones disponibles para el cliente en esta página: un vínculo en el cual hacer clic para ver el catálogo de mascotas y un vínculo a la sección Exclusivo para Miembros.

Página de registro

La página de registro permite al cliente registrarse o crear una cuenta de membresía nueva. Incluye el formulario que los clientes deben rellenar para obtener una cuenta de membresía. La Figura 12-2 muestra la página de registro. Esta página tiene dos botones de envío diferentes: uno para registrarse con una cuenta de membresía existente y uno para crear una cuenta de membresía nueva.

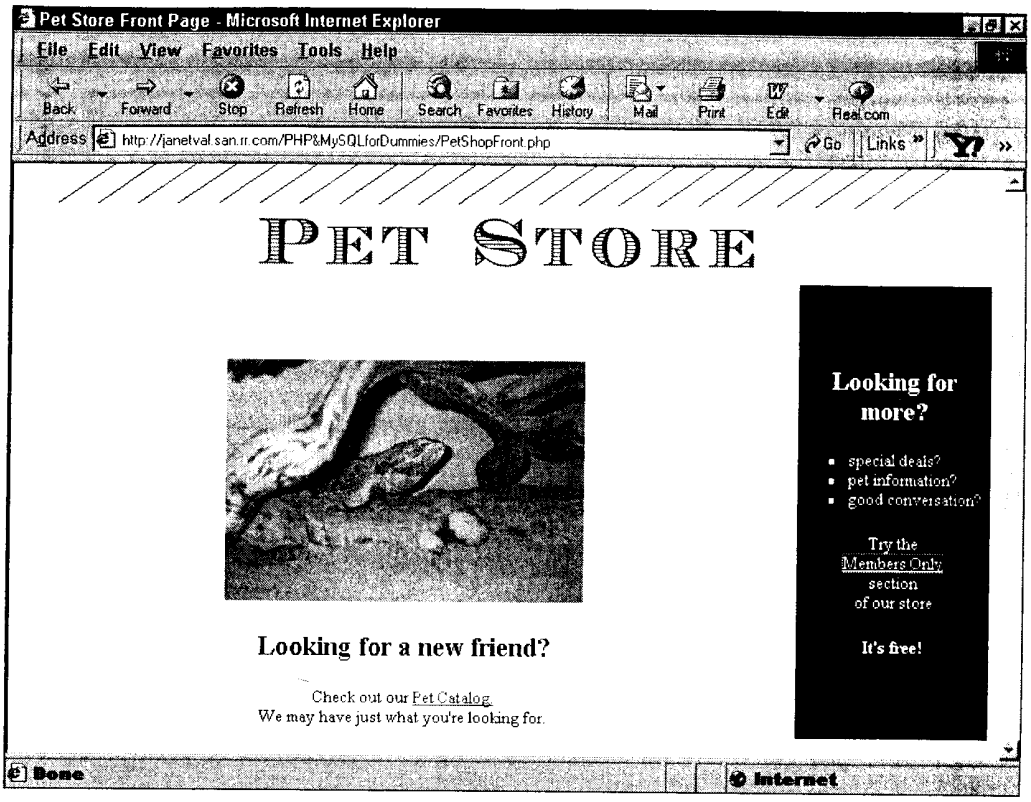


Figura 12-1: Página de apertura del sitio web de la Tienda de Mascotas.

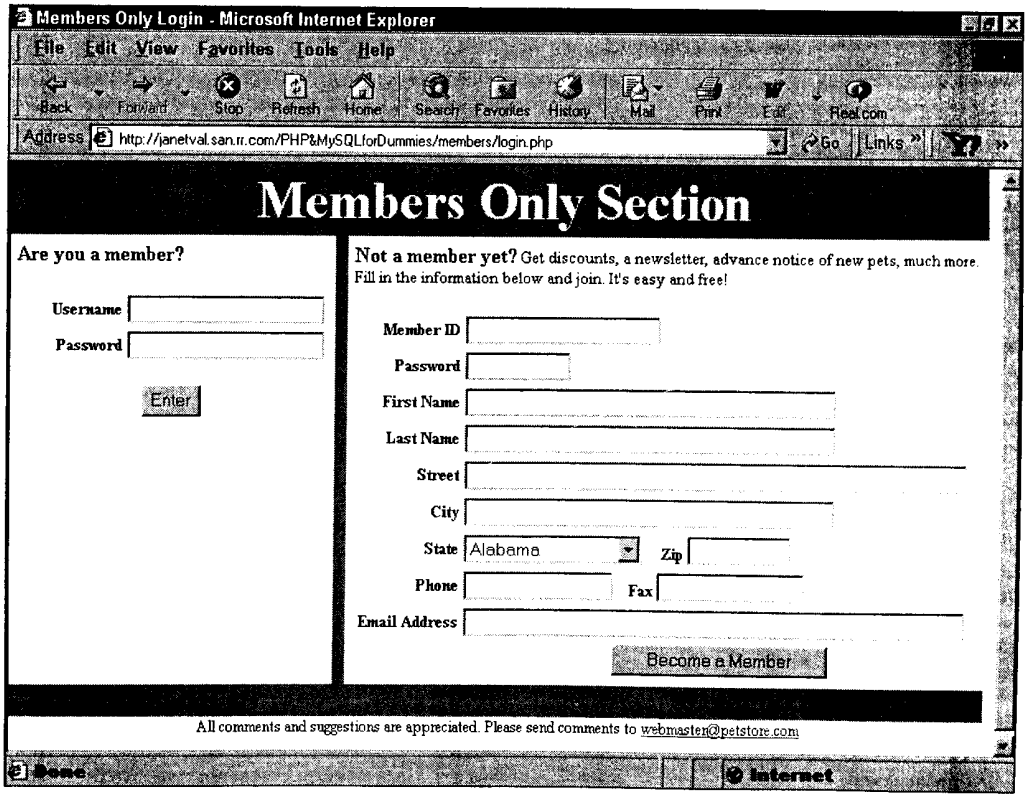


Figura 12-2: La página donde los clientes se registran o crean una nueva cuenta de membresía.

Si un cliente comete un error en la página de registro, ya sea en la sección de registro o en la sección para miembros nuevos, el formulario aparecerá nuevamente, con un mensaje de error. Por ejemplo, suponga que un cliente comete un error al digitar su dirección electrónica: Se le olvidó digitar `.com` al final de la dirección. La Figura 12-3 muestra la pantalla que verá después de haber enviado el formulario con un error en él. Observe que el mensaje de error aparece justo encima del formulario.

Cuando los miembros se registran exitosamente con una identificación de membresía y contraseña válidas, van a la primera página de la sección Exclusivo para Miembros. Cuando los miembros nuevos envían exitosamente un formulario con información que parece razonable, van a la página de de bienvenida para los miembros nuevos (consulte la siguiente sección). Además, se envía un mensaje electrónico al nuevo miembro con el siguiente contenido:

```
Una nueva cuenta de membresia ha sido creada para usted. Su
        identificacion de membresia y contraseña son:

gperez
secreta

Apreciamos su interes en nuestra Tienda de Mascotas en
        PetStore.com

Si tiene preguntas o problemas, envie un mensaje a
        webmaster@petstore.com
```

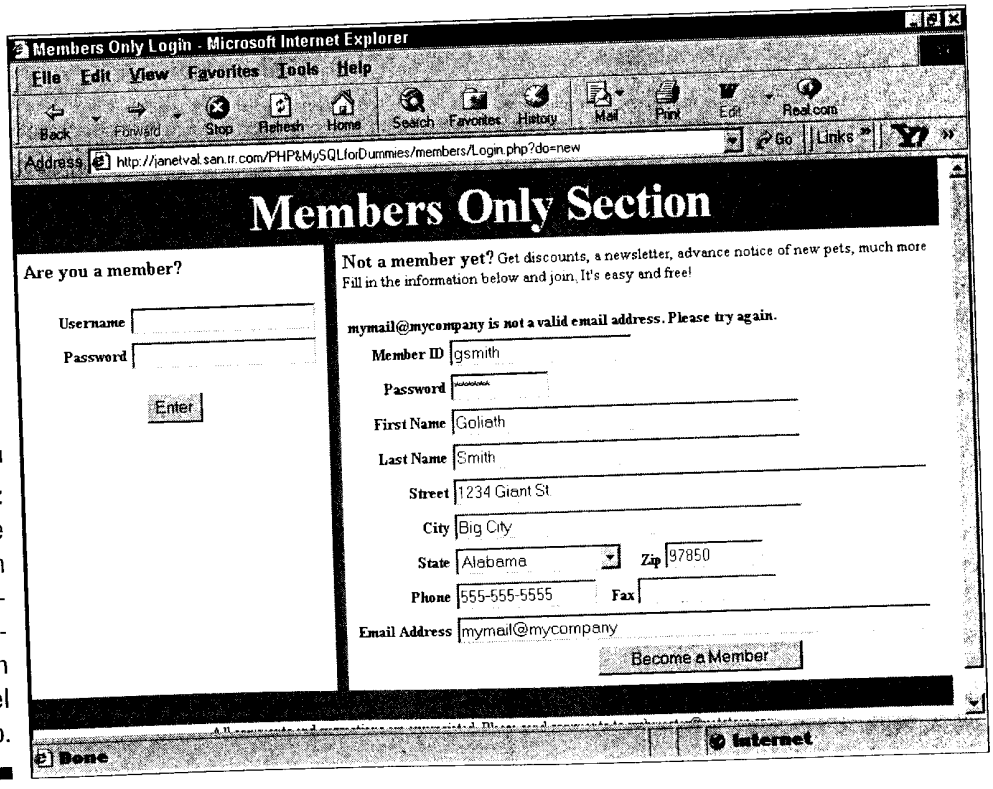


Figura 12-3: Página que muestra un mensaje como resultado de un error en el formulario.

Este mensaje electrónico contiene la contraseña del cliente. Creo que es útil tanto para el cliente como para el negocio enviar al cliente una copia impresa de su contraseña. Los clientes seguramente olvidarán su contraseña. Parece ser una de las reglas. Un mensaje electrónico con su contraseña podría ayudarlos cuando la olviden, y les ahorrará tanto a ellos como a usted algunos problemas. Por supuesto, los mensajes electrónicos no son necesariamente seguros, de modo que mandar las contraseñas por correo electrónico no es una buena idea para algunas cuentas, tales como cuentas bancarias en línea. Pero, para esta aplicación de la Tienda de Mascotas, en la cual sólo los descuentos no autorizados y la información sobre el cuidado de mascotas corren riesgo, enviar la contraseña por correo electrónico es un riesgo razonable.

Página de bienvenida a los miembros nuevos

La página de bienvenida a los miembros nuevos saluda al cliente y le ofrece información útil. El cliente ve que la cuenta se ha instalado y luego puede entrar a la sección Exclusivo para Miembros inmediatamente. La Figura 12-4 muestra una página de bienvenida.

Figura 12-4: Página de bienvenida a los miembros nuevos.

Sección Exclusivo para Miembros

Uno o más páginas web conforman los contenidos de la sección Exclusivo para Miembros. Cualquiera que sea el contenido, las páginas no difieren de cualquier otra página web o programa PHP, excepto por algunos enunciados PHP al inicio de cada archivo, los cuales impiden que quienes no son miembros vean las páginas.

Escribir los programas

Después de saber cómo se verán las páginas y lo harán, puede escribir los programas. En general, se crea un programa para cada página, aunque a veces tiene más sentido separar los programas en más de un archivo o combinar programas en una página. (Para ver los detalles sobre cómo organizar las aplicaciones, consulte el Capítulo 10.)

Como recomiendo en el Capítulo 10, guarde la información necesaria para conectarse a la base de datos en un archivo separado e incluya ese archivo en todos los programas que necesitan acceder a la base de datos. Almacene el archivo en una ubicación segura, bajo un nombre engañoso. Para esta aplicación, la siguiente información está almacenada en un archivo llamado `dogs.inc`:

```
<?php
  $usuario="catalogo";
  $usuario="localhost";
  $clave="";
  $basededatos="Directoriodemiembros";
?>
```

La aplicación de registro de los miembros tiene varias tareas básicas:

1. **Mostrar la página inicial de la tienda.** Esta proporciona un vínculo a la página de registro.
2. **Mostrar una página donde los clientes pueden rellenar una identificación de membresía y una contraseña para registrarse.**
3. **Revisar la identificación de membresía y la contraseña que el cliente digita contra la identificación de membresía y la contraseña en la base de datos.** Si la identificación y la contraseña están bien, el cliente entra a la sección Exclusivo para Miembros. Si la identificación y/o la contraseña están equivocadas, el cliente vuelve a la página de entrada.
4. **Mostrar una página donde los clientes pueden completar la información necesaria para obtener una cuenta de membresía.**
5. **Revisar la información digitada por el cliente en busca de campos vacíos o formatos incorrectos.** Si se encuentra información mala, mostrar el formulario nuevamente para que el cliente pueda corregirla.

6. **Cuando la información está correcta, agregar el nuevo miembro a la base de datos.**
7. **Mostrar la página de bienvenida a los miembros nuevos.**

Las tareas se realizan en tres programas:

- `Paginainicialtiendademascotas.php`: Muestra la página inicial de la tienda (tarea 1).
- `Login.php`: Lleva a cabo las tareas de registro y creación de cuentas para miembros nuevos (tareas 2 — 6).
- `Nuevomiembro.php`: Muestra la página que da la bienvenida al miembro nuevo (tarea 7).

Escribir Página inicial de la tienda de mascotas

La página inicial de la tienda no necesita ningún enunciado PHP. Simplemente despliega una página web con un vínculo al Catálogo de Mascotas y un vínculo a la sección Exclusivo para Miembros del sitio web. Los enunciados en HTML (Lenguaje de Marcado de Hipertexto, por sus siglas en inglés) son suficientes para hacerlo. La Lista 12-1 muestra el archivo HTML que describe la página inicial de la tienda.

Lista 12-1: Archivo HTML para la página inicial de la tienda

```
<?php
  /* Programa: TiendaMascotas.php
   * Descripcion: Muestra la pagina de apertura para la
           Tienda de Mascotas.
   */
?>
<html>
<head><title>Pagina inicial de la Tienda de Mascotas
          </title></head>
<body topmargin="0" leftmargin="0" marginheight="0"
      marginwidth="0">
<table width="100%" height="100%" border="0"
       cellspacing="0" cellpadding="0">
  <tr>
    <td align="center" valign="top" height="30" colspan="2">
      <img src="images/awning-top.gif" alt="awning">
    </td>
  </tr>
  <tr>
    <td align="center" valign="top" colspan="2">
      <img src="images/Name.gif" alt="Pet Store">
    </td></tr>
  <tr>
    <td width="80%" align="center">
```

```
        <p style="margin-top: 40pt">
        <img src="images/lizard-front.jpg" alt="foto de
             lagartija "
          height="186" width="280">
        <p><h2>¿Busca un nuevo amigo?</h2>
        <p>Consulte nuestro
             <a href="catalogo.php">Catalogo de Mascotas.</a>
          <br> Probablemente tenemos lo que necesita.
      </td>
      <td width="20%" bgcolor="black">
        <div style="color: white; link: white">
        <p style="text-align: center; font-size: 15pt">
        <b>¿Busca algo <br>mas?</b></p>
        <ul>
         <li>¿ofertas especiales?
         <li>¿informacion sobre mascotas?
         <li>¿buena conversacion?
        </ul>
        <p style="text-align: center">Pruebe la seccion
        <br><a href="login.php"
                 style="color: white">exclusiva para
            miembros</a>
        <br><br>de nuestra tienda
        <p style="text-align: center"><b>¡Es gratis!</b></p>
      </td>
    </tr>
  </table>
  </body></html>
```

Observe el vínculo hacia el programa PHP de registro. Cuando el cliente hace clic en el vínculo, la página de registro aparece.

Escribir Login

La página de registro (vea la Figura 12-2) es producida por el programa `Login.php`, como se muestra en la Lista 12-2. El programa usa un enunciado switch para crear dos secciones: una para el registro y otra para crear una cuenta nueva. El programa crea una sesión que se usa en las páginas web exclusivas para miembros. El formulario de registro no se incluye en este programa; está en un archivo separado llamado `login_form.inc`, el cual se trae a este programa cada vez que se necesite el formulario, usando enunciados `include`.

Lista 12-2: Programa de registro

```
<?php
/* Programa: Login.php
 * Descripcion: Programa de registro para la seccion Exclusivo
                para Miembros de la Tienda de Mascotas. Brinda dos
                opciones: (1) registrarse usando un nombre de entrada
                existente y (2) introducir un nuevo nombre de
```

```
            registro. Los nombres de registro y las claves se
            almacenan en una base de datos MySQL.
*/
 session_start();                                                    # 9
 include("dogs.inc");                                                #10
 switch (@$_GET['do'])                                               #11
 {
   case "registro":                                                  #13
     $conexion = mysql_connect($usuario, $usuario,$clave)            #14
              or die ("No se pudo conectar al servidor.");
     $db = mysql_select_db($basededatos, $conexion)
              or die ("No se pudo seleccionar la base de
            datos.");                                                #17

     $sql = "SELECT nombreEntradaFROM Miembro
              WHERE Nombreregistro='$_POST[fnombreusuario]'";
                                                                     #20
     $resultado = mysql_query($sql)
                or die("No se pudo ejecutar la consulta.");
                                                                     #22
     $num = mysql_num_rows($resultado);                              #23
     if ($num == 1)  // nombre de registro fue encontrado
                                                                     #24
     {
        $sql = "SELECT nombreEntradaFROM Miembro
                WHERE Nombreregistro='$_POST[fnombreusuario]'
                AND clave=clave('$_POST[fclave]')";
        $result2 = mysql_query($sql)
                 or die("No se pudo ejecutar la consulta 2.");
                                                                     #30
        $num2 = mysql_num_rows($result2);
        if ($num2 > 0)  // clave correcta                            #32
        {
          $_SESSION['auth']="si";                                    #34
          $nombreregistro=$_POST['fnombreusuario'];
          $_SESSION['nombreregistro'] = $nombreregistro;
                                                                     #36
          $hoy = date("A-m-d h:m:s");                                #37
          $sql = "INSERT INTO Registro
           (nombreEntrada,fechadeCreacion)
                   VALORS ('$nombreregistro','$hoy')";
          mysql_query($sql) or die("No se puede ejecutar la
           consulta.");
          header("Ubicacion: Member_page.php");                      #41
        }
        else    // clave no es correcta                             #43
        {
          unset($do);                                                #45
          $mensaje="El nombre de registro,
           '$_POST[fnombreusuario]'
                    existe, pero usted no ha digitado la clave
           correcta. Por favor,  intentelo de nuevo.<br>";
          include("login_form.inc");                                 #49
        }
     }                                                               #51
     elseif ($num == 0) // nombre de registro no encontrado
                                                                     #52
     {
        unset($do);                                                 #54
        $mensaje = "El nombre de registro digitado no existe.
           Por favor, intentelo de nuevo.<br>";
        include("login_form.inc");
     }
   break;                                                            #59
```

```
                                                                    #61
case "new":                                                         #62
  foreach($_POST as $campo => $valor)
  {                                                                 #64
    if ($campo != "fax")
    {                                                               #66
      if ($valor == "")
      {
        unset($_GET['do']);
        $mensaje_nuevo = "Hace falta informacion obligatoria.
        Por favor, intentelo de nuevo.";
        include("login_form.inc");
        exit();
      }
    }
    if (ereg("(Nombre)",$campo))                                    #75
    {
     /*if (!ereg("^[A-Za-z' -]{1,50}$",$valor))
     {
       unset($_GET['do']);
       $mensaje_nuevo = "$campo no es un nombre valido.
                        Por favor, intentelo de nuevo.";
       include("login_form.inc");
       exit();
     }*/
    }
    $$campo = strip_tags(trim($valor));                             #86
  } // end foreach
  if (!ereg("^[0-9]{5,5}(\-[0-9]{4,4})?$",$Codigopostal))
                                                                    #88
  {
    unset($_GET['do']);
    $mensaje_nuevo = "$Codigopostal no es un codigo postal
        valido. Por favor, intentelo de nuevo.";
    include("login_form.inc");
    exit();
  }
  if (!ereg("^[0-9)(xX -]{7,20}$",$Telefono))                       #96
  {
    unset($_GET['do']);
    $mensaje_nuevo = "$Telefono no es un numero telefonico
        valido. Por favor, intentelo de nuevo.";
    include("login_form.inc");
    exit();
  }
  if ($fax != "")                                                   #104
  {
    if (!ereg("^[0-9)(xX -]{7,20}$",$fax))
    {
      unset($_GET['do']);
      $mensaje_nuevo = "$fax no es un numero de fax valido.
        Por favor, intentelo de nuevo.";
      include("login_form.inc");
      exit();
    }
  }
  if (!ereg("^.+@.+\\..+$",$email))                                 #115
  {
    unset($_GET['do']);
    $mensaje_nuevo = "$email no es una direccion electronica
        valida. Por favor, intentelo de nuevo.";
    include("login_form.inc");
    exit();                                                         #122
  }
  /* revisar si el nombre de registro ya existe */
```

```
        $conexion = mysql_connect($usuario,$usuario,$clave)
                or die ("No se pudo conectar al servidor.");
        $db = mysql_select_db($basededatos, $conexion)
                or die ("No se pudo seleccionar la base de
            datos.");
        $sql = "SELECT nombreEntradaFROM Miembro
                WHERE Nombreregistro='$nombrenuevo'";
        $resultado = mysql_query($sql)
                or die("No se pudo ejecutar la consulta.");
        $num = mysql_numrows($resultado);
        if ($num > 0)                                          #133
        {
          unset($_GET['do']);
          $mensaje_nuevo = "$nombrenuevo ya existe. Seleccione otra
              identificacion de membresia.";
          include("login_form.inc");
          exit();
        }
        else                                                   #141
        {
          $hoy = date("A-m-d");                                #143
          $sql = "INSERT INTO Miembro
              (Nombreregistro,Fechacreacion,
                    clave,nombre,apellido,calle,ciudad,
                    estado,codigopostal,telefono,fax,email) VALORES
                  ('$nombrenuevo','$hoy',clave('$contranueva'),
                   '$nombre','$apellido','$calle','$ciudad',
              $estado','$codigopostal','$telefono','$fax','$email')
              ";
          mysql_query($sql);                                   #150
          $_SESSION['auth']="si";                              #151
          $_SESSION['nombreregistro'] = $nombrenuevo;
              #152
          /* enviar mensaje electronico al miembro nuevo */
              #153
          $emess = "Una nueva cuenta de membresia ha sido creada
              para usted. ";
          $emess.= "Su identificacion de membresia y clave nuevas
              son: ";
          $emess.= "\n\n\t$nombrenuevo\n\t$contranueva\n\n";
          $emess.= "Apreciamos su interes en nuestra Tienda de
              Mascotas ";
          $emess.= " en PetStore.com. \n\n";
          $emess.= "Si tiene preguntas o problemas,";
          $emess.= " envie un mensaje a webmaster@petstore.com";
          $ehead="De: member-desk@petstore.com\r\n";           #161
          $subj = "Su nueva cuenta de membresia para la Tienda de
              Mascotas ";
          $mailsend=mail("$email","$subj","$emess","$ehead");
          header("ubicacion: New_member.php");                 #164
        }
      break;                                                   #166

      default:                                                 #168
        include("login_form.inc");
    }
?>
```

