

Tema 3

Uso de formularios



El protocolo **HTTP** (Hypertext Transfer Protocol) se define como un estándar de Internet que determina la manera en que cliente y servidor deben comunicarse, así como la estructura de los mensajes que pueden intercambiar. De esta manera, el navegador realizará una petición HTTP siempre que el usuario escriba una URL en la barra de direcciones del navegador, acepte un formulario o realice cualquier otra tarea que lo lleve a un nuevo destino.

El cliente, por tanto, realiza una petición enviando un mensaje, que podemos dividir básicamente en dos partes: **cabecera** y **cuerpo**. La cabecera suele incluir diferentes líneas de texto ASCII, y acaba con un retorno de carro y avance de línea adicional

```
GET    /dwes/pagina.html HTTP/1.1
Host:  www.mipagina.com
Connection: close
User-agent: Mozilla/4.0
Accept-language: sp
```

Entre todas estas líneas destacamos la primera de ellas, conocida como **línea de petición**, que incluye la **versión** del protocolo, la **URL** que define el destino del mensaje, y el **método** utilizado, que permitirá identificar el tipo de operación que el cliente desea hacer con el mensaje. Los métodos más comunes son **GET** y **POST**, y ambos cumplen la misma misión. Entonces, ¿qué los diferencia? El primero transmite la información agregando parámetros a la URL, lo que lo hace poco aconsejable cuando se va a transmitir información sensible. Por otro lado, el método **POST** oculta esta información, enviándola como parte del cuerpo del mensaje.

CABECERA

versión HTTP
página solicitada
método utilizado (GET, POST)

CUERPO

vacío si se usa GET
si usamos POST contiene
información para el servidor

El servidor recibirá la petición y la procesa generando una respuesta, que estará constituido, en primer lugar por la **línea de estado** que, además de información sobre la versión del protocolo utilizado, incluirá un código de respuesta. El primer dígito de dicho código nos dará una pista sobre el resultado de la petición.

2	Petición satisfecha con éxito. El código 200 OK indicará esta situación.
3	Se ha producido una redirección. Se incluirá en la respuesta la nueva URL.
4	Se ha producido un error. Los más habituales son: 400 Bad Request , que indica que la petición no ha podido ser atendida, y el error 404 Not Found indicará que el elemento solicitado no se encuentra en el servidor.
5	Indica que se ha producido un error en el servidor. El más habitual es 500 Internal Server Error .

A continuación aparecerán seis **líneas de cabecera** y, finalmente el **cuerpo** de la entidad, que contendrá en sí la respuesta.

```
HTTP/1.1 200 OK
Connection: close
Date: Mon, 27 Oct 2017 20:00:00 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 27 Oct 2017 19:00:00 GMT
Content-Length: 6821
Content-Type: text/html

datos...
```

Se dice que HTTP es un **protocolo sin estado**, porque el servidor no guarda información alguna sobre los clientes. Esto es, si un cliente solicita la misma información dos veces seguidas, el servidor responderá siempre como si nunca antes hubiese mantenido una comunicación con dicho cliente.

Formularios

Nuestro objetivo será el de construir páginas dinámicas que interaccionen con los usuarios. Esto llevará a nuestra aplicación, no sólo a tener que reaccionar a las acciones del ratón, sino que además deberá proporcionar respuestas en función de la información proporcionada por los usuarios a través del teclado. Utilizaremos para este cometido **formularios**, que no forman parte de PHP, sino que son elementos propios del lenguaje HTML.

Los formularios están constituidos por diferentes **campos**. Estos permitirán a los usuarios proporcionar información que, posteriormente será enviada al servidor e interpretada de forma apropiada, generando una respuesta que será enviada de vuelta a la máquina cliente y visualizada en el navegador.

Recordamos que, la etiqueta utilizada para la definición de un formulario es **<form>**, encargada de encerrar todos los campos que sean necesarios, tales como botones, cuadros de texto, listas desplegables, etc. Completamos la etiqueta con dos atributos obligatorios: **action** y **method**.

El atributo **action** definirá la URL a la que enviar la información contenida en el formulario y que, posteriormente deberemos procesar a través de un lenguaje como PHP. Generalmente, la URL será diferente, aunque en ocasiones podremos tener el formulario y el script de proceso en el mismo documento web. Por otro lado, el atributo **method** permitirá indicar el método HTTP utilizado para el intercambio de información. Sus dos posibles valores son, los ya conocidos **GET** y **POST**.

Dijimos anteriormente que el método **GET** envía la información en forma de parámetros codificados en la URL. De esta manera, los valores del formulario se adjuntan a la URL tras un signo de interrogación (?) en parejas identificador (nombre asociado al campo del formulario) y valor, y se concatenan mediante **&**.

