Tutorial de PHP y MySQL COMPLETO

Fuente:

http://es.tldp.org/Manuales-LuCAS/manual_PHP/manual_PHP/

TEMA 1

1. **Instalación de Apache+PHP+MySQL**
    - Instalación en Windows
    - Someter PHP a prueba
    - Someter MySQL a prueba

2. **Desarrollar una aplicación Web con Base de datos**
    - Planear su aplicación Web con base de datos
        - Identificar lo que se espera de la aplicación
        - Tome en cuenta a los usuarios
        - Hacer el sitio que sea fácil de usar
        - Dejar espacio para expansiones
        - Escríbalo
    - Diseñar la base de datos
        - Escoger los datos
        - Organizar los datos
        - Organizar datos en las tablas
        - Cómo crear relaciones entre tablas
    - Diseñar la base de datos de la aplicación
        - Proceso del diseño del Catálogo de mascotas
        - Tablas de la base de datos
        - Tablas para la base de datos Sólo para miembros
    - Diseñar la aplicación
        - Construir la base de datos
        - Escribir los programas

TEMA 2

3. **La base de datos MySQL**
    - Construcción de la base de datos
    - Cómo comunicarse con MySQL
    - Contruir consultas en SQL
    - Enviar consultas en SQL
    - Crear una base de datos nueva
    - Cómo borrar una base de datos

- Cómo agregar tablas a una base de datos
- Consultas SQL para crear una tabla
- Cómo cambiar la estructura de la base de datos
- Mover datos hacia dentro y fuera de la base de datos
  - Agregar información
  - Cómo agregar una fila a la vez
  - Cómo recuperar información
  - Como recuperar información específica
  - Cómo recuperar datos en un orden específico
  - Cómo recuperar datos de una fuente específica
  - Cómo combinar información de tablas
  - Cómo actualizar información
  - Cómo eliminar información
- Cómo proteger sus datos
  - Controlar el acceso a sus datos
  - Comprender los nombres de las cuentas y hostnames
  - Echemos un vistazo a los archivos
  - Respaldar sus datos
  - Restaurar sus datos

TEMA 3

4. **PHP General**
   - Agregar una sección PHP a una página HTML
   - Variables y Operadores
   - Constantes
   - Cadenas entre comillas sencillas versus cadenas entre comillas dobles
   - Sentencias de Control
   - Las Tablas o arreglos
     - Cómo crear tablas o arreglos
     - Cómo moverse por un arreglo
     - Cómo ordenar arreglos
     - Tablas o arreglos multidimensionales
   - Las Funciones
   - La instrucción return
   - Parámetros de las funciones
   - Funciones variable
   - Recursión
   - Cómo ahorrarnos líneas de código
   - Tiempo y fecha
   - Almacenar una marca de tiempo en una varible
   - Bloques de construcción PHP para programas
   - Funciones PHP y MySQL
     - Hacer una conexión
     - Conectarse al servidor MySQL

- Seleccionar la base de datos correcta
- Enviar consultas SQL
- Enviar una consulta SELECT
- Extraer y usar los datos
- Extraer una fila de datos
- Usar un ciclo para obtener todas las filas de datos
- Usar funciones para extraer datos

TEMA 4

5. **Formularios**
    - Obtener información del usuario
        - Usar formularios HTML
        - Hacer que los formularios sean dinámicos
        - Construir listas de selección
        - Crear listas de botones de opción
        - Construir listas de casillas para marcar
        - Revisar la información del formulario
            - Revisar en busca de campos vacios
            - Verificar el formato de la información
    - Los Formularios
    - Descarga de archivos desde un formulario
        - Funciones de acceso a ficheros

TEMA 5

6. **Proyecto**

Tema 1

Instalación de Apache+PHP+MySQL en Windows

En este capítulo describiremos el proceso de instalación de la base de datos MySQL, de un servidor web Apache con PHP, en una máquina con sistema operativo Windows.

Lo primero que debemos hacer es conseguirnos los programas necesarios, y que mejor para ello que dirigirnos a las páginas web, de los programas en cuestión:

- **Apache**: www.apache.org
  - apache_1_3_x_win32.exe
- **MySQL**: www.mysql.com
  - mysql-shareware-3.22.34-win.zip
- **PHP**: www.php.net
  - php-3.0.x-win32.zip

O tan solo instalar Wamp Server: : http://wamp-server.softonic.com/descargar

Someter PHP a prueba

Una vez que ya tenemos instalados *PHP* y *MySQL*, y el servidor *Apache* configurado para usarlos, podemos comenzar a escribir nuestro primer script en PHP.

Dicho esto, vamos a someter a prueba a PHP. El código que queramos que sea interpretado por el servidor, lo pondremos entre las marcas *<php* y *?>* o *<?* y *?>* , para que sepa diferenciarlo de las restantes etiquetas HTML. Vamos a poner un ejemplo, que lo guardaremos como *ejemplo1.php* en la raíz del servidor web, es decir en */wamp/www (carpeta de publicación)*:

Como el servidor Web, PHP y el archivo ejemplo1.php están en la misma computadora en la que está haciendo la prueba, puede digitar en la barra de dirección del navegador: *localhost/ejemplo1.php*

ejemplo1.php	Ver documento anexo de código fuente

Ahora si ponemos esta URL en nuestro navegador veremos una línea con el texto *"Hola. Este es mi primer script en PHP".*

Lo primero que apreciamos en el script son sus delimitadores. En la primera línea del script vemos **`<?php`** que nos indica que comienza un script en PHP, y en la última colocamos **`?>`** para indicar el final del script. Hay que destacar que todas las líneas que se encuentre entre estos delimitadores deben acabar en *punto y coma,* excepto las sentencias de control (if, swicht, while, etc.).

Como en toda programación, es importante poner muchos comentarios, para lo cual si queremos comentar una sola línea tenemos que poner al principio de la línea `//`, si lo que queremos es comentar varias utilizaremos los delimitadores `/* - */`.

Para que el servidor envié texto utilizaremos la instrucción **`echo`**, aunque también podemos utilizar **`printf`** de uso similar al del *C o Perl.*

Vemos que la palabra ***myvar*** comienza con el signo dólar (***$***). Este símbolo le indica a *PHP* que es una variable. Nosotros le hemos asignado un texto a esta variable, pero también pueden contener números o tablas (arrays). Es importante recordar que todas las variables comienzan con el *signo dólar*.

También ha observado que el texto que le asignamos a la variable termina con
, esto no se imprime sirve para indicarle al navegador una nueva línea.

Finalmente, debajo de las líneas deberá ver una gran tabla que muestra toda la información asociada con PHP en tu sistema.

Muestra la información PHP, la ruta y los nombres de los 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.

Someter MySQL a prueba

Si ya sabe que PHP está corriendo bien, pruebe si tienes acceso a MySQL usando PHP. Sólo sigue los siguientes pasos:

Captura el siguiente código y llama al archivo **mysql_up.php**

mysql_up.php	Ver documento anexo de código fuente

Si tu cuenta MySQL no requiere contraseña, no digites nada entre las comillas, como sigue:

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

Debería ver una tabla con una larga lista de nombres y valores de variables. 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 probando el mensaje.

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

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

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

Desarrollar una aplicación Web con base de datos

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 planteamiento. Construir las partes de la aplicación se hace después del planeamiento. Los pasos de 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.

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.

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 puedas 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, realice 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 a 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 planteamiento, las tareas que desea que su aplicación realice son todavía bastante generales. Puede cumplir cada una de estas tareas de muchas formas diferentes. Ahora debe examinar las tareas de cerca y detallar con exactitud cómo las cumplirá la aplicación.

Por ejemplo, si 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 las mascotas se almacenen 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 mascota (por ejemplo, perros, gatos, 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 muestran la información sobre las mascotas que concuerdan con las palabras de la búsqueda. Por ejemplo, un cliente puede digitar gato para ver la lista de todos los gatos disponible. 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

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 alimentos 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 la aplicación:

- ✓ **Mostrar 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 en la sección Solo 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 del 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 pide al cliente su nombre de usuario y la contraseña.**

  - ➢ **Comparar la contraseña y e nombre de usuario digitados por el cliente con los nombre 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.**

A esta altura debe tener una idea bastante clara de lo que desea de su aplicación con base de datos. Al inicio del proyecto empiece con un plan tan completo como le sea posible, de esa manera se mantendrá concentrado.

Tome en cuenta a los usuarios

Identificar lo que usted espera que haga su aplicación con base de datos para la Web es sólo un aspecto del planeamiento. También debe tomar en cuenta lo sus usuarios querrán de ella. Por ejemplo, digamos que su meta es recopilar la 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, necesita 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:

- ✓ Recibir un boletín
- ✓ Para participar en una rifa con un buen premio.
- ✓ Para recibir descuentos especiales.
- ✓ Para ser notificados sobre productos nuevos o mejoras en los productos, tan pronto estén disponibles.
- ✓ Para tener acceso a información valiosa.

Supongamos 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 tomas 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 en forma diferente.

Dejar espacio para expansiones

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

Escríbalo

Escriba el plan. Mientras lo desarrolla, su plan es lo más importante en su mente y está totalmente claro. Pero en unas cuantas semanas le sorprendería 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 como lo diseñó.

Diseñar la base de datos

Después de haber determinado con exactitud qué hará su aplicación con base de datos para la Web, ya está listo para diseñar la base de datos que guardarán 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 e línea necesita una base de datos que pueda guardar la información sobre el cliente y la información sobre los pedidos.

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

- ✓ El nombre de la mascota
- ✓ Una descripción de la mascota
- ✓ Una foto de la mascota
- ✓ El costo de la mascota

Para la sección Sólo para miembros, le conviene guardar la 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 la 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.

Organizar los datos

MySQL es un SGBD (Sistema Gestor de Base de Datos), lo cual significa que los datos se organizan en tabla. Usted puede establecer relaciones entre las tablas en la base de datos.

Organizar datos en tablas

Las tablas de un SGBD relacional se organizan como cualesquiera otras tablas que usted ya conoce, es decir, en filas y columnas.

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

- Clientes
- Productos
- Libros

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.

En el uso de las bases de datos, un objeto es una entidad, y una entidad tiene atributos.

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

1. Nombre su base de datos.
2. Identifique los objetos.
3. Defina y nombre una tabla para cada objeto.
4. Identifique los atributos de cada objeto.

5. Defina y nombre las columnas para cada uno de los atributos que identificó en el paso 4.
6. Identifique la clave primaria.
7. Defina los valores predeterminados.
8. Identifique columnas con datos obligatorios.

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 guardas lo 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***.

Diseñar las bases de datos de la aplicación

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
- ✓ 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 base de datos.

Se debe diseñar la base de datos ***CatalogodeMacotas*** siguiendo los pasos presentados en la sección "Organizar los datos en tablas":

1. **Nombre de la base de datos.**

   La base de datos para el Catálogo de mascotas se llama ***CatalogodeMacotas.***

2. **Identifique los objetos.**

   La lista de la información es:
   - El nombre de la mascota
   - Una descripción de la mascota
   - Una foto de la mascota

- El costo de la mascota

Toda esta información es sobre mascotas, así 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 llamado ***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, etc. Sin embargo, es muy posible que su tienda de mascotas tenga más de un gato 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 e catálogo impreso y e 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, pájaro, gato – 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.

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

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.
- ***Descripciónmascota***: 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***.

- ***Descripcintipo***: 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 tabal ***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 la tablas.

8. Identifique columnas con datos obligatorios.

A las siguientes columnas no se les permitirá nunca estar vacías:

Idmascota

Nombremascota
Colormascota
TipodeMascota

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
- ✓ Fecha de inscripción del miembro
- ✓ Entradas del miembro

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

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 México, puede asumir que la dirección del miembro es una dirección en el formato de direcciones postales de México.

- **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 ítems 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 para cada uno de los atributos que identificó en el paso 4.

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 ***Tiempoentrada*** juntos.

7. Defina los valores predeterminados.

No hay valores predeterminados para ninguna de las tablas.

8. Identifique 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.

Tablas de la base de datos

El diseño de la base de datos para la aplicación del Catálogo de mascotas incluye tres tablas:

- ***Mascota,***
- ***TipodeMascota***
- ***ColordelaMascota***

Las tablas siguientes muestran la organización de estas tablas.

Nombre de la base de datos: *CatalogodeMascotas*

Tabla 1 *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(25)	Categoría de la mascota
Descripcionmascota	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 2 *TipodeMascota*		
Nombre de la variable	**Tipo**	**Descripción**
Tipomascota	CHAR(15)	Nombre de la categoría de la mascota (llave primaria)
Descripcionmascota	VARCHAR(255)	Descripción de la categoría

Tabla 3 *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 Sólo para miembros incluye dos tablas, llamadas ***Miembro*** y ***Entrada***.

El diseño de la base de datos es el siguiente:

Nombre de la base de datos: ***DirectoriodeMiembros***

Tabla 1 *Miembro*		
Nombre de la variable	**Tipo**	**Descripción**
Nombreentrada	VARCHAR(20)	Nombre para entrar especificado por usuario (llave primaria)
Clave	CHAR(255)	Contraseña especificada por el usuario
Fechacreacion	DATE	Fecha en el miembro se inscribió y registro 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
Codigopostal	CHAR(10)	Código postal del miembro
email	VARCHAR(50)	Dirección electrónica del miembro
Teléfono	CHAR(15)	Número telefónico del miembro
Fax	CHAR(15)	Número de fax del miembro

Tabla 2 *Entrada*		
Nombre de la variable	**Tipo**	**Descripción**
Nombreentrada	VARCHAR(20)	Nombre para entrar especificado por el usuario (llave primaria 1)
Tiempoentrada	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 de construir la base de datos; luego escribirá los programas en PHP.

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 o 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 nombre y define las tablas que constituyen la base de datos. Para construir, 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 tados. Las instrucciones detalladas sobre cómo construir la base de datos se verá en la siguiente sección.

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 pueda insertarle y extraerle datos.