Algunas de las líneas en la Lista 12-2 tienen números de líneas al final de los renglones. El siguiente listado se refiere a los números de línea en la lista para comentar el programa y cómo funciona:

9 Inicia una sesión. La sesión debe iniciarse al principio del programa, aun cuando el usuario no se haya registrado todavía.

10 Incluye el archivo que establece las variables necesarias para conectarse a la base de datos. El programa se llama `dogs.inc`, el cual es un nombre engañoso que parece más seguro que llamarlo `miscontraseñas.inc`.

11 Comienza un enunciado `switch`. El enunciado `switch` contiene tres secciones, con base en el valor que se pasó para do, obtenido del arreglo incorporado `$_GET`. La primera sección corre cuando la pareja de valores pasados para `do` es `registro`; la segunda sección corre cuando el valor pasado para `do` es `nuevo`; y la tercera sección es la predeterminada que corre si no se pasó ningún valor para `do`. La tercera sección simplemente crea la página de registro y sólo corre cuando el cliente se vincula a la página de registro por primera vez.

13 Inicia el bloque `case` para la sección de registro: la sección que corre cuando el cliente se registra. La sección de registro del formulario envía `do=login` en el URL, lo cual hace que esta sección del enunciado `switch` corra.

14 Las líneas 14 — 17 se conectan a MySQL y seleccionan la base de datos.

19 Las líneas 19 — 22 buscan en la tabla `Miembros` de la base de datos una fila con el nombre de registro digitado por el cliente.

23 Verifica si se encontró una fila con un campo `Nombreregistro` que contiene la identificación de membresía digitada por el cliente. `$num` será igual a `0` o `1`, dependiendo de si se halló la fila o no.

24 Inicia un bloque `if` que se ejecuta si la identificación de membresía se encontró. Eso significa que el usuario envió una identificación de membresía que sí está en la base de datos. Este bloque luego chequea si la contraseña enviada por el usuario es correcta para la identificación de membresía dada. Este bloque se documenta en más detalle en la lista siguiente:

26 Las líneas 26 — 28 crean una consulta que busca una fila tanto con la identificación de membresía como con la contraseña enviadas por el cliente. Observe que la contraseña enviada en el formulario (`$fclave`) se codifica usando la función MySQL, `password()`. Las contraseñas en la base de datos están codificadas, de modo que la contraseña que usted está tratando de hacer concordar también debe estar en código, o no funcionará.

29 Las líneas 29 — 31 ejecutan la consulta y revisan si se encontró una concordancia. `$num2` equivale a `1` o `0`, dependiendo de si se halló una fila con la identificación de membresía y la contraseña.

32 Comienza un bloque `if` que se ejecuta si la contraseña es correcta. Este sería un registro exitoso. Las líneas 32 — 41 se ejecutan, y realizan las siguientes tareas: 1) Las dos variables de la sesión, `aut` y `nomregistro`, se almacenan en el arreglo `SESSION`. 2) `$hoy` se crea con la fecha y hora de hoy, en el formato correcto esperado por la tabla en la base de datos. 3) Una fila para el registro se introduce en la tabla Registro. 4) La primera página de la sección Exclusivo para Miembros se envía al miembro.

43 Inicia un bloque `else` que se ejecuta si la contraseña no es correcta. Este sería un registro no exitoso. Las líneas 45 — 49 se ejecutan, y

realizan las siguientes tareas: 1) Eliminar la variable del formulario `$do`. Esto evita cualquier confusión posterior. 2) Establecer el mensaje de error apropiado en `$mensaje`. 3) Mostrar la página de registro nuevamente. La página de registro mostrará el mensaje de error.

Observe que el ciclo que inicia en la línea 43 deja al usuario saber cuándo tiene un nombre de registro verdadero, pero la contraseña equivocada. Si la seguridad de sus datos es muy importante, tal vez quiera escribir este ciclo en forma diferente. Brindar esa información podría resultar útil para alguien que esté tratando de ingresar ilegalmente. El intruso ahora sólo necesita averiguar la contraseña. Para mayor seguridad, tenga una condición que dé el mismo mensaje de error cada vez que el nombre de entrada o la contraseña sean incorrectos. En este ejemplo, yo prefiero brindar la información porque es útil para el miembro legítimo (quien tal vez no recuerde si ya instaló una cuenta o no), y no estoy protegiendo ninguna información vital.

51 Termina el bloque que se ejecuta cuando la identificación de membresía se encuentra en la base de datos.

52 Inicia un bloque `if` que se ejecuta cuando la identificación de membresía no se encuentra en la base de datos. Este realmente podría ser un ciclo else, en lugar de uno `elseif`, pero creo que es más claro para los humanos con la condición `if` en el enunciado. Este bloque elimina la variable del formulario `$do`, crea el mensaje de error apropiado y también muestra la página de registro nuevamente, la cual incluye el mensaje de error.

59 Termina el bloque `case` que se ejecuta cuando el cliente envía la identificación de membresía y la contraseña para registrarse. El bloque de registro se extiende desde la línea 13 hasta esta línea.

61 Empieza el bloque case que se ejecuta cuando el cliente completa el formulario para obtener una cuenta de membresía nueva. El formulario envía `do=new` en el URL, lo cual hace que el programa salte a esta sección del enunciado `switch`.

62 Inicia un ciclo `foreach` que circula por cada campo en el formulario del miembro nuevo. El ciclo revisa en busca de campos obligatorios en blanco y verifica que el nombre y el apellido tengan caracteres aceptables. Los enunciados en el ciclo se documentan más detalladamente en la siguiente lista:

64 Verifica si el campo es el campo de fax. El campo para el fax no es obligatorio. No se chequea para ver si está en blanco, porque no hay problema si lo está.

66 Revisa si el campo está en blanco. Si es así, el bloque `if` realiza las siguientes tareas: 1) Elimina `$do`. 2) Crea un mensaje de error que explica el problema. 3) Muestra el formulario de registro otra vez, incluyendo el mensaje de error. 4) Detiene el programa y espera a que el usuario envíe el formulario nuevamente con el campo relleno.

75 Verifica si el campo es el campo del apellido o del nombre. Si lo es, revisa el formato del campo en busca de caracteres permitidos. Si

encuentra algún caracter que no es permitido, realiza las siguientes tareas: 1) Elimina `$do`. 2) Crea un mensaje de error que explica el problema. 3) Muestra el formulario de registro otra vez, incluyendo el mensaje de error. 4) Detiene el programa y espera a que el usuario envíe el formulario nuevamente con el formato correcto.

86 Recorta espacios adicionales de todos los valores del campo después de haberlos revisado en la línea 66 para asegurarse de que no estén vacíos. Retira cualquier etiqueta HTML que encuentra en cualquiera de los campos. Crea una variable para cada uno de los campos de la siguiente manera. Suponga que en el primer ciclo del ciclo foreach, la variable es `$ _POST[Nombreregistro] = gperez`. El ciclo foreach establece $clave="Nombreregistro" y `$valor="gperez"`. Por lo tanto, el enunciado en la línea 86 es equivalente a `$Nombreregistro=strip_tags(trim("gperez"))`. La `$$clave` es `$Nombreregistro` porque `$clave=Nombreregistro`.

87 Termina el ciclo `foreach`..

88 Las líneas 88 — 122 son un arreglo de bloques `if` que revisan los campos para cerciorarse de que tengan el formato correcto. Si alguno de los campos revisados no tiene el formato correcto, el bloque realiza las tareas siguientes: 1) Elimina `$do`. 2) Crea un mensaje de error que explica el problema. 3) Muestra el formulario de registro otra vez, incluyendo el mensaje de error. 4) Detiene el programa y espera a que el usuario envía el formulario nuevamente con el formato correcto.

124 Las líneas 124 — 132 revisan si la identificación de membresía enviada por el cliente ya es un `Nombreregistro` en la tabla Miembros de la base de datos. El `Nombreregistro` debe ser único. `$num` equivale a 0 o 1, dependiendo de si `Nombreregistro` se encuentra en la base de datos.

133 Inicia un bloque `if` que se ejecuta si `Nombreregistro` ya está en la base de datos. El nuevo miembro no puede agregarse si la identificación de membresía no es única. El bloque realiza las siguientes tareas: 1) Elimina `$do`. 2) Crea un mensaje de error que explica el problema. 3) Muestra el formulario de registro nuevamente, incluyendo el mensaje de error. 4) Detiene el programa y espera a que el usuario vuelva a enviar el formulario con una identificación de membresía diferente.

141 Comienza un bloque else si el `Nombreregistro` no está en la base de datos. Esta es una aplicación exitosa para una cuenta de membresía. El bloque inserta una fila nueva en la tabla `Miembros` para la cuenta del nuevo miembro y manda un mensaje electrónico al cliente sobre la cuenta nueva. Los enunciados en el bloque se documentan más detalladamente en la siguiente lista:

143 Establece `$hoy` en la fecha de hoy, en el formato correcto para el campo `Fechacreacion` de la tabla `Miembros`.

144 Crea una consulta `INSERT` para añadir la fila del miembro nuevo. Observe que la contraseña está codificada como `clave('$contranueva')` al introducirse. Esta es una medida de

seguridad, la cual impide que alguien se fije en la base de datos y pueda ver la contraseña. Si usted está totalmente seguro de que nadie que no debiera verá la base de datos, la codificación no es realmente necesaria.

150 Ejecuta la consulta `INSERT`.

151 En la líneas 151 y 152, las dos variables de sesión, `$aut y $nomregistro`, se almacenan en el arreglo `SESSION`.

154 Las líneas 154 — 163 mandan un mensaje electrónico al nuevo miembro, en el cual verifican la identificación de membresía y la contraseña. Observe que el mensaje electrónico se crea en la variable `$emess` a lo largo de varias líneas. Empieza en la línea 154 y se añade (usando.=) a cada línea hasta terminar, en la línea 160. Esto es para facilitar a los humanos la lectura, no porque PHP lo necesite. A diferencia del contenido HTML, el cual ignora los espacios extras y los finales de las líneas, estas y otras cosas tienen un efecto en el mensaje electrónico. Por ejemplo, si yo creara un mensaje largo (con espacios adicionales para sangrarlo, y permitirme así leerlo), esos espacios aparecerían en el mensaje electrónico. Por eso, establezco el mensaje en varias líneas en las cuales puedo usar sangrías para facilitar la lectura del programa. La línea 163 usa la función PHP `mail` para enviar el mensaje electrónico. La función `mail` se documenta en el Capítulo 14.

164 Envía al cliente a la página de miembros nuevos.

165 Termina el bloque `else` para una aplicación exitosa de la cuenta del miembro nuevo.

166 Termina el bloque `case` para la sección Miembros Nuevos de la página de registro.

168 Inicia el bloque case para la condición predeterminada. Si `$do` no está configurado en `"registro"` ni `"nuevo"`, el programa salta hasta este bloque. Como los dos formularios en la página de registro están configurados en `$do`, este bloque sólo se ejecuta la primera vez que este programa corre: cuando el usuario se vincula a él desde la página inicial de la tienda y aún no ha enviado ninguno de los formularios. Esta sección sólo tiene un enunciado: el enunciado que despliega la página de registro.

Este programa muestra la página de registro en muchos lugares. Esto se hace con los enunciados include que llaman al archivo `login_form.inc`. Este archivo incluye el HTML que produce la página de registro. El programa `Login.php` no produce ningún output. Todo el output es producido por `login_form.inc`. Este tipo de orden en la aplicación se comenta en el Capítulo 10. Este es un buen ejemplo del uso de archivos `include`. Sólo imagine este programa, el cual ya es bastante largo, si los enunciados en `login_form.inc`, mostrados en la Lista 12-3, se incluyeran en el programa de registro en cada lugar donde se incluye `login_form`. Vaya, ese sería un enredo que sólo un PC podría entender.

Lista 12-3: Archivo que crea la página de registro

```
<?php
 /* File: login_form.inc
  * Descripcion: Muestra la p·gina de registro. La pagina
          despliega dos formularios: uno para digitar un
          nombre de registro y una contrase√a existentes, y
          otro para la informacion necesaria para solicitar
          una cuenta nueva.
*/
include("functions12.inc");                                    # 8
?>
<html>
<head><title>Registro exclusivo para miembros</title></head>
<body topmargin="0" leftmargin="0" marginheight="0"
      marginwidth="0">
<table border="0" cellpadding="5" cellspacing="0">
   <tr><td colspan="3" bgcolor="gray" align="center">
       <font color="white" size="+10">
      <b>Secci%n Exclusivo para miembros</b></font></td></tr>
   <tr>
       <td width="33%" valign="top">
         <font size="+1"><b>øEs usted un miembro?</b></font>
         <p>
         <!-formulario para registro del cliente -->
         <form action="Login.php?do=login" method="POST">
         <table border="0">
                                                               #25
<?php
           if (isset($mensaje))
            echo "<tr><td colspan='2'>$mensaje </td></tr>";
?>
          <tr><td align=right><b>Nombreusuario</b></td>
            <td><input type="text" name="fnombreusuario"
                    size="20" maxsize="20">
            </td></tr>
          <tr><td width="120"
          align="right"><b>Contrase√a</b>
            </td>
            <td><input type="password" name="fcontrase√a"
                    size="20" maxsize="20"></td></tr>
         <tr><td align="center" colspan="2">
             <br><input type="submit" name="log"
                    value"Enter">
             </td></tr>
        </table>
        </form>
       </td>
       <td width="1" bgcolor="gray"></td>
       <td width="67%">
         <p><font size="+1"><b>øTodav√a no es
           miembro?</b></font>
             Obtenga descuentos, un boletin, avisos
           anticipados sobre nuevas mascotas, y mucho mas.
           Complete la informacion a continuacion y
           suscribase. ûEs facil, y gratis! </b>
        <!-formulario para completarse por miembro nuevo -->
       <form action="Login.php?do=new" method="POST">
```

```
        <p>
         <table border="0" width="100%">
<?php
          if (isset($mensaje_nuevo))                          #55
             echo "<tr><td
          colspan='2'><b>$mensaje_nuevo</b>
                        </td></tr>";
?>
          <tr><td align="right"><b>identificacion de
          membresГa</b></td>
              <td><input type="text" name="nombrenuevo"
                  value"<?php echo @$nombrenuevo ?>"
                  size="20" maxlength="20"></td></tr>
          <tr><td align="right"><b>Contrasena</b></td>
              <td><input type="password" name="contranueva"
                         value"<?php echo @$ contranueva ?>"
 size="10" maxlength="8"></td></tr>
          <tr><td align="right"><b>Nombre </b></td>
              <td><input type="text" name=" nombre"
                        value"<?php echo @$ nombre ?>"
                         size="40" maxlength="40"></td></tr>
          <tr><td align="right"><b>Apellido </b></td>
              <td><input type="text" name="Apellido"
                         value"<?php echo @$Apellido ?>"
                         size="40" maxlength="40"></td></tr>
          <tr><td align="right"><b>Calle</b></td>
              <td><input type="text" name="calle"
                        value"<?php echo @$calle ?>"
                         size="55" maxlength="50"></td></tr>
          <tr><td align="right"><b>Ciudad</b></td>
              <td><input type="text" name="ciudad"
                         value"<?php echo @$ciudad ?>"
                         size="40" maxlength="40"></td></tr>
          <tr><td align="right"><b>Estado</b></td>
              <td><select name="estado">
<?php
                   $Nombreestado=extraerNombreEstado();
          #86
                   $Codigoestado=extraerCodigoEstado();
          #87
                   for ($n=1;$n<=50;$n++)
                   {
                      $estado=$Nombreestado[$n];
                      $scodigo=$Codigoestado[$n];
                      echo "<opcion value'$scodigo'";
                      if ($scodigo== "AL")
                       echo " seleccionado";
                     echo ">$estado\n";
                   }
?>
                  </select>
                      <b>Codigo
          postal</b>
                  <input type="text" name="Codigopostal"
                         value"<?php echo @$Codigopostal ?>"
                         size="10" maxsize="10">
              </td></tr>
```

```
            <tr><td align=right><b>Telefono</b></td>
                <td><input type="test" name="Telefono"
                           value"<?php echo @$Telefono ?>"
                           size="15" maxlength="20">
                       <b>Fax</b>
                    <input type="text" name="fax"
                        value"<?php echo @$fax ?>"
                        size="15" maxlength="20"></td></tr>
            <tr><td align=right><b>email</b></td>
                <td><input type="test" name="email"
                           value"<?php echo @$email ?>"
                           size="55" maxlength="67"></td></tr>
            <tr><td> </td>
                <td align="center">
                    <input type="submit"
                           value"Afiliese "></td>
            </tr>
          </table>
        </form>
      </td>
   </tr>
   <tr><td colspan="3" bgcolor="gray"> </td></tr>
</table>
<div align="center"><font size="-1">
    Todos los comentarios y sugerencias son bienvenidos. Por
          favor, envíe sus comentarios a <a
          href="mailto:webmaster@petstore.com">
webmaster@petstore.com</A> </font></div>

</body></html>
```

Observe los siguientes puntos sobre `login_form`:

- La mayoría de los enunciados están en HTML, con unas pocas secciones en PHP aquí y allá.
- Los dos formularios que empiezan en las líneas 23 y 51 ponen al mismo programa en acción, pero agregan una cadena diferente al URL: `do=login` o `do=new`.
- Los mensajes de error se muestran en la página de registro usando pequeñas secciones PHP. Cada formulario tiene su sección, y el mensaje tiene nombres diferentes para los dos formularios: `$mensaje` y `$mensaje_nuevo`. En la línea 26, la variable `$mensaje` se prueba. Si tiene un valor, el mensaje se muestra, pero no aparece si no tiene ningún valor. Si no hubo ningún error en el formulario, el mensaje nunca se configuró, y no se desplegó ningún mensaje. Un enunciado similar en la línea 55 muestra mensajes de error del formulario para miembros nuevos.