`http://www.mipagina.com/dwes/pagina.html?par1=val1&par2=val2`

No sucederá igual si empleamos el protocolo **POST**. Sea como fuere, el servidor almacenará los nombres de los parámetros, junto con sus valores, en las variables globales **\$_GET** ó **\$_POST**, según el método utilizado. Estas variables, no son más que arrays asociativos, que trataremos más en profundidad en próximos temas.

Existe otro atributo interesante que utilizaremos a lo largo del curso: **enctype**. Este nos permite especificar el tipo de codificación empleada al enviar el formulario. Tendremos que utilizarlo cuando deseemos adjuntar archivos para ser enviados al servidor, en cuyo caso deberemos asignarle el valor **multipart/form-data**.

Etiqueta INPUT

Definido el formulario, podemos incorporar diferentes campos. Entre ellos, el más versátil se identifica mediante la etiqueta **<input />**. Esta etiqueta está vacía, por lo que no debemos escribir su equivalente de cierre, sino incluir una barra inclinada al final de la misma.

Siempre que añadamos un campo al formulario será necesario especificar el tipo de información que se solicita al usuario en cada caso, lo cual es importante para el proceso de

validación del formulario. Utilizamos para ello y de forma **obligatoria** el atributo **type**. Dada su importancia, resumimos en la siguiente tabla sus posibles valores, el significado y uso.

Valor	Descripción
button	Define un botón. Se utiliza generalmente con JavaScript.
checkbox	El campo del formulario será un checkbox. Debe ir acompañado siempre de un valor expresado mediante el atributo value .
color	Muestra en pantalla un selector de color.
date	Muestra un selector de fecha (día, mes y año).
email	Define un campo que únicamente aceptará direcciones de email válidas.
file	Muestra en el formulario un botón que permite seleccionar archivos de la máquina cliente que, posteriormente se subirán al servidor.
hidden	Oculto un campo. Resultará especialmente útil a la hora de intercambiar información con el servidor. Aún estando oculto, la información asociada al campo se enviará al servidor.
image	Añade una imagen a un botón de envío del formulario. Se utiliza conjuntamente con los atributos src (obligatorio), width y height .
month	Muestra un campo para la selección de un mes. Se le puede asignar un valor por defecto a través del valor value .
number	Define un campo de entrada numérica. Se definen restricciones utilizando los atributos min , max , step y value .
password	Muestra un campo para la entrada de contraseñas donde el texto que escribimos queda oculto.
radio	Crea un conjunto de opciones de entre las que el usuario podrá elegir tan sólo una. Se deberán añadir tantos campos de tipo radio como opciones se deseen mostrar al usuario y, todos ellos deberán tener el mismo valor en atributo name .
range	Muestra en el formulario un control de tipo <i>slide</i> para la inserción de un valor numérico. Se podrán añadir restricciones con los atributos min , max , step y value .
reset	Muestra un botón para resetear los campos del formulario.
submit	Muestra un botón de envío del formulario.

Valor	Descripción
text	Muestra un campo que admite texto (los saltos de línea son eliminados automáticamente).
time	Muestra en el formulario un selector de hora y minutos.
url	Define un campo de texto para la inserción de URLs. El navegador Safari detecta automáticamente este tipo de campos en dispositivos móviles y adapta el teclado convenientemente para la inserción de direcciones web.
week	Muestra un selector de semana y año.

Debes recordar que el uso del atributo **name** es **obligatorio** en los campos de un formulario, ya que nos permitirá definir el nombre del parámetro que identificará cada uno de los valores introducidos en el formulario. Se resumen a continuación algunos atributos que también resultan de interés.

Valor	Descripción
checked	Atributo booleano que nos permite indicar si un elemento del formulario estará preseleccionado al cargar la página. Este atributo tiene sentido únicamente cuando el tipo del campo es checkbox o radio .
disabled	Indicará que el campo del formulario está deshabilitado y, por tanto, no sólo no podremos modificar su valor, sino que <u>tampoco será enviado al servidor</u> para su posterior proceso. Es un atributo booleano, por lo que no es necesario darle valor alguno.
autofocus	Atributo booleano que permitirá especificar qué campo del formulario tendrá el foco al cargar la página.
placeholder	Muestra un texto descriptivo en el campo. Funciona únicamente con los campos de tipo text , url , email y password .
autocomplete	Tomará los valores on y off , permitiéndonos especificar cuándo un determinado campo del formulario debe autocompletar su contenido o no. Este atributo funcionará únicamente con campos de tipo text , url , email , password , range , color y todos los relacionados con fechas . También podrá utilizarse en la etiqueta <form> , en cuyo caso su funcionalidad afectará a todos los campos del formulario.

