CAPÍTULO 11

Gestión de formularios

La forma más corriente en la que los usuarios de un sitio web interactúan con PHP y MySQL es a través de formularios HTML. Estos se introdujeron muy pronto en el desarrollo de la World Wide Web, en 1993, incluso antes de la aparición del comercio electrónico, y han seguido siendo un pilar fundamental desde entonces, debido a su simplicidad y facilidad de uso.

Por supuesto, se han hecho mejoras a lo largo de los años para añadir funcionalidad extra a la gestión de formularios HTML, por lo que este capítulo te pondrá al día sobre el estado del arte de los formularios HTML y te mostrará los mejores procedimientos para implementar formularios que tengan buena usabilidad y seguridad. Además, como verás más adelante, la especificación HTML5 ha mejorado aún más el uso de formularios.

Creación de formularios

La gestión de formularios es un proceso que consta de varias partes. La primera es la creación de un formulario en el que el usuario pueda introducir los detalles necesarios. Estos datos se envían al servidor web, donde se interpretan, a menudo con alguna comprobación de errores. Si el código PHP identifica uno o más campos que requieren que se vuelvan a rellenar, el formulario puede volver a mostrarse con un mensaje de error. Cuando el código admite los datos precisos de la entrada, realiza alguna acción en la que generalmente está involucrada la base de datos, como puede ser introducir detalles sobre una compra.

Para crear un formulario, debes contar al menos con los siguientes elementos:

- Una etiqueta de apertura `<form>` y otra de cierre `</form>`.
- Un tipo de presentación que especifique uno de los métodos GET o POST.
- Uno o más campos `input` (de entrada).
- El URL de destino a la que deben enviarse los datos del formulario.

El Ejemplo 11-1 muestra un formulario muy sencillo creado con PHP, que debes escribir y guardar como *formtest.php*.

Ejemplo 11-1. formtest.php, un sencillo gestor de formularios PHP

```
<?php // formtest.php
  echo <<<_END
    <html>
      <head>
        <title>Form Test</title>
      </head>
      <body>
      <form method="post" action="formtest.php">
        What is your name?
        <input type="text" name="name">
        <input type="submit">
      </form>
      </body>
    </html>
_END;
?>
```

Lo primero que hay que observar en este ejemplo es que, como ya se ha visto en este libro, en lugar de entrar y salir del código PHP, el constructor `echo <<<_END...._END` se utiliza siempre que se debe generar código HTML de varias líneas.

Dentro de esta composición de varias líneas está el código estándar de inico de HTML, que muestra su título e inicia el cuerpo del documento. Continúa por el formulario, que está configurado para enviar los datos que contiene, mediante el método POST, al programa PHP *formtest.php*, que es el nombre del propio programa.

El resto del programa lo único que hace es cerrar todos los elementos que abrió: el formulario, el cuerpo del documento HTML y la declaración `echo <<<_END` de PHP. El resultado de abrir el programa en un navegador web se muestra en la Figura 11-1.

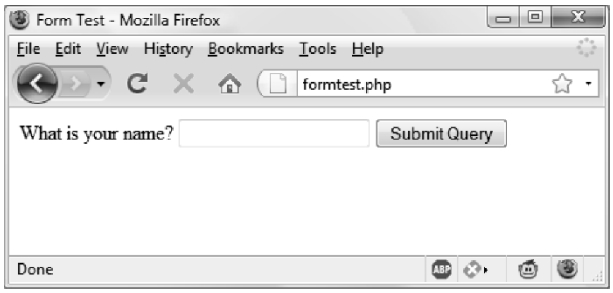

Figura 11-1. Resultado de abrir formtest.php en un navegador

Extracción de los datos enviados

El Ejemplo 11-1 es solo una parte de las que componen el proceso de gestión de formularios. Si introduces un nombre y haces clic en el botón Submit Query (enviar consulta) , no ocurrirá absolutamente nada excepto que el formulario se muestra de nuevo. Entonces, ahora es el momento de añadir un código PHP para procesar los datos enviados por el formulario.

El Ejemplo 11-2 es una ampliación del programa anterior para incluir el procesamiento de datos. Escribe o modifica *formtest.php* agregando las nuevas líneas, guárdalo como *formtest2.php* y prueba el programa. El resultado de ejecutar este programa e introducir un nombre se muestra en la Figura 11-2.

Ejemplo 11-2. Versión actualizada de formtest.php

```
<?php // formtest2.php
  if (isset($_POST['name'])) $name = $_POST['name'];
  else $name = "(Not entered)";

  echo <<<_END
    <html>
      <head>
        <title>Form Test</title>
      </head>
      <body>
        Your name is: $name<br>
        <form method="post" action="formtest2.php">
          What is your name?
          <input type="text" name="name">
          <input type="submit">
        </form>
      </body>
    </html>
_END;
?>
```

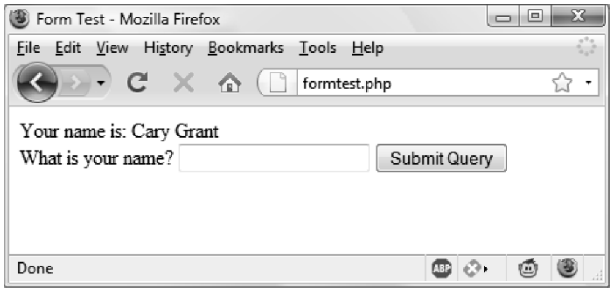

Figura 11-2. formtest.php con la gestión de datos

Los únicos cambios son un par de líneas al principio que verifican el campo `name` de la matriz asociativa `$_POST` y hacen `echo` de ella al usuario. En el Capítulo 10 se introdujo la matriz asociativa `$_POST`, que contiene un elemento para cada campo de un formulario HTML. En el Ejemplo 11-2, el nombre de entrada que se ha utilizado era `name` y el método de formulario era POST, así que el elemento `name` de la matriz $_POST contiene el valor en `$_POST['name']`.