- Una lista desplegable de selección (comenzada en la línea 84) se brinda para que el cliente seleccione el estado; esto protege contra errores de digitación. Observe que las líneas 86 y 87 llaman funciones. Estas no son funciones PHP; son mías. Las funciones se incluyen en el programa en la línea 8; crean series de una lista de nombres de estados y una lista con códigos de dos letras para los estados. Usando funciones, no hacen falta las dos listas de los 50 estados en el programa. Las funciones se pueden usar repetidamente para muchos programas. El archivo `function12.inc` contiene las dos funciones, como sigue:

```
<?php
function extraerCodigoEstado()
{
  $Codigo_estado = arreglo(1=> "AL",
       "AK",
       "AZ",
       ...
       "WY" );
  return $Codigoestado;
}

function extraerNombreEstado()
{
  $Nombre_estado = arreglo(1=> "Alabama",
       "Alaska",
       "Arizona",
       ...
       "Wyoming" );
  return $Nombreestado;
}
?>
```

  Luego, un ciclo `for` crea 50 opciones para la lista de selección, usando los dos arreglos de estados.

Después de correr `Login.php`, si el usuario resulta exitoso en su registro, la primera página de la sección Exclusivo para Miembros se despliega. Si el usuario tiene éxito al abrir su cuenta nueva, el programa `New_member.php` correrá.

Escribir Nuevomiembro

La página de bienvenida a los nuevos miembros los saluda por su nombre y brinda información sobre sus cuentas. Después, los miembros tienen la opción de entrar a la sección exclusiva para miembros o regresar a la página principal. La Lista 12-4 muestra el programa que presenta la página que los miembros nuevos ven.

Lista 12-4: Programa que da la bienvenida a los miembros nuevos

```
<?php
 /* Programa: nuevo_miembro.php
  * Descripcion: Muestra la pagina de bienvenida. Saluda al
          miembro por su nombre y da al usuario la opcion de
          entrar a la seccion restringida o regresar a la
          pagina principal.                                    # 7
  */
  session_start();                                             # 9

  if (@$_SESSION['auth'] != "si")
  {
     header("Location: login.php");
     exit();                                                   #14
  }
  include("dogs.inc");
  $conexion = mysql_connect($usuario,$usuario,$clave)
              or die ("No se pudo conectar al servidor.");     #16

  $db = mysql_select_db($base de datos, $clave)
              or die ("No se pudo seleccionar la base de       #18
          datos.");
  $sql = "SELECT Nombre,Apellido FROM Miembro
                WHERE
          Nombreregistro='{$_SESSION['nombreregistro']}'";
  $resultado = mysql_query($sql)
              or die("No se pudo ejecutar la consulta 1.");
  $fila = mysql_fetch_array($resultado,MYSQL_ASSOC);
  extract($fila);
  echo "<html>
        <head><title>Bienvenida a los miembros nuevos
           </title></head>
        <body>
        <h2 align='center' style='margin-top: .7in'>
        Bienvenido(a) $Nombre $Apellido</h2>\n";               #30
  ?>
  <p>Su nueva cuenta le permite ingresar a la seccion Exclusivo
          para Miembros de nuestro sitio web. Encontrara
          descuentos especiales y promociones, una enorme
          base de datos con informacion y anecdotas sobre
          animales, consejos de los expertos, avisos
          anticipados sobre nuevas mascotas en venta, un
          tablero de mensajes donde puede hablar con otros
          miembros, y mucho mas.
  <p>Su nueva identificacion de membresia y clave le fueron
          enviados por correo electronico. Guardelos
          cuidadosamente para uso futuro.<br>

  <div align="center">
  <p style="margin-top: .5in"><b>¡Nos alegra contar con su
          presencia!</b>
  <form action="login.php" method="POST">
     <input type="submit"
          valor="Entrar a la seccion Exclusivo para
           Miembros">

  </form>
  <form action="miembrosTiendaMascotas.php" method="POST">
     <input type="submit" valor="Ir a la pagina principal de la
           Tienda de Mascotas ">

  </form>
  </div>
  </body></html>
```

Observe los siguientes puntos sobre `nuevomiembro.php.php`:

- Se inicia una sesión en la línea 7. Esto hace que las variables de sesión almacenadas en `Login.php` estén disponibles en este programa.
- El programa revisa, empezando en la línea 9, si el cliente se ha registrado. `$aut` se configura en sí en `Login.php` cuando el cliente se registra exitosamente o crea una cuenta nueva, y se almacena en el arreglo `$_SESSION`. Si `$aut` no es igual a sí, el cliente no está registrado. Si un cliente trate de correr el programa `New_member.php` sin correr el programa `Login.php` primero, `$_SESSION[aut]` no será igual a sí, y el usuario será enviado a la página de registro.
- El programa extrae el nombre y apellido del cliente de la base de datos, comenzando por el enunciado para la conexión con la base de datos, en la línea 15. En la línea 19/20, la consulta se crea usando `$_SESSION[nomregistro]` para buscar la información del miembro. La variable de sesión `nomregistro` que contiene la identificación de membresía se configuró en el programa de registro.
- La sección PHP termina en la línea 30. El resto del programa es HTML.
- El programa usa dos formularios distintos para proporcionar dos botones de envío diferentes. Los enunciados del formulario en las líneas 41 y 45 inician programas distintos.

El cliente controla lo que sucede después. Si el cliente hace clic en el botón para regresar a la página principal, se corre el programa `tiendamascotas.php`. Si el cliente hace clic en el botón de envío para la sección Exclusivo para Miembros, la primera página de la sección exclusiva para miembros se muestra.

Escribir la sección Exclusivo para Miembros

Las páginas web en la sección Exclusivo para Miembros no son diferentes de otras páginas web. Usted sólo quiere restringirlas a los miembros registrados. Para hacerlo, inicia una sesión y verifica si están registrados al comienzo de cada página. Los enunciados para la parte superior de cada programa son

```
session_start();
if (@$_SESSION['auth'] != "si")8
{
   header("Ubicacion: Login.php");
   exit();
}
```

Cuando `session_start` se ejecuta, PHP chequea si hay una sesión existente. Si es así, configura las variables de `sesion`. Una de las variables de sesión es `$aut`. Cuando el usuario se registra, `$_SESSION[aut]` se establece en `si`. Si `$_SESSION[aut]` no es igual a `si`, el usuario no está registrado, y el programa lleva al usuario a la página de registro.

Planear para crecer

El plan original de una aplicación generalmente incluye todas las cosas maravillosas que el usuario podría desear que hiciera. En términos más realistas, generalmente es importante poner la aplicación a disposición de los usuarios lo más pronto posible. En consecuencia, las aplicaciones usualmente se hacen públicas con un subconjunto de la funcionalidad planeada. Más funcionalidad se añade después. Por eso es importante escribir su aplicación con el crecimiento futuro en mente.

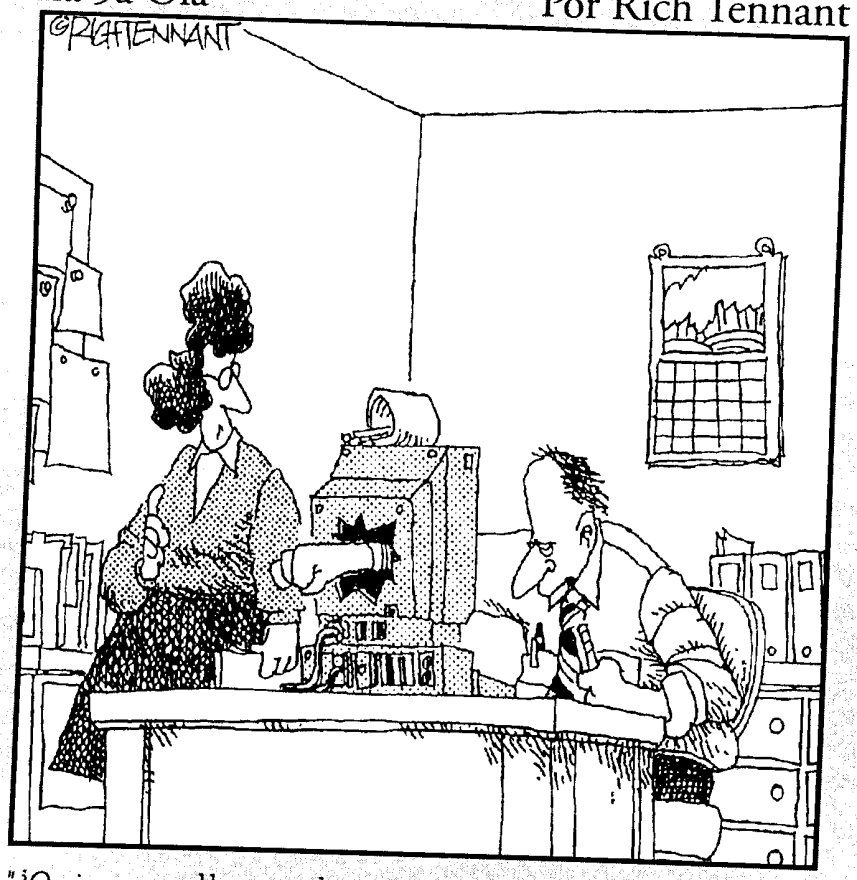
Si se fija en la aplicación de registro de este capítulo, estoy segura de que verá muchas cosas que podrían agregarse. Estas son algunas posibilidades:

- **Enviar una contraseña olvidada por correo electrónico.** Los usuarios a menudo olvidan sus contraseñas. Muchas aplicaciones de registro cuentan con un vínculo en el cual los usuarios pueden hacer clic para que sus contraseñas sean mandadas por correo electrónico.
- **Cambiar la contraseña.** Los miembros tal vez quieran cambiar su contraseña. La aplicación podría ofrecer un formulario para los cambios de contraseñas.
- **Actualizar la información.** Los miembros podrían mudarse o cambiar su número telefónico o dirección electrónica. La aplicación podría brindar una manera para que los miembros cambien su propia información.
- **Crear una lista de miembros.** Tal vez usted desee presentar una lista de todos los miembros en la base de datos, formateada elegantemente. Esto probablemente no sea algo que querrá poner a la disposición de otros miembros, sino sólo para usted mismo. En algunas situaciones, sin embargo, podría querer ofrecer la lista a todos los miembros.

Usted puede añadir fácilmente cualquiera de estas capacidades a la aplicación. Por ejemplo, puede agregar un botón al formulario de registro que diga *Olvidé mi contraseña*, el cual le enviará la contraseña por correo electrónico a la dirección en la base de datos. El botón puede correr el programa de registro con una sección para mandar la contraseña por correo electrónico o correr un programa diferente que envíe lo haga. Igualmente, puede añadir botones para cambiar la contraseña o actualizar la información del cliente. No es necesario esperar hasta que una aplicación tenga todos los detalles y acabados para dejar que sus clientes la usen. Puede escribirla paso a paso.

Parte V
Los diez más

La 5a Ola Por Rich Tennant



"¿Quiere que llame a la compañía y les pida que envíen otra copia de prueba de su sistema de software para bases de datos, o ya sabe lo que va a escribir?"

En esta parte. . .

Los capítulos en esta parte contienen pistas, consejos y advertencias basados en mi experiencia. Tal vez le sirvan como un atajo en su viaje hacia convertirse en un desarrollador web seguro de sí mismo. Sinceramente, espero que así sea.

Capítulo 13

Diez cosas que querrá hacer usando funciones PHP

En este capítulo

- Aprender sobre muchas funciones útiles
- Entender lo que las funciones pueden hacer

Uno de los aspectos más fuertes de PHP lo constituyen sus muchas funciones incorporadas. En este capítulo, incluyo las funciones PHP que yo uso con más frecuencia. Describo algunas de ellas en otras partes de este libro; algunas las menciono brevemente, y otras no aparecen antes. De ninguna manera son estas todas las funciones que existen. Hay muchos cientos de funciones en el lenguaje PHP. Para una lista completa de todas las funciones, vea la documentación de PHP en `www.php.net`.

Comuníquese con MySQL

PHP tiene muchas funciones diseñadas específicamente para interactuar con MySQL. Describo las siguientes funciones MySQL a fondo a lo largo del libro, particularmente en el Capítulo 8:

```
mysql_connect();      mysql_select_db();      mysql_fetch_array()
mysql_close();        mysql_num_rows();       mysql_query()
```

Las siguientes funciones podrían ser útiles, pero no las comento o las comento apenas brevemente en capítulos anteriores:

- `mysql_insert_id()`: Para uso con una columna MySQL `AUTO-INCREMENT`. Esta función extrae el último número insertado en la columna.
- `mysql_fetch_row($result)`: Extrae una fila de la ubicación para los resultados temporales. La fila se pone en un arreglo con números como claves. Es igual que `mysql_fetch_array($row,MYSQL_NUM)`.

- `mysql_affected_rows($result)`: Devuelve el número de filas que fueron afectadas por una consulta; por ejemplo, el número de filas borradas o actualizadas.
- `mysql_num_fields($result)`: Devuelve el número de campos en un resultado.
- `mysql_field_name($result, N)`: Devuelve el nombre de la fila indicado por N. Por ejemplo, `mysql_field_name($result,1)` indica el nombre de la segunda columna en el resultado. La primera columna es 0.

Si usa alguna de las funciones anteriores con MySQL 4.1, sus nombres son algo distintos. En lugar de empezar con `mysql_`, los nombres de las funciones empiezan con `mysqli_`.

Envíe un mensaje electrónico

PHP proporciona una función que manda un mensaje electrónico desde su programa PHP. El formato es

```
mail(direccion,asunto,mensaje,encabezados);
```

Estos son los valores que usted necesita rellenar:

- *direccion:* La dirección electrónica que recibirá el mensaje.
- *asunto:* Una cadena que va en la línea de asunto del mensaje electrónico.
- *mensaje:* El contenido que va dentro del mensaje.
- *encabezados:* Una cadena que establece valores para los encabezados. Por ejemplo, tal vez tenga una cadena de encabezados como la siguiente:

  ```
  "From: member-desk@petstore.com\r\nbcc: mama@miempresa.com"
  ```

  El encabezado configuraría el encabezado De a la dirección electrónica dada, y además enviará una copia ciega del mensaje electrónico a *mamá*.

El siguiente es un ejemplo de enunciados PHP que usted puede usar en su programa para montar y enviar un mensaje electrónico:

```
$para = "janet@valade.com";
$asunto = "Prueba";
$mensaje = "Esta es una prueba de la funcion mail";
$encabezados = bcc:techsupport@miempresa.com\r\n
$enviarmail = mail($para,$asunto,$mensaje,$encabezados);
```

A veces usted podría tener un problema con su correo electrónico. PHP tiene una configuración que debe estar correcta antes de que la función mail se pueda conectar al software de correo electrónico de su sistema. La función predeterminada ge-

neralmente está correcta, pero si sus mensajes no parecen llegar a su destino, revise la configuración de mail en PHP al buscar el siguiente output de `phpinfo()`:

```
Sendmail_path                (en Unix/Linux)
SMTP                         (en Windows)
```

Podría estar configurado incorrectamente. Usted puede cambiar la configuración editando el archivo `php.ini`. Busque las siguientes líneas:

```
[mail function]
; For Win32 only.
SMTP = localhost

; For Win32 only.
sendmail_from = me@localhost.com

; For Unix only.
;sendmail_path =
```

Los usuarios de Windows deben cambiar las primeras dos configuraciones. La primera configuración es donde se pone el nombre de su servidor de correo saliente. Sin importar cómo usted envíe el correo electrónico (por medio de una LAN en su trabajo, un cable módem en casa, un PSI vía módem), manda el correo con un servidor SMTP, el cual tiene una dirección que usted debe conocer.

Si envía directamente desde su PC, debería poder encontrar el nombre del servidor de correo saliente que está usando en su software de correo electrónico. Por ejemplo, en Microsoft Outlook Express, escoja Tools➪Accounts➪ Properties y luego seleccione la pestaña Servers. Busque el nombre de su servidor de correo saliente. Si no lo encuentra, puede preguntar a su administrador de correo electrónico por el nombre. Si usa un PSI, puede preguntar allí. El nombre probablemente tenga un formato similar al siguiente:

```
mail.nombrepsi.net
```

La segunda configuración es la dirección del remitente que se envía con todos sus mensajes electrónicos. Cambie la configuración a la dirección electrónica que desea usar como dirección de remitente, como sigue:

```
sendmail_from = Janet@Valade.com
```

La tercera configuración es paras los usuarios de Unix. La configuración predeterminada generalmente es la correcta. Si no funciona, usted debe hablar con el administrador de su sistema sobre la ruta correcta hacia su servidor de correo saliente. Esto usualmente aplica a Linux también.

No olvide quitar el punto y coma al inicio de las líneas. El punto y coma hace que la línea se convierta en un comentario, de modo que la configuración no está activa hasta que usted no elimine el punto y coma.

Use sesiones PHP

Las funciones para abrir o cerrar una sesión se presentan a continuación. Explico todas estas funciones en el Capítulo 9.

```
session_start();      session_destroy()
```

Detenga su programa

Ocasionalmente, usted sólo desea detener el programa: acabarlo, hacerlo desistir. Hay dos funciones para hacerlo: `exit()` y `die()`. En realidad, estos son dos nombres diferentes para la misma función. Exit (salir) probablemente sea exacto, pero a veces es más divertido decir die (morir). Ambas funciones mostrarán un mensaje cuando se detienen si usted proporciona uno. El formato es

```
exit("cadena del mensaje");
```

Cuando `exit` se ejecuta, la cadena del mensaje aparece.

Maneje arreglos

Los arreglos son muy útiles en PHP, particularmente para extraer los resultados de funciones de la base de datos y para variables de formularios. Explico las siguientes funciones de series en otras partes del libro, principalmente en el Capítulo 7:

```
array();    extract();      sort();      asort();
rsort();    arsort();       ksort();     krsort();
```

Estas son otras funciones útiles:

- `array_reverse($nombrevar)`: Devuelve un arreglo con los valores en orden invertido.
- `array_unique($nombrevar)`: Elimina los valores duplicados de un arreglo.
- `in_array("cadena",$nombrevar)`: Revisa un arreglo `$varname` en busca de una cadena `"cadena"`.
- `range(valor,valor2)`: Crea un arreglo que contiene todos los valores entre `valor1` y `valor2`. Por ejemplo, *range* (`'a','z'`) crea un arreglo que contiene todas las letras entre la a y la z.
- `explode("sep","cadena")`: Crea un arreglo de cadenas en la cual cada artículo es una subcadena de `cadena`, separado `sep`. Por ejemplo,

`explode(" ",$cadena)` crea un arreglo en la cual cada palabra en `$cadena` es un valor separado. Se parece a la función `split` en Perl.

- `implode("glue",$cadena)`: Crea una cadena que contiene todos los valores en `$cadena` con `glue` entre ellos. Por ejemplo, `implode(", ",$array)` crea una cadena: `valor1, valor2, valor3`, etc. Se parece a la función `join` en Perl.

Y hay muchas otras útiles funciones de series. PHP puede hace prácticamente cualquier cosa que a usted se le ocurra hacer con un arreglo.