El plan desarrollado esquematiza los programas que deben 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 s 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 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 del formulario.

Tiene métodos para poner los datos en las bases de datos MySQL, y tiene métodos para extraer datos de una base de datos MySQL. Posteriormente se verán las instrucciones necesarias para escribir programas en PHP.

Tema 2

La base de datos MySQL

Construcción de la base de datos

Después de terminar el diseño de su base datos, estarás listo para convertirlo en una base de datos funcional. En este tema 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 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.

Cómo comunicarse con MySQL

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

- ✓ Crear base de datos nuevas.
- ✓ Sabe dónde están guardadas las bases de datos.
- ✓ Guardar 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.

Construir consultas en SQL

SQL (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 parecidas a oraciones en inglés. En general no es necesario comprender ningún lenguaje técnico 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 se tratarán 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 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 * FROM `miembro` ORDER BY `Nombreentrada` ASC
LIMIT 0 , 30
```

Esta consulta recupera todos los apellidos y nombres de los miembros que viven en Ecatepec y los coloca en orden alfabético por el apellido.

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

Mayúsculas: Las palabras del lenguaje SQL completamente en mayúsculas; los elementos de información variable (como los nombres de las columnas) en minúsculas. Se hace con la intención de leer los ejemplos con mayor facilidad, no porque MySQL necesite este formato. No importa si las palabras en SQL están en mayúscula o minúscula; para MySQL `select` es igual que `SELECT`. Con el sistema UNIX o LINUX, es otra historia.

Espacios: Las palabras en SQL tienen que separarse por uno o más espacios. No importa cuántos espacios use. 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 ***Mx*** y ***Ecatepec*** están encerradas en comillas dobles (“). Mx y Ecatepec son series de caracteres llamadas cadenas de texto o cadenas de caracteres. 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. Cuando compara números (enteros, por ejemplo) almacenados en columnas numéricas, no hace falta encerrar los números entre comillas.

Enviar consultas en SQL

El programa ***mysql_send.php*** tiene una función simple: Ejecutar consultas y mostrar resultados. Digite el programa en el directorio donde está desarrollando su aplicación web:

mysql_send.php	Ver documento anexo de código fuente

Debe cambiar las líneas 5, 6 y 7 del programa antes de poder usarlo. Estas son las líneas:

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

Cambie **hostname** al nombre de la computadora donde MySQL está instalado; si la base de datos MySQL está instalada en la misma computadora que su sitio web, puede usar **localhost** como el **hostname**.

Cambie **mysqlusuario** 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, pero debe darle una contraseña ya que una cuenta llamada **root** sin contraseña no es nada segura.

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 mostrada a continuación:

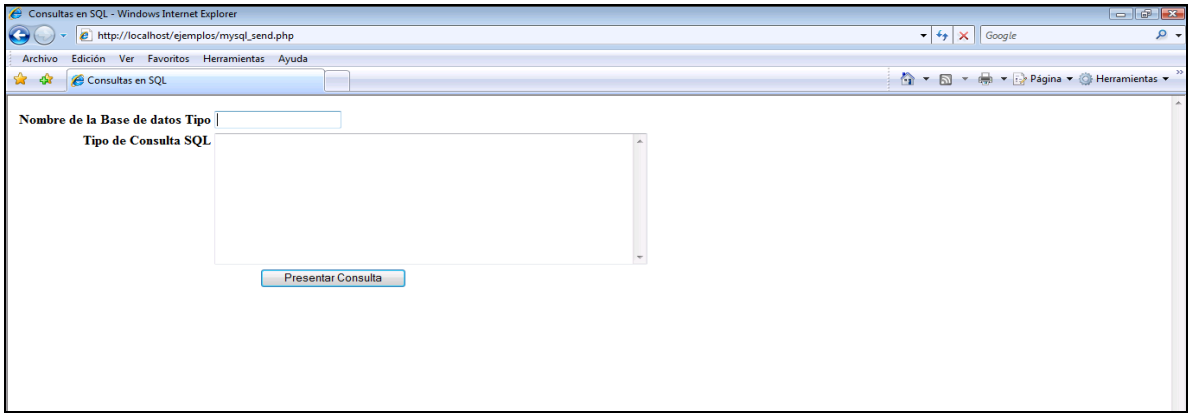

Fig. 1 Página Web de consulta SQL producida por ***mysql_send.php***

2. **En el campo Nombre de la Base de datos Tipo: *SHOW DATABASE***
3. **Haga clic en el botón "Presentar Consulta"**

En esta consulta no es necesario digitar nombre de una base de datos, al momento de dar clic en el botón, aparecerá una lista de las bases de datos existentes. En la mayoría de los casos, verá una base de datos llamada ***Text***, que automáticamente se instala cuando se instala MySQL. Además, probablemente verá una base de datos llamada ***mysql***, que MySQL usar 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.

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, eliminar y visualizar bases de datos y tablas.

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 propuestos anteriormente:

```
CREATE DATABASE CatalogodeMascotas

CREATE DATABASE DirectoriodeMiembros
```

A MySQL y PHP no les importa que todas sus tablas estén en una sola base de datos, en lugar de organizarlas 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.

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 las bases de datos:

- ***CatalogodeMascotas***; tiene diseñada tres tablas: `Mascota`, `TipoMascota` y `ColordelaMascota`.
- ***DirectoriodeMiembros***: está diseñada 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 1, debe digitar el nombre de la base de datos en el campo superior. Sino no lo hace verá el mensaje de error `No Database Selected`.

La consulta para agregar una tabla empieza con:

```
CREATE TABLE `nombrebasededatos`.`nombredetabla`(

    campo1 – tipo,

    campo2 – tipo,

PRIMARY KEY(campoN))
```

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, 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.
- ✓ **UNSIGNED**: Use esta definición para indicar que los valores en este campo numérico nunca serán número 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 diferentes 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. Debe especificar la clave primaria usando el siguiente formato:

```
CREATE TABLE `DirectoriodeMiembros`.`Miembro` (
  loginName VARCHAR(20) NOT NULL,
  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 las 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 siguiente lista 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 solo línea a MySQL no le importa cuántas líneas use. Sin embargo, el formato mostrado en una lista es más fácil de leer. Este formato amigable a los humanos también le ayuda a detectar errores de digitación.

Consulta SQL para crear una tabla

```
CREATE TABLE 'DirectoriodeMiembros'.'Miembros' (
    '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('Nombreentrada'))
```

Observe que la lista de los nombre de las columnas en la lista anterior 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.

Cómo cambiar la estructura de la base de datos

El formato básico de esta consulta es *ALTER TABLE nombretabla*, seguido por los cambio específicos que está solicitando, la siguiente tabla muestra los cambios que es posible hacer.

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 especiales
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 definición nombre_columna nuevo_nombre_columna	Cambia la definición de una columna y renombre 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	Renombre 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, por lo que decide cambiarlo

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

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

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**
- ✓ **Actualizar información**
- ✓ **Recuperar información**
- ✓ **Eliminar información**

Agregar información

La consulta *LOAD* de SQL puede leer datos de un archivo grande o pequeño de texto. O sea, si sus datos ya están en un archivo dentro de su computadora, 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 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.

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 'carácter'
FIELDS ENCLOSED BY 'carácter'
LINES TERMINATED BY 'carácter'
```

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.

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, use la siguiente consulta para ver y revisar todos los datos de la tabla:

```
SELECT * FROM Mascota
```

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 `directoriodemiembros`.`entrada` (`Nombreentrada`,
`Tiempoentrada`) VALUES ('Lflores', '2010-07-06 23:00:17');
```

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.**
- ✓ **Se permiten una lista de columnas parcial.**
- ✓ **No es obligatorio usar una lista de columnas**
- ✓ **La lista de columnas y la lista de valores deben ser de la misma longitud.**

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

```
INSERT INTO `directoriodemiembros`.`miembro` (
`Nombreentrada`,
`Clave`,
`Fechacreacion`,
`Apellido`,
`Nombre`,
```

```
`Calle`,
`Ciudad`,
`Estado`,
`Codigopostal`,
`email`,
`Telefono`,
`Fax`
)
VALUES (
'Glopez','1234','2010-07-
06','Gonzalez','Alicia','hgkd','dfsf','Michoacan','81546',
'fkn@edu.mx','5465412','4554654'
), (
'Ljimenes','5678','2010-07-
06','Lopez','juan','jhkljlkj','khkjjlk','morelos','874845',
'kjfkjf@edu.mx','54656465','5646546'
);
```

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? ¿Hay un lagarto a la venta? 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 mucha 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 la información:

- ✓ **Puede solicitar únicamente la información (las columnas) que necesite para responder su pregunta.**
- ✓ **Puede solicitar la información en algún orden en particular.**
- ✓ **Puede solicitar información de objetos seleccionados (las filas) en su tabla.**

Cómo recuperar información específica

Para recuperar información específica, enumere las columnas que contiene 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 AS precioConImpuesto FROM Mascota
```

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

Por ejemplo, tal vez quiere saber el valor más bajo en la columna o el valor más alto, esto se realiza a través de la funciones *MAX()* y *MIN()*. Hay más de 100 funciones SQL, 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

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 la información, use la siguiente frase:

```
GROUP BY nombre_columna
```

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 ***TipoMascota***:

```
SELECT  *  FROM  Mascota GROUP BY TipoMascota
```

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 Ecatepec, 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. El formato básico de la cláusula *WHERE* es

```
WHERE expresión AND|OR expresión AND|OR expresión ……
```

expresión especifica un valor que debe compararse con los valores guardados en la base de datos. Solo se seleccionan las filas que contengan información que concuerden 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 selecciona la fila. Cuando se usa OR, sólo una de las expresiones conectadas por OR debe concordar para seleccionar la fila.

Algunas expresiones comunes se muestran en la siguiente tabla:

Expresiones de la cláusula WHERE		
Expresión	**Ejemplo**	**Resultado**
`columna = valor`	`codigopostal = "12345"`	Sólo selecciona las filas donde 12345 estén almacenados 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
`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 900001 o 30044

`columna LIKE valor` `valor` puede contener el comodín % (que concuerda con cualquier cadena) y (que concuerda con cualquier carácter)	`codigopostal LIKE "9%"`	Selecciona 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 carácter)	`codigopostal NOT LIKE "9%"`	Selecciona las filas donde el código postal no empieza con 9

Puede combinar cualquiera de las expresiones de la tabla anterior con *AND* y *OR*. En algunos casos deberá usar paréntesis para aclarar los criterios de selección. Por ejemplo, puede usar la siguiente para responder a sus necesidades:

```
SELECT Apellido, Nombre FROM Miembro
  WHERE Apellido LIKE "B%"
    AND Ciudad ="Nezahualcóyotl"
    AND (teléfono LIKE "%2%" OR fax LIKE "%2%")
```

Observe los paréntesis en la última línea. Sin ellos no podrá obtener los resultados que usted desea. 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 Netzahualcóyotl y cuyos números telefónicos tienen un 2 en ellos, y todos los miembros cuyo número de fax tiene un 2 en ellos independientemente de si vive o no en Netzahualcóyotl e independientemente si su nombre empieza o no con B. Cuando se procesara el último *OR*, se seleccionaría 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 expresiones 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 2 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 Estado de México, use esta consulta:

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

Algunas consulta *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

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*, cláusula *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 `nombreentrada`
FROM `miembro`
UNION ALL SELECT `nombreentrada`
FROM `entrada`
LIMIT 0 , 30
```

El resultado de esta consulta es el apellido y el nombre de 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*, 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.O. 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 producida 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 Luis y una fila para Sara y la `tabla2` contiene sólo una fila para Sara, una conjunción interna contendría dos filas – una fila para Luis y una fila para Sara – aunque la fila para Luis 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 usa 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

EJEMPLO:

    SELECT * FROM `miembro` , `entrada`
    WHERE `miembro`.`nombreentrada` = `entrada`.`nombreentrada`
    LIMIT 0 , 30
```

```
Y estas consultas son conjunciones externas:

SELECT listanombrecolumna FROM tabla1 LEFT JOIN tabla2

     ON tabla1.col1 = tabla2.col2

EJEMPLO:

    SELECT * FROM `miembro`  LEFT JOIN `entrada`

    ON `miembro`.`nombreentrada` = `entrada`.`nombreentrada`

    LIMIT 0 , 30


SELECT listanombrecolumna FROM tabla1 RIGHT JOIN tabla2

     ON tabla1.col1 = tabla2.col2

EJEMPLO:

    SELECT * FROM `miembro`
    RIGHT JOIN `entrada` ON `miembro`.`nombreentrada` =
    `entrada`.`nombreentrada`
    LIMIT 0 , 30
```

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
Siamés	Gato
Persa	Gato
Canario	Pájaro

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

NombreMascota	ColorMascota
Siamés	Café
Siamés	Blanco
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 =
Color.NombreMascota
```

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

NombreMascota	TipoMascota	NombreMascota	ColorMascota
Siamés	Gato	Siamés	Café
Siamés	Gato	Siamés	Blanco

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

```
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***)*,* `ColorMascota` – pero con filas diferentes:

NombreMascota	TipoMascota	NombreMascota	ColorMascota
Siamés	Gato	Siamés	Café
Siamés	Gato	Siamés	Blanco
Persa	Gato	<NULL>	<NULL>
Canario	Pájaro	<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 ***Mascota*** 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
Siamés	Gato	Siamés	Café
Siamés	Gato	Siamés	Blanco
<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 ve 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.Noembreentrada 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 columnas 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 = "Av. Morelos 123", teléfono =
"(55)55555555"
WHERE Nombreentrada = "granchico"
```

Cómo eliminar información

Mantenga información en su base de datos a 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. Los datos no se podrán recuperar.