La función `isset` de PHP se usa para probar si se le ha asignado un valor a `$_POST['name']`. Si no se ha puesto nada, el programa asigna el valor `(Not entered)`; de lo contrario, almacena el valor que se ha introducido. Más abajo, se ha añadido una línea después de la sentencia `<body>` para mostrar ese valor, que se almacena en `$name`.

Valores por defecto

A veces es conveniente ofrecer a los visitantes de tu sitio valores predeterminados en un formulario web. Por ejemplo, supongamos que pones un widget de calculadora de pagos de préstamos en un sitio web de bienes raíces. Podría tener sentido introducir valores por defecto de, digamos, 25 años y el 6 % de interés, para que el usuario pueda escribir simplemente la suma principal que va a pedir prestada o la cantidad que puede permitirse pagar cada mes.

En este caso, el HTML para esos dos valores sería algo así como el Ejemplo 11-3.

Ejemplo 11-3. Establecimiento de valores por defecto

```
<form method="post" action="calc.php"><pre>
      Loan Amount <input type="text" name="principle">
Monthly Repayment <input type="text" name="monthly">
  Number of Years <input type="text" name="years" value="25">
    Interest Rate <input type="text" name="rate"  value="6">
                  <input type="submit">
</pre></form>
```

Si deseas probar este ejemplo (y los otros de código HTML), escríbelo y guárdalo con la extensión de archivo *.html* (o *.htm*), como *test.html* (o *test.htm*), y luego carga ese archivo en tu navegador.

Echa un vistazo a las entradas tercera y cuarta. Al rellenar el atributo `value`, se visualiza un valor por defecto en el campo, que los usuarios pueden modificar si lo desean. Con valores por defecto sensibles, a menudo puedes hacer que tus formularios web sean más fáciles de usar al hacer que la mecanografía innecesaria sea la mínima posible. El resultado del código anterior es el de la Figura 11-3. Por supuesto, esto se creó para ilustrar los valores por defecto y, debido a que no se ha escrito el programa *calc.php*, el formulario no hará nada si se envía.

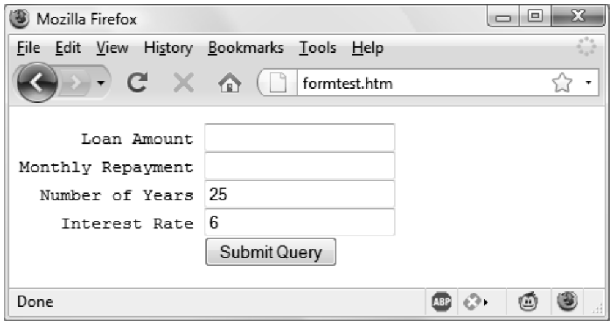

Figura 11-3. Utilización de valores por defecto para campos de formulario seleccionados

Los valores por defecto también se utilizan para campos ocultos, por ejemplo si deseas pasar información adicional desde tu página web a tu programa, además de los que introducen los usuarios. Echaremos un vistazo a campos ocultos más adelante en este capítulo.

Tipos de entradas

Los formularios HTML son muy versátiles y te permiten enviar una amplia gama de tipos de entradas, desde cuadros de texto y áreas de texto a casillas de verificación y botones de selección, etc.

Cuadros de texto

El tipo de entrada que probablemente utilizarás con más frecuencia es el cuadro de texto. Acepta una amplia gama de texto alfanumérico y otros caracteres en un cuadro de una sola línea. El formato general de una entrada de cuadro de texto es el siguiente:

```
<input type="text" name="name" size="size" maxlength="length"
  value="value">
```

Ya hemos visto los atributos de `name` y `value`, pero vamos a introducir aquí dos más: `size` y `maxlength`. El atributo `size` especifica el ancho del cuadro (en caracteres de la fuente que se va a utilizar) como aparecería en la pantalla, y `maxlength` especifica el número máximo de caracteres que el usuario puede introducir en el campo.

Los únicos atributos requeridos son `type`, que le indica al navegador web el tipo de entrada que debe esperar, y `name`, para dar a la entrada un nombre que se utilizará para procesar el campo una vez recibido el formulario.

Áreas de texto

Cuando necesites aceptar la entrada de más de una línea corta de texto, utiliza el área de texto. Es similar a un cuadro de texto, pero, debido a que permite varias líneas, tiene algunos atributos diferentes. Su formato general es el siguiente:

```
<textarea name="name" cols="width" rows="height" wrap="type">
</textarea>
```

Lo primero que hay que tener en cuenta es que `<textarea>` tiene su propia etiqueta y no es un subtipo de la etiqueta `<input>`. Por lo tanto, es necesario cerrar `</textarea>` para finalizar la entrada.

En lugar de un atributo por defecto, si tienes texto por defecto que mostrar, debes colocarlo antes del cierre `</textarea>`, y este se presentará y el usuario podrá editarlo:

```
<textarea name="name" cols="width" rows="height" wrap="type">
This is some default text.
</textarea>
```

Para controlar la anchura y la altura, utiliza los atributos `cols` y `rows`. Ambos usan el carácter de la fuente que se utilice para determinar el tamaño del área. Si los omites, se creará un cuadro de entrada por defecto que variará en dimensiones en función del navegador que se utilice, por lo que siempre debes definirlos para estar seguro de cómo aparecerá el formulario.