Revise en busca de variables

A veces, usted sólo necesita saber si una variable existe. Las siguientes funciones se pueden usar para probar si una variable está configurada actualmente:

```
isset($varname);  // verdadero si la variable esta configurada
!isset($varname); // verdadero si la variable no esta configurada
empty($varname);  // verdadero si el valor es 0 o no esta
          configurado
```

Formatee valores

En ocasiones, necesita formatear los valores en las variables. En el Capítulo 6, le explico cómo formatear números en el formato de pago usando `number_format()` y `sprintf()`. En ese mismo capítulo, también comento `unset()`, la cual elimina los valores de una variable. En esta sección, describo algunas capacidades adicionales de `sprintf()`.

La función `sprintf()` le permite formatear cualquier cadena o número, incluyendo valores de variables. El formato general es

```
$newvar = sprintf("format",$nombrevar,$nombrevar,...);
```

donde `format` da instrucciones para el formato y `$nombrevar` contiene los valores a formatear. `format` puede contener literales e instrucciones para formatear los valores en `$nombrevar`. En realidad, el formato puede contener sólo literales. El siguiente enunciado es válido:

```
$newvar = sprintf("Tengo una mascota ");
```

Este enunciado da como output la cadena literal. Sin embargo, usted también pude agregar variables, usando los siguientes enunciados:

```
$nperros = 5;
$ngatos = 2;
$newvar = sprintf("Tengo %s perros y %s gatos",$nperros,$ngatos);
```

El `%s` es una instrucción de formato que le dice a `sprintf` que inserte el valor de la variable como una cadena. Así, el output es `Tengo 5 perros y 2 gatos`. El carácter `%` le indica a `sprintf` que una instrucción de formato inicia ahí. La instrucción de formato tiene el siguiente formato:

```
%pad-width.dectype
```

Estos son los componentes de las instrucciones de formato:

- `%`: Señala el comienzo de la instrucción de formato.
- *`pad`*: Un carácter de relleno usado para completar el número cuando sea necesario. Si no se especifica ningún carácter, se usa un espacio. `pad` puede ser un espacio, un 0 o cualquier carácter precedido por una comilla simple (`'`). Por ejemplo, es común rellenar números con 0, así: 01 ó 0001.
- `-`: Símbolo que indica justificar los caracteres hacia la izquierda. Si no se incluye, los caracteres se justifican hacia la derecha.
- *`width`*: Número de caracteres por usar para el valor. Si el valor no rellena el ancho (`width`), se usa el carácter de relleno para rellenar el valor. Por ejemplo, si el ancho es `5`, el carácter de relleno es `0` y el valor `1`, el resultado es `00001`.
- *`.dec`*: Número de espacios decimales a usarse para un número
- *`type`*: Tipo de valor. Use `s` para la mayoría de los valores. Use `f` para los números a los que desea formatear con espacios decimales.

Algunos enunciados `sprintf` posibles son

```
sprintf("Tengo $%03.2f. ¿Acaso %s tiene algo?",$dinero,$nombre);
sprintf("%'.-20s%3.2f",$producto,$precio);
```

El resultado de estos enunciados es

```
Tengo $030.00. ¿Acaso Tom tiene algo?
Gatito..... 30.00
```

Compare cadenas con patrones

En capítulos anteriores, uso expresiones regulares o patrones para hacer concordar cadenas. (Le explico las expresiones regulares en el Capítulo 6.) Las siguientes funciones usan expresiones regulares para encontrar y a veces reemplazar patrones en las cadenas:

- `ereg("patron",$nombrevar)`: Revisa si el `patron` se encuentra en `$nombrevar`. `eregi` es la misma función excepto que no pone atención a las mayúsculas y minúsculas.
- `ereg_replace("patron","cadena",$nombrevar)`: Busca el patrón en `$nombrevar` y lo reemplaza con la `cadena`. `eregi_replace` es la misma función excepto que no pone atención a las mayúsculas y minúsculas.

Averigüe sobre las cadenas

Ocasionalmente, usted necesita saber cosas sobre una cadena, tal como su longitud o si el primer carácter es una letra O mayúscula. PHP ofrece muchas funciones para revisar las cadenas:

- `strlen($nombrevar)`: Devuelve la longitud de la cadena.
- `strpos("cadena","subcadena")`: Devuelve la posición en `cadena` donde `subcadena` empieza. Por ejemplo, `strpos("hola","el")` devuelve `1`. Recuerde que la primera posición para PHP es 0. `strrpos()` encuentra la última posición en `cadena` donde empieza `subcadena`.
- `substr("cadena",n1,n2)`: Devuelve la subcadena de `cadena` que empieza en `n1` y cuenta con `n2` caracteres de longitud. Por ejemplo, `substr("hola",2,2)` indica `ll`.
- `strtr($nombrevar,"str1","str2")`: Busca `str` en la cadena `$varname` y lo reemplaza con `str2` cada vez que lo encuentra.
- `strrev($nombrevar)`: Devuelve la cadena con los caracteres invertidos.

Existen muchas, muchas otras funciones de cadenas. Vea la documentación en `www.php.net`.

Cambie las mayúsculas o minúsculas de las cadenas

Cambiar las letras mayúsculas a minúsculas y viceversa no es tan fácil. Agradezca a PHP por proporcionar funciones para hacerlo por usted:

- `strtolower($nombrevar)`: Cambia cualquier letra mayúscula en la cadena a minúscula.
- `strtoupper($nombrevar)`: Cambia cualquier letra minúscula en la cadena a mayúscula.
- `ucfirst($nombrevar)`: Cambia la primera letra de la cadena a mayúscula.
- `ucwords($nombrevar)`: Cambia la primera letra de cada palabra de la cadena a mayúscula.

Capítulo 14

Diez descuidos frecuentes en PHP

En este capítulo

- Reconocer errores comunes en PHP
- Interpretar los mensajes de error

Le garantizo que usted hará todas las cosas que yo menciono en este capítulo. No es posible escribir programas sin cometer estos errores. El truco es aprender a reconocerlos, tirar de sus cabellos y decir "Ay no, otra vez", y luego arreglarlos. Un mensaje de error que verá muchas veces es

```
Parse error: parse error in c:\test.php on line 7
```

Esta es la manera en que PHP dice "¿Perdón?" Significa que no entiende algo. Este mensaje amablemente señala el archivo y el número de la línea donde PHP se confundió. A veces señala el error directamente, pero otras veces la confusión de PHP es el resultado de un error anterior en el programa.

Faltan puntos y comas

Todos los enunciados en PHP terminan con un punto y coma (;). PHP no deja de leer un enunciado hasta no toparse con un punto y coma. Si usted omite el punto y coma al final de una línea, PHP continúa leyendo el enunciado en la siguiente línea. Por ejemplo, considere este enunciado:

```
$test = 1
echo $test;
```

Por supuesto, el enunciado no tiene sentido para PHP cuando lee las dos líneas como un solo enunciado, de modo que se queja con un mensaje de error, tal como el molesto

```
Parse error: parse error in c:\test.php on line 2
```

En poco tiempo, usted escribirá la dirección de su casa con puntos y comas al final de cada línea.

No hay suficientes signos de igual

Cuando usted pregunta si dos valores son iguales en un enunciado de comparación, necesita dos signos de igual (==). Usar un solo signo de igual es un error común. Es perfectamente razonable, pues usted ha usado un signo de igual para transmitir el significado de *igual* desde que estaba en primer grado, cuando descubrió que 2 + 2 = 4. Este es un error difícil de reconocer porque no causa un mensaje de error. Simplemente hace que su programa haga cosas extrañas, como ciclos infinitos o bloques if que nunca se ejecutan. Yo no dejo de asombrarme ante la cantidad de tiempo que puedo pasar observando

```
$test = 0;
while ( $test = 0 )
{
   $test++;
}
```

sin encontrar por qué es un ciclo sin fin.

Nombres de variables mal escritos

Este es otro descuido en PHP que no produce un mensaje de error, sólo comportamiento extraño del programa. Si usted escribe mal el nombre de una variable, PHP la considera una variable nueva, y hace lo que usted le pide que haga. Esta es otra astuta manera de escribir un ciclo infinito:

```
$test = 0;
while ( $test == 0 )
{
   $Test++;
}
```

Recuerde: Para PHP, `$test` no es la misma variable que `$Test`.

Faltan signos de dólar

Un signo de dólar faltante en el nombre de una variable es realmente difícil de detectar, pero al menos generalmente produce un mensaje de error, de modo que usted sabrá dónde buscar el problema. Usualmente da como resultado el ya conocido error parse:

```
Parse error: parse error in test.php on line 7
```

Comillas revoltosas

Usted puede tener demasiadas, muy pocas o el tipo equivocado de comillas. Se tienen demasiadas cuando se colocan comillas dentro de comillas, tal como

```
$test = "<table width="100%">";
```

PHP verá la segunda comilla doble (") — antes de 100 — como el final de las comillas dobles (") y leerá el 1 como una instrucción, lo cual no tiene sentido. *¡Voilá!* Otro error parse. La línea debe ser así

```
$test = "<table width='100%'>";
```

o así

```
$test = "<table width=\"100%\">";
```

Usted cuenta con muy pocas comillas cuando olvida terminar una cadena con las comillas, por ejemplo

```
$test = "<table width='100%'>;
```

PHP continuará leyendo las líneas como parte de la cadena entre comillas hasta encontrar otras comillas dobles ("); esto podría no suceder por varias líneas. Esta es una ocasión en la cual el error parse que señala el lugar donde PHP se confundió no señala el error mismo. El error real ocurrió algunas líneas antes, cuando usted olvidó terminar la cadena.

Se tiene el tipo equivocado de comillas cuando se usa una comilla simple (') cuando debió usarse una comilla doble (") o viceversa. La diferencia entre las comillas simples y las dobles es algo importante, y lo explico en el Capítulo 6.

Output invisible

Algunos enunciados, tales como el enunciado header, deben ejecutarse antes de que el programa produzca cualquier output. Si usted trata de usar este tipo de enunciados después de enviar el output, fallarán. Los enunciados siguientes fallarán porque el mensaje header no es el primer output:

```
<html>
<?php
   header("Location: http://empresa.com");
?>
```

<html> no está en una sección PHP y, por lo tanto, se envía como output HTML. Los siguientes enunciados funcionarán:

```
<?php
   header("Location: http://empresa.com");
?>
<html>
```

Los siguientes enunciados fallarán:

```
 <?php
   header("Location: http://empresa.com");
?>
<html>
```

porque hay un único espacio en blanco antes de la etiqueta PHP de apertura. El espacio en blanco es output para el explorador, aunque la página web resultante se vea vacía. Por lo tanto, el enunciado header falla porque hay output antes. Este es un error común, y es difícil de detectar.

Arreglos numerados

PHP cree que el primer valor en un arreglo es el número cero (0). Por supuesto, los humanos tienden a creer que las listas empiezan con el número uno (1). Esta forma fundamentalmente diferente de ver las listas provoca que nosotros los humanos creamos que un arreglo no está funcionando correctamente, cuando en realidad está perfectamente bien. Por ejemplo, considere los siguientes enunciados:

```
$test = 1;
while ( $test <= 3 )
{
   $serie[] = $test;
   $test++;
}
echo $serie[3];
```

No se despliega nada por medio de estos enunciados. Yo me precipito a la conclusión de que hay algo malo con mi ciclo. De hecho, mi ciclo está bien. Sólo da como resultado la siguiente serie:

```
$serie[0]=1
$serie[1]=2
$serie[2]=3
```

Y no establece nada en `$serie[3]`.

Incluir enunciados PHP

Cuando un archivo se inserta usando un enunciado `include` en una sección PHP, me parece razonable que los enunciados en el archivo sean tratados como enunciados PHP. Después de todo, PHP añade los enunciados al programa en el punto donde yo los incluyo. Sin embargo, PHP no lo toma de esa forma. Si un archivo llamado `file1.inc` contiene los siguientes enunciados:

```
if ( $test == 1 )
        echo "Hola";
```

y yo lo incluyo con los siguientes enunciados en mi programa principal:

```
<?php
$test = 1;
include ("file1.inc");
?>
```

es de esperar que la palabra `Hola` aparezca en la página web. Sin embargo, la página web en realidad despliega esto:

```
if ( $test == 1 ) echo "Hola";
```

Claramente, el archivo que se incluyó se ve como HTML. Para enviar `Hola` a la página web, `file1.inc` debe contener los siguientes enunciados:

```
<?php
if ( $test == 1 )
        echo "Hola";
?>
```

Parejas faltantes

Los paréntesis y llaves vienen en parejas y deben usarse así. Abrir con un (que no tiene un) de cierre o una { sin la otra } ocasionará un mensaje de error. Uno de mis favoritos es usar un paréntesis de cierre donde se necesitan dos, como en el enunciado siguiente:

```
if ( isset($test)
```

Este enunciado necesita un paréntesis de cierre al final. Es mucho más difícil detectar que uno de sus bloques no se cerró cuando hay un bloque dentro de otro. Por ejemplo, considere lo siguiente:

```
while ( $test < 3 )
{
if ( $test2 != "si" )
{
if ( $test3 > 4 )
{
echo "ir a";
}
}
```

Puede ver que hay tres llaves de apertura y sólo dos de cierre. Imagine que hay 100 líneas de código dentro de estos bloques. Puede ser difícil dar con el problema; especialmente si usted cree que la última llave de cierre está cerrando el ciclo `while`, pero PHP la toma como una que cierra el ciclo `if` para `$test2`. En algún punto más adelante en el programa, PHP podría usar una llave de cierre para cerrar el ciclo `while` que usted ni siquiera está viendo. Puede ser difícil rastrear el problema en un programa grande.

Usar sangría en los bloques facilita ver adónde pertenecen las llaves y los paréntesis de cierre. Además, yo uso comentarios a menudo para saber dónde estoy, tal como

```
while ( $test < 3 )
{
 if ( $test2 != "si" )
 {
  if ( $test3 > 4 )
  {
   echo "ir a";
  } // cierra el bloque if para $test3
 } // cierra el bloque if para $test2
} // cierra el bloque while
```

Paréntesis y corchetes confusos

No estoy segura si este es un problema común para todos o sólo para mí, porque me rehúso a admitir que mi vista no está tan bien como antes. Aunque PHP no tiene problemas para distinguir entre los paréntesis y las llaves, mis ojos no son tan confiables. Especialmente cuando miro la pantalla de un PC al final de una maratón de programación de diez horas, puedo confundir fácilmente (con {. Usar el símbolo equivocado ocasionará un mensaje de error parse.

Parte VI
Apéndices

La 5a Ola — Por Rich Tennant

En esta parte. . .

Esta parte le brinda las instrucciones para instalar MySQL y PHP. El Apéndice C presenta información sobre la instalación y configuración que podría serle útil si necesita instalar Apache.

Apéndice A

Instalación de MySQL

Aunque MySQL corre en muchas plataformas, le describo cómo instalarlo en Linux, Unix, Windows y Mac: juntos abarcan la mayoría de los sitios web en Internet. Asegúrese de leer las instrucciones completamente antes de empezar la instalación.

MySQL se puede instalar con más facilidad desde paquetes binarios: paquetes precompilados y listos para instalar. Los binarios están disponibles para la mayoría de los sistemas operativos: Linux, Windows, Mac, FreeBSD, muchas modalidades de Unix y otros. Si un paquete de este tipo está disponible para su sistema operativo, úselo. Sólo instale MySQL de la fuente si es completamente inevitable, como cuando no existe un binario para su sistema operativo o si usted necesita alguna funcionalidad que no está compilada en los binarios (por ejemplo, un conjunto de caracteres diferente).

Si tiene problemas para iniciar el servidor MySQL después de instalarlo, revise el registro de errores en busca de información útil. El registro de errores se localiza en el directorio de datos y tiene la extensión `.err`.

En Windows

En la mayoría de los casos, cuando usted descarga e instala MySQL, el servidor se inicia automáticamente. Si no sucede así, o si usted necesita detenerlo y reiniciarlo por otra razón, puede hacerlo manualmente usando la utilidad `WinMySQLadmin` que se instala con MySQL, tal y como lo describo en la sección venidera: "Iniciar el servidor MySQL". También puede usar WinMySQLadmin para configurar MySQL de modo que se inicie cada vez que se inicia su PC.

Descargar e instalar MySQL

Para instalar MySQL en Windows, siga estos pasos:

1. **Diríja su explorador web a `www.mysql.com`, la página de inicio de MySQL.**
2. **Haga clic en el vínculo del número de versión Production.**

   Busque la sección debajo del encabezado Database Server. Al momento de escribir esto, la versión de producción es 4.0.17.

   MySQL 4.0.x tiene soporte de PHP 4 o 5. El soporte para MySQL 4.1.x se brinda a partir de PHP 5.
3. **Desplácese por la pantalla hasta llegar al encabezado Windows Downloads.**
4. **Haga clic en el vínculo de descarga para el binario de Windows llamado `Windows 95/98/NT/2000/XP/2003`.** Este binario incluye un instalador.

   Se abre un recuadro de diálogo.
5. **Seleccione la opción para guardar el archivo.**

   Se abre un recuadro de diálogo que le permite seleccionar dónde desea guardar el archivo.
6. **Navegue hasta donde quiera guardar el archivo (por ejemplo, `c:\downloads`). Luego haga clic en Save.**

   Después de descargar, verá un archivo Zip en la ubicación de la descarga (por ejemplo, `c:\downloads`), el cual contiene los archivos de MySQL. El archivo se llama `mysql-`, seguido por el número de la versión y `-win.zip`: por ejemplo, `mysql-4.0.17-win.zip`.
7. **Use su utilidad Zip favorita para descomprimir los archivos y guardarlos en una ubicación temporal (por ejemplo, `c:\downloads\mysql`).**

   Dos utilidades Zip populares son PKZIP en `www.pkware.com` y WinZip en `www.winzip.com`.
8. **Navegue hasta el directorio temporal donde se guardan los archivos descomprimidos. Luego haga doble clic en `setup.exe`.**

   Nota: Si está instalando desde un sistema Windows NT/2000/XP, asegúrese de estar registrado a una cuenta con privilegios administrativos.

   La pantalla de apertura que se muestra en la Figura A-1 aparecerá.
9. **Haga clic en Next.**

   Se despliega la licencia.
10. **Haga clic en el botón I Agree para continuar.**

    Verá una pantalla que muestra el directorio donde se instalará MySQL.

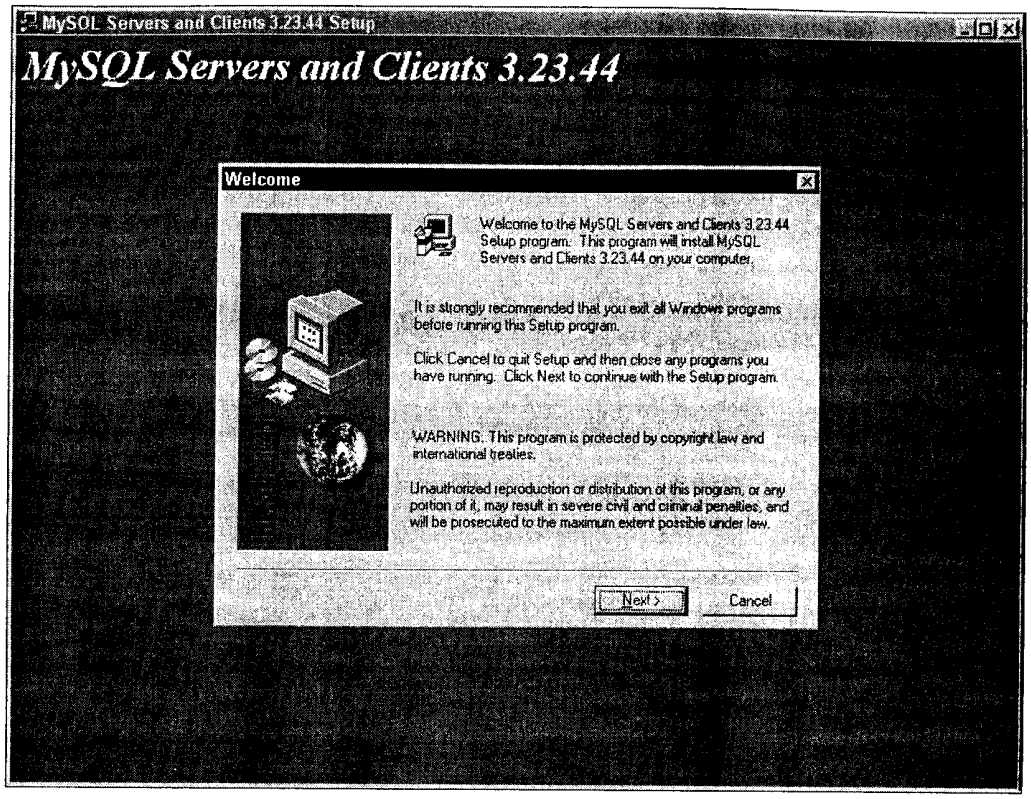


Figura A-1: La pantalla de apertura en el programa de configuración de MySQL.