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
```

Cómo proteger los 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 datos, la cual contiene información importante digitada por usted y por sus usuarios. Por tanto, debe protegerla.

Controlar el acceso a sus datos

Usted controla 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 pontro 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 su PC autorizada o 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 autorizada 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.

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 MySQL sólo cuando se conecte desde el hostname. Cuando se prepara una consulta *GRANT* o *REVOKE* (se describen más adelante), 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 Windows (también llamado a veces nombre de entrada). Cambiar el nombre de entrada de MySQL no afecta de ninguna manera el nombre de entrada en Windows, y viceversa.

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

Es posible configurar una cuenta con el nombre de la cuenta en blanco y con el *hostname* en blanco, esto permitirá 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.

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.

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 *GRAN* 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 vera el siguiente mensaje:

```
grant command denied
```

Los permisos se pueden otorgar y remover individualmente o todos a la vez. La siguiente tabla enumera algunos de los permisos que puede asignar o revocar.

Permisos para cuentas MySQL	
Permiso	**Descripción**
ALL	Todos los permisos
ALTER	Puede alterar la estructura de las tablas
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 administrativa, tales como apagar el servidor MySQL. No es probable que quiera que cualquiera además de usted tenga tales privilegios.

Respaldar sus datos

Al crear una cuenta nueva se especifica la contraseña

Usted debe tener por lo menos una copia de su valiosa base de datos, y no en el mismo lugar en el que está la copia que está usando actualmente.

- ✓ Almacene una copia en una ubicación fácilmente accesible, incluso podría ser en la misma computadora, para reemplazar rápidamente una base de datos en uso que se haya dañado.

- ✓ Almacene una segunda copia en otra computadora; les será útil en caso de la que la computadora falle y la primera copia de respaldo no esté disponible.

- ✓ Almacene una tercera copia en un lugar físico completamente diferente, o en un medio portátil.

Si usted es el administrador de MySQL, es el responsable de hacer los respaldos. MySQL brinda un programa llamado *mysqldump* que puede usar para hacer las copias de respaldo; *mysqldum* crea un archivo de texto con todas las instrucciones SQL necesarias para recrear toda su base de datos.

Siga estos pasos para hacer una copia de respaldo de su base de datos en Windows:

1. **Abra una ventana de comando**

   Ejecute cmd en el campo ejecutar, para ir al modo 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
```

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 a menudo.

Tema 3

PHP General

Los programas son la parte de aplicación de su aplicación con base de datos para la Web, realizan las tareas, crean y muestran las páginas web, aceptan y procesan la información que envías los usuarios, almacenan la información en la base de datos, extraen la información de la base de datos llevan a cabo todas la 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.

Agregar una sección PHP a una página HTML

PHP es un socio de HTML (Lenguaje de Marcado de Hipertexto) 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 de desplegar gráficos en sus páginas web y reproducir archivos de música. Pero el HTML, solo 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 las páginas web están 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`. Los enunciados en lenguaje PHP se encierran en etiquetas PHP con la siguiente forma:

```
<?php                                ¿>
```

Si usted recuerda, se ha digitado el programa ***ejemplo1.php***, en el tema1:

ejemplo1.php	Ver documento anexo de código fuente

La sección PHP que usted a su archivo HTML consta de un arreglo de enunciados PHP, 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.

Le recomiendo escribir sus programas PHP con un editor que enumere las líneas. Hallará muchos editores que sirven para editar PHP en phpeditors.linuxbackup.co.uk.

Observe que los enunciados dentro de los bloques están sangrados. La sangría no es necesaria par PHP. Se usa estrictamente para facilitar la lectura.

En general, a PHP no le importa si las palabras claves del enunciado están en mayúscula o minúscula. Echo, echo, ECHO y eCHo son todos iguales para PHP.

Variables y operadores

Variables

Como vimos antes todas la variables deben ser precedidas por ***signo dólar*** *($)*, y le asignamos contenido con el **signo igual** (=). Con las variables, PHP **distingue** entre mayúsculas y minúsculas, por lo que no es lo mismo **`$myvar`** que **`$Myvar`**, éstas son dos variables totalmente distintas. ***ejemplo4.php***

ejemplo4.php	Ver documento anexo de código fuente

Al denominar variables, use nombres que indiquen claramente qué información está en la variable.

El uso de la barra invertida, como en **`\n`**, no es obligatorio, pero ayuda a la depuración del código que enviamos al navegador, además del **`\n`** existen otros usos:

`\"` Carácter dobles comillas
`\\` Carácter barra invertida
`\n` Nueva línea
`\r` Retorno de carro
`\t` Tabulador horizontal

Usted puede ir más allá y destruir la variable usando este enunciado:

```
unset ($mivar);
```

Constantes

Las constantes son similares a las variables, con la salvedad de que no llevan el signo dólar delante, y sólo la podemos asignar una vez. Para definir una constantes usaremos la función ***`define`*** como sigue:

```
define ("nombreconstante", valorconstante");
```

Por ejemplo, fijar una constante con el nombre de la empres, use el siguiente enunciado (insertar éste código en el ejemplo 2): ***ejemplo5.php***

ejemplo5.php	Ver documento anexo de código fuente

```
define ("EMPRESA", "Tienda de mascotas ABD");
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.

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.

PHP crea diversas constantes al arrancar, como PHP_VERSION que contiene la versión de PHP, TRUE que le asigna 1 o FALSE que le asigna 0.

Operadores Aritméticos:

`$a + $b` **Suma**
`$a - $b` **Resta**
`$a * $b` **Multiplicación**
`$a / $b` **División**
`$a % $b` **Resto de la división de $a por $b**
`$a++` **Incrementa en 1 a $a**
`$a--` **Resta 1 a $a**

Operadores de Cadenas:

Para concatenación de cadenas se usa el punto.

```
$a = "Hola";
$b = $a . "Mundo"; // Ahora $b contiene "Hola Mundo"
```

En este punto hay que hacer una distinción, la interpretación que hace PHP de las simples y dobles comillas. En el segundo caso PHP interpretará el contenido de la cadena.

```
$a = "Mundo";
echo = 'Hola $a'; //Esto escribirá Hola $a
echo = "Hola $a"; //Esto escribirá Hola Mundo
```

Operadores de Comparación:

`$a < $b` $a **menor que** $b
`$a > $b` $a **mayor que** $b
`$a <= $b` $a **menor o igual que** $b
`$a >= $b` $a **mayor o igual que** $b
`$a == $b` $a **igual que** $b
`$a != $b` $a **distinto que** $b

Operadores Lógicos:

`$a AND $b` Verdadero si ambos son verdadero
`$a && $b` Verdadero si ambos son verdadero
`$a OR $b` Verdadero si alguno de los dos es verdadero
`$a !! $b` Verdadero si alguno de los dos es verdadero
`$a XOR $b` Verdadero si sólo uno de los dos es verdadero
`!$a` Verdadero si $a es falso, y recíprocamente

Operadores de Asignación:

`$a = $b` **Asigna** a $a el contenido de $b
`$a += $b` Le **suma** a $b a $a
`$a -= $b` Le **resta** a $b a $a
`$a *= $b` **Multiplica** $a por $b y lo asigna a $a
`$a /= $b` **Divide** $a por $b y lo asigna a $a
`$a .= $b` **Añade** la cadena $b a la cadena $a

Cadenas entre comillas sencillas versus cadenas entre comillas dobles

Las cadenas de comillas sencillas y las que tienen comillas dobles se manejan en forma diferente. Estas son algunas 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";
$resultado 2 = '$edad';
echo $resultado1;
echo $resultado2;
```

La salida es:
12
$edad

Sentencias de control

Las sentencias de control permiten ejecutar bloque de códigos dependiendo de unas condiciones. Para PHP el 0 es equivalente a Falso y cualquier otro número es Verdadero.

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, despliegue la página web exclusiva para miembros.
- ✓ ¿El cliente vive en el Estado de México? Si es así, despliegue el mapa de las tiendas en el Edo. de México.

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 secreta. ¿Falso o verdadero? Si es verdadero, muestre la página web exclusiva para miembros.
- ✓ El cliente vive en el Estado de México. ¿Falso o verdadero? Si es verdadero, despliegue un mapa con la ubicación de las tiendas del Estado de México.

IF...ELSE

La sentencia **IF...ELSE** permite ejecutar un bloque de instrucciones si la condición es **Verdadera** y otro bloque de instrucciones si ésta es **Falsa**. Es importante tener en cuenta las instrucciones si ésta es **Falsa**. Es importante tener en cuenta que la condición que evaluemos ha de estar encerrada entre **paréntesis** (esto es aplicable a todas las sentencias de control).

```
if (condición) {
      Este bloque se ejecuta si la condición es VERDADERA
   } else {
      Este boque se ejecuta si la condición es FALSA
   }
```

Existe una forma sencilla de usar la sentencia IF cuando no tenemos que usar el ELSE y solo tenemos que ejecutar una línea de código.

```
   if ($a > 4) echo "$a es mayor que 4";
```

IF...ELSEIF...ELSE

La sentencia IF...ELSEIF...ELSE permite ejecuta varias condiciones en cascada. Para este caso veremos un ejemplo, en el que utilizaremos los operadores lógicos.

```
   <?php
     if ($nombre == ""){
      echo "Tú no tienes nombre";
     } elseif (($nombre=="eva") OR ($nombre=="Eva")) {
      echo "Tu nombre es EVA";
     } else {
      echo "Tu nombre es " . $nombre;
     }
    ?>
```

SWITCH...CASE...DEFAULT

Una alternativa a IF...ELSEIF...ELSE, es la sentencia SWITCH, la cual evalúa y compara cada expresión de la sentencia CASE con la expresión que evaluamos, si llegamos al final de la lista de CASE y encuentra una condición Verdadera, ejecuta el código de bloque que haya en DEFAULT. Si encontramos una condición verdadera debemos ejecutar un BREAK para que la sentencia SWITCH no siga buscando en la lista de CASE. Veamos un ejemplo.

```
<?php
  switch ($dia) {
   case "Lunes":
   echo "Hoy es Lunes";
   break;

   case "Martes":
   echo "Hoy es Martes";
   break;

   case "Miercoles":
   echo "Hoy es Miércoles";
```

```
   break;

   case "Jueves":
   echo "Hoy es Jueves";
   break;

   case "Viernes":
   echo "Hoy es Viernes";
   break;

   case "Sábado":
   echo "Hoy es Sábado";
   break;

   case "Domingo":
   echo "Hoy es Domingo";
   break;
   default:
   echo "Esa cadena no corresponde a ningún día de la semana";
   }
?>
```

WHILE

La sentencia WHILE ejecuta un bloque de código mientras se cumpla una determinada condición.

```
<?php
  $num = 1;
  while ($num < 5) {
  echo $num,"<br>";
  $num++;
  }
?>
```

Podemos romper un bucle WHILE utilizando la sentencia **BREAK**.

```
   <?php
     $num = 1;
     while ($num < 5) {
       echo $num,"<br>";
         if ($num == 3){
          echo "Aquí nos salimos \n";
          break;
        }
     $num++;
     }
   ?>
```

DO...WHILE

Esta sentencia es similar a WHILE, salvo que con esta sentencia primero ejecutamos el bloque de código y después se evalúa la condición, por lo que el bloque de código se ejecuta siempre al menos una vez.

```
<?php
  $num = 1;
  do {
    echo $num;
```

```
      if ($num == 4){
         echo "Aquí nos salimos \n";
         break;
      }
  $num++;
  } while ($num < 5);
?>
```

FOR

El bucle FOR no es estrictamente necesario, cualquier bucle FOR puede ser sustituido fácilmente por otro WHILE. Sin embargo, el bucle FOR resulta muy útil cuando debemos ejecutar un bloque de código a condición de que una variable se encuentre entre un valor mínimo y otro máximo. El bucle FOR también se puede romper mediante la sentencia **BREAK**.

```
<?php
for ($num = 1; $num <=10; $num++){
   echo $num;
   if ($num == 6){
      echo "Aquí nos salimos \n";
      break;
      }
   }
?>
```

Las tablas o arreglos

Las tablas (o array en inglés y/o arreglos), son muy importantes en PHP, ya que generalmente, las funciones que devuelven varios valores, **como las funciones ligadas a las bases de datos, lo hacen en forma de tabla.**

Los arreglos son variables complejas. Un arreglo almacena un grupo de valores bajo el único nombre de variable. Un arreglo es útil para almacener valores relacionados.

Cómo crear tablas o 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 llamdo $mascotas:

En PHP disponemos de dos tipos de tablas. El primero sería el clásico, utilizando **índices, *ejempo6.php***:

ejemplo6.php	Ver documento anexo de código fuente

Un arreglo se puede considerar como una lista de parejas de claves y valores.

Un segundo tipo, usted específica la clave entre corchetes (arreglos **asociativos**). En el arreglo siguiente, las claves son los números 1, 2 y 3:

```
<?php
  $mascota[1] = "pony";
  $mascota[2] = "canario";
  $mascota[3] = "persa";
```

```
  echo ("yo tengo un " . $mascota[2] . "<BR>\n");
?>
```

Sin embargo también puede usar palabras como claves. Por ejemplo, los enunciados siguientes crean un arreglo de capitales estatales:

```
$capitales['MC']="Estado de México";
$capitales['VZ']="Veracruz";
$capitales['NL']="Nuevo León";
```

o bien,

```
$capitales = array("MC"=>"Estado de México"; "VZ"=>"Veracruz!; "NL"="Nuevo León");
```

Esta es una forma de asignar elementos a una tabla, pero una forma más formal es utilizando la función **`array`**, ***ejemplo7.php***

ejemplo7.php	Ver documento anexo de código fuente

Si no se especifica, el primer índice es el **cero**, pero podemos utilizar el operador **=>** para especificar el índice inicial. (Sustitúyelo en la línea del array, y prueba, y observa qué otros cambios debes hacer, para que se ejecute el programa correctamente):

```
$mascotas = array("pony", "canario", "persa", "poodle");
```

Cómo moverse por un arreglo

Existen dos formas 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 fina, de valor en valor.

Ahora bien, recorrer una tabla y mostrar su contenido es sencillo utilizando los índices, pero ¿cómo hacerlo en las tablas asociativas?