Por último, puedes controlar cómo se ajustará el texto introducido en el cuadro (y cómo se enviará dicho ajuste al servidor) mediante el atributo `wrap`. La Tabla 11-1 muestra los tipos de empaquetados disponibles. Si se omite el atributo `wrap`, se utiliza un empaquetado soft.

Tabla 11-1. Tipos de empaquetados disponibles en una entrada <textarea>

Tipo	Acción
`Off`	El texto no tiene empaquetado, y las líneas aparecen exactamente como las ha tecleado el usuario.
`Soft`	El texto va empaquetado, pero se envía al servidor como una cadena larga sin retorno de carro ni avance de línea.
`Hard`	El texto va empaquetado y se envía al servidor en formato empaquetado con retorno de carro y avance de línea.

Casillas de verificación

Cuando deseas ofrecer una serie de opciones diferentes a los usuarios, de las cuales puede seleccionar uno o más elementos, las casillas de verificación son el camino a seguir. Aquí está el formato a utilizar:

```
<input type="checkbox" name="name" value="value" checked="checked">
```

Por defecto, las casillas de verificación son cuadradas. Si incluyes el atributo `checked`, la casilla ya está marcada cuando se muestra el navegador. La cadena que asignes al atributo deben ser un par de comillas dobles o simples o el valor `"checked"`, o no debe haber ningún valor asignado (solo `checked`). Si no incluyes el atributo, el cuadro se muestra sin marcar. A continuación se muestra un ejemplo de cómo crear una casilla sin marcar:

```
I Agree <input type="checkbox" name="agree">
```

Si el usuario no marca la casilla, no se enviará ningún valor. Pero si lo hace, se enviará un valor de `"on"` para el campo denominado `agree`. Si prefieres que se envíe tu propio valor en lugar de la palabra *on* (como puede ser el número 1), puedes utilizar la siguiente sintaxis:

```
I Agree <input type="checkbox" name="agree" value="1">
```

Por otro lado, si deseas ofrecer un boletín a tus lectores al enviar un formulario, es posible que quieras que la casilla de verificación ya esté marcada como valor predeterminado:

```
Subscribe? <input type="checkbox" name="news" checked="checked">
```

Si deseas permitir que se seleccionen grupos de elementos a la vez, asígnales el mismo nombre. Sin embargo, solo se enviará el último elemento marcado, a menos que pases una matriz como nombre. Así, el Ejemplo 11-4 permite al usuario seleccionar sus helados favoritos (consulta la Figura 11-4 para ver cómo se muestra en un navegador).

Ejemplo 11-4. Posibilidad de varias opciones de casillas de verificación

```
   Vanilla <input type="checkbox" name="ice" value="Vanilla">
 Chocolate <input type="checkbox" name="ice" value="Chocolate">
Strawberry <input type="checkbox" name="ice" value="Strawberry">
```

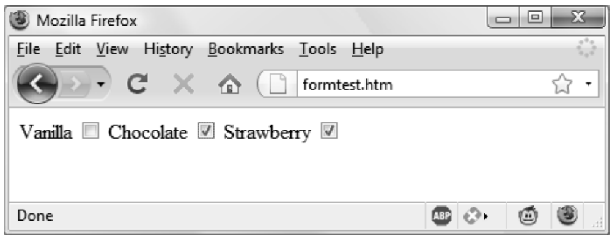

Figura 11-4. Uso de casillas de verificación para realizar selecciones de forma rápida

Si solo se ha seleccionado una de las casillas de verificación, como por ejemplo la segunda, solo se enviará ese elemento (al campo denominado `ice` se le asignará el valor `"Chocolate"`). Pero si se seleccionan dos o más, solo se enviará el último valor y se ignorarán los valores anteriores.

Si *deseas* un comportamiento exclusivo, para que solo se pueda presentar un artículo, entonces en este caso debes usar botones de selección (ver la siguiente sección). De lo contrario, para permitir la presentación de múltiples artículos tienes que alterar ligeramente el código HTML, como en el Ejemplo 11-5 (observa la adición de los corchetes, [], a continuación de los valores de `ice`).

Ejemplo 11-5. Envío de varios valores con una matriz

```
   Vanilla <input type="checkbox" name="ice[]" value="Vanilla">
 Chocolate <input type="checkbox" name="ice[]" value="Chocolate">
Strawberry <input type="checkbox" name="ice[]" value="Strawberry">
```

Ahora, cuando se envía el formulario, si se ha verificado alguno de estos elementos, se enviará una matriz llamada `ice` que contiene todos los valores seleccionados. Puedes extraer el valor enviado individualmente, o la matriz de valores a una variable como esta:

```
$ice = $_POST['ice'];
```

Si el campo `ice` se ha contabilizado como un valor único, `$ice` será una sola cadena, tal como "Strawberry". Pero si `ice` se definió en forma de matriz (como en el Ejemplo 11-5), `$ice` será una matriz, y su número de elementos será el número de valores enviados. La Tabla 11-2 muestra los siete posibles conjuntos de valores que este HTML podría presentar para una, dos o las tres selecciones. En cada caso, se crea una matriz de uno, dos o tres elementos.