11. **Si desea instalar MySQL en el directorio predeterminado, `c:\mysql`, haga clic en Next. Si desea instalar MySQL en un directorio diferente, haga clic en Browse, seleccione un directorio y haga clic en Aceptar; luego haga clic en Next.**

    Verá una pantalla en la cual puede escoger el tipo de instalación.

12. **Seleccione Typical y luego haga clic en Next.**

    La instalación de MySQL comienza. Una vez que esté completa, aparecerá un mensaje.

El servidor podría o no haberse iniciado durante la instalación. Si está corriendo, usted debería ver un semáforo en la bandeja de su sistema (en la parte inferior de la pantalla) con una luz verde encendida. Si no está corriendo, revise la próxima sección.

Iniciar el servidor MySQL

Puede iniciar y detener su servidor manualmente, aunque es más probable que usted desee que el servidor MySQL corra cada vez que su PC corra. Para ver cómo configurar MySQL de modo que se inicie cuando su PC se inicia, vea la siguiente sección.

En Windows 98/Me

Puede iniciar y detener su servidor con WinMySQLadmin, un programa que fue instalado con MySQL. Este programa es el responsable de mostrar el semáforo en la bandeja de su sistema. Si el semáforo aparece, WinMySQL está corriendo. Si el semáforo no está en su bandeja, usted debe iniciar WinMySQLadmin.

Si WinMySQLadmin no está corriendo, tal vez usted pueda iniciarlo desde su menú de inicio: Escoja Incio➪Programas➪Inicio➪WinMySQLadmin. Si no puede encontrarlo en su menú de inicio, use Windows Explorer para navegar hacia el directorio bin en el directorio donde está instalado MySQL (por ejemplo, `c:\mysql\bin`) y haga doble clic en WinMySQLadmin. Una vez iniciado, usted debería poder ver el semáforo en la bandeja del sistema.

Para iniciar o detener el servidor MySQL usando WinMySQLadmin, siga estos pasos:

1. **Haga clic derecho sobre el semáforo en la bandeja de su sistema.**

   Verá un menú corto.

2. **Seleccione su sistema operativo: Windows 9x.**
3. **Para iniciarlo o detenerlo, haga clic en Start the Server o Stop the Server, respectivamente.**
4. **Salga de WinMySQLadmin haciendo clic derecho en la ventana WinMySQLadmin, y luego haga clic en Hide Me.**

En Windows NT/2000/XP

Puede iniciar su servidor MySQL directamente al navegar hacia el subdirectorio bin en el directorio donde está instalado MySQL (tal como `c:\mysql\bin`), y luego haga doble clic en el archivo `mysqld.exe`.

El procedimiento anterior podría no funcionar si el programa WinMySQLadmin no está corriendo (si no hay un semáforo en la bandeja de su sistema). En este caso, puede iniciar el programa WinMySQL haciendo doble clic en él o iniciar el servidor MySQL con una consola DOS Window. Si tiene problemas, a menudo resulta útil iniciar el servidor con una consola porque algunos útiles mensajes de error se desplegarán.

Para iniciar el servidor MySQL con una consola, siga estos pasos:

1. **Abra una ventana command prompt.**

   Por ejemplo, escoja Inicio➪Programas➪Accesorios➪Linea de comandos.

2. **Cambie al directorio bin para MySQL.**

   Por ejemplo, `CD c:\mysql\bin`.

3. **Digite** mysqld — console.

Aparecerá un mensaje diciéndole que el servidor se ha iniciado. La ventana permanece abierta y lista para recibir cualquier otro mensaje del servidor. No cierre la ventana.

Tal vez usted pueda detener el servidor usando `WinMySQLadmin` si está corriendo al hacer clic derecho en el semáforo, seleccionar Win NT desde el menú que aparece, y luego hacer clic en Stop the Server. Si esto no funciona, puede usar los siguientes pasos:

1. **Abra una ventana command prompt.**

   Por ejemplo, escoja Inicio⇨Programas⇨Accesorios⇨Línea de comandos.

2. **Cambie al directorio bin para MySQL.**

   Por ejemplo, `CD c:\mysql\bin`.

3. **Digite** mysqladmin -u root –p shutdown.

   La `-u` será seguida del nombre de una cuenta: en este caso, root, la cual se instaló cuando MySQL fue instalado. Si usted usa la `-p`, se le pedirá una contraseña. A menos que haya instalado una contraseña, no necesitará una; por esa razón, deje la `-p` por fuera.

Configurar el servidor para que se inicie cuando se inicie el PC

En la mayoría de los casos, usted querrá configurar su servidor MySQL para que corra cuando su PC esté corriendo.

En Windows 98/Me

Si desea configurar su servidor MySQL de modo que se inicie cada vez que su PC lo haga, inicie `WinMySQLadmin` tal y como describo en la sección anterior y luego siga estos pasos:

1. **Haga clic en la pestaña my.ini Setup.**
2. **Haga clic en Crear accesos directos en el menú Inicio, en la esquina inferior izquierda de la pantalla.**
3. **Salga de WinMySQLadmin haciendo clic derecho en la ventana de WinMySQLadmin y luego escoja Hide Me.**

En Windows NT/2000/XP

Para configurar el servidor MySQL de modo que se inicie cada vez que el PC se inicie, configúrelo como un servicio. El procedimiento de instalación podría haberlo configurado como un servicio. Puede verificar si MySQL es un servicio de la siguiente manera:

1. **Abra la ventana de servicios en Inicio➪Configuración➪Panel de control➪Herramientas Administrativas➪Servicios.**

   Se abre una ventana que muestra una lista de todos los servicios.

2. **Desplácese por la lista para ver si MySQL aparece en ella.**

   Si no es así, no está configurado como un servicio. Vea cómo configurarlo como un servicio más adelante en esta sección.

3. **Seleccione Tipo de Inicio.**

   Debería decir Automatico. Si no es así, haga lo siguiente:

   1. Haga clic derecho en MySQL.
   2. Escoja Propiedades del menú que aparece.
   3. Seleccione Automatico de la lista desplegable Tipo de Inicio a la mitad de la ventana.
   4. Haga clic en Aceptar.

Si MySQL no apareció en la lista de servicios, puede configurarlo como un servicio así:

1. **Abra una ventana command prompt.**

   Por ejemplo, escoja Inicio➪Programas➪Accesorios➪Línea de Comandos.

2. **Cambie al directorio bin para MySQL.**

   Por ejemplo, `CD c:\mysql\bin`.

3. **Digite** mysqld — install.

   Aparece un mensaje que le dice que el servicio se instaló satisfactoriamente.

Si necesita remover el servidor MySQL de la lista de servicios, como cuando está a punto de actualizar a una versión más nueva, siga los pasos 1 y 2 anteriores. Para el paso 3, digite **mysqld — remove**.

En Linux/Unix

Usar RPM (sólo para Linux)

MySQL se puede instalar en Linux usando RPM. Aunque RPM quiere decir Red Hat Package Manager, RPM está disponible en la mayoría de las versiones de Linux, no sólo en Red Hat. Usar RPM es la forma más fácil de instalar en Linux. Si instalar desde un archivo RPM no funciona para usted, trate de usar un paquete listo para instalar, llamado binario, el cual también es fácil de instalar; las instrucciones de instalación se presentan en la sección "Desde archivos binarios". Si ninguno de es-

tos métodos funciona para usted, siempre puede instalar MySQL de los archivos fuente. Para hacerlo, siga las instrucciones en la sección "Desde archivos fuente".

Puede descargar el archivo RPM usando las siguientes instrucciones. Sin embargo, el archivo RPM podría ya estar en el CD donde venía su sistema operativo Linux. Instalar el archivo RPM desde un CD le ahorra la molestia de descargarlo (puede saltarse los Pasos 1 — 9 en la siguiente lista), pero si la versión de MySQL en su CD no es la más reciente, tal vez prefiera descargar un archivo RPM de todos modos.

Para instalar MySQL en Linux desde un archivo RPM, siga estos pasos:

1. **Dirija su explorador web a** `www.mysql.com`**, la página de inicio de MySQL.**
2. **Haga clic en el vínculo del número de versión de Production.**

   Fíjese en la lista del lado derecho de la pantalla, titulada MySQL Products. Busque la sección bajo el encabezado Database Server. Hasta el momento de escribir este libro, la versión de producción es 4.0.17.
3. **desplácese por la pantalla hasta encontrar el encabezado Linux x86 RPM Downloads.**

   Esta sección enumera varias descargas para Linux.
4. **Haga clic en el vínculo de descarga para el servidor estándar.** Debería ser la primera entrada.

   Aparece un recuadro de diálogo.
5. **Seleccione la opción para guardar el archivo.**

   Se abre un recuadro que le permite seleccionar dónde desea guardar el archivo.
6. **Navegue hacia donde desea guardar el RPM (por ejemplo,** `/usr/src/mysql`**). Luego haga clic en Save.**
7. **Repita los Pasos 5 — 7 para descargar el archivo RPM para Client Programs en la misma ubicación de descarga.**
8. **Cambie al directorio donde guardó la descarga.**

   Por ejemplo, digite `cd /usr/src/mysql`. Verá dos archivos en el directorio: uno llamado `MySQL-server-`, seguido por el número de la versión `e.i386.rpm`, y un segundo archivo de nombre muy parecido con `client` como parte de su nombre. Por ejemplo: `MySQL-4.0.15-0.i386.rpm` y `MySQL-client-4.0.15-0.i386.rpm`.
9. **Instale RPM digitando este comando:**

   ```
   rpm -i listofpackages
   ```

   Por ejemplo, usando la muestra en el Paso 10, el comando sería este:

   ```
   rpm -i MySQL-server-4.0.17-0.i386.rpm MySQL-client-
         4.0.17-0.i386.rpm
   ```

Este comando instala los paquetes de MySQL. Abre la cuenta MySQL y el nombre de grupo que usted necesita, y también crea el directorio de datos en `/var/lib/mysql`. Además, inicia el servidor MySQL y crea las entradas apropiadas en `/etc/rc.d` de modo que MySQL se inicie automáticamente cada vez que su PC se inicia.

Es necesario usar una cuenta que tenga permisos para correr el comando rpm exitosamente, tal como una cuenta root.

10. **Para comprobar que MySQL esté funcionando bien, digite esto:**

```
bin/mysqladmin --version
```

Debería ver el número de versión de su servidor MySQL.

Desde archivos binarios

Los archivos binarios compilados y listos para usar están disponibles para varias versiones de Linux y Unix. Si ninguna de las versiones funciona para su máquina Linux/Unix, puede instalar MySQL desde archivos fuente, pero es mejor usar un binario si es posible. Al momento de escribir este libro, los archivos binarios de MySQL estaban disponibles para las siguientes versiones de Unix, pero en cualquier momento podría haber más:

- Solaris
- HP-UX
- AIX
- SCO
- SGI Irix
- Dec OSF
- QNX
- BSDi
- OpenBSD
- FreeBSD

Para instalar una versión de archivo binario de MySQL en Linux o Unix, siga estos pasos:

1. **Dirija su explorador web a `www.mysql.com`, la página de inicio de MySQL.**
2. **Haga clic en el vínculo del número de versión de Production.**

   Fíjese en la lista del lado derecho de la pantalla, titulada MySQL Products. Busque la sección bajo el encabezado Database Server. Al momento de escribir esto, la versión de producción es 4.0.17.

3. **Desplácese por la pantalla hasta llegar al encabezado para Linux o para su versión de Unix (por ejemplo, Solaris Downloads).**

   Cada sección enumera varias descargas para ese sistema operativo.

4. **Localice el paquete correcto para su versión de sistema operativo.**

   Para Linux, probablemente desee la versión binaria Intel libc6. Para Unix, seleccione la versión correcta del sistema Unix: por ejemplo, Solaris 9 (SPARC, 64-bit).

5. **Haga clic en el enlace de descarga para la versión estándar de su sistema operativo.**

   Se abre un recuadro de diálogo.

6. **Seleccione la opción para guardar el archivo.**

   Se abre un recuadro que le permite seleccionar dónde desea guardar el archivo.

7. **Navegue hasta donde desea instalar MySQL. Luego haga clic en Save.**

   La ubicación estándar es `/usr/local`; es mejor usar esta ubicación si es posible.

8. **Una vez completa la descarga, cambie al directorio de descargas: por ejemplo,** `cd /usr/local`.

   Se ve un archivo llamado `mysql-`, seguido del número de la versión, el nombre del sistema operativo y `.tar.gz`; por ejemplo, `mysql-4.0.17-sun-solaris2.9-sparc.64bit-tar.gz`. Este archivo es un tarball.

9. **Cree una identificación de usuario y de grupo para que MySQL corra usando estos comandos:**

   ```
   groupadd mysql
   useradd -g mysql mysql
   ```

   La sintaxis para los comandos podría diferir un poco en distintas versiones de Unix, o podrían llamarse `adduser` y `addgroup`.

   Nota: Debe estar usando una cuenta que está autorizada para agregar usuarios y grupos.

10. **Desempaque el tarball digitando esto:**

    ```
    gunzip -c nombredelarchivo | tar -xvf -
    ```

    Por ejemplo:

    ```
    gunzip -c mysql-4.0.17-sun-solaris2.9-sparc-64bit.tar.gz | tar -xvf -
    ```

    Nota: Debe estar usando una cuenta que le permita crear archivos en `/usr/local`.

11. **Cree un vínculo hacia el directorio nuevo de modo que pueda referirse a él con un nombre más corto, en lugar de su nombre actual, difícil de digitar. Digite lo siguiente:**

```
ln -s newdirectoryname mysql
```

Por ejemplo:

```
ln -s mysql-4.0.17-sun-solaris2.9-sparc-64bit mysql
```

Ahora se puede referir al directorio como mysql, en vez de su nombre largo.

12. **Cambie al nuevo directorio digitando** `cd mysql`.

    Debería ver varios subdirectorios, incluyendo `/bin` y `/scripts`.

13. **Agregue la ruta al directorio bin (por ejemplo,** `/usr/local/mysql/bin`**) a la ruta de su sistema de modo que los programas MySQL se puedan acceder desde cualquier programa que necesite tener acceso a MySQL.**

    Debería hacer esto editando el archivo que configura las variables del sistema cuando su PC se inicia.

14. **Digite lo siguiente:**

    ```
    scripts/mysql_install_db
    ```

    Este comando corre un script que inicia su base de datos MySQL.

15. **Asegúrese de que la propiedad y membresía del grupo de sus directorios MySQL estén correctas. Establezca la propiedad con estos comandos:**

    ```
    chown -R root  /usr/local/mysql
    chown -R mysql /usr/local/mysql/data
    chgrp -R mysql /usr/local/mysql
    ```

    Estos comandos hacen que `root` sea el propietario de todos los directorios de MySQL, excepto los datos, y hacen que `mysql` sea el propietario de los datos. Todos los directorios MySQL pertenecen al grupo `mysql`.

16. **Configure su PC de manera que MySQL se inicie automáticamente cuando su máquina se inicia, copiando el archivo** `mysql.server` **desde** `/usr/local/mysql/support-files` **a la ubicación donde su sistema tiene sus archivos de inicio.**

17. **Para probar MySQL, puede iniciar su servidor manualmente, sin reiniciar su PC, digitando lo siguiente:**

    ```
    bin/safe_mysqld --user=mysql &
    ```

18. **Para comprobar que MySQL esté corriendo bien, digite**

    ```
    bin/mysqladmin --version
    ```

    Debería ver el número de versión de su servidor MySQL.

Desde archivos fuente

Antes de decidirse a instalar MySQL de archivos fuente, verifique si hay archivos binarios para su sistema operativo. Los archivos binarios MySQL son paquetes precompilados y listos para instalar para la instalación de MySQL. Los archivos binarios MySQL son muy convenientes y confiables.

Se instala MySQL al descargar los archivos fuente, compilarlos e instalar los programas compilados. Este proceso suena terriblemente técnico e intimidante, pero no lo es. Lea todos los pasos siguientes antes de empezar el procedimiento de instalación.

Para instalar MySQL del código fuente, siga estos pasos:

1. **Dirija su explorador web a `www.mysql.com`, la página de inicio de MySQL.**
2. **Haga clic en el vínculo del número de versión de Production.**

   Fíjese en la lista al lado derecho de la pantalla, llamada MySQL Products. Busque la sección bajo el encabezado Database Server. Hasta el momento de escribir esto, la versión de producción es 4.0.17.
3. **Desplácese hasta el final de la pantalla, al encabezado Source Downloads.**

   Esta sección enumera varias descargas.
4. **Localice la versión tarball y haga clic en el enlace de descarga junto a ella.**

   Se abre un recuadro de diálogo.
5. **Seleccione la opción para guardar el archivo.**

   Se abre un recuadro que le permite seleccionar dónde desea guardar el archivo.
6. **Navegue hasta donde desea instalar MySQL y luego haga clic en Save.**

   La ubicación estándar es `/usr/local`. Es mejor usar la ubicación estándar si es posible.
7. **Una vez completa la descarga, cambie al directorio de descargas: por ejemplo, `cd-/usr/local`.**

   Verá un archivo llamado `mysql-`, seguido del número de la versión y.`tar.gz`.; por ejemplo, `mysql-4.0.17.tar.gz`. Este archivo es un tarball.
8. **Cree una identificación de usuario y grupo para que MySQL corra usando los siguientes comandos:**

   ```
   groupadd mysql
   useradd -g mysql mysql
   ```

   La sintaxis para los comandos puede diferir ligeramente en distintas versiones de Unix, o podrían llamarse `adduser y addgroup`.

   Nota: Debe usar una cuenta que esté autorizada para agregar usuarios y grupos.

9. **Desempaque el tarball digitando**

   ```
   gunzip -c filename | tar -xvf -
   ```

   Por ejemplo:

   ```
   gunzip -c mysql-3.23.44.tar.gz | tar -xvf -
   ```

   Verá un directorio nuevo llamado `mysql-version`: por ejemplo, mysql-4.0.15.

   Debe estar usando una cuenta que tenga permiso para crear archivos en `/usr/local`.

10. **Cambie al directorio nuevo.**

    Por ejemplo, digite **cd mysql-4.0.17.**

11. **Digite lo siguiente:**

    ```
    ./configure --prefix=/usr/local/mysql
    ```

    Verá varias líneas de output. El output le dirá cuándo la configuración ha terminado. Esto podría tomar algún tiempo.

12. **Digite** make.

    Verá muchas líneas de output. El output le dirá cuándo make esté listo. `make` podría correr por algún tiempo.

13. **Digite** make install.

    `make install` terminará rápidamente.

    Nota: Tal vez necesite correr este comando como `root`.

14. **Digite lo siguiente:**

    ```
    scripts/mysql_install_db.
    ```

    Este comando corre un script que inicializa su base de datos MySQL.

15. **Cerciórese de que la propiedad y membresía del grupo de sus directorios MySQL sean correctas. Establezca la propiedad con estos comandos:**

    ```
    chown -R root  /usr/local/mysql
    chown -R mysql /usr/local/mysql/data
    chgrp -R mysql /usr/local/mysql
    ```

    Estos comandos hacen que `root` sea el propietario de todos los directorios de MySQL, excepto los datos, y hacen que `mysql` sea el propietario de los datos. Todos los directorios de MySQL pertenecen al grupo `mysql`.