La manipulación de las tabas asociativas se hace a través de funciones que actúan sobre un puntero interno que indica la posición. Por defecto, el puntero se sitúa en el primer elemento añadido en la tabla, hasta que es movido por una función:

`current` - devuelve el valor del elemento que indica el puntero
`pos` - realiza la misma función que **current**
`reset` - mueve el puntero al **primer** elemento de la tabla
`end` - mueve el puntero al **último** elemento de la tabla
`next` - mueve el puntero al elemento **siguiente**
`prev` - mueve el puntero al elemento **anterior**
`count` - devuelve el número de elementos de una tabla.

Veamos un ejemplo de las funciones anteriores***, ejemplo8.php***:

ejemplo8.php	Ver documento anexo de código fuente

Recorrer una tabla con las funciones anteriores se hace un poco tedioso, para ello se recomienda utilizar la función ***each()***. **ejemplo9.php**

ejemplo9.php	Ver documento anexo de código fuente

La función ***each()*** devuelve el valor del elemento actual, en este caso, el valor del elemento actual y su clave, y desplaza el puntero al siguiente, cuando llega al final devuelve **FALSO**, y termina el bucle ***while()***.

La función ***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 enunciado;
}
```

`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 los estados de la república y repite una lista:

```
$capitales = array ("MC" => "Estado de México", "VZ" => "Veracruz",
"NL" => "Nuevo León" );
ksort ($capitales);
foreach($capitales as $estado => $ciudad)
{
  echo "$ciudades, $estado>br>";
}
```

Los enunciados anteriores dan la siguiente salida en la página web:

Estado de México, MC
Nuevo León, NL
Veracruz, VZ

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 la salida. Entonces, usted puede usar el siguiente enunciado echo:

```
echo "$ciudad<br>";
```

La salida con estos cambios son:

Estado de México

Nuevo León
Veracruz

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 con foreach.

Cómo ordenar arreglos

Una de las características más útiles de los arreglos es que PHP las puede clasificar. PHP almacena originalmete 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 clave.

PHP puede distribuir los arreglos de varias formas. Para distibuir 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 número primero, seguidos por las mayúsculas y, finalmente, las minúsculas. Por ejemplo, considere el arreglo `$mascotas`, creado en la sección anterior:

```
$mascota[0] = "pony";
$mascota[1] = "canario";
$mascota[2] = "persa";
```

Después de usar el enunciado `sort` siguiente:

```
sort ($mascotas);
```

el arreglo se transforma en:

```
$mascota[0] = "canario";
$mascota[1] = "persa";
$mascota[2] = "pony";
```

Si usa `sort()` para ordentar 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 los estados de la república creada en la sección anterior:

```
$capitales['MC']="Estado de México";
$capitales['VZ']="Veracruz";
$capitales['NL']="Nuevo León";
```

Después del siguiente enunciado de clasificación:

```
asort($capitales);
```

el arreglo se convierte en:

```
$capitales['MC']="Estado de México";
$capitales['NL']="Nuevo León";
$capitales['VZ']="Veracruz";
```

Observe que las claves permanecieron con el valor cuando los elementos fueron reordenados. Ahora los elementos están en orden alfabético, y la clave del estado correcto todavía está con el estado de la república correspondiente. Si las claves hubieran sido números estarían ahora eb un orden diferente. Por ejemplo, si el arreglo original fuera:

```
$capitales['1']="Estado de México";
$capitales['2']="Veracruz";
$capitales['3']="Nuevo León";
```

después de un enunciado asort, la nueva serie sería:

```
$capitales['MC']="Estado de México";
$capitales['NL']="Nuevo León";
$capitales['VZ']="Veracruz";
```

Dudo que usted quiera usar `asort` en un arreglo con claves numéricas.

Hay varios otros enunciados `sort` que ordenan de otras formas. Vea la tabla siguiente que incluye todos los enunciados sort disponibles:

Manera 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($nombrearreglo)	Ordena según valor en orden inverso; asigna números nuevos como claves
ksort($nombrearreglo)	Ordena según clave
krsort($nombrearreglo)	Ordena según clave en orden inverso
usort($nombrearreglo,nombrefuncion)	Ordena según función

Tablas o arreglos multidimensionales

En la sección anterior, se describió 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.

Las tablas multidimensionales son simplemente tablas en las cuales cada elemento es a su vez otra tabla. ***ejemplo10.php***

ejemplo10.php	Ver documento anexo de código fuente

La función ***list()*** es más bien un operador de asignación, lo que hace es asignar valores a unas lista de variables. En este caso los valores son extraídos de una tabla por la función ***each().***

Las funciones

Muchas veces, cuando trabajamos en el desarrollo de una aplicación, nos surge la necesidad de ejecutar un mismo bloque de código en diferentes partes de nuestra aplicación. Una **Función** no es más que un bloque de código al que le pasamos una serie de parámetros y nos devuelve un valor.

Como todos los lenguajes de programación, PHP trae una gran cantidad de funciones para nuestro uso, pero las funciones más importantes son las que nosotros creamos.

Para declarar una función debemos utilizar la instrucción ***function*** seguido del nombre que le vamos a dar, y después entre paréntesis la lista de argumentos separados por comas, aunque también habrá funciones que no tomen ningún argumento.

```
function nombre_de_funcion (arg_1, arg_2, ..., arg_n) {
   bloque de código
}
```

Cualquier instrucción válida de PHP puede aparecer en el cuerpo (lo que antes hemos llamado (bloque de código) de una función, incluso otras funciones y definiciones de clases.

La instrucción RETURN

Cuando invocamos una función, la ejecución del programa pasa a ejecutar las líneas de código que contenga la función, y una vez terminado, el programa continúa su ejecución desde el punto en que fue llamada la función.

Existe una manera de terminar la ejecución de la función aunque aún haya código por ejecutar, mediante el uso de la instrucción ***return*** terminamos la ejecución del código de una función y devolvemos un valor. Podemos tener varios ***return*** en nuestra función, pero por lo general, cuantos más ***return*** tengamos menos reutilizable será nuestra función.

```
<?php
function mayor ($x, $y)
{
   if ($x > $y) {
      return $x." es mayor que".$y;
   } else {
      return $y." es mayor que".$x;
   }
}
?>
```

Aunque quedaría mejor:

```
<?php
function mayor ($x, $y)
{
   $msg = "";
   if ($x > $y) {
      $msg = $x." es mayor que".$y;
   } else {
      $msg = $y." es mayor que".$x;
   }
```

```
    return $msg;
}
?>
```

Con la instrucción ***return*** puede devolverse cualquier tipo de valor, incluyendo tablas y objetos. PHP solo permite a las funciones devolver un valor, y para solventar este pequeño problema, si queremos que nuestra función devuelva varios tenemos que utilizar una tabla (array).

Parámetros de las funciones

Existen dos formas de pasar los parámetros a una función, por **valor** o por **referencia**.

Cuando pasamos una variable por **valor** a una función, ocurra lo que ocurra en ésta en nada modificará el contenido de la variable. Mientras que si lo hacemos por **referencia**, cualquier cambio acontecido en la función sobre la variable lo hará para siempre.

En PHP, por defecto, las variables se pasan por valor. Para hacerlo por referencia debemos anteponer un **ampersand** (**&**) a la variable. ***ejemplo11.php***

ejemplo11.php	Ver documento anexo de código fuente

Si queremos que un parámetro de una función se pase siempre por referencia debemos anteponer un ***ampersand (&)*** al nombre del parámetro en la definición de la función.

En PHP podemos definir valores por defecto para los parámetros de una función. Estos valores tienen que ser una expresión constante, y no una variable o miembro de una clase. Además cuando usamos parámetros por defectos, éstos deben estar a la derecha de cualquier parámetro sin valor por defecto, de otra forma PHP nos devolverá un error.

```
<?php
function suma ($x=1, $y)
{
    $x = $x + 1;
    return $x+$y;
}
?>
```

Si ejecutáramos esta función nos daría error, ya que hemos dado a **$x** el valor **1** por defecto y la hemos colocado a la izquierda de un parámetro que no tiene valor por defecto. La forma correcta es:

```
<?php
function suma ($y, $x=1)
{
    $x = $x + 1;
    return $x+$y;
}
?>
```

Llegados a este punto, damos un paso atrás y volvemos a las variables, para distinguir entre variables estáticas (**static**) y globales (**global**). Las variables estáticas se definen dentro de una función, la primera vez que es llamada dicha función la variable se inicializa, guardando su valor para posteriores llamadas. ***ejercicio12.php***

ejemplo12.php	Ver documento anexo de código fuente

Las variables globales, no se pueden declarar dentro de una función, lo que hacemos el llamar a una variable que ya ha sido declarada, tomando el valor que tenga en ese momento, pudiéndose modificar en la función. ***ejemplo13.php***

ejemplo13.php	Ver documento anexo de código fuente

Funciones Variable

PHP soporta el concepto de **funciones variables**, esto es significa que si una variable tiene unos paréntesis añadidos al final, PHP buscará una función con el mismo nombre que el contenido de la variable, e intentará ejecutarla. ***ejemplo14.php***

ejemplo14.php	Ver documento anexo de código fuente

Recursión

PHP también permite la recursión, es decir, una función se puede llamar así misma. Para aclarar el concepto de recursión, vamos a crear una función que comprueba si un número es entero o no.

Un número que no sea entero (7.4), tiene una parte entera y otra decimal (comprendida entre 0 y 1), lo que vamos a hacer para comprobar si un número es entero o no, será restarle 1 al número en cuestión hasta que nos que quedemos sin parte entera, y entonces comprobaremos si tiene parte decimal (un poco tedioso todo esto). ***ejemplo15.php***

ejemplo15.php	Ver documento anexo de código fuente

Cómo ahorrarnos líneas de código

En las lecciones anteriores hemos aprendido el uso básico de las funciones de PHP para trabajar con MySQL. En esta lección y sucesivas vamos a ver nuevas funciones que nos facilitan y potencian nuestras páginas web.

Por lo general, todos nuestros script tienen partes de código iguales, las funciones ***include()*** y ***require()*** nos van ahorrar muchas de estas líneas de código. Ambas funciones hacen una llamada a un determinado fichero pero de dos maneras diferentes, con ***include()***, insertamos lo que contenga el fichero que llamemos de manera literal en nuestro script, mientras que con ***require()***, le decimos que el script necesitará parte de código que se encuentra en el fichero que llama ***require().***

Como todo esto es un poco tedioso, veamos unos ejemplos que nos lo aclara:

```
<?php
include ("header.inc");
echo "Hola Mundo";
include ("footer.inc");
?>
```

Si tenemos en cuenta que el fichero **header.inc** contiene:

```
<html>
<body>
```

y el fichero **footer.inc** contiene:

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

Nuestro script sería equivalente a:

```
<html>
<body>
<?php
echo "Hola Mundo";
?>
</body>
</html>
```

Ahora veamos el script de ejemplo para la función ***require()***:

```
<?php
require ("config.inc");
include ("header.inc");
echo $cadena;
include ("footer.inc");
?>
```

Donde el fichero **config.inc** tendría algo como esto:

```
<?php
$cadena = "Hola Mundo";
?>
```

Tiempo y fecha

En esta lección vamos a ver como algunas funciones relacionadas con el tiempo y la fecha, así como algunos ejemplos prácticos.

time

Devuelve el número de segundos transcurridos desde el 1 de Enero de 1970. A esta forma de expresar fecha y hora se le denomina **tmestamp.**

date(formato, timestamp)

La función **date** devuelve una constante (formato, timestamp)

La función **date** devuelve una cadena formateada según los código de formato. Si no le pasamos la variable ***timestamp*** nos devuelve la cadena formateada para la fecha y la hora actual.

Los códigos de formato para la función date son:

CODIGO	DESCRIPCIÓN
a	am o pm
A	AM o PM
d	Día del mes con ceros
D	Abreviatura del día de la semana (inglés)
F	Nombre del mes (inglés)
h	Hora en formato 1-12
H	Hora en formato 0-23
i	Minutos
j	Día del mes sin ceros
l	Dia de la semana
m	Número de mes (1-12)
M	Abreviatura del mes (inglés)
s	Segundos

y	Año con 2 dígitos
Y	Año con 4 dígitos
z	Dia del año (1-365)

Para ver algunos ejemplos supongamos que ahora es el 17 de junio de 2010 a las 14 horas 30 minutos y 22 segundos:

- `date("d-m-Y") -> 17-06-2010`
- `date("H:i:s") -> 14:30:22`
- `date("Y") -> 2010`
- `date("YmdHis") -> 2010617143022`
- `date("d/m/y H:i a") -> 17/06/10 14:30 pm`
- `date(d-m-Y H:i, time()) -> el momento actual`

mktime(hora, min, seg, mes, dia, año)

La función ***mktime*** devuelve una variable de tipo ***timestamp*** a partir de las coordenadas dadas. La principal utilidad de esta función es la de añadir o quitar una determinada cantidad de fecha u horas a una dada.

```
<?PHP
function restarDias($numdias, $date) {
   if (isset($date)) {
      $date = time();
   }
   list($hora, $min, $seg, $dia, $mes, $anno) = explodest($hora, $min,
   $seg, $dia, $mes, $anno) = explode( " ", date( "H i s d m Y"));
   $d = $dia - $numdias;
   $fecha = date("d-m-Y", mktime($hora, $min, $seg, $mes, $d, $anno));
   return $fecha;
   }
   echo restarDias(5)."<BR>"; echo restarDias(10)."<BR>";
?>
```

chekdate (mes, dia, año)

La función ***chekdate*** comprueba si una fecha es válida, si es así devuelve TRUE y si no lo es FALSE. Una fecha se considera válida si el año está entre 1900 y 32767, el mes entre 1 y 12, y el día es menor o igual que número de días total del mes en cuestión.