Valor	Descripción
multiple	Atributo booleano que permite indicar que el campo del formulario aceptará más de un valor. Puede utilizarse únicamente con los campos de tipo file (los archivos se seleccionarán haciendo clic sobre ellos mientras se mantiene pulsada la tecla CTRL) y email (las direcciones de correo se deberán introducir separadas por comas).
pattern	Es un atributo bastante importante y útil, ya que nos permite definir, utilizando expresiones regulares , un patrón que deberá cumplir la información introducida en el campo del formulario para que éste se valide correctamente y sea enviado al servidor.
readonly	Atributo booleano que permite especificar si un campo es de sólo lectura. Sea como fuere, la información contenida en dicho campo se enviará al servidor para su proceso.
required	Es también un atributo booleano que indicará al navegador que el campo es obligatorio y, por tanto no podrá enviarse el formulario hasta que éste tenga un valor.
size	Tiene únicamente sentido con los campos de tipo text , url , email , url y password , ya que permite especificar la longitud del texto (en caracteres) que éstos admitirá.
value	Tiene diferente sentido, según el tipo del campo utilizado. Para los campos de tipo button , reset y submit , definirá el texto sobre el botón. Si definimos campos de tipo text , password o hidden , el atributo permitirá especificar el valor por defecto. Por último, si el tipo del campo es checkbox , radio o image , el atributo definirá el valor asociado a la entrada.

Otro de los atributos que se han añadido con la especificación de HTML5 es **list**, cuyo uso depende directamente de la etiqueta **<datalist>**. Ésta, junto a la etiqueta **<option>**, permite definir un conjunto de posibles sugerencias para el campo de entrada donde estemos utilizando el atributo **list**. Actualmente no está soportado en Safari.

```
<form action="proceso.php" method="GET">
  <label for="nav">Introduce el navegador:</label>
  <input name="nav" list="navegadores" />
  <datalist id="navegadores">
    <option value="Edge"></option>
    <option value="Firefox"></option>
    <option value="Google Chrome"></option>
    <option value="Safari"></option>
  </datalist>
</form>
```

Etiqueta TEXTAREA

Aunque hemos podido comprobar la versatilidad de la etiqueta `<input>`, que permite crear diez tipos diferentes de controles de formulario, algunas aplicaciones web utilizan otros elementos que no se pueden crear de esta manera. Un ejemplo de ello son las áreas de texto, bastante útiles cuando deseamos introducir una gran cantidad de texto. La etiqueta `<textarea>` es la encargada de definir este tipo de campos.

```
<textarea cols="40" rows="10"></textarea>
```

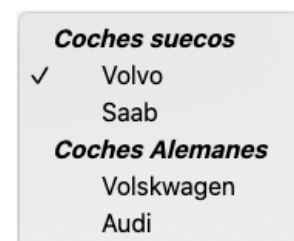
Además de **autofocus**, **disabled**, **form**, **maxlength**, **name**, **placeholder**, **readonly** y **required** esta etiqueta permite el uso de los atributos **cols** y **rows**, que definen respectivamente el ancho y alto (en caracteres) del área de texto. El valor por defecto para el ancho es **20** y para el alto **2**.

Etiqueta SELECT

Al igual que sucede con las áreas de texto, HTML define una etiqueta específica para crear listas desplegables. La encargada de ello es `<select>`, que permite el uso de los atributos **autofocus**, **disabled**, **form**, **multiple**, **name**, **required** y **size**, donde éste último permite especificar el número de opciones visibles en la lista desplegable.

Esta etiqueta se utiliza junto a `<option>` que nos permitirá especificar cada una de las opciones para dicha lista. Esta etiqueta deberá ir acompañada del atributo **value**, con el que tendremos que especificar el valor para cada opción. Independientemente de este, esta etiqueta permite también el uso de los atributos **disabled**, y **selected**.

Junto con listas desplegables podemos utilizar también la etiqueta `<optgroup>`, que nos permitirá agrupar opciones relacionadas dentro de una lista desplegable. El único atributo que suele utilizarse con esta etiqueta (y que además es **obligatorio**) es **label**, que define el nombre de cada agrupación. Los navegadores muestran de forma destacada el título de cada agrupación, de forma que el usuario pueda localizar más fácilmente la opción deseada.



Coches suecos	
✓	Volvo
	Saab
Coches Alemanes	
	Volkswagen
	Audi

```
<select name="coches">
  <optgroup label="Coches suecos">
    <option value="vol">Volvo