16. **Configure su PC para que MySQL se inicie automáticamente cuando su máquina se inicie, copiando el archivo** `mysql.server` **desde** `/usr/local/mysql/support-files` **a la ubicación donde su sistema tiene sus archivos de inicio.**

17. **Para probar MySQL, puede iniciar su servidor manualmente, sin reiniciar su PC, digitando lo siguiente:**

```
bin/safe_mysqld --user=mysql &
```

18. **Para comprobar que MySQL esté corriendo bien, digite:**

```
bin/mysqladmin --version
```

Debería ver el número de versión de su servidor MySQL.

En Mac

Puede descargar MySQL usando un paquete binario Mac OS X 10.2 (Jaguar) PKG. Si su sistema operativo es OS X 10.1 o anterior, no podrá usar este paquete. Necesitará descargar un tarball e instalar MySQL desde el código fuente, tal y como describo en la sección anterior.

1. **Dirija su explorador web a `www.mysql.com`, la página de inicio de MySQL.**
2. **Haga clic en el vínculo del número de versión de Production.**

   Fíjese en la lista al lado derecho de la pantalla, titulada MySQL Products. Busque la sección bajo el encabezado Database Server. Hasta el momento de escribir esto, la versión de producción es 4.0.17
3. **Desplácese por la pantalla hasta encontrar la sección con el encabezado Mac OS Package Installer Downloads.**

   Esta sección enumera varias descargas.
4. **Localice la versión estándar y luego haga clic en el enlace de descarga junto a ella.**

   Se abre un recuadro de diálogo.
5. **Seleccione la opción para guardar el archivo.**

   Se abre un recuadro que le permite seleccionar dónde desea guardar el archivo.
6. **Navegue hasta donde desea instalar MySQL y luego haga clic en Save.**

   La ubicación estándar es `/usr/local`. Es mejor usar la ubicación estándar si es posible.
7. **Una vez completa la descarga, cambie al directorio de descargas: por ejemplo, `/usr/local`.**

   Verá un paquete llamado `mysql-standard`, seguido del número de la versión y dmg, tal como `mysql-standard-4.0.17.dmg`. Si el archivo descargado no tiene la `extension.dmg`, cambie el nombre de archivo para darle la `extension.dmg`.

8. **Cree una identificación de usuario y de grupo para que MySQL corra.** En la mayoría de las versiones Mac más nuevas, este usuario y grupo ya existen.

9. **Monte la imagen del disco al hacer doble clic en su icono en el Finder.**

10. **Haga doble clic en el icono del paquete para instalar MySQL PKG.**

    El instalador del paquete correrá e instalará el paquete. Instala MySQL en el directorio `/usr/local/mysql-`, seguido del número de versión. También instala un vínculo simbólico `/usr/local/mysql/` que señala hacia el directorio donde está instalado MySQL. Además, inicializa la base de datos corriendo el `script mysql_install_db`, que crea una cuenta en MySQL llamada root.

11. **Tal vez necesite cambiar el propietario del directorio `mysql`.**

    El directorio donde MySQL está instalado (por ejemplo, `/usr/local/mysql-4.0.17`) debería pertenecer a root. El directorio de datos, (tal como, `/usr/local/mysql-4.0.17/data`) debería ser propiedad de la cuenta `mysql`. Ambos directorios deberían pertenecer al grupo `mysql`. Si el usuario y el grupo no están correctos, cámbielos con los siguientes comandos:

    ```
    sudo chown -R root /usr/local/mysql-4.0.17
    sudo chown -R mysql /usr/local/mysql-4.0.17/data
    sudo chown -R root /usr/local/mysql-4.0.17/bin
    ```

12. **Inicie el servidor MySQL usando los siguientes comandos:**

    ```
    cd /usr/local/mysql
    sudo ./bin/mysqld_safe
    si es necesario, digite su contraseña
    Oprima Ctrl-Z
    bg
    Oprima Ctrl-D o digite exit
    ```

    Esto inicia el servidor manualmente, lo cual significa que usted debe iniciar el servidor MySQL cada vez que reinicia su PC. Para que su servidor se inicie cada vez que su PC lo haga, debe instalar MySQL Startup Item, el cual se incluye en la imagen del disco de instalación en un paquete de instalación separado. Para instalar Startup Item, haga doble clic en el icono MySQLStartupItem.pkg.

Para detener el servidor MySQL, cambie al subdirectorio bin en el directorio donde MySQL está instalado y digite

```
mysqladmin -u root -p shutdown
```

La `-p` provoca que `mysqladmin` le pida una contraseña. Si la cuenta no requiere de una contraseña, no incluya `-p`.

Configurar MySQL

MySQL lee un archivo de configuración cuando se inicia. Si usted usa lo predeterminado o un instalador, probablemente no necesite agregar nada al archivo de configuración. Sin embargo, si instala MySQL en una ubicación no estándar o si desea que las bases de datos se almacenen en un lugar diferente al predeterminado, quizás necesite editar el archivo de configuración.

El archivo se llama `my.ini` o `my.cnf`. Se localiza en su directorio del sistema (tal como WINNT) si usa Windows y en `/etc` en Linux/Unix/Mac. El archivo se ve más o menos así:

```
[mysqld]
basedir=D:/mysql4
datadir=D:/mysql4/data
#port=3306
```

La línea `basedir` le dice al servidor MySQL dónde está instalado MySQL. La línea `datadir` le indica al servidor dónde se localizan las bases de datos. El # al inicio de la última línea la convierte en inactiva. Usted podría eliminar el # y cambiar el número de puerto para decirle al servidor que busque consultas de bases de datos en un puerto diferente.

Apéndice B

Instalación de PHP

Aunque PHP corre en muchas plataformas, le describo cómo instalarlo en Unix/Linux/ Mac y Windows; esto incluye la mayoría de los sitios web en Internet. PHP corre con varios servidores web, pero estas instrucciones se concentran principalmente en Apache e Internet Information Servers (IIS), pues juntos manejan casi el 90 por ciento de los sitios web en Internet. Si usted necesita instrucciones para otros sistemas operativos o servidores web, consulte el sitio web de PHP (`www.php.net`).

Esta sección le brinda las instrucciones de instalación para PHP 5. Si está instalando una versión anterior, hay algunas diferencias; por eso, lea el archivo `install.txt` que se proporciona con al distribución de PHP.

Instalar PHP en Unix/Linux/Mac con Apache

En Unix/Linux

Puede instalar PHP como un módulo de Apache o como un intérprete independiente. Si está usando PHP como lenguaje de script en las páginas web para interactuar con una base de datos, instale PHP como un módulo de Apache. PHP es más rápido y más seguro como un módulo. En este libro no comento el uso de PHP como intérprete independiente.

Usted instala PHP al descargar los archivos fuente, compilarlos e instalar los programas compilados. Este proceso no es tan técnico e intimidante como suena. Proporciono instrucciones paso a paso en las secciones siguientes. Lea todos los pasos antes de empezar el procedimiento de instalación.

Sólo para usuarios de Linux: PHP para Linux está disponible en un RPM además de en archivos fuente. Podría estar en el formato RPM en su CD de distribución. Sin embargo, cuando instala PHP desde un RPM, usted no puede controlar las opciones con las que se instala PHP. Por ejemplo, necesita instalar PHP con el soporte para MySQL activado, pero RPM tal vez no tenga el soporte para MySQL

activado. MySQL es popular, por lo cual muchos RPMs tienen el soporte para él activado, pero eso está fuera de su control. Además, un RPM generalmente permite todas las opciones más populares, de manera que un RPM podría activar opciones que usted no necesita. En consecuencia, el modo más simple y eficiente de instalar PHP podría ser desde la fuente. Si está familiarizado con los RPMs, siéntase libre de buscar un RPM e instalarlo. Hay RPMs disponibles. No obstante, presento los pasos para la instalación desde el código fuente, no desde RPMs.

Antes de la instalación

Antes de iniciar la instalación de PHP, revise lo siguiente:

- **El módulo Apache mod_so está instalado.** Generalmente lo está. Para mostrar una lista de todos los módulos, digite lo siguiente:

  ```
  httpd -l
  ```

  Es probable que usted deba estar en el directorio donde httpd se localiza antes de que el comando funcione. El output generalmente muestra una larga lista de módulos. Para PHP, usted debe preocuparse únicamente por `mod_so`. Si mod_so no está cargado, Apache debe reinstalarse usando la `opcion enable-module=so`.

- **La utilidad apxs está instalada.** `apxs` se instala cuando se instala Apache. Usted debería poder encontrar un archivo llamado `apxs`. Si Apache estuviera instalado en Linux desde un RPM, apxs tal vez no esté instalado. Algunos RPMs para Apache constan de dos RPMs: uno para el servidor Apache básico y otro para las herramientas de desarrollo de Apache. Posiblemente, el RPM con las herramientas de desarrollo, el cual instala `apxs`, debe instalarse.

- **La versión de Apache es reciente.** Consulte en el Apéndice C la información sobre las versiones de Apache.

  Para verificar la versión, digite lo siguiente:

  ```
  httpd --v
  ```

  Es probable que usted deba estar en el directorio donde se localiza httpd antes de que el comando funcione.

Instalación

Para instalar PHP en Unix/Linux con el servidor Web Apache, siga estos pasos:

1. **Dirija su explorador web hacia www.php.net, la página de inicio de PHP.**
2. **Haga clic en Downloads.**
3. **Haga clic en la última versión del código fuente de PHP, la cual es la versión 5.0.0 al momento de escribir esto.**

   El archivo que está a punto de descargar contiene muchos archivos comprimidos en un solo archivo: un tarball.

Se abre un recuadro de diálogo.

4. **Seleccione la opción para guardar el archivo.**

   Se abre un recuadro de diálogo que le permite seleccionar dónde se guardará el archivo.

5. **Navegue hasta donde desea guardar el código fuente (por ejemplo, `/usr/src`). Luego haga clic en Save.**

6. **Después de la descarga, cambie al directorio de descargas (por ejemplo, `cd-/usr/src`).**

   Verá un archivo llamado `php-`, seguido del nombre de versión y `tar.gz`.

7. **Desempaque el tarball. El comando para la versión5.0.0 de PHP es**

   ```
   gunzip -c php-5.0.0.tar.gz | tar -xf -
   ```

   Se crea un directorio nuevo llamado `php-5.0.0` con varios subdirectorios.

8. **Cambie al nuevo directorio que fue creado cuando desempacó el tarball. Por ejemplo:**

   ```
   cd php-5.0.0
   ```

9. **Digite el comando `configure`.**

   Use uno de los dos comandos `configure` siguientes:

   ```
   ./configure --with-mysql=DIR --with-apxs
   ./configure --with-mysqli=DIR --with-apxs
   ```

   Use `mysql` si está usando MySQL 4.0 o una versión anterior; use `mysqli` si está usando MySQL 4.1 o posterior. DIR es la ruta hacia el directorio MySQL apropiado. Al usar `with-mysql`, use la ruta hacia el directorio donde está instalado `mysql`, por ejemplo:

   ```
   --with-mysql=/user/local/mysql
   ```

   Al usar `with-mysqli`, use la ruta hacia el archivo llamado `mysql_config`.

   Si está usando Apache 2, use la opción `with-apxs2`. (Consulte en el Apéndice C la información sobre cómo usar Apache 2.)

   Verá muchas líneas de output. Espere hasta que el comando configure se haya completado. Esto podría tardar algunos minutos. Si el comando configure falla, aparecerá un mensaje informativo. Generalmente, el problema se debe a la falta de software. Verá un mensaje de error que le indica que el software xyz software no se puede encontrar o que la versión 5.6 de xyz se requiere, pero se encontró la versión 4.2 de xyz. Usted necesita instalar o actualizar el software que PHP necesita.

   Si la utilidad `apxs` no está instalada en el lugar esperado, verá un mensaje de error que indica que apxs no pudo encontrarse. Si obtiene este mensaje, verifique la ubicación donde `apxs` está instalada (`find / -name apxs`) e incluya la ruta en la opción `with-apxs` del comando `configure`: `--with-apxs=/usr/sbin/apxs` o `/usr/local/apache/bin/apxs`. Si está usando Apache 2, la opción es `--with-apxs2=/usr/sbin/apxs`.

10. **Digite** make.

    Verá muchas líneas de output. Espere hasta que haya terminado. Esto podría tardar algunos minutos.

11. **Digite** make install.

En Mac OS X

Con el lanzamiento de PHP 4.3, usted puede instalar PHP en Mac OS X tan fácilmente como en Unix/Linux. Se instala al descargar los archivos fuente, compilarlos e instalar los programas compilados. Este proceso no es tan técnico ni intimidante como suena. Le proporciono instrucciones paso a paso en las secciones siguientes. Lea todos los pasos antes de empezar el procedimiento de instalación para asegurarse de entender todo claramente y tener todo listo, de manera que no sea necesario detenerse en medio de la instalación.

Antes de la instalación

Si quiere usar PHP con Apache para su sitio web, Apache debe estar instalado. La mayoría de los sistemas Mac OS X vienen con Apache ya instalado. Para más información sobre Apache, consulte el Apéndice C.

Antes de iniciar la instalación de PHP, revise lo siguiente:

- **La versión de Apache es reciente.** Consulte en el Apéndice C la información sobre las versiones de Apache. Para verificar la versión, digite lo siguiente:

  ```
  httpd --v
  ```

  Es probable que usted deba estar en el directorio donde se localiza httpd antes de que el comando funcione

  Hasta el momento de escribir este libro, PHP con Apache 2 todavía se consideraba experimental. Para uso en la producción de sitios web, tal vez sea mejor usar Apache 1.3 en lugar de Apache 2. Consulte en el Apéndice C la discusión sobre las versiones de Apache. Manténgase al día sobre la condición de PHP con Apache 2 revisando el sitio web de PHP en `www.php.net/manual/en/install.apache2.php`.

- **El módulo Apache mod_so está instalado.** Generalmente lo está. Para mostrar una lista de todos los módulos, digite lo siguiente:

  ```
  httpd -l.
  ```

  Es probable que usted deba estar en el directorio donde `httpd` se localiza antes de que el comando funcione. El output generalmente muestra una larga lista de módulos. Para PHP, usted sólo debe preocuparse por `mod_so`. Si `mod_so` no está cargado, Apache debe reinstalarse.

- **La utilidad apxs está instalada.** `apxs` normalmente se instala cuando se instala Apache. Para determinar si está instalado en su PC, debe buscar un

archivo llamado `apxs`, generalmente en el directorio `/usr/sbin/apxs`. Si encuentra el archivo, `apxs` está instalada. Si no lo encuentra, no lo está.

- **Los archivos del CD Developer's Tools están instalados.** Este CD es complementario a la distribución principal de Mac OS X. Si usted no puede hallar el CD, puede descargar las herramientas del sitio web Apple Developer Connection en `developer.apple.com/tools/macosxtools.html`.

Instalación

Para instalar PHP en Mac, siga estos pasos:

1. **Dirija su explorador web hacia `www.php.net`, la página de inicio de PHP.**
2. **Haga clic en Downloads.**
3. **Haga clic en la última versión del código fuente de PHP, la cual es la versión 5.0.0 al momento de escribir esto.**

   Se abre un recuadro de diálogo.
4. **Seleccione la opción para guardar el archivo.**

   Se abre un recuadro de diálogo que le permite seleccionar dónde se guardará el archivo.
5. **Navegue hasta donde desea guardar el código fuente (por ejemplo, `/usr/src`). Luego haga clic en Save.**
6. **Después de descargarlo, cambie al directorio de descargas (por ejemplo, `cd-/usr/src`).**

   Verá un archivo llamado `php-`, seguido del nombre de versión y tar.gz. Este archivo contiene varios archivos comprimidos en un solo archivo. El archivo probablemente haya sido desempacado automáticamente por StuffIt Expandir, por lo cual podría ver el directorio `php-5.0.0`. Si es así, avance hasta el Paso 8.
7. **Desempaque el tarball.**

   El comando para hacerlo en la versión 5.0.0 de PHP es

   ```
   tar xvfz php-5.0.0.tar.gz
   ```

   Se crea un directorio nuevo llamado `php-5.0.0`, con varios subdirectorios.
8. **Cambie al nuevo directorio que fue creado cuando desempacó el tarball.**

   Por ejemplo, puede usar un comando como el siguiente:

   ```
   cd php-5.0.0
   ```
9. **Digite el comando `configure`:**

   El comando `configure` consiste en./`configure` seguido de todas las opciones necesarias. El conjunto mínimo de opciones es este:

   - **Opciones de ubicación:** Como Mac almacena los archivos en lugares diferentes a las ubicaciones predeterminadas de PHP, usted necesita decirle a PHP adónde se localizan los archivos. Use las siguientes opciones:

```
--prefix=/usr
--sysconfdir=/etc
--localstatedir=/var
--mandir=/usr/share/man
```

- Opcion zlib: `--with-zlib`
- **Opción Apache:** Si está instalando PHP para usar con Apache, use la siguiente opción: `--with-apxs o --with-apxs2.`

Por lo tanto, el comando de configuración que usted más probablemente debería usar es

```
./configure --prefix=/usr --sysconfdir=/etc
    --localstatedir=/var --mandir=/usr/share/man
     --with-apxs --with-zlib
```

También necesita usar una opción para incluir el soporte para MySQL. Use una de las opciones siguientes:

```
--with-mysql=DIR
--with-mysqli=DIR
```

Use `mysql` si está usando MySQL 4.0 o una versión anterior; use `mysqli` si está usando MySQL 4.1 o posterior. `DIR` es la ruta hacia el directorio MySQL apropiado. Cuando usa `with-mysql`, use la ruta hacia el directorio donde está instalado `mysql`, como sigue:

```
--with-mysql=/user/local/mysql
```

Puede digitar el comando configure en una línea. Si usa más de una línea, digite a \ al final de cada una.

Verá muchas líneas de output. Espere hasta que el comando configure se haya completado. Esto podría tomar algunos minutos.

Si la utilidad `apxs` no está instalada en la ubicación esperada, verá un mensaje de error, el cual indica que apxs no se pudo encontrar. Si recibe este mensaje, verifique la ubicación donde apxs está instalada (`find / -name apxs`) e incluya la ruta en la opción `with-apxs` del comando configure: `--with-apxs=/usr/sbin/apxs.`

Quizás deba usar muchas otras opciones, tales como las opciones para la base de datos que está usando o las opciones que cambian los directorios donde PHP está instalado. Estas `opciones` configure se comentan en la sección "Opciones de instalación", más adelante en este Apéndice.

10. **Digite** make.

    Verá muchas líneas de output. Espere hasta que haya terminado. Esto podría tomar algunos minutos.

11. **Digite** make install

Opciones de instalación

Las secciones anteriores le dan los pasos para instalar PHP rápidamente con las opciones necesarias para las aplicaciones en este libro. Sin embargo, tal vez usted prefiera instalar PHP en forma diferente. Por ejemplo, todos los programas y archivos PHP se instalan en sus ubicaciones predeterminadas, pero quizás usted necesite instalar PHP en lugares distintos. O tal vez esté planeando aplicaciones que necesiten software adicional. Puede usar opciones de línea de comandos adicionales si debe configurar PHP para sus necesidades específicas. Sólo agregue las opciones al comando mostrado en el Paso 13 de las instrucciones de instalación para Unix/Linux o el Paso 9 de las instrucciones de instalación para Mac. En general, el orden de las opciones en la línea del comandos no importa. La Tabla B-1 muestra las opciones usadas más comúnmente para PHP. Para ver una lista de todas las opciones posibles, digite./`configure --help`.