```
<?PHP
if (checkdate(23, 6, 2010)) {
   echo "La fecha es correcta";
} else {
   echo "La fecha es incorrecta";
}
?>
```

Para el ejemplo anterior nos daría que la fecha es incorrecta, febrero nunca tiene un día 31.

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 inglés. Por ejemplo, puede crear una marca de tiempo para enero 01 del 2011, así:

```
$Fechaclave = strtotime("january 01 2011");
```

`strtotime` reconoce las siguientes palabras y abreviaturas:

- ✓ **Nombre de meses**: Los nombres de 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**: + o –
- ✓ **Todos los numerous**
- ✓ **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.

Bloques de construcción PHP para programas

En resumen 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. 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 relaciona, tal como la descripción, la foto y e 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, con la sentencia swicht.
- ✓ **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 esto

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: 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. Escfibir un bloque de enunciados que despliegue la información del producto y usar ese bloque repetidmente, es muyco más eficiente que escribit 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.

Funciones de PHP/MySQL

PHP y MySQL funcionan muy bien juntos. Una de las características más fuerte 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 consulta SQL a la base de datos. No hace falta conocer los detalles de la comunicación co MySQL; PHP se encarga de eso. Usted nada más necesita saber sobre las consultas SQL y cómo usar las funciones de PHP.

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.

Las funciones de PHP que se usan con MySQL 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 verán:

```
mysql_connect($conectar);
mysql_query("enunciado SQL",$conectar);
```

Hacer una conexión

Antes de poder almacenar o extraer datos, usted debe conectarse a la base de datos. Todo lo que necesita saber es el nombre y la ubicación de la base de datos.

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. La conexión permanece abierta hasta que usted específicamente la cierre o el programa termine.

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 de la computadora 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 ("dirección","cuentamysql","password")
       or die ("mensaje");
```

- ✓ `dirección`: El nombre de la computadora donde MySQL está instalado; por ejemplo: `servidordb.miempresa.com`. Pero si la base de datos MySQL está en la misma computadora, use `localhost`. Si esta información está en blanco (""), PHP asume que es `localhost`.

- ✓ `cuentamysql`: Es el nombre de cuenta MySQL. Puede dejar esta información en blanco (""), lo cual significa que cualquier cuenta puede conectarse; pero, esa es una mala idea por razones de seguridad.
- ✓ `password`: Es la contraseña de su cuenta MySQL. Si su cuenta MySQL no requiere de una contraseña, no digite nada entre la comillas: "".
- ✓ `mensaje`: El mensaje que se envía al buscador si la conexión falla. La conexión falla si su computadora o la red están caídos o si el servidor MySQL no está corriendo. También puede fallar si la información brindada no es la correcta.

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 cierrar así:

```
mysql_close($nombreconexion);
```

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");
```

- ✓ `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.

Por ejemplo, usted puede seleccionar la base de datos `Catalogodemascotas` con el siguiente enunciado:

```
$db = mysql_select_db("Catalogodemasctas",$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, se recomienda 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");
```

Enviar consultas SQL

Una vez abierta la conexión con el servidor MySQL, y cuando PHP sepa con cuál base de datos desea interactura, usted envía su consulta SQL. La consulta es una solicitud al servidor MySQL de que almacene, actualice o recupere algunos datos. (Consulte la información de la tema 2).

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 seleccionar la base de datos");
```

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 seleccionar la base de datos");
```

La variable 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 `$resultado` variable confiene información sobre si la consulta se ejecutó satisfactoriamente o no. Si fue exitosa, `$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 lo datos obtenidos en sí. Algunas consultas que sí devuelven datos son SELECT y SHOW.

El uso de comillas dobles y simples pueden resultar algo confuso al asignar la cadena de consulta a `$consulta`. En realidad, las comillas se usan en dos niveles: las comilla necesarias para asignar la cadena a `$consulta` y las comillas que son aparte de la consulta en lenguaje SQL. Las siguientes reglas le ayudarán a evitar problemas con las comillas:

- ✓ Use comillas dobles a comienzao 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 = 'Pérez'";
$consulta = "UPDATE Miembro SET apellido = '$apellido'";
```

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 consulte 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

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 anteiores, el identificador de `resultado` se pono en la variable `$resultado`. El siguiente paso después de ejecutar la función es mover los datos desde su ubicación temporal hacia variables que se pueden usar en le 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 consuta `select` podría dar como resultado más de una fila de datos en la tabla temporal. 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 `mysql_fetch_array` es:

```
$fila = mysql_fetch_array ($identificadoderesultado, 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`.

- ✓ `($identificadoderesultado`: La variable que señala hacia la ubicación temporal de los resultados.
- ✓ `tipodearreglo`: El tipo de serie en la cual se ponen los resultados. Pueden ser uno de dos tipos de series o ambos tipo. 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 fial, 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 com clave y uno con el nombre dela columna como clave. Si no se especifica el tipo de serie en el llamdo 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 usario en
el formulario
$consulta = "SELECT contraseña FROM Miembro
         WHERE nombreentrada =' glopez ' ";
$resultado = mysql_query($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 páginas web exclusivas para miembros
        }
else
        {
  echo "Clave inválida<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 `glopez`).
- ✓ 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.

```
$entradadelusuario = "secreta": //contraseña digitada por el usario en
el formulario
$consulta = "SELECT contraseña FROM Miembro
         WHERE nombreentrada =' glopez ' ";
$resultado = mysql_query($consulta);
         or die ("No se pudo ejecutar la consulta.");
$fila = mysql_fetch_array($resultado,MYSQL_ASSOC);
extract($fila);
         if ($entradadelusuario == $clave
        {
echo "Entrada aceptada<br>");
  enunciados que muestran las páginas web exclusivas para miembros
        }
else
        {
  echo "Clave inválida<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 fial de datos y la procesa. El ciclo se repite hasta que todas las filas se hayan recuperado. Puede usar un ciclo while o un ciclo for para recuperar la información.

La manera más común de procesar la información es usar el ciclo while como sigue:

```
while ( $fila = mysql_fetch_array($resultado))
{
   bloque
```

```
    }
```

Este ciclo se repite hasta haver 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`, `Tipomascota`, `Descripcionmascota` y `precio`. La tabla siguiente muestra un conjunto de datos de muestra para la tabla `Mascota`

Datos de muestra para la tabla Mascota			
Nombremascota	**Tipomascota**	**Descripcionmascota**	**Precio**
Pony	Caballo	Muy pequeño; la mitad de un caballo estándar.	5000
Siamés	Gato	Pequeño, pelo café/negro	2000
Angel	Pez	Extremidades finas como de velo	200
San Bernardo	Perro	Tamaño grande, pelo crinado	3000
Burro	Caballo	Tamaño mediado, peludo, orejas largas	1300
Perico	Ave	Mediano, plumas casi siempre verde	500
Iguana	Reptil	Tamaño regular, cuerpo alargado	300

El programa `listademascotas.php`, en la tabla anterior selecciona todos los caballos de la tabla ***Mascotas*** y despliega la información en una tabla HTML en la página web. La variable `$tipomascota` contiene información que un usuario digitó en un formulario.

listademascotas.php	Ver documento anexo de código fuente

Para observar los resultados, debe tener creada la bases de datos ***CatalogodeMascotas***, la tabla ***Mascotas*** y tener los registros de la tabla anterior.

La figura 2 muestra la página web desplegada por el programa `listadeMascotas.php`. La página web muestra los elementos `Mascot`a para el `Tipomascota` caballo; el despliegue está formateado con una tabla HTML.

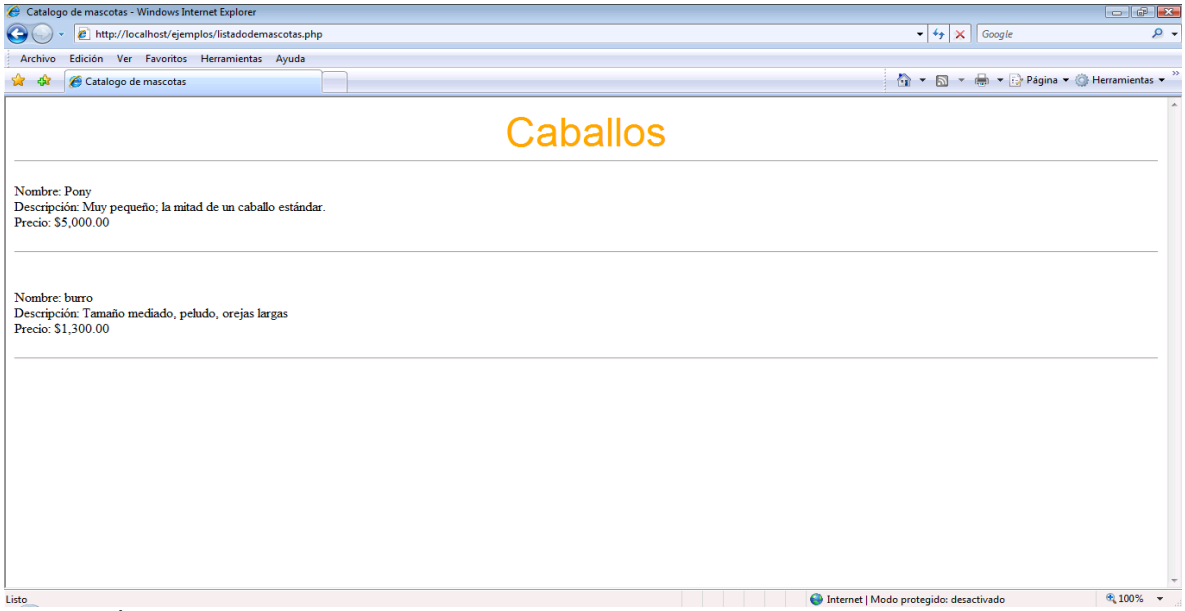

Figura 2 Página Web resultante de ***listadodemascotas.php***

El programa `listadodemascota.php` usa un ciclo `while` para extraer todas las filas de la ubicación temporal. En algunos casos, quizás usted necesite usar el ciclo `for`. Por ejemplo, si debe usar un número en su ciclo es más útil que un `while`.

Para usar un ciclo `for`, necesita saber cuántas filas de datos de 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 `listadeMascotas.php` muestra los elementos Mascota del tipo Caballo. Suponga que usted desea enumerar cada elemento. El siguiente programa listanumerada.php, despliega una lista numerada usando un ciclo for.

listanumerada.php	Ver documento anexo de código fuente

La figura 3 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.

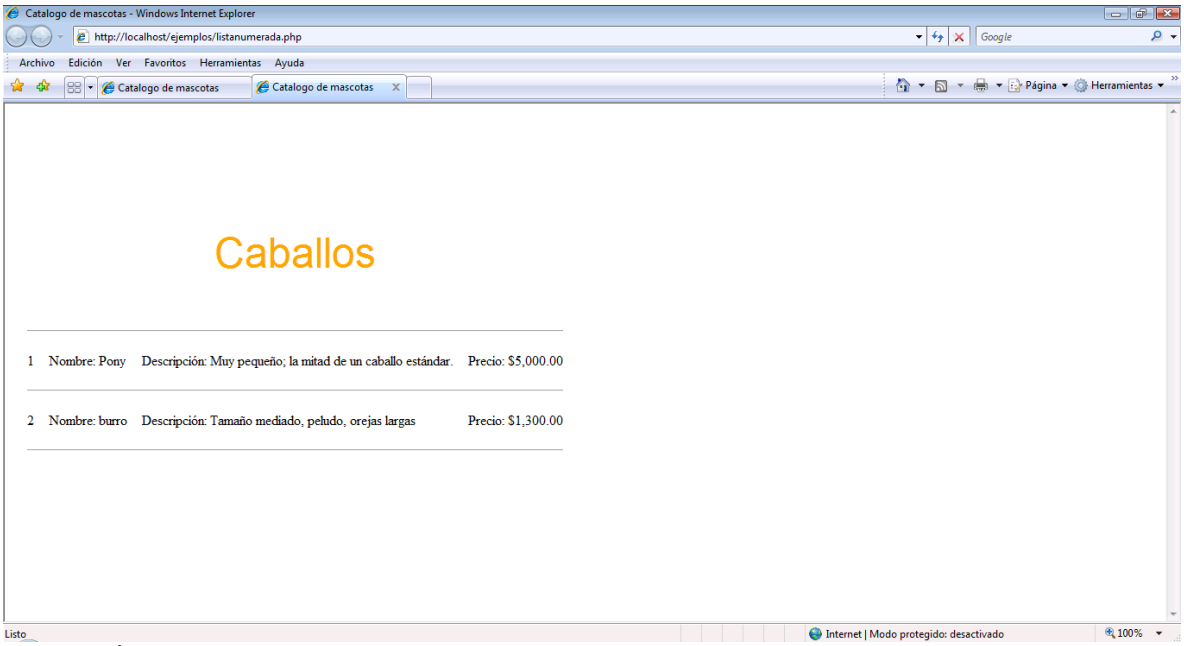

Figura 3 Página Web resultante de ***listanumerada.php***

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 de su aplicación. Las funciones (bloques de enunciados que realizan ciertas tareas específicas) están diseñadas para tales situaciones.

Por ejemplo, considere el 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 lo necesite.