Tabla 11-2. Los siete posibles conjuntos de valores para la matriz $ice

Envío de un valor	Envío de dos valores	Envío de tres valores
`$ice[0] => Vanilla`	`$ice[0] => Vanilla`	`$ice[0] => Vanilla`
	`$ice[1] => Chocolate`	`$ice[1] => Chocolate`
`$ice[0] => Chocolate`		`$ice[2] => Strawberry`
	`$ice[0] => Vanilla`	
`$ice[1] => Strawberry`	`$ice[1] => Strawberry`	
	`$ice[0] => Chocolate`	
	`$ice[1] => Strawberry`	

Si `$ice` es una matriz, el código PHP para mostrar su contenido es bastante sencillo y podría verse así:

```
foreach($ice as $item) echo "$item<br>";
```

Este código usa el constructor `foreach` estándar de PHP para recorrer la matriz `$ice` y pasar el valor de cada elemento a la variable `$item`, que luego se muestra mediante el comando `echo <br>` es solo un recurso de formato HTML para forzar una nueva línea, después de cada uno de los sabores, en la pantalla.

Botones de selección

Los botones de selección llevan el nombre de los pulsadores de preselección de los aparatos de radio antiguos, en los que cualquier botón que se ha pulsado previamente vuelve a su situación inicial cuando se pulsa otro botón. Se utilizan cuando se desea que solo se devuelva un valor individual de una selección de dos o más opciones. Todos los botones de un grupo deben usar el mismo nombre y, ya que solo se devuelve un único valor, no es necesario que pases una matriz.

Por ejemplo, si tu sitio web ofrece una selección de plazos de entrega para los artículos comprados en tu tienda, puedes usar HTML como en el Ejemplo 11-6 (ver la Figura 11-5 para ver su aspecto). Por defecto, los botones de selección son redondos.

Ejemplo 11-6. Uso de botones de selección

```
8am-Noon<input type="radio" name="time" value="1">
Noon-4pm<input type="radio" name="time" value="2" checked="checked">
 4pm-8pm<input type="radio" name="time" value="3">
```

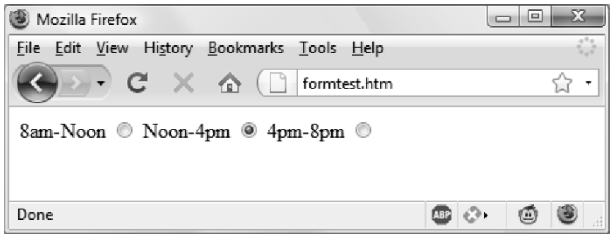

Figura 11-5. Selección de un valor único con los botones de selección

Aquí, la segunda opción, Noon-4pm, se ha seleccionado por defecto. Este valor predeterminado garantiza que el usuario elegirá al menos un plazo de entrega, que puede cambiar a una de las otras dos opciones si lo prefiere. En caso contrario, si no se marca ninguno, el usuario podría olvidar seleccionar una opción, y no se enviaría ningún valor en relación con el margen de tiempo de entrega.

Campos ocultos

A veces es conveniente tener campos de formulario ocultos para poder realizar un seguimiento del estado de la entrada del formulario. Por ejemplo, es posible que tengas la necesidad de saber si ya se ha presentado un formulario. Para ello, puedes añadir algo de HTML a tu PHP, como lo siguiente:

```
echo '<input type="hidden" name="submitted" value="yes">'
```

Esta es una simple declaración `echo` de PHP que añade un campo `input` al formulario HTML. Supongamos que el formulario se creó fuera del programa y se ha mostrado al usuario. La primera vez que el programa PHP recibe la entrada, esta línea de código no se ha ejecutado, por lo que no debe haber ningún campo con nombre `submitted`. El programa PHP vuelve a crear el formulario y añade el campo `input`. Así que cuando el visitante vuelve a enviar el formulario, el programa PHP lo recibe con el campo `submitted` ajustado a `"yes"`. El código puede simplemente verificar si el campo está presente:

```
if (isset($_POST['submitted']))
{...
```

Los campos ocultos también pueden ser útiles para almacenar otros detalles, como una cadena de identificador de sesión que puedes crear para identificar a un usuario, etc.

Nunca trates los campos ocultos como si fueran seguros, porque no lo son. Alguien podría ver fácilmente el HTML que los contiene mediante la función View Source del navegador. Un atacante malicioso también puede crear un mensaje que elimine, añada o cambie un campo oculto.

<select>

La etiqueta `<select>` te permite crear una lista desplegable de opciones, que ofrece selecciones únicas o múltiples. Se ajusta a la siguiente sintaxis:

```
<select name="name" size="size" multiple="multiple">
```

El atributo `size` es el número de líneas a mostrar. Al hacer clic en la pantalla se despliega una lista que muestra todas las opciones. Si utilizas el atributo `multiple`, un usuario puede seleccionar varias opciones de la lista pulsando la tecla Ctrl al hacer clic. Por lo tanto, para preguntar a un usuario por su verdura favorita de una selección de cinco, puedes usar HTML como el del Ejemplo 11-7, que ofrece una sola selección.