Tabla B-1 Opciones de configuración para PHP

Opción	*Le dice a PHP que*
`prefix=`*`PREFIX`*	Establezca el directorio principal de PHP en `PREFIX`. El `PREFIX` predeterminado es `/usr/local`.
`exec-prefix=`*`EPREFIX`*	Instale los archivos dependientes de la arquitectura en `EPREFIX`. El `EPREFIX` predeterminado es `PREFIX`.
`bindir=`*`DIR`*	Instale los ejecutables por el usuario en `DIR`. El predeterminado es `EPREFIX/bin`.
`infodir=`*`DIR`*	Instale la documentación de información en `DIR`. El predeterminado es `PREFIX/info`.
`mandir=`*`DIR`*	Instale los archivos man en `DIR`. El predeterminado es `PREFIX/man`.
`with-config-file-path=`*`DIR`*	Busque el archivo de configuración (php.ini) en DIR. Sin esta opción, PHP busca el archivo de configuración en una ubicación predeterminada, generalmente `/usr/local/lib`.
`disable-libxml`	Desactive el soporte XML que se incluye de manera predeterminada.
`enable-ftp`	Permita el soporte FTP.
`enable-magic-quotes`	Active el escape automático de las comillas con una diagonal inversa.

(continúa)

Tabla B-1 *(continuación)*

Opción	*Le dice a PHP que*
`with-apxs=FILE`	Construya un módulo Apache compartido usando la utilidad apxs ubicada en `FILE`. El `FILE` predeterminado es apxs.
`with-apxs2=FILE`	Construya un módulo Apache 2 compartido usando la utilidad apxs ubicada en `FILE`. El `FILE` predeterminado es apxs.
`with-mysql=DIR`	Active el soporte para MySQL 4.0 o bases de datos anteriores. El `DIR` predeterminado donde se ubica MySQL es `/usr/local`.
`with-mysqli=DIR`	Active el soporte para MySQL 4.1 o bases de datos posteriores. DIR debe ser la ruta hacia el archivo llamado `mysql_config` que se instaló con 4.1.
`with-openssl=DIR`	Active el soporte para OpenSSL para un servidor seguro. Requiere de una versión OpenSSL 0.9.5 o posterior.
`with-oci8=DIR`	Active el soporte para Oracle 7 o posterior. El DIR predeterminado se halla dentro de la variable ambiental, ORACLE_HOME.
`with-oracle=DIR`	Active el soporte para versiones anteriores de Oracle. El `DIR` predeterminado está en la variable ambiental, `ORACLE_HOME`.
`with-pgsql=DIR`	Active el soporte para las bases de datos PostgreSQL. El `DIR` predeterminado donde se localiza PostgreSQL es `/usr/local/pgsql`.
`with-servlet=DIR`	Incluya soporte servlet. DIR es el directorio de instalación base para JSDK. La extensión Java debe construirse como un.dll compartido.

Configurar Apache para PHP

Usted debe configurar Apache para que reconozca y corra archivos PHP. Hay un archivo de configuración Apache, `httpd.conf`, en su sistema, probablemente en `/etc` o en `/usr/local/apache/conf`. Debe editar este archivo antes de que PHP pueda funcionar adecuadamente.

Siga estos pasos para configurar su sistema para PHP:

1. **Abra el archivo** `httpd.conf` **de modo que pueda hacer cambios.**
2. **Configure Apache para cargar el módulo PHP.**

   Encuentre la lista de instrucciones `LoadModule`. Busque la siguiente línea:

   ```
   LoadModule php5_module libexec/libphp5.so.
   ```

   Si esta línea no está ahí, agréguela. Si hay un signo de numeral (#) al inicio de la línea, quítelo.
3. **Configure Apache para que reconozca las extensiones PHP.**

   Debe decirle Apache cuáles archivos podrían contener código PHP. Busque una sección que describa `AddType`. Tal vez vea una o más líneas `AddType` para otro software. Busque la línea `AddType` para PHP, como sigue:

   ```
   AddType application/x-httpd-php.php
   ```

   Si la encuentra con un signo de número (#) al inicio de la línea, elimine dicho signo. Si no encuentra esta línea, agréguela a los enunciados `AddType`. Esta línea le dice a Apache que busque código PHP en todos los archivos con la extensión.php. Usted puede especificar cualquier extensión o serie de extensiones.
4. **Inicie (si no está corriendo) o reinicie (si ya está corriendo) el servidor Apache httpd.**

   Puede iniciar o reiniciar el servidor usando un script que fue instalado en su sistema durante la instalación. Este script podría ser apachectl o httpd.apache, y podría estar localizado en `/bin o /usr/local/apache/bin`. Por ejemplo, usted podría iniciar el servidor digitando `apachectl start`, reiniciarlo usando `apachectl restart` o detenerlo usando `apachectl stop`. A veces, reiniciar no es suficiente; debe detener el servidor antes, y luego iniciarlo.

En Windows

PHP corre en Windows 98/Me y Windows NT/2000/XP. No corre en Windows 3.1. A partir de PHP 4.3.0, tampoco se da soporte a Windows 95.

Para instalar PHP 5 en Windows con soporte para MySQL, debe descargar un archivo Zip que contiene todos los archivos necesarios para PHP. Los siguientes pasos le muestran cómo instalar PHP en Windows:

1. **Dirija su explorador web hacia** `www.php.net`.
2. **Haga clic en Download.**
3. **Vaya a la sección Windows Binaries. Haga clic en el vínculo para descargar el paquete Sip para la versión más reciente de PHP (hasta el momento de escribir esto, 5.0.0).**
4. **Haga clic en el enlace a un sitio web espejo desde el cual podrá descargar el archivo y escoger el sitio más cercano a su ubicación.**

Se abre un recuadro de diálogo.

5. **Seleccione la opción para guardar el archivo.**

   Se abre un recuadro de diálogo que le permite seleccionar dónde se guardará el archivo.

6. **Navegue hasta donde desea que el archivo se descargue. Esta debería ser una ubicación temporal, tal como un directorio para descargas. Luego haga clic en Save.**

   Una vez completa la descarga, verá un archivo en la ubicación para descargas, el cual contiene todos los archivos necesarios. El nombre del archivo es `php`, seguido del número de versión y `win32.zip`. Para la versión actual, el archivo se llama `php5.0.0-Win32.zip`.

7. **Extraiga los archivos del archivo.`zip` hacia el directorio donde desea instalar PHP, tal como `c:\php`.**

   Si hace doble clic en el archivo.`zip`, debería abrir el software de su PC que extrae los archivos de archivos.`zip`, tal como WinZip o PKZIP. Seleccione el artículo del menú para extraer y seleccione el directorio hacia donde se extraerán los archivos. `C:\php` es una buena opción para la instalación, porque muchos archivos de configuración asumirán que ahí es donde PHP está instalado; por eso, resulta mucho más probable que las configuraciones predeterminadas estén correctas.

   Es mejor no instalar PHP en un directorio con un espacio en la ruta, tal como Program Files/PHP. A veces esto causa problemas.

   Ahora usted tiene un directorio con varios subdirectorios que contienen los archivos que necesita.

8. **Copie el archivo necesario para MySQL hacia el directorio principal de PHP.**

   El archivo se localiza en el subdirectorio `ext` en el directorio, donde PHP está instalado. Copie uno de los archivos siguientes, dependiendo de cuál versión de MySQL esté usando:

```
ext\php_mysql.dll   (para MySQL 4.0 o anterior)
ext\php_mysqli.dll  (para MySQL 4.1 o posterior)
```

   Copie los dos archivos en el directorio principal de PHP, como `c:\php`.

   Se requiere de otro archivo para darle soporte a MySQL, el cual se llama `libmysql.dll`. Este archivo ya debería estar ubicado en el directorio principal de PHP. Si no está ahí, debe encontrarlo y copiarlo ahí. Si no está en su directorio PHP, usualmente está instalado con MySQL: encuéntrelo en el directorio donde MySQL se instaló, tal vez un subdirectorio bin, como `c:\mysql\bin`.

   Ocasionalmente, PHP necesita archivos DLL que no puede hallar. Cuando esto sucede, PHP despliega un mensaje de error cuando usted corre el programa PHP, el cual le dice que no puede encontrar un DLL en particular. Generalmente puede hallar el DLL en el subdirectorio `ext` y copiarlo al directorio principal de PHP.

9. **Configure su servidor web.**

   La siguiente sección le brinda las instrucciones para configurar su servidor web.

10. **Configure PHP.**

    Siga las instrucciones en la sección que se encuentra más adelante en este capítulo.

Configurar su servidor web para PHP

Su servidor web necesita estar configurado para reconocer los scripts en PHP y correrlos. Siga los pasos en la sección para su servidor web:

Configurar Apache

Debe editar un archivo de configuración de Apache, llamado `httpd.conf`, antes de que PHP pueda correr adecuadamente. Para configurar Apache para PHP, siga estos pasos:

1. **Abra `httpd.conf` para editarlo.**

   Podrá editarlo escogiendo Start⇨Programs⇨Apache HTTPD Server⇨ Configure Apache Server⇨Edit Configuration.

   Si Edit Configuration no está en su menú de Inicio, encuentre el archivo `httpd.conf` en su disco duro, generalmente en el directorio donde Apache está instalado, en un subdirectorio `conf` (por ejemplo, `c:\program files\Apache group\Apache\conf`). Abra este archivo en un editor, como Notepad o WordPad.

2. **Establezca un apodo para el directorio donde PHP está instalado.**

   Un enunciado `ScriptAlias` se usa para establecer un nombre para el directorio donde PHP está instalado. Busque enunciados `ScriptAlias` en el archivo `httpd.conf`. Quizás vea algunos para otro software. Si no ve uno para PHP, añada lo siguiente:

   ```
   ScriptAlias /php/ "c:/php/"
   ```

   El primer argumento es el nombre, y el segundo es lo que representa. En este enunciado, el nombre `/php/` se usa para significar `c:/php/`. Observe que Apache prefiere las diagonales. Observe también que la ruta del directorio con un carácter especial (los dos puntos) está encerrada entre comillas dobles.

3. **Configure Apache para que corra PHP cuando encuentre un archivo que sea un programa PHP.**

Un enunciado `Action` se usa para decirle a Apache que corra PHP al toparse con un archivo que sea un programa PHP. Si usted no halla el enunciado `Action` para PHP, agregue lo siguiente:

```
Action application/x-httpd-php /php/php-cgi.exe
```

Antes de PHP 5, el intérprete de PHP era `named php.exe`. Si usted está instalando una versión anterior de PHP, la línea debería terminar con `php.exe`, en lugar de `php-cgi.exe`. Si encuentra una línea action para PHP en su archivo `httpd.conf`, asegúrese de que use el nombre del intérprete de PHP correcto para la versión de PHP que está instalando.

Observe que el enunciado Action usa el nombre definido en el enunciado `ScriptAlias`. Localiza a `php-cgi.exe` en `/php/`, lo cual significa `c:/php/`. Si usted cambia el enunciado ScriptAlias para que diga `c:/php27/`, el enunciado `Action` entonces buscaría `php-cgi.exe in c:/php27`.

4. **Debe decirle a Apache cuáles archivos son programas PHP.**

   Busque una sección que describa `AddType`. Esta sección podría contener una o más líneas `AddType` para otro software. La línea `AddType` para PHP es

   ```
   AddType application/x-httpd-php.php
   ```

   Busque esta línea. Si la encuentra con un signo de libras (#) al inicio de la línea, elimine el signo. Si no halla la línea, añádala a la lista de enunciados `AddType`. Puede específica cualquier extensión o serie de extensiones.

   Esta línea le indica a Apache que los archivos con la extensión.`php` son archivos del tipo `application/x-httpd-php`. Apache entonces se fija en el enunciado `Action` del Paso 2 y sabe que los archivos de este tipo deberían correrse usando el programa `/php/php.exe`.

5. **Inicie (si no está corriendo) o reinicie (si ya está corriendo) Apache.**

   Puede iniciarlo como un servicio en Windows NT/2000 escogiendo Start➪Programs➪Apache HTTPD Server➪Control Apache Server, y luego seleccione Start o Restart.

   O bien, puede iniciarlo en Windows 98/ME escogiendo Start➪Programs➪Apache Web Server➪Management.

   En ocasiones, reiniciar Apache no es suficiente; debe detenerlo antes y luego iniciarlo. Además, su PC indudablemente está configurado para que Apache se inicie cuando su PC lo haga. Por lo tanto, usted puede apagar y luego iniciar su PC para reiniciar Apache.

Configurar IIS

Para configurar IIS de modo que funcione con PHP, siga estos pasos:

1. **Ingrese a IIS Management Console.**

Debería poder entrar escogiendo Start➪Programs➪Administrative Tools➪Internet Services Manager o Start➪Settings➪Control Panel ➪Administrative Tools➪Internet Services Manager.

2. **Haga clic derecho en su sitio web (como en Default Web Site).**
3. **Seleccione Properties.**
4. **Seleccione la pestaña Home Directory.**
5. **Haga clic en el botón Configuration.**
6. **Escoja la pestaña App Mappings.**
7. **Haga clic en Add.**
8. **En la casilla Executable, digite la ruta hacia el intérprete de PHP: por ejemplo, digite** c:\php\php-cgi.exe.
9. **En la casilla Extension, digite**.php.

   Esta será la extensión que se asociará con los scripts PHP.
10. **Seleccione la casilla Script Engine.**
11. **Haga clic en Aceptar.**

    Repita los Pasos 6 — 10 si desea alguna extensión adicional, además de.php, para que PHP la procese, como por ejemplo.phtml.

Configurar PHP

PHP usa configuraciones en un archivo llamado `php.ini` para controlar parte de su comportamiento. PHP busca `php.ini` cuando se inicia y usa las configuraciones que encuentra. Si PHP no puede encontrar el archivo, usa un conjunto de configuraciones predeterminadas.

La ubicación predeterminada para el archivo php.ini es una de las siguientes, a menos que usted la cambie durante la instalación:

- **Windows:** El directorio del sistema, dependiendo de la versión de Windows, como sigue:
  - Windows 98/Me/XP: `windows`
  - Windows NT/2000: `winnt`
- **Unix/Linux/Mac:** `/usr/local/lib`

Si el archivo `php.ini` no se instala durante la instalación, usted debe instalarlo ahora. Un archivo de configuración con los controles predeterminados, llamado `php.ini-dist`, se incluye en la distribución de PHP. Copie este archivo a la ubicación apropiada, tal como las ubicaciones predeterminadas mencionadas anteriormente, y cambie su nombre a `php.ini`.

Si tiene una versión anterior de PHP instalada (tal como PHP 4.3), haga una copia de respaldo del archivo php.ini antes de sobrescribirlo con el archivo `php.ini` para PHP 5. Así podrá ver las configuraciones que está usando actualmente y cambiarlas en el nuevo archivo php.ini para que concuerden con las configuraciones actuales.

Para configurar PHP, siga estos pasos.

1. **Abra el archivo `php.ini` para editarlo.**
2. **Active el soporte para `mysql` o `mysqli`.**

   Busque una lista de extensiones. Encuentre la línea para `mysql` (si está usando MySQL 4.0 o anterior) o para `mysqli` (si está usando MySQL 4.1 o posterior), como sigue:

   ```
   ;extension=php_mysql.dll
   ;extension=php_mysqli.dll
   ```

   Observe el punto y coma (;) al inicio de las líneas. Para activar la extensión, remueva el punto y coma. Si la línea de extensión no está en su archivo `php.ini`, agréguela.

3. **Sólo si está usando PHP con el servidor web IIS, desactive force redirect. Encuentre la línea:**

   ```
   ;cgi.force_redirect = 1
   ```

   Debe eliminar el punto y coma de modo que las configuraciones estén activas, y también debe cambiar el 1 por un 0. Después de los cambios, la línea lucirá así:

   ```
   cgi.force_redirect = 0
   ```

4. **Guarde el archivo `php.ini`.**
5. **Tal vez deba reiniciar el servidor Apache antes de que las nuevas configuraciones se hagan efectivas.**

En general, las configuraciones predeterminadas restantes permiten que PHP corra bien, pero tal vez usted necesite editar algunas de ellas por razones específicas. Comento las configuraciones en el archivo `php.ini` a lo largo del libro cuando discuto un tema que requiera que usted cambie las configuraciones. Por ejemplo, las acciones de manejo de errores de PHP se pueden cambiar mediante configuraciones en el archivo `php.ini`. Las configuraciones posibles para el manejo de errores y sus efectos se discuten en el Capítulo 4.

Apéndice C

Instalación y configuración de Apache

Apache es un servidor web de fuente abierta. Un servidor web entrega los archivos en el sitio web al visitante que desea ver las páginas web.

En la mayoría de los casos, es mejor conseguir Apache en la Web e instalarlo usted mismo. Aunque Apache tal vez ya esté instalado en su PC (porque la mayoría de las distribuciones de Linux y Mac OS X lo incluyen), es probable que no sea la última versión. O bien, Apache quizás no haya sido instalado con las opciones que usted necesita. Debido a esto, a menudo es mejor instalar Apache usted mismo.

Seleccionar una versión de Apache

Apache está disponible actualmente en dos versiones: Apache 1.3 y Apache 2. Apache 2 es la versión más nueva, lanzada en abril del 2002. Apache 2 no tiene soporte en las instalaciones de Windows 9x; requiere de Windows NT/2000/XP.

Muchas distribuciones de Linux vienen con Apache 2 instalado. Sin embargo, hasta el momento de escribir este libro, el sitio web PHP advierte contra el uso de Apache 2 con PHP en un ambiente de producción. Revise la página web para la condición actual de PHP con Apache 2 en `www.php.net/manual/en/install.apache2.php#install.apache2.unix`.

Por el momento, las versiones actuales son

- Apache 2.0.47
- Apache 1.3.28

Trate de instalar la versión más actual de Apache, para que su servidor Apache incluya las más recientes opciones de seguridad y los arreglos de errores. Ya no se están agregando características nuevas a Apache 1.3, pero las pulgas se continúan reparando y los asuntos de seguridad se están resolviendo. Nuevas versiones de Apache 1.3 continúan publicándose, aunque menos frecuentemente que para Apache 2.

Instalar Apache

Apache puede descargarse e instalarse en su servidor web gratuitamente. Está disponible para casi todos los sistemas operativos, incluyendo Windows, Linux, muchas versiones de Unix y Mac.

En Linux/Unix

Para instalar Apache en Linux y Unix, se descarga el código fuente, se compila y se instala. Esto suena intimidante pero es mucho más fácil de lo que parece.

Antes de la instalación

Antes de instalar Apache, revise los siguientes requisitos:

- **Espacio en disco:** Necesitará hasta 50MB de espacio en disco durante la instalación. Apache probablemente use 10MB después de la instalación, aunque la cantidad varía dependiendo de las opciones usadas y los módulos instalados.
- **Compilador C:** Su PC tiene un compilador C que cumple con ANSI instalado. GNU C (gcc) es una buena opción.

Instalación

Para instalar Apache desde archivos fuente, siga estos pasos:.

1. **Dirija su explorador web hacia `httpd.apache.org`, la página de inicio de Apache.**
2. **Haga clic en el vínculo From a Mirror bajo Download, al lado izquierdo de la página.**
3. **Desplácese hasta la sección Mirror.**

   Un espejo específico se selecciona para usted. Si no desea usar ese espejo, seleccione otro. O si tiene problemas para descargarlo, regrese a esta página y seleccione otro.
4. **desplácese más hacia abajo en la misma página, hasta la sección para Apache 2 o para Apache 1.3, dependiendo de cuál quiera instalar. Localice y resalte el archivo que desea descargar.**

   Por ejemplo, en este momento, la versión más reciente de Apache 1.3 para Linux es `apache-1.3.28.tar.gz`.
5. **Haga clic en la última versión para descargarla.**
6. **Seleccione la opción para guardar el archivo.**
7. **Navegue hasta donde desea guardar el código fuente (por ejemplo, `/usr/src`). Luego haga clic en Save.**

8. **Después de la descarga, cambie al directorio de descargas (por ejemplo, `cd-/usr/src`).**

   Verá un archivo llamado apache-, seguido del nombre de versión y `tar.gz`. Este archivo se llama un tarball porque contiene muchos archivos comprimidos dentro del archivo tarball por parte de un programa llamado `tar`.

   Asegúrese de estar usando una cuenta que tiene permiso para escribir en `/usr/src`, tal como `root`.