El programa `obtienedatos.php` muestra cómo usar una función para extraer datos. La función 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 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.

obtienedatos.php	Ver documento anexo de código fuente

La página web muestra:

Siamés
Descripción: Pequeño, pelo café/negro
Precio: $2000.00

Observe lo siguiente sobre el programa

- ✓ 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 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 "`Siamés`". 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` siguiente usa una función para el mismo propósito. La función devuelve un arreglo multidimensional con los datos de mascotas para todas las mascotas del tipo especificado.

obtienemascota.php	Ver documento anexo de código fuente

El programa `obtienemascota.php` procede como sigue:

1. **Se conecta con el servidor MySQL en el programa principal.**
2. **Llama a la función** `extraertipodemascota`. Pasa "`Caballo`" como una cadena de caracteres y también establece `$Info_mascota` para que pueda recibir los datos devueltos por la función.
3. **La función selecciona la base de datos** `Catalogodemascotas`.
4. **La función envía una consulta para extraer todas las filas con** `Caballo` **en la columna** `Tipodemascota`. Los datos se almacenan en una tabla en una ubicación temporal. La variable `$resultado` identifica la ubicación de la tabla temporal.
5. **Establecer un contador.** `$j` es un contador que se incrementa en cada ciclo. Empieza el 1 antes del ciclo.
6. **Inicia un ciclo** `while`. La función intenta extraer una fila de la tabla temporal de datos y resulta exitosa. Si no hubiera filas para extraer en la ubicación temporal, el ciclo `while` terminaría.
7. **Inicio un ciclo** `foreach`. El ciclo se mueve por la fila y procesa cada campo.
8. **Almacena los valores en el arreglo.** `$Info_mascota` 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 `$j`, es ahora igua a 1. Todos los campos en la fila se almacenan en `$Info_mascota` con el nombre de la columna como clave.
9. **Incrementa el contador.** `$j` aumenta en 1.
10. **Llega al final del ciclo** `while`.
11. **Regresa al inicio del ciclo** `while`.
12. **Repite los paso 6 al 11 para cada fila en los resultados.**
13. **Devuelve** `$serie` **al programa principal.** `$serie` contiene todos los datos para todas las filas seleccionadas.
14. `$Info_mascotas` **recibe datos de la función.** Todos los datos se pasan.
    La figura 4 muestra la estructura de `$Info_mascota` después de que la función ha terminado de ejecutarse.

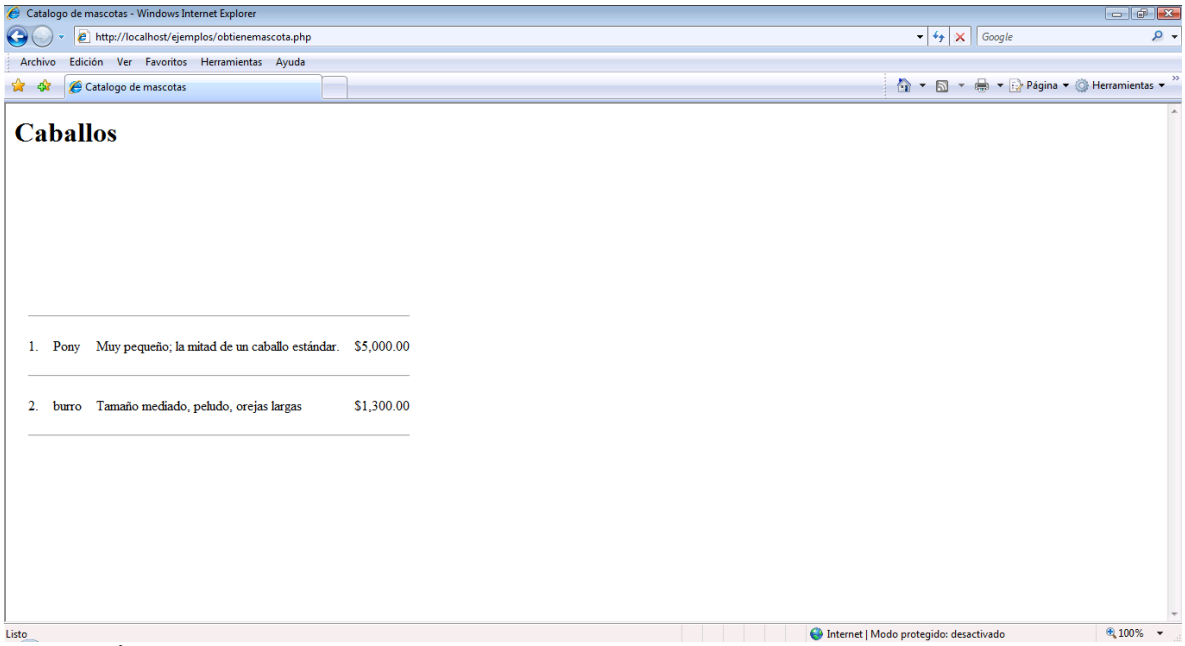

Figura 4 Página Web resultante de ***obtienemascota.php***

15. **El programa principal envía las Descripciones de las mascotas a explorador en una tabla HTML**. Los datos apropiados se insertan desde el arrego `$Info_mascota`.

Tema 4

Usar formularios HTML

Obtener información del usuario

Muchas aplicaciones están diseñadas para formular preguntas que los usuario responden digitando información. A veces, la información se almacena en una base de datos; otras, la información se usa en enunciados condicioneales 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**
- ✓ **Inscribirse**
- ✓ **Registrarse**
- ✓ **Ver información seleccionada**

Las preguntas se formulan mediante formularios HTML. El usuario responde las preguntas digitando información en el formulario o seleccionando elementos 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, e 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. 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.

Los formularios HTML son muy importantes para los sitios web interactivos. 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 sección PHP
?>
<html>
   <form action = "procesaformulario.php" method = "POST">
     <input type = "text" nombre = "Nombre">
     <input type = "submit" valor = "enviar nombre">
   </form>
</html>
<?php
    //enunciados en la sección PHP
?>
```

Cualquiera de estos métodos produce el formulario de la figura 5.

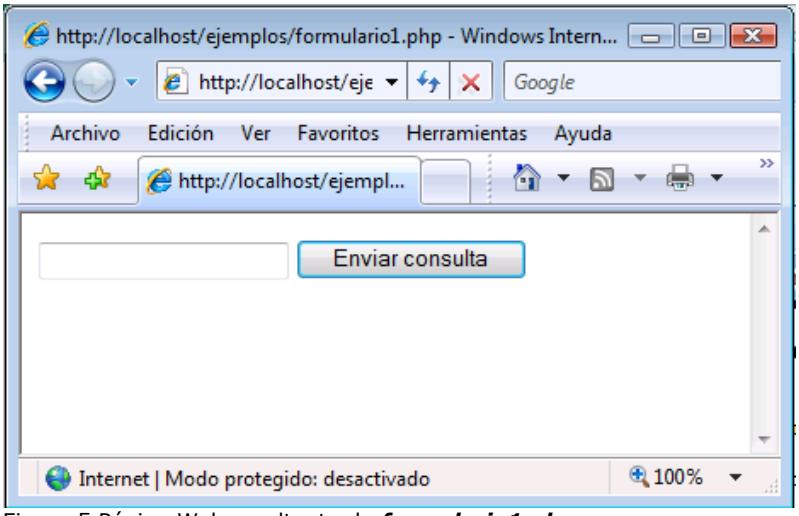

Figura 5 Página Web resultante de ***formulario1.php***

Cuando el usuario hace clic en el botón Enviar consulta (`type = "submit"`), el programa `procesaformulario.php` especificado en `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 siguiente tabla muestra las series incorporadas que contienen la información del formulario.

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, $_COOKIE.

En estos arreglos incorporados, cada índice de serie es el nombre del campo de entrada del formulario. Por ejemplo, si el usuario digitó `Juan López` en el campo de entrada del formulario en la figura 5 e hizo clic en el botón `"Enviar consulta"`, el programa `procesaformulario.php` corre y puede usar una variable de arreglo en el siguiente formato:

```
$_POST['Juan López]
```

Un formulari del tipo POST son desplegados por el programa `procesarformulario.php`. Cuando el formulario es enviado, se correl el siguiente programa:

```
<?php
  /* Nombre del programa: procesaformulario.php
   * Descripción: El programa muestra toda la información 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>
```

El programa anterior, se escribe para procesar la información de cualquier formulario que use el método `POST`. Suponga que usted tiene un formulario levemente más complicado, tal como el programa `muestraformulario.php`:

muestraformulario.php	Ver documento anexo de código fuente

Observe lo siguiente en cuanto a muestraformulario.php, tal y como aparece:

- ✓ **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 formate con una tabla HTML.** Las tablas son una parte importante del HTML.
- ✓ **El script hace un ciclo por el arreglo $etiquetas con un enunciado foreach.** El código HTML para una fila de la tabla es la salida de cada ciclo. Los valores de serie apropiados se usan en el código HTML.

Por razones de seguridad incluya `maxlength` (define el número de caracteres que los usuarios pueden digitar en el campo) en su enunciado HTML. 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 el usuario llena el formulario mostrado en la figura 6 y lo envía, el programa `procesaformulario.php` corre y produce la siguiente salida:

```
primernombre = Ana
segundonombre = de la Cruz
apellido = Feria
calle = Av. Morelos 123
estado = Ecatepec
codigopostal = 55123
```

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.

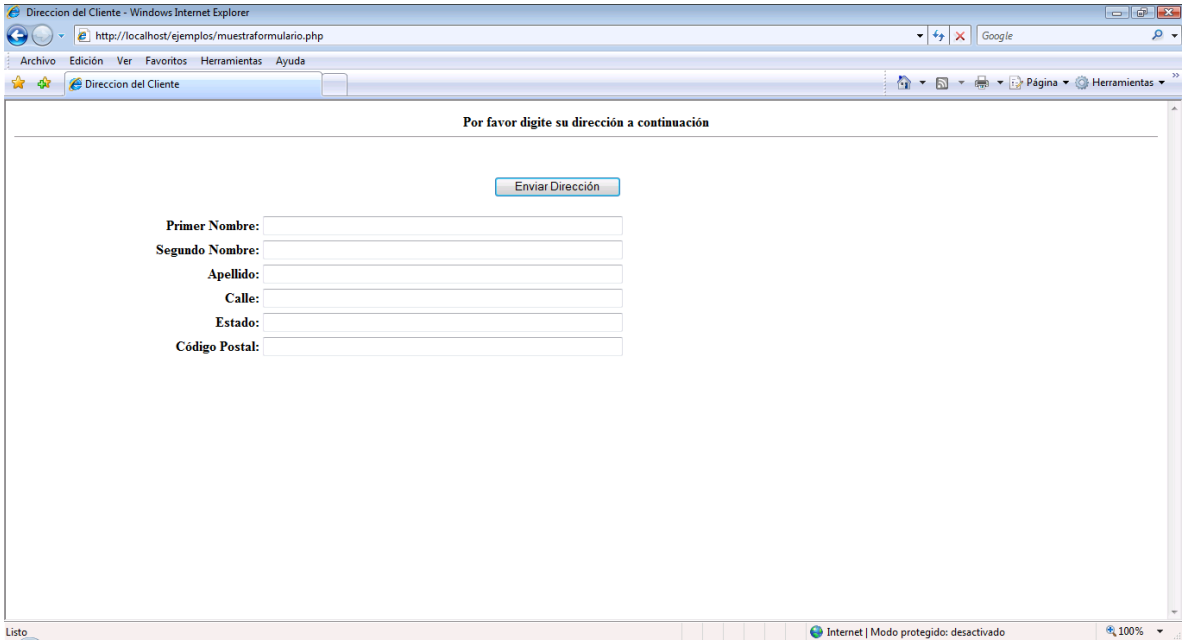

Figura 6 Página Web resultante de ***muestraformulario.php***

Hacer que los formularios sean dinámicos

Si tiene información de los usuarios almacenada en una base de datos. 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. El programa `muestradirección.php`, el cual despliega un formulario con información de la base de datos. Este formulario es muy parecido al formulario mostrado en la figura 6, excepto que este formulario tiene información en el (recuperada de la base de datos) y los campos en el formulario de la figura 6 están vacíos.

muestradireccion.php	Ver documento anexo de código fuente

Observe lo siguiente de la ejecución del programa `muestradireccion.php`:

- ✓ **El enunciado del formulario tansfiere 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. Verificar los datos en un formulario y guardar información en la base de datos se discute más adelante, en la sección "`Revisar la Información`" y "`Poner información en una base de datos`".
- ✓ **Cada campo de entrada en el formulario recibe un nombre.** La información en el campo de entrada se almacena en una variable que tiene el mismo nombre que el campo de entrada.
- ✓ **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 entrada.** El parámetro `valor` muestra el valor apropiado del arreglo `$fila`, la cual contiene datos de la base de datos.

La figura 7 muestra la página web resultante del programa `muestradireccion.php`. La información en el formulario es la información que está almacenada en la base de datos.

Figura 7 Página Web resultante de ***muestradireccion.php***

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 municipios 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, programa `lista1.php`:

listas1.php	Ver documento anexo de código fuente

La figura 8, muestra la lista de selección que estos enunciados producen, 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 9, y el usuario puede seleccionar cualquier elemento de la lista.

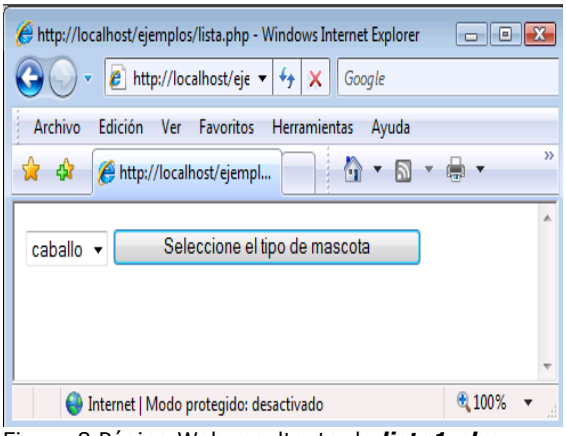

Figura 8 Página Web resultante de ***lista1.php***

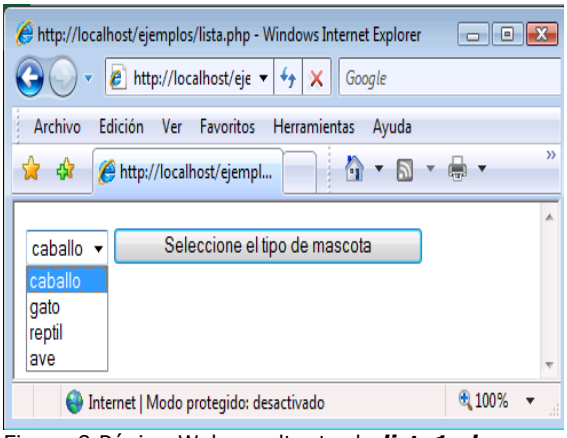

Figura 9 Página Web resultante de ***lista1.php***

Sin embargo, con las variables las varibles 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.

El programa `construirselect.php` construye una lista de selección de categorías de mascotas a partir de la base de datos.

construirselect.php	Ver documento anexo de código fuente

Observe lo siguiente del programa construirselect.php:

- ✓ **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 mascotas varias veces si apareciera varias veces en la base de datos.
- ✓ **Uso de** `ORDER BY` **en la consulta**: Los tipos de mascotas se ordenan alfabéticamente.
- ✓ **Enunciado** `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 opiton 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.
- ✓ **Enunciado** `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 desplegable producida por este programa está en orden alfabético, como se aprecia en la figura 10:

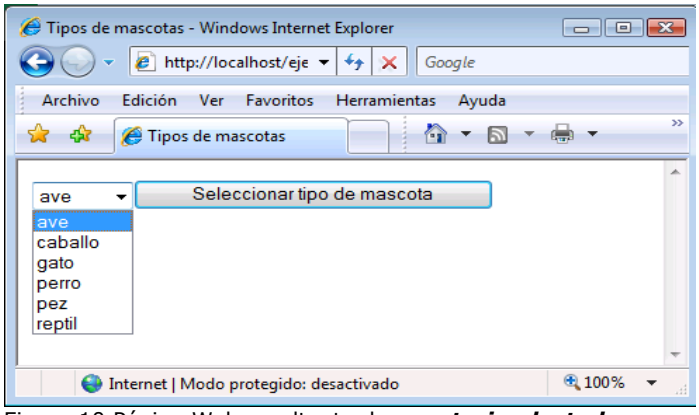

Figura 10 Página Web resultante de ***construirselect.php***

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, y quiere que la fecha de hoy esté seleccionada en forma predeterminada cuando la casilla se despliegue. El programa `seleccionar fecha. php`, el cual despliega un formulario para seleccionar una fecha y selecciona la fecha de hoy automáticamente.

seleccionarfecha.php	Ver documento anexo de código fuente

La página web producida por el programa `seleccionarfecha.php` se muestra en la figura 11. La fecha apareces 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 desplegarse. La lista de selección para el día muestra los 31 días al desplegarse. La lista de selección para el año muestra cinco años.

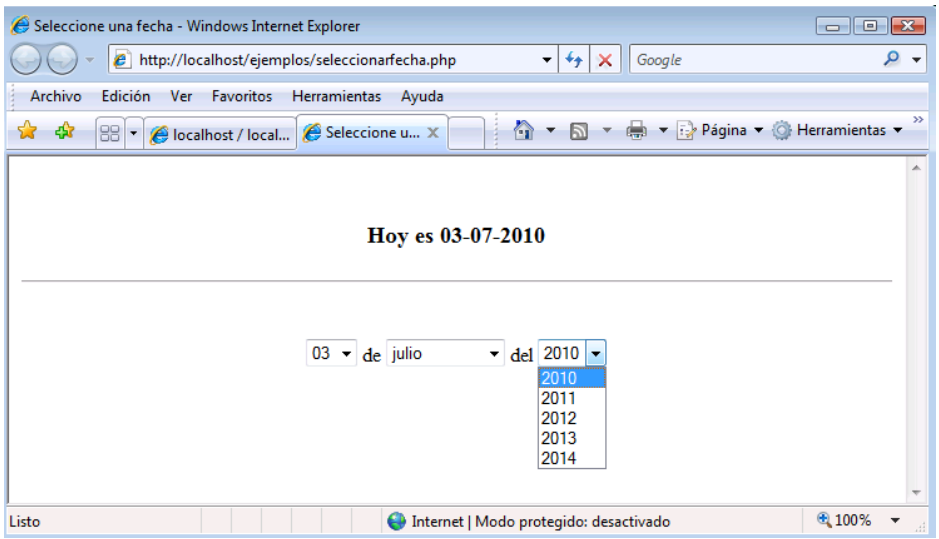

Figura 11 Página Web resultante de ***seleccionarfecha.php***

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 concuerdan con los números de los meses.
2. **Crea variables que contienen la fecha actual**. `$hoy` contiene la fecha actual formateada que se usa para mostrar la fecha en la página web.
3. **Muestra la fecha actual en la parte superior de la página web.**
4. **Construye el campo de selección para el día**. Usa el procedimiento descrito para el mes. Sin embargo, sólo se usan números para esta lista de selección. El ciclo se repite 31 veces.
5. **Construye el campo de selección para el mes.**
    - **a.** Crea una variable que contiene el día actual.
    - **b.** Hace `echo` de la etiqueta `select`, a la cual debería hacérsele `echo` una sola vez.
    - **c.** Emplieza un ciclo `for` que se repite 31 veces.
    - **d.** Dentro del ciclo, hace `echo` de la etiqueta opción usando el primer valor del arreglo `$nombremes`.
    - **e.** Si el número del mes que se está procesando es igual al número del mes actual, añade la palabra `"selected"` a la etiqueta `option`.
    - **f.** Repite el ciclo 11 veces más.
    - **g.** Hace `echo` de la etiqueta `select` de cierre para el campo de selección, a la cual debería hacérsele `echo` una sola vez.
6. **Construye el campo de selección para el año.**
    a. Crea la variable `$annioinicio`, la cual contiene el año actual.
    b. Hace `echo` de la etiqueta `select`, a la que se le debe hacer `echo` sólo una vez.
    c. Inicia el ciclo `for`. El valor inicial para el ciclo es `$annioinicio`. El valor final para el ciclo es `$annioinicio + 4`.
    d. Dentro del ciclo, hace `echo` de la etiqueta `option`, usando el valor inicial del ciclo `for`, que es el año actual.
    e. Si el número de años que se está procesando es igual al número del mes actual, añade la palabra `"selected"` a la etiqueta `option`.
    f. Repite el ciclo hasta que el valor final sea igual a `$annioinicio + 4`.
    g. Hace `echo` de la etiqueta de cierre `select` para el campo de selección, a la que se debe hacer `echo` una sola vez.
7. **Hace echo de la etiqueta final para el formulario**

Crear listas de botones de opción

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 mascotas de su interés.

El formato para los botones de opción en los formularios es:

```
<input type="radio" name = "mascotas" value = "caballo">
```

Puede construer 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 secció anterio. El programa `construyeradio.php`, el cual crea una lista de botones de opción con base en los tipos de mascotas.

construyeradio.php	Ver documento anexo de código fuente

Este programa es muy parecido al de la figura 12.

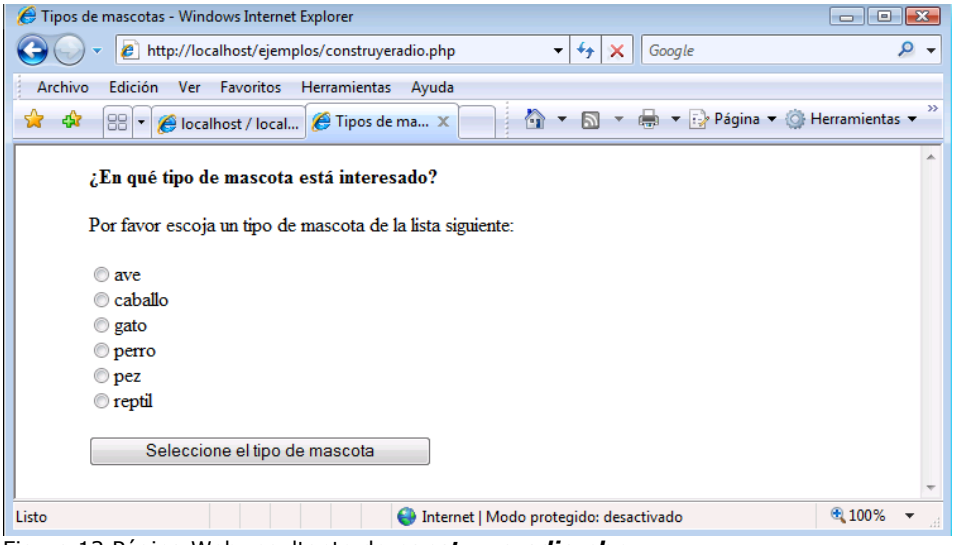

Figura 12 Página Web resultante de ***construyeradio.php***

Construir listas de casillas para marcar

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 `construyecheckbox.php` crea una lista de casillas para marcar.

construyercheckbox.php	Ver documento anexo de código fuente

Este programa es muy similar al programa anterior que construye una lista de botones de opción. Sin embargo, observe que el campo de salida usa un arreglo: `interes['$Tipomascota']`. 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. La figura 13, muestra la página web producida.

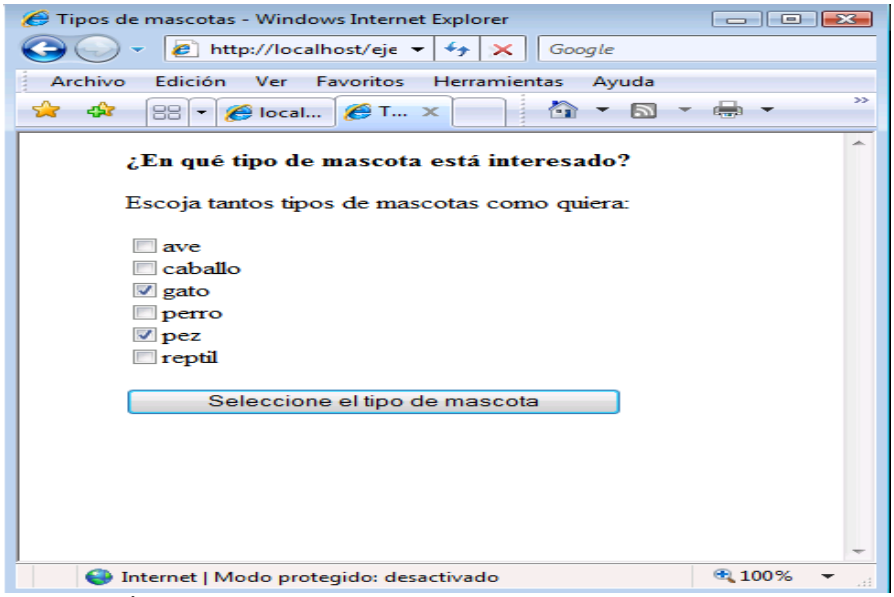

Figura 13 Página Web resultante de ***construyecheckbox.php***

Revisar la información del formulario

El usuario completa un formulario HTML, tal vez con errores, o quizá haya digitado algo sin sentido o maliciosa que puede causarle problemas a usted o a otras personas que usan su sitio web. Antes de utilizar la información del usuario o almacenarla en su base de datos, es mejor revisarla, para cerciorarse de que sea la información solicitada, es decir, "Validar" la información.

Validar los datos incluye lo siguiente:

- ✓ **Revisar en busca de campos vacíos.** Si está en blanco se le dice al usario que la información es obligatoria, y el formulario reaparece para que el usuario pueda digitar la información faltante.
- ✓ **Verificar el formato de la información.** Revisar si está en el formato correcto.

Revisar en busca de campos vacíos

Cuando usted crea un formulario, puede decidir cuáles campos son obligatorios y cuáles son opcionales. En general para revisar en busca de campos vacíos es:

```
if ($Nombreentrada =="")
{
   echo "No digitó su Nombre de entrada. Es obligatorio. <br>\n";
   mostrar el formulario;
   exit();
}
echo "Bienvenido al club exclusivo para miembros.";
```

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 decir, sin el enunciado `exit`, el programa desplegaría el formulario y luego continuaría haciendo eco del enunciado de bienvenida.

En muchos casos, usted quiere verificar todos los campos en el formulario. Puede hacerlo a 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 campo<br>\n";
   mostrar el formulario;
   exit();
}
echo "Bienvenido";
```

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 usuario que tiene 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 campo<br>\n";
   mostrar el formulario;
exit();
      }
   }
}
echo "Bienvenido";
```

Observe que el enunciado condicional `if` externo es verdaderamente sólo si el campo no es e campo fax y no es el campo segundo nombre. Para estos campos, el programa no llega al enunciado `if` interno, el cual revisa en busca de campos en blanco.

El programa `chequeodedatos.php` procesa un formulario con cuatro campos: `primernombre`, `segundonombre`, `apellido` y `teléfono`. Todos los campos son obligatorios, excepto `segundonombre`.

chequeodedatos.php	Ver documento anexo de código fuente

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 `$arreglo_blanco`.
4. **Si no se encontró ningún campo en blanco, salta al mensaje de bienvenida.**
5. **Si encontró uno o más campos en blanco:**
   a. Muestra un mensaje de error. Este mensaje explica al usuario que falta alguna información obligatoria.
   b. Muestra una lista de información faltante. Hace un ciclo por `$serie_blanco` y muestra la(s) etiqueta(s).
   c. 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.

d. 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 14, 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.

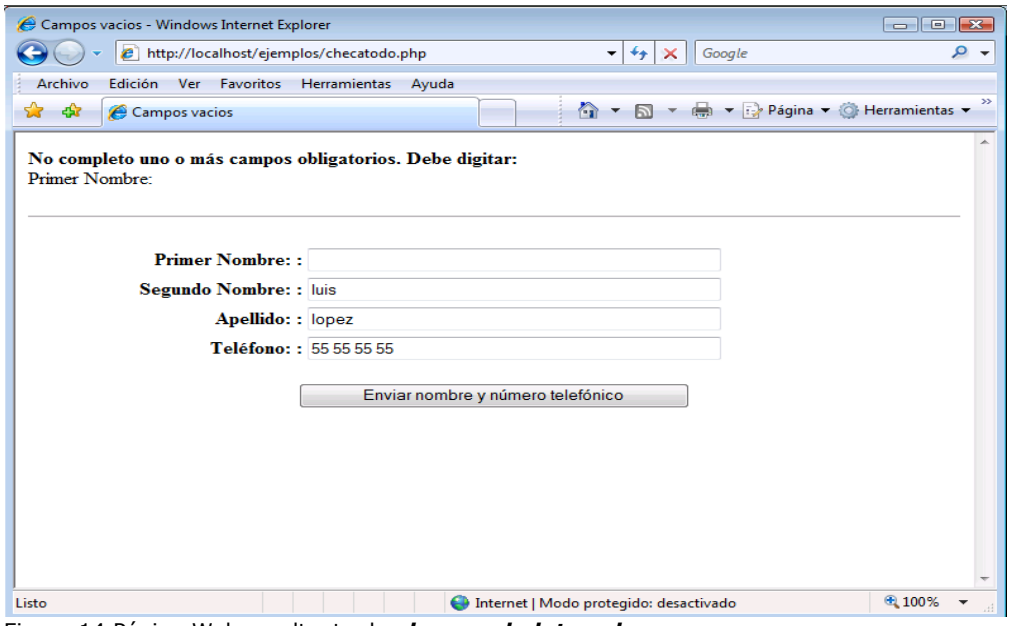

Figura 14 Página Web resultante de ***chequeodedatos.php***

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 cuando el formulario es enviado, hacérselos saber al usuario y luego solicitarle que vuelva a digitar la información.

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. Una etiqueta particularmente peligrosa sería una etiqueta de script que permite a un usuario introducir un programa en el campo de un formulario.

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.

En general, estos son los enunciados que se usan para revisar campos:

```
if ( !ereg ( " patrón ", $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 entrada 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. Además, es difícil imaginar un nombre que tenga más de 50 caracteres. Así, usted puede usar los siguientes enunciados para checar 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 formate aceptable. Puede checar el formato haciendo unos cuantos cambios simples al programa `chequeodedatos.php`, ahora en el programa `checatodo.php`

checatodo.php	Ver documento anexo de código fuente

Estas son las diferencias entre este programa y el programa anterior:

- ✓ **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 15, es el resultado obtenido cuando el usuario accidentalmente digita su primer nombre en el campo del segundo nombre y también digita sin sentido el número telefónico. Observe que aparecen dos mensajes de error, los cuales muestran que el campo del primer nombre está en blanco y el camp del número telefónico contiene información incorrecta.

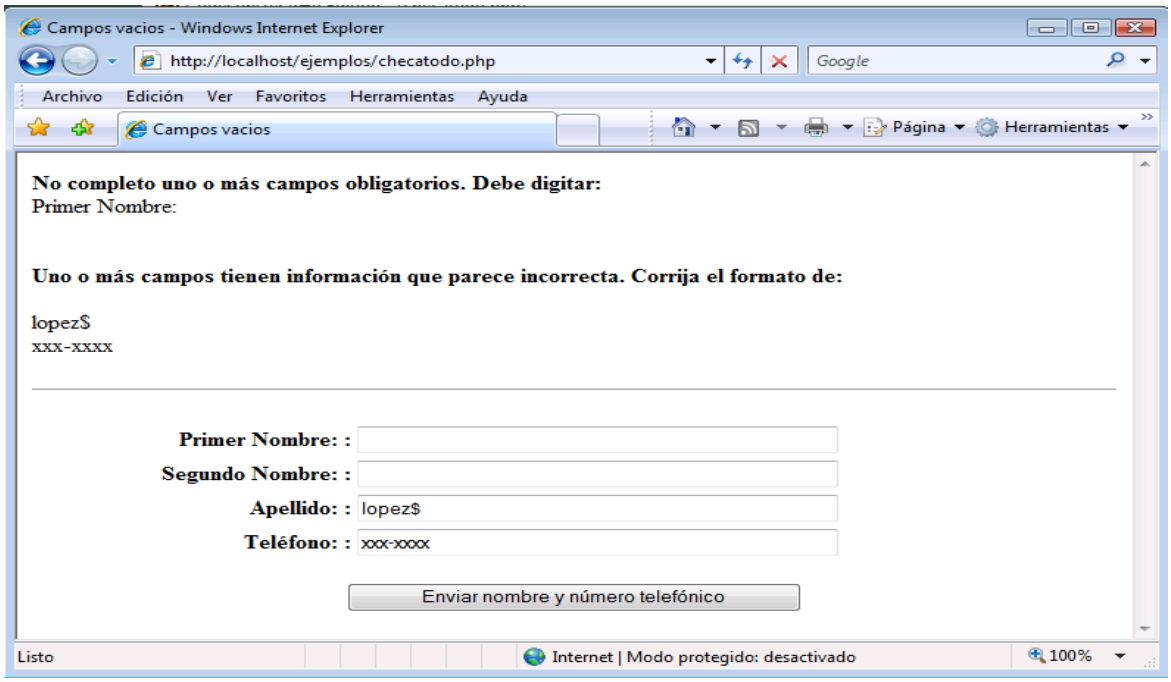

Figura 15 Página Web resultante de ***checatodo.php***

Descarga de archivos con formularios

Vamos a ver un caso especial, como descargar un archivo desde un formulario. Para ello utilizaremos una etiqueta ***INPUT*** de tipo ***FILE***, soportada a partir de las versiones de los navegadores Nestcape Navigato 2.0 e Internet Explorer 4.0.

El formulario debe usar el método ***post***, y el atributo enctype debe tener el valor ***multipart/form-data***. Además al formulario debemos añadirle un campo oculto de nombre ***MAX_FILE_SIZE***, al cual le daremos el valor en byte del tamaño máximo del archivo a descargar.

```
<FORM ENCTYPE="multipart/form-data" ACTION="7-3.php" METHOD="post">
   <INPUT TYPE="hidden" name="MAX_FILE_SIZE" value="100000">
   <INPUT NAME="archivo" TYPE="file">
   <INPUT TYPE="submit" VALUE="Descargar Archivo">
</FORM>
```

Cuando el formulario es enviado, PHP detectará automáticamente que se está descargando un archivo y lo colocará en un directorio temporal en el servidor. Dicho directorio será el que esté indicado en el archivo de configuración ***php.ini***, o en su defecto en el directorio temporal del sistema.

Cuando PHP detecta que se está descargando un archivo crea varias variables con el prefijo del nombre del archivo pero con distintas terminaciones. La variable terminada en ***_name*** contiene el nombre original del archivo, la terminad en ***_size*** el tamaño en bytes de éste, y la variable terminada en ***_type*** nos indicará el tipo de archivo si éste es ofrecido por el navegador.

Si el proceso de descarga no ha sido correcto la variable archivo tomará el valor ***none*** y ***_size*** será `0`, y si el proceso ha sido correcto, pero la variable terminada en ***_size*** da `0`, quiere decir que el archivo a descargar supera el tamaño máximo indicado por ***MAX_FILE_SIZE***.

Una vez descargado el archivo, lo primero que debemos hacer es moverlo a otro lugar, pues sino, no se hace nada con él, cuando acabe la ejecución de la página se borrará.

Veamos un ejemplo de todo lo dicho.

```
<HTML>
<BODY>
<?PHP
if ($enviar) {if ($enviar) {
   if ($archivo != "none" AND $archivo_size != 0){
      echo "Nombre: $archivo_name <BR>\n";
      echo "Tamaño: $archivo_size <BR>\n";
      echo "Tipo: $archivo_type <BR>\n";
      /* para Windows
      if (! copy ($archivo, "C:\\TEMP\\".$archivo_name)) {
         echo "<h2>No se ha podido copiar el archivo</h2>\n";
      }
      */
      /* para Linux/Unix */
      if (! copy ($archivo, "/tmp/".$archivo_name)) {
         echo "<h2>No se ha podido copiar el archivo</h2>\n";
      }
   } elseif ($archivo != "none" AND $archivo_size == 0) {
      echo  "<h2>Tamaño  de  arcft:  75">echo  "<h2>Tamaño  de  archivo
      superado</h2>\n";
   } else {
      echo "<h2>No ha escogido un archivo para descargar</h2>\n";
   }
```

```
   echo "<HR>\n";
}
?>
<FORM ENCTYPE="multipart/form-data" ACTION="<?php echo $PHP_SELF ?>"
METHOD="post">
   <INPUT type="hidden" name="MAX_FILE_SIZE" value="100000">
   <p><b>Archivo a descargar<b><br>
   <INPUT type="file" name="archivo" size="35"></p>
   <p><INPUT type="submit" name="enviar" value="Aceptar"></p>
</FORM>
</BODY>
</HTML>
```

Funciones de acceso a ficheros

Posiblemente durante nuestra tarea de programación nos surja la necesidad de obtener datos de un fichero, o bien, de crear uno. PHP nos provee de una extensa gama de funciones de acceso a ficheros.

En esta lección sólo vamos a las funciones básicas, abrir ***(fopen),*** cerrar **(fclose),** leer ***(fgets)*** y escribir ***(fputs).*** Estas cuatro nos solventaran la mayoría de problemas que nos surjan con respecto al acceso a ficheros.

fopen (archivo, modo)

Con esta función abrimos un fichero, bien sea local o una dirección de internet (http:// o ftp://).

La función ***fopen*** nos devuelve un valor numérico (indicador de archivo) de tipo ***integer*** que nos servirá para hacer referencia al archivo abierto.

Con ***fopen*** podemos abrir un archivo de los siguientes modos:

r solo lectura
r+ lectura y escritura
w solo escritura. Si no existe el archivo lo crea, si ya existe lo machaca.
w+ lectura y escritura. Si no existe el archivo lo crea, si ya existe lo machaca.
a solo lectura. Si no existe el archivo lo crea, si ya existe empieza a escribir al final del archivo.
a+ lectura y escritura. Si no existe el archivo lo crea, si ya existe empieza a escribir al final del archivo.

```
<?PHP
//abreo"><?PHP
//abre un archivo utilizando el protocolo HTTP
if ( ! fopen("http://www.ciberaula.com/", "r")) {
   echo "El archivo no se puede abrir\n";
   exit;
}
?>
```

Los modos ***r, r+ , w , w+*** colocan el puntero de lectura/escritura a principio del fichero, los modos ***a , a+*** lo colocan al final.

fgets (indicador_archivo, longitud)

La función ***fgets*** nos devuelve una cadena con la longitud específica del fichero al que apunta el indicador de archivo.

```
<?PHP
//abre un archivo e imprime cada linea
$archivo = fopen("dlass="codigo">$archivo = fopen("data.txt" , "r");
```

```
if ($archivo) {
   while (!feof($archivo)) {
      $linea = fgets($archivo, 255);
      echo $linea; }
}
fclose ($archivo);
?>
```

La función ***feof*** devuelve TRUE si puntero de lectura/escritura se encuentra al final del fichero, y FALSE en caso contrario.

fputs (indicador_archivo, cadena)

La función ***fputs*** escribe una cadena en el fichero indicado. Para escribir en una archivo este debe haber sido previamente abierto. La función ***fputs*** devuelve TRUE si se ha escrito con éxito, en caso contrario devuelve FALSE.

```
<?PHP
//abre un archivo y escribe en él
$archivo = fopen("data.txt" , "w");
if ($archivo) {
   fputs ($archivo, "Hola Mundo");
}
fclose ($archivo);
?>
<);
?>
```

fclose (indicador_archivo)

Con esta función cerramos el fichero que nos marca el indicador de archivo, devuelve TRUE si el fichero se cierra correctamente y FALSE sino se ha podido cerrar.

file_exists (fichero)

Esta función devuelve TRUE si el archivo especificado existe, y FALSE en caso contrario.

```
<?PHP
if (file_exists("data.txt")) {
   echo "El fichero existe";
} else {
   echo "El fichero NO existe";
}
?>
```

copy (origen, destino)

La función ***copy*** copia un fichero de un lugar (origen) a otro (destino), devuelve TRUE si la copia a tenido éxito y FALSE en caso contrario.

```
<?PHP
if (copy("data.txt", "/tmp/data.txt")) {
   echo "El fichero ha sido copiado con éxito";
} else {
   echo "El fichero NO se higo" style="margin-left: 50">echo "El
   fichero NO se ha podido copiar";
}
?>
```

Tema 5

Proyecto

´

Desarrollar la aplicación

Archivo HTML para la página inicial de la tienda. El programa `tiendamascota.php`, el cual almacena un nombre y número telefónico de un formulario:

tiendamascota.php	Ver documento anexo de código fuente

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 se inicia.

Mostrar tipos de mascotas

La página del tipo de mascotas, muestra al cliente una lista de todos los tipos de mascotas actualmente disponibles en el catálogo. El programa catalogo.php produce la página web del tipo de mascotas.

catalogo.php	Ver documento anexo de código fuente

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 mascotas seleccionado.

Mostrar las mascotas

La página de mascotas muestra a cliente una lista de mascotas de todas las mascotas del tipo seleccionado actualmente disponibles en al catálogo.

mostrar.php	Ver documento anexo de código fuente

Escoger categoría

Programa similiar a los ejercicios.

escogercategoria.php	Ver documento anexo de código fuente

Escoger nombre

La página de mascotas muestra a cliente una lista de mascotas de todas las mascotas del tipo seleccionado actualmente disponibles en al catálogo.

escogernombre.php	Ver documento anexo de código fuente

Agregar mascota

El último programa acepta los datos del formulario en el segundo programa. Si 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.

agregar.php	Ver documento anexo de código fuente

Archivos .inc

Todos los archivos .inc	Ver documento anexo de código fuente

Realizar el procedimiento para utilizar la sentencia ***include*** (archivos con extensión .inc), en los archivos php:

Abrir el archivo de configuración de PHP, php.ini
Modificar la línea con las sentencia para Windows:

```
;include_path = ".;c:\php\
```

Borra el caracter ; para descomentarla, quedando así:

```
include_path = ".;c:\php\includes"
```

Nota: puedes modificar la ruta a tus necesidades, o puedes utilizar la propuesta, pero debes crear las carpetas que se indican en la unidad especificada.