Ejemplo 11-7. Uso de <select>

```
Vegetables
<select name="veg" size="1">
  <option value="Peas">Peas</option>
  <option value="Beans">Beans</option>
  <option value="Carrots">Carrots</option>
  <option value="Cabbage">Cabbage</option>
  <option value="Broccoli">Broccoli</option>
</select>
```

Este HTML ofrece cinco opciones, con la primera, *Peas*, preseleccionada (debido a que es el primer elemento). La Figura 11-6 muestra la salida donde se ha hecho clic en la lista para desplegarla, y la opción *Carrots* está resaltada. Si deseas tener una opción por defecto diferente que se ofrezca primero (como *Beans*), usa el atributo `selected`, así:

```
<option selected="selected" value="Beans">Beans</option>
```

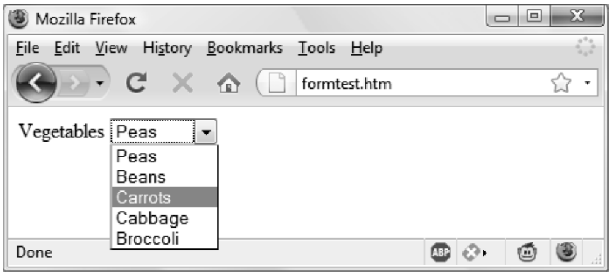

Figura 11-6. Creación de una lista desplegable con <select>

También puedes permitir que los usuarios seleccionen más de un elemento, como en el Ejemplo 11-8.

Ejemplo 11-8. Uso de <select> con el atributo múltiple

```
Vegetables
<select name="veg" size="5" multiple="multiple">
  <option value="Peas">Peas</option>
  <option value="Beans">Beans</option>
  <option value="Carrots">Carrots</option>
  <option value="Cabbage">Cabbage</option>
  <option value="Broccoli">Broccoli</option>
</select>
```

Este HTML no es muy diferente; el tamaño se ha cambiado a `"5"` y se ha añadido el atributo múltiple. Pero, como puedes ver en la Figura 11-7, ahora es posible para el usuario seleccionar más de una opción utilizando la tecla Ctrl al hacer clic. Puedes omitir el atributo `size` si lo deseas, y la salida será la misma; sin embargo, con una lista más grande el cuadro desplegable puede mostrar más elementos, así que recomiendo que escojas un número adecuado de filas y te quedes con él. También recomiendo no usar varias casillas de selección más pequeñas que dos filas en altura: algunos navegadores pueden no mostrar correctamente las barras de desplazamiento necesarias para acceder a ellas.

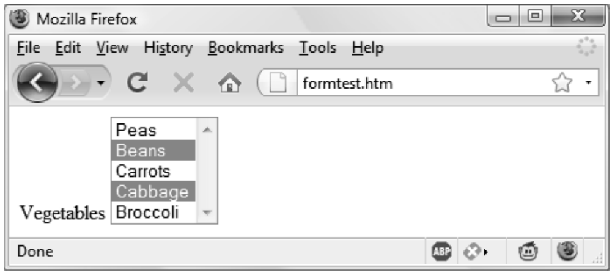

Figura 11-7. Uso de <select> con el atributo múltiple

También puedes utilizar el atributo `selected` dentro de una selección múltiple y, de hecho, puedes tener más de una opción preseleccionada si lo deseas.

Etiquetas

Puedes mejorar aún la experiencia de usuario con la etiqueta `<label>`. Con ella, puedes rodear un elemento del formularioy hacerlo seleccionable al hacer clic en cualquier parte visible que se encuentra entre las etiquetas `<label>` de apertura y cierre.

Así, si volvemos al ejemplo de la selección de un plazo de entrega, se podría permitir al usuario hacer clic en el botón de selección en sí *y* en el texto asociado, de esta manera:

```
<label>8am-Noon<input type="radio" name="time" value="1"></label>
```

El texto no se subrayará como un hipervínculo al hacer esto, pero cuando el puntero del ratón pase por encima cambiará a una flecha en lugar de presentar el cursor de texto, lo que indica que todo el elemento es clicable.

El botón de envío

Para que coincida con el tipo de formulario que se envía, puedes cambiar el texto del botón de envío a lo que desees mediante el atributo `value`, así:

```
<input type="submit" value="Search">
```

También puedes sustituir el botón de texto estándar por una imagen gráfica de tu elección con un código HTML como este:

```
<input type="image" name="submit" src="image.gif">
```

Desinfección de entradas

Ahora volvemos a la programación PHP. Nunca se puede insistir lo suficiente en que la gestión de los datos del usuario es un campo minado para la seguridad, y que es esencial aprender a tratar todos estos problemas con la mayor precaución desde el principio. En realidad no es tan difícil de desinfectar de los posibles intentos de *hacking*, pero debe hacerse.

Lo primero que hay que recordar es que, independientemente de las limitaciones que hayas impuesto a un formulario HTML para limitar los tipos y tamaños de las entradas, es un asunto trivial para un hacker usar la función *View Source* de su navegador para extraer el formulario y modificarlo para presentar entradas maliciosas a tu sitio web.

Por lo tanto, nunca debes confiar en ninguna variable que obtengas de las matrices `$_GET` o `$_POST` hasta que las hayas desinfectado. Si no lo haces, los usuarios pueden intentar inyectar JavaScript en los datos para interferir con el funcionamiento de tu sitio, o incluso intentar añadir MySQL para comprometer tu base de datos.

Por lo tanto, en lugar de usar solo código como el siguiente cuando se lee en la entrada del usuario:

```
$variable = $_POST['user_input'];
```

también debes utilizar una o más de las siguientes líneas de código. Por ejemplo, para evitar que se inyecten caracteres de escape en una cadena que se presentará a MySQL, utiliza lo siguiente. Recuerda que esta función tiene en cuenta el conjunto de caracteres de la conexión MySQL, por lo que debe usarse con un objeto de conexión `mysqli` (en este caso, `$connection`), como se discutió en el Capítulo 10:

```
$variable = $connection->real_escape_string($variable);
```

Recuerda que la forma más segura de proteger MySQL de la piratería informática es usar marcadores de posición y declaraciones preparadas, como se ha descrito en el Capítulo 10. Si lo haces para todos los accesos a MySQL, no es necesario escapar de los datos que se transfieren hacia dentro o hacia fuera de la base de datos. Sin embargo, todavía necesitarás desinfectar la entrada cuando la incluyas en HTML.

Para deshacerte de barras oblicuas no deseadas, primero debes comprobar si la característica de comillas mágicas de PHP está activada (que saldrán de las comillas si les añades barras oblicuas), si es así, llama a `stripslashes`, de este modo:

```
if (get_magic_quotes_gpc())
$variable = stripslashes($variable);
```

Y para eliminar cualquier HTML de una cadena, usa lo siguiente:

```
$variable = htmlentities($variable);
```

Por ejemplo, lo que viene a continuación cambiaría una cadena de código HTML interpretable como `<b>hi</b>` en `<b>hi</b>`, que luego se muestra como texto, y no se interpretará como etiquetas HTML.

Por último, si deseas eliminar HTML por completo de una entrada, utiliza lo siguiente (pero asegúrate de utilizarlo antes de llamar a `htmlentities`, que reemplaza cualquier paréntesis angular utilizado como parte de las etiquetas HTML):

```
$variable = strip_tags($variable);
```

De hecho, hasta que sepas exactamente qué desinfección necesitas para un programa, el Ejemplo 11-9 muestra un par de funciones que reúnen todas estas comprobaciones para proporcionar un muy buen nivel de seguridad.

Ejemplo 11-9. Las funciones sanitizeString y sanitizeMySQL

```
<?php
  function sanitizeString($var)
  {
    if (get_magic_quotes_gpc())
      $var = stripslashes($var);
    $var = strip_tags($var);
    $var = htmlentities($var); return $var;
  }

  function sanitizeMySQL($connection, $var)
  {
    $var = $connection->real_escape_string($var);
    $var = sanitizeString($var); return $var;
  }
?>
```

Añade este código al final de tus programas PHP, luego puedes llamarlo para cada entrada de usuario a desinfectar, así:

```
$var = sanitizeString($_POST['user_input']);
```

O, cuando tienes una conexión MySQL abierta y un objeto de conexión `mysqli` (en este caso, llamado `$connection`):

```
$var = sanitizeMySQL($connection, $_POST['user_input']);
```

Si usas la versión procedimental de la extensión `mysqli`, necesitarás modificar la función `sanitizeMySQL` para llamar a la función `mysqli_real_escape_string`, así (en cuyo caso `$connection` será entonces un gestor, no un objeto):

```
$var = mysqli_real_escape_string($connection, $var);
```

Programa de ejemplo

Veamos cómo un programa PHP de la vida real se integra con un formulario HTML mediante la creación del programa *convert.php* que aparece en el Ejemplo 11-10. Escríbelo tal como aparece y pruébalo.

Ejemplo 11-10. Programa para convertir valores entre grados Fahrenheit y Celsius

```
<?php // convert.php
  $f = $c = '';

  if (isset($_POST['f'])) $f = sanitizeString($_POST['f']);
  if (isset($_POST['c'])) $c = sanitizeString($_POST['c']);

  if (is_numeric($f))
  {
    $c = intval((5 / 9) * ($f - 32));
    $out = "$f &deg;f equals $c &deg;c";
  }
  elseif(is_numeric($c))
  {
    $f = intval((9 / 5) * $c + 32);
    $out = "$c &deg;c equals $f &deg;f";
  }
  else $out = "";

  echo <<<_END
<html>
  <head>
    <title>Temperature Converter</title>
  </head>
```

```
  <body>
    <pre>
      Enter either Fahrenheit or Celsius and click on Convert

      <b>$out</b>
      <form method="post" action="">
        Fahrenheit <input type="text" name="f" size="7">
           Celsius <input type="text" name="c" size="7">
                   <input type="submit" value="Convert">
      </form>
    </pre>
  </body>
</html>
_END;

  function sanitizeString($var)
  {
    if (get_magic_quotes_gpc())
      $var = stripslashes($var);
    $var = strip_tags($var);
    $var = htmlentities($var); return $var;
  }
?>
```

Cuando llamas a *convert.php* en un navegador, el resultado se parece a la Figura 11-8.

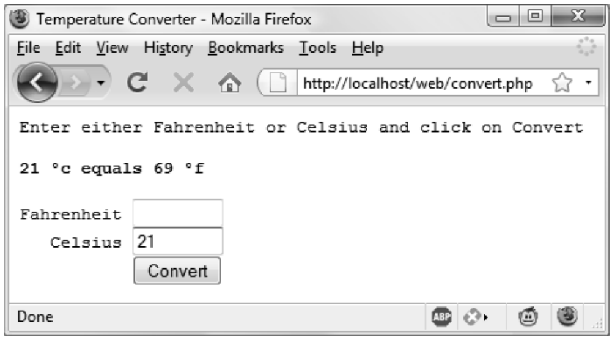

Figura 11-8. Programa de conversión de temperatura funcionando

Si desglosamos el programa, la primera línea inicializa las variables `$c` y `$f` en caso de que no se fijen en el programa. Las dos líneas siguientes obtienen los valores del campo llamado `f` o del campo llamado `c`, para un valor de entrada Fahrenheit o Celsius. Si el usuario introduce ambos, el valor Celsius simplemente se ignora y el valor Fahrenheit se convierte. Como medida de seguridad, también se utiliza la nueva función `sanitizeString` del Ejemplo 11-9.

Por lo tanto, habiendo presentado valores o cadenas vacías tanto en `$f` como en `$c`, la siguiente parte del código constituye una estructura `if...elseif...else` que primero prueba si `$f` tiene un valor numérico. Si no, comprueba `$c`; si `$c` tampoco tiene valor numérico, la variable `$out` adquiere el valor de una cadena vacía (veremos más sobre esto en un momento).

Si se encuentra que `$f` tiene un valor numérico, a la variable `$c` se le asigna una expresión matemática sencilla que convierte el valor de `$f` de Fahrenheit a Celsius. La fórmula utilizada es Celsius = (5 / 9) × (Fahrenheit – 32). La variable `$out` contiene un mensaje que explica la conversión.

Por otro lado, si se encuentra que `$c` tiene un valor numérico, se lleva a cabo una operación complementaria para convertir el valor de `$c` de Celsius a Fahrenheit y se asigna el valor del resultado a `$f`. La fórmula utilizada es Fahrenheit = (9 / 5) × Celsius + 32. Al igual que con el anterior, la cadena `$out` está configurada para contener un mensaje sobre la conversión.

En ambas conversiones, se llama a la función `intval` de PHP para convertir el resultado de la conversión a un valor entero. No es necesario, pero se ve mejor.

Con toda la aritmética hecha, el programa ahora produce el HTML, que comienza con el encabezamiento y el título básicos y luego contiene algún texto introductorio antes de mostrar el valor de `$out`. Si no se realiza ninguna conversión de temperatura, `$out` tendrá un valor de `NULL` y no se mostrará nada, que es exactamente lo que queremos que ocurra cuando el formulario aún no se ha enviado. Pero si se ha hecho una conversión, `$out` contiene el resultado, que se visualiza.

Después de esto, llegamos al formulario, que está configurado para enviarlo mediante el método POST al programa mismo (representado por un par de comillas dobles, de modo que el archivo se puede guardar con cualquier nombre). Dentro del formulario, hay dos entradas para un valor Fahrenheit o Celsius. A continuación se presenta un botón de envío con el texto Convert (Convertir) y el formulario se cierra.

Después de imprimir el HTML para cerrar el documento, llegamos finalmente a la función `sanitizeString` del Ejemplo 11-9. Trata de jugar con el ejemplo introduciendo diferentes valores en los campos; para divertirte un poco, ¿puedes encontrar un valor para el cual Fahrenheit y Celsius sean iguales?

Todos los ejemplos de este capítulo han utilizado el método POST para enviar datos del formulario. Recomiendo este método, ya que es el más limpio y seguro. Sin embargo, los formularios se pueden cambiar fácilmente para usar el método GET siempre que los valores se obtengan de la matriz `$_GET` en lugar de la matriz `$_POST`. Las razones para hacer esto pueden ser las de hacer que el resultado de una búsqueda se pueda marcar o se pueda acceder a él desde un enlace en otra página.

Mejoras en HTML5

Con HTML5, los desarrolladores pueden utilizar una serie de mejoras útiles para la gestión de formularios y hacer que el uso de estos sea más fácil que nunca, incluidos nuevos atributos: color, fecha, cronómetros y nuevos tipos de entradas, aunque algunas de estas características aún no están disponibles en los principales navegadores.

Atributo autocomplete

Puedes aplicar el atributo `autocomplete` al elemento `<form>`, o a cualquiera de los atributos de `color`, `date`, `email`, `password`, `range`, `search`, `tel`, `text` o los tipos `url` del elemento `<input>`.

Con la función de autocompletar activada, se hace una rellamada a las entradas de usuario anteriores y se introducen automáticamente en los campos como sugerencias. También puedes desactivarla. A continuación se muestra cómo activar la función autocompletar para todo un formulario y cómo desactivarla para campos específicos (resaltados en negrita):

```
<form action='myform.php' method='post' autocomplete='on'>
<input type='text'     name='username'>
<input type='password' name='password' autocomplete='off'>
</form>
```

Atributo autofocus

El atributo `autofocus` proporciona un enfoque inmediato a un elemento cuando se carga una página. Se puede aplicar a cualquier elemento `<input>`, `<textarea>` o `<button>`, así:

```
<input type='text' name='query' autofocus='autofocus'>
```

Los navegadores que utilizan interfaces táctiles (como Android, iOS o Windows Phone) normalmente ignoran el atributo `autofocus` y dejan que el usuario toque un campo para darle enfoque; de lo contrario, el *zoom*, el enfoque y los teclados emergentes que este atributo generaría podrían convertirse rápidamente en algo muy molesto.

Debido a que esta característica hará que el foco se mueva hacia un elemento de entrada, la tecla Backspace (Retroceso) ya no llevará al usuario de vuelta a una página web (aunque la flecha Alt-Left (Alt-Izquierda) y Alt- Right (Alt- derecha) todavía se moverán hacia atrás y hacia adelante dentro del historial de navegación).

Atributo placeholder

El atributo `placeholder` te permite colocar en cualquier campo de entrada en blanco una sugerencia útil para explicar a los usuarios lo que deben introducir. Lo usas así:

```
<input type='text' name='name' size='50' placeholder='First &
  Last name'>
```

El campo de entrada mostrará el texto que aparece en el marcador como un aviso hasta que el usuario comience a escribir, en cuyo momento el texto del marcador desaparece.

Atributo required

El atributo `required` garantiza que se ha completado un campo antes de enviar el formulario. Úsalo de esta forma:

```
<input type='text' name='creditcard' required='required'>
```

Cuando el navegador detecta un intento de envío de un formulario donde hay una entrada `required` incompleta se visualiza un mensaje que solicita al usuario que complete el campo.

Atributos de sustitución

Con los atributos de sustitución, puedes sustituir las parametrizaciones del formulario elemento por elemento. Así, por ejemplo, mediante el atributo `formaction`, puedes especificar que un botón de envío debe enviar un formulario a un URL diferente del especificado en el formulario (donde los URL por defecto y el ignorado están en negrita), como lo siguiente:

```
<form action='url1.php' method='post'>
<input type='text' name='field'>
<input type='submit' formaction='url2.php'>
</form>
```

HTML5 también ofrece soporte para los atributos de sustitución `formenctype`, `formmethod`, `formnovalidate` y `formtarget`, que puedes usar exactamente de la misma manera que `formaction` para anular una de estas configuraciones.

Atributos width y height

Si usas estos nuevos atributos, puedes alterar las dimensiones para entradas de tipo imagen, así:

```
<input type='image' src='picture.png' width='120' height='80'>
```

Atributos min y max

Con los atributos `min` y `max`, puedes especificar valores mínimos y máximos para los valores de las entradas. Utiliza los atributos de este modo:

```
<input type='time' name='alarm' value='07:00' min='05:00'
  max='09:00'>
```

El navegador dejará, dentro del rango permitido, seleccionar valores hacia arriba y hacia abajo, o simplemente rechazará valores fuera de ese rango.

Atributo step

A menudo usado con `min` y `max`, el atributo `step` permite recorrer los valores numéricos o de fecha, como este caso:

```
<input type='time' name='meeting' value='12:00'
  min='09:00' max='16:00' step='3600'>
```

Cuando pases por los valores de fecha o de hora, cada unidad representa 1 segundo.

Atributo form

Con HTML5, ya no es necesario colocar los elementos `<input>` dentro de los elementos `<form>`, ya que se puede especificar el formulario al que se aplica una entrada proporcionando el atributo `form`. El siguiente código muestra la creación de un formulario, pero con la entrada fuera de las etiquetas `<form>` y `</form>`:

```
<form action='myscript.php' method='post' id='form1'>
</form>

<input type='text' name='username' form='form1'>
```

Para ello, debes asignar un ID al formulario mediante el atributo `id` y referirte a este ID en el atributo `form` del elemento `input`. Este atributo resulta muy útil para añadir campos de entrada ocultos, ya que no se puede controlar la disposición del campo dentro del formulario, o para utilizar JavaScript para modificar formularios y entradas sobre la marcha.

Atributo list

En HTML5 se pueden adjuntar listas a las entradas para permitir fácilmente a los usuarios la selección de una lista predefinida, que se puede utilizar de esta manera:

```
Select destination:
<input type='url' name='site' list='links'>

<datalist id='links'>
  <option label='Google' value='http://google.com'>
  <option label='Yahoo!' value='http://yahoo.com'>
  <option label='Bing'   value='http://bing.com'>
  <option label='Ask'    value='http://ask.com'>
</datalist>
```

Tipo de entrada color

El tipo de entrada `color` llama a un selector de color para que puedas simplemente hacer clic en el color de tu elección. Lo usas así:

```
Choose a color <input type='color' name='color'>
```

Tipos de entradas number y range

Los tipos de entradas `number` y `range` restringen la entrada a un número y, opcionalmente, también especifican un rango permitido como este:

```
<input type='number' name='age'>
<input type='range' name='num' min='0' max='100' value='50'
  step='1'>
```

Selectores de fecha y hora

Al elegir un tipo de entrada `date`, `month`, `week`, `time`, `datetime` o `datetimelocal`, aparecerá un selector en los navegadores compatibles desde los que el usuario puede realizar una selección, como esta, que introduzca la hora:

```
<input type='time' name='time' value='12:34'>
```

El siguiente capítulo te mostrará cómo utilizar las cookies y la autenticación para almacenar las preferencias de los usuarios y mantenerlos conectados, y cómo mantener una sesión de usuario completa.

Preguntas

1. Puedes enviar datos de formulario mediante el método POST o GET. ¿Qué se usan las matrices asociativas para pasar estos datos a PHP?
2. ¿Cuál es la diferencia entre un cuadro de texto y un área de texto?
3. Si un formulario tiene que ofrecer tres opciones a un usuario, cada una de las cuales es mutuamente excluyente para que solo se pueda seleccionar uno de los tres, ¿qué tipo de entrada utilizarías si se te ha dado la opción de elegir entre casillas de verificación y botones de selección?
4. ¿Cómo puedes enviar un grupo de selecciones desde un formulario web utilizando solo un solo nombre de campo?
5. ¿Cómo puede enviar un campo de formulario sin mostrarlo en el navegador?
6. ¿Qué etiqueta HTML se utiliza para encapsular un elemento de formulario y texto de soporte o haciendo que toda la unidad sea seleccionable con un clic del ratón?
7. ¿Qué función de PHP convierte HTML en un formato que se puede mostrar, pero que un navegador no lo interpreta como HTML?
8. ¿Qué atributo de formulario se puede utilizar para ayudar a los usuarios a completar los campos de entrada?
9. ¿Cómo puede asegurarse de que se rellena una entrada antes de que se envíe un formulario?

Consulta "Respuestas del Capítulo 11" en la página 713 en el Apéndice A para comprobar las respuestas a estas preguntas.