9. **Verifique el archivo descargado para cerciorarse de que no ha sido alterado. Para verificar el archivo, siga estos pasos:**

   a. Descargue dos archivos de `www.apache.org/dist/httpd/`: Uno de los archivos por descargar se llama `KEYS`. El segundo archivo tiene el mismo nombre de archivo, incluyendo el número de versión, que la fuente, pero el nombre de archivo termina en.`asc`.

   b. Digite una de las líneas siguientes, dependiendo de cuál versión de PGP esté instalada en su PC:

```
pgp <KEYS
gpg --import KEYS
```

      Aparecerán varias líneas de output.

   c. Digite una de las siguientes líneas, con el número correcto de versión.

```
pgp apache-1.3.28.tar.gz.asc
gpg --verify apache-1.3.28.tar.gz.asc
```

      Debería ver algo parecido a lo siguiente:

```
Good signature from user "Sander Striker"
```

      Esto es lo que usted está buscando. Varios mensajes probablemente aparecerán, pero el mensaje anterior es el importante. Probablemente también vea un mensaje que indica que la relación entre la clave y la identificación de la clave no se puede verificar. Esto está bien.

      Si no obtiene un mensaje indicando que la identificación está bien, el archivo podría haber sido alterado y puede ser peligroso. En este caso, repita el proceso empezando con el Paso 1 y seleccione un espejo diferente de dónde descargarlo.

10. **Desempaque el tarball.**

    El comando para desempacar el tarball para la versión 1.3.28 es el siguiente:

```
gunzip -c apache-1.3.28.tar.gz | tar -xf -
```

    Un directorio nuevo llamado `apache-1.3.28` se crea con varios subdirectorios, el cual contiene todos los archivos que usted acaba de desempacar del tarball.

11. **Cambie al nuevo directorio que creó cuando desempacó el tarball.**

    Por ejemplo, puede usar un comando como el siguiente:

```
cd apache-1.3.28
```

12. **Digite el comando** `configure`.

    El comando `configure` consiste en `./configure` seguido de todas las opciones necesarias. Si puede usar todas las opciones predeterminadas, puede usar `configure` sin ninguna opción. Sin embargo, para usar Apache con PHP como un módulo, use el comando configure como sigue:

    ```
    ./configure --enable-module=so
    ```

    Una de las opciones de instalación más importantes que tal vez quiera usar es `prefix`, la cual establece una ubicación diferente donde usted desea que se instale Apache. En forma predeterminada, Apache se instala en `/usr/local/apache o usr/local/apache2`. Puede cambiar la ubicación de instalación con la siguiente línea:

    ```
    ./configure --prefix=/software/apache
    ```

    Puede ver una lista de todas las opciones disponibles digitando la siguiente línea:

    ```
    ./configure --help
    ```

    Este script puede tardar algún tiempo en terminar de correr. Conforme corre, despliega output. Cuando termina, aparece el aviso del sistema. Si `configure` encuentra un problema, aparece un mensaje de error descriptivo.

13. **Digite el siguiente comando:**

    ```
    make
    ```

    Este comando construye el servidor Apache. Puede tardar varios minutos para terminar de correr. Conforme corre, muestra mensajes que le indican qué está haciendo. Podría haber pausas largas ocasionalmente conforme completa alguna acción. Cuando termina, regresa al aviso del sistema. Si tiene algún problema, muestra un mensaje de error descriptivo.

14. **Digite el siguiente comando:**

    ```
    make install
    ```

    Este comando instala el software de Apache en las ubicaciones apropiadas, con base en el comando `configure` que usted usó en el Paso 11.

15. **Inicie el servidor web Apache.**

    Consulte los detalles en la sección siguiente llamada "Iniciar y detener Apache".

16. **Digite el URL de su sitio web (por ejemplo,** `www.mysite.com` **o** `localhost`**) en un explorador para probar a Apache.**

    Si todo sale bien, verá un mensaje de Apache que le dice que Apache está funcionando.

Iniciar y detener Apache

Un programa llamado `apachectl` está disponible para controlar el servidor. En forma predeterminada, el programa se almacena en un subdirectorio lla-

mado `bin` en el directorio donde Apache está instalado. Algunas distribuciones de Linux podrían colocarlo en otro directorio.

El script requiere de una palabra clave. Las palabras claves más comunes son `start`, `stop` y `restart`. La sintaxis general es la siguiente:

```
ruta/apachectl palabra clave
```

Por ejemplo, si Apache se instaló en el directorio predeterminado, digite la línea siguiente para iniciar Apache:

```
/usr/local/apache/bin/apachectl start
```

Iniciar Apache

El script `apachectl` inicia el servidor Apache, el cual luego corre en el fondo mientras busca solicitudes HTTP. En forma predeterminada, el servidor Apache compilado se llama `httpd` y se almacena en el mismo directorio que el script `apachectl`, aunque usted puede cambiar su nombre y ubicación al instalar Apache. El script `apachectl` sirve como una interfaz hacia el servidor compilado, llamado `httpd`.

Puede correr el servidor `httpd` directamente, pero es mejor usar `apachectl` como interfaz. El script `apachectl` maneja y revisa los datos que el comando httpd requiere. Use el script `apachectl` para iniciar Apache con el siguiente comando:

```
/usr/local/apache/bin/apachectl start
```

El script `apachectl` contiene una línea que corre `httpd`. En forma predefinida, `apachectl` busca a `httpd` en la ubicación predeterminada: `/usr/local/apache/bin` o `/usr/local/apache2/bin`. Si usted instaló Apache en una ubicación no estándar, tal vez necesite editar `apachectl` para usar la ruta correcta. Abra `apachectl` y luego busque la siguiente línea:

```
HTTPD='/usr/local/apache2/bin/httpd'
```

Cambie la ruta a la ubicación donde instaló httpd. Por ejemplo, la línea nueva podría ser esta:

```
HTTPD='/usr/mystuff/bin/httpd'
```

Después de iniciar Apache, puede verificar si Apache está corriendo al fijarse en los procesos en su PC. Digite el comando siguiente para mostrar una lista de los procesos que están corriendo:

```
ps -A
```

Si Apache está corriendo, la lista de los procesos incluirá algunos procesos `httpd`.

Obtener información de Apache

Puede usar opciones con el servidor `httpd` para obtener información sobre Apache. Por ejemplo, puede averiguar cuál versión de Apache está instalada cambiando al directorio con `httpd` y digitando

```
httpd -v
```

O, probablemente,.`/httpd -v`. Puede averiguar cuáles módulos están instalados con Apache al digitar lo siguiente:

```
httpd -l
```

Para ver todas las opciones que están disponibles, digite esto:

```
httpd -h
```

Reiniciar Apache

Cada vez que usted cambia el archivo de configuración, las nuevas directrices se hacen efectivas la próxima vez que Apache inicia. Si Apache está apagado cuando usted hace los cambios, puede iniciar Apache tal y como describí anteriormente en "Iniciar Apache." Sin embargo, si Apache está corriendo, no puede usar start para reiniciarlo. Si usa start, obtendrá un mensaje de error que indica que Apache ya está corriendo. Puede usar el siguiente comando para reiniciar Apache cuando ya está corriendo:

```
/usr/local/apache2/bin/apachectl restart
```

Aunque el comando `restart` generalmente funciona, a veces no lo hace. Si reinicia Apache y las nuevas configuraciones no parecen estar en efecto, trate de detener Apache e iniciarlo nuevamente. A veces esto resuelve el problema.

Detener Apache

Para detener Apache, use el comando siguiente:

```
/usr/local/apache/bin/apachectl stop
```

Puede revisar si Apache se detuvo al verificar los procesos que están corriendo en su PC usando el siguiente comando:

```
ps -A
```

El output de `ps` no debería incluir ningún proceso `httpd`.

En Windows

Puede instalar Apache en casi cualquier versión de Windows, aunque es mejor con Windows NT/2000/XP.

Instalación

Para instalar Apache, siga estos pasos:

1. **Dirija su buscador web hacia** `httpd.apache.org`**, la página de inicio de Apache.**
2. **Haga clic en el vínculo From a Mirror bajo Download, al lado izquierdo de la página.**
3. **Desplácese hasta la sección Mirror.**

   Un espejo específico está seleccionado para usted. Si no desea usar ese espejo, seleccione otro. O si tiene problemas para descargar desde dicho espejo, regrese a esta página y seleccione otro.
4. **desplácese más abajo en la misma página, hasta la sección para Apache 2 o para Apache 1.3, dependiendo de cuál quiera instalar. Localice y resalte la línea para Win 32 Binary (MSI installer).**

   Por ejemplo, en este momento, la versión más reciente de Apache 1.3 para Windows es `apache-1.3.28`.
5. **Haga clic en el nombre de archivo para descargarlo.**
6. **Seleccione la opción para guardar el archivo.**
7. **Navegue hasta donde desea guardar el instalador. Debería tratarse de un directorio temporal, tal como un directorio de descargas. Luego haga clic en Save.**

   Una vez completa la descarga, verá un archivo en la ubicación de descargas, el cual contiene todos los archivos necesarios. El archivo se llama `apache`, seguido del número de versión y `win32-x86-no_src.msi`. Para la versión actual, el archivo se llama `apache_1.3.28-win32-x86-no_src.msi`
8. **Haga doble clic en el archivo descargado.**

   Se inicia el asistente de instalación de Apache, y aparece una pantalla de bienvenida.
9. **Haga clic en Next.**

   Aparece el acuerdo de la licencia.
10. **Seleccione I Accept the Terms en el acuerdo de la licencia y luego haga clic en Next.**

    Si no acepta los términos, no podrá instalar el software.

    Se despliega una pantalla con información sobre Apache.
11. **Haga clic en Next.**

    Aparece una pantalla que le pide información.
12. **Digite la información solicitada y luego haga clic en Next.**

    La información solicitada es

- **Nombre del dominio:** Digite el nombre de su domino, tal como **MiBellaCompañía.com**. Si está instalando Apache para propósitos de prueba y sólo planea acceder a él desde la misma máquina donde está instalado, puede digitar **localhost.**
- **Nombre del servidor:** Digite el nombre del servidor donde está instalando Apache, tal como **www.MiBellaEmpresa.com** o **s1.micompañía.com**. Si está instalando Apache para propósitos de prueba y sólo planea acceder a él desde la misma máquina donde está instalado, puede digitar **localhost.**
- **Dirección electrónica:** Digite la dirección electrónica donde desea recibir mensajes electrónicos sobre el servidor web, tal como **Servidorweb@ MiBellaEmpresa.com.**
- **Modo de funcionamiento:** Seleccione si desea que Apache corra como un servicio, con lo cual iniciaría automáticamente cuando el PC se inicia, o si desea iniciar Apache manualmente cuando desea usarlo. En la mayoría de los casos, la primera opción es mejor: correr Apache como un servicio.

Se despliega la pantalla del tipo de instalación.

13. **Seleccione un tipo de instalación y haga clic en Next.**

    En la mayoría de los casos, debería seleccionar Complete. Sólo los usuarios avanzados que entienden Apache bien deberían seleccionar Custom.

    Aparece una pantalla que muestra dónde se instalará Apache.

14. **Seleccione el directorio donde desea instalar Apache y haga clic en Next.**

    Verá el directorio de instalación predeterminado para Apache, generalmente `C:\Program Files\Apache Group`. Si esto está bien, haga clic en Next. Si desea instalar Apache en un directorio diferente, haga clic en Change y seleccione uno distinto, haga clic en Aceptar y haga clic en Next.

    Se despliega una pantalla que dice que el asistente está listo para instalar Apache.

15. **Haga clic en Install.**

    Si hace falta, puede regresar y cambiar cualquier información que digitó antes de proceder con la instalación.

    Una pantalla muestra el progreso mientras Apache se está instalando. Cuando la instalación está completa, aparece una pantalla que indica que el asistente ha completado la instalación satisfactoriamente.

16. **Haga clic en Finish para salir del asistente de instalación.**

Apache está instalado en su PC con base en su sistema operativo. Si lo instala en Windows NT/2000/XP, se instala automáticamente como un servicio que se inicia automáticamente cuando su PC se inicia. Si lo instala en Windows 95/98/Me, necesita iniciarlo manualmente o configurarlo de modo que se inicie automáticamente cuando su PC lo haga. Consulte la siguiente sección: "Iniciar y detener Apache", para más información.

Iniciar y detener Apache

Cuando instala Apache en Windows NT/2000/XP, se instala automáticamente como un servicio y se inicia. Está listo para usarse. Usted puede probarlo digitando el nombre de su sitio web (o **localhost**) en la ventana de su explorador. Verá una página web de bienvenida que dice: "Si puede ver esto, significa que la instalación del software del servidor web Apache en este sistema fue exitosa". En Windows 95/98/Me, debe iniciar Apache manualmente, usando el menú.

Apache instala elementos del menú para detener e iniciar Apache durante la instalación. Puede encontrar este menú en ➪Programs➪Apache HTTP Server➪Control Apache Server.

El menú que usa para iniciar y detener Apache brinda los siguientes elementos:

- **Iniciar:** Se usa para iniciar Apache cuando no está corriendo. Si hace clic en este elemento cuando Apache está corriendo, verá un mensaje de error que dice que Apache ya se ha iniciado.
- **Detener:** Se usa para detener Apache cuando está corriendo. Si hace clic en este elemento cuando Apache no está corriendo, verá el mensaje de error correspondiente.
- **Reiniciar:** Se usa para reiniciar Apache cuando está corriendo. Si hace cambios a la configuración de Apache, debe reiniciar Apache para que los cambios se hagan efectivos.

Obtener información de Apache

A veces, usted desea conocer información sobre su instalación de Apache, tal como la versión que está instalada. Puede obtener esta información de Apache abriendo una ventana command prompt (Start➪Programs➪Accessories➪Command Prompt), cambiando al directorio donde Apache está instalado (tal como, cd C:\Apache), y tener acceso a Apache con opciones. Por ejemplo, para averiguar cuál versión de Apache está instalada, digite lo siguiente en la ventana command prompt:

```
Apache -v
```

Para averiguar cuáles módulos están compilados en Apache, digite

```
Apache -l
```

También puede iniciar y detener Apache directamente, como sigue:

```
Apache -k start
Apache -k stop
```

Puede ver todas las opciones disponibles digitando lo siguiente:

```
Apache -h
```

En Mac

Instalar Apache en Mac es muy parecido a instalarlo en Unix/Linux. Se descarga el código fuente y se compila. Para instalar Apache en Mac, siga estos pasos:

1. **Descargue el código fuente, guárdelo en un directorio y cambie al directorio donde el archivo descargado se guardará.**

   Siga los Pasos 1 — 8 de las instrucciones para Unix/Linux.

   Verá un archivo llamado `httpd`, seguido del nombre de versión y `tar.gz`, tal como `httpd-1.3.28.tar.gz`. Este archivo es el tarball: un archivo que contiene todos los archivos necesarios, comprimidos en uno solo.

2. **Desempaque el tarball usando un comando parecido a lo siguiente:**

   ```
   gnutar -xzf /httpd_1.3.28.tar.gz
   ```

   Después de desempacar el tarball, verá un directorio llamado `httpd_1.3.28`. Este directorio contiene varios subdirectorios y muchos archivos.

3. **Use un comando cd para cambiar al directorio nuevo creado cuando desempacó el tarball (por ejemplo, `cd httpd_1.3.28`).**

4. **Digite el comando siguiente:**

   ```
   ./configure --enable-module=most --enable-shared=max
   ```

   Este comando puede tardar algún tiempo en correr.

5. **Digite el siguiente comando para construir el servidor Apache:**

   ```
   make
   ```

   Este comando puede tardar unos cuantos minutos en correr. Muestra mensajes mientras está corriendo, con pausas ocasionales para que un proceso termine de correr.

6. **Digite el siguiente comando para instalar Apache:**

   ```
   sudo make install
   ```

7. **Inicie el servidor web Apache.**

   Consulte los detalles en la sección "Iniciar y detener Apache", bajo Unix/Linux.

8. **Digite el URL para su sitio web (por ejemplo, `www.misitio.com` o `localhost`) en un buscador para probar Apache.**

   Si todo sale bien, verá una página web que le indica que Apache está funcionando.

Configurar Apache

Cuando Apache inicia, lee información de un archivo de configuración. Si Apache no puede leer el archivo de configuración, no puede iniciar. A menos que

usted le diga a Apache que use un archivo de configuración diferente, buscará el archivo `conf/httpd.conf` en el directorio donde Apache está instalado.

Cambiar configuraciones

Apache se comporta de acuerdo con comandos, llamados directrices, en el archivo de configuración. Usted puede cambiar algunos de los comportamientos de Apache editando el archivo de configuración de Apache y reiniciar Apache, de modo que lea las nuevas directrices.

El archivo de configuración es un archivo de texto que contiene comandos llamados directrices. Apache se comporta según las directrices en este archivo. En la mayoría de los casos, los controles predeterminados le permiten a Apache iniciar y correr en su sistema. Sin embargo, quizás usted necesite cambiar las configuraciones en algunos casos. Algunas razones por las cuales querrá hacerlo son

- **Instalar PHP:** Si instala PHP, necesita configurar Apache para reconocer los programas en PHP. El Apéndice B describe cómo cambiar la configuración de Apache para PHP.
- **Cambiar su espacio web:** Apache busca los archivos de páginas web en un directorio específico y sus subdirectorios, a menudo llamados su espacio web. Usted puede cambiar la ubicación de su espacio web.
- **Cambiar el puerto donde Apache escucha:** De manera predeterminada, Apache escucha las solicitudes de los archivos en el puerto 80. Puede configurar Apache para escuchar en un puerto diferente.

Para cambiar cualquier configuración, edite el archivo `httpd.conf`. En Windows, puede acceder a este archivo por medio del menú en Start➪Programs➪Apache HTTPD Server➪Configure Apache Server➪Edit the Apache httpd.conf File. Al hacer clic en este elemento del menú, el archivo `httpd.conf` se abre en Notepad.

El archivo `httpd.conf` tiene comentarios (empezando con #) que describen las directrices, pero usted debería asegurarse de entender su función antes de cambiar cualquiera de ellas. Todas las directrices se documentan en el sitio web de Apache.

Al agregar o cambiar rutas o nombres de archivos, use diagonales, incluso cuando el directorio está en Windows. Apache puede entenderlo. Además, la ruta o los nombres no necesitan estar entre comillas a menos que incluyan caracteres especiales. Los dos puntos (:) son un carácter especial; la raya (_) y el guión (-) no lo son. Por ejemplo, para indicar un directorio en Windows, usted usaría algo como esto:

```
"c:/temp/mydir"
```

Acuérdese de reiniciar Apache después de cambiar cualquier configuración. Las configuraciones no se hacen efectivas hasta que Apache no se reinicie. A veces, usar el comando de reinicio no funciona para cambiarlas. Si las configuraciones nuevas no parecen estar en efecto, trate de detener el servidor con stop y luego empiécelo de nuevo con start.

Cambiar la ubicación de su espacio web

De manera predeterminada, Apache busca los archivos de su página web en el subdirectorio `htdocs`, en el directorio donde Apache está instalado. Usted puede cambiarlo con la directriz `DocumentRoot`. Busque la línea que empieza con `DocumentRoot`, tal como la siguiente:

```
DocumentRoot "C:/Program Files/Apache Group/Apache/htdocs"
```

Cambie la ruta o el nombre de archivo a la ubicación donde desea almacenar los archivos de su página web. No incluya una diagonal (/) al final de la ruta del directorio. Por ejemplo, la siguiente podría ser su directriz nueva:

```
DocumentRoot /usr/mysrver/Apache2/webpages
```

Cambiar el número del puerto

De manera predeterminada, Apache escucha en el puerto 80. Tal vez usted quiera cambiarlo, por ejemplo, si está configurando un segundo servidor Apache para propósitos de prueba. El puerto se establece usando la directriz `Listen` como sigue:

```
Listen 80
```

Con Apache 2, la directriz `Listen` es obligatoria. Si no se incluye dicha directriz, Apache 2 no se iniciará.

Puede cambiar el número de puerto como sigue:

```
Listen 8080
```

Recuerde reiniciar Apache después de cambiar cualquier directriz.

Índice

• Símbolos •

• A •

• B •

• C •

• D •

• E •

• F •

• G •

• H •

• M •

• N •

• O •

• P •

• Q •

• R •

• S •

• T •

•U•

•V•

•W•

•X•

•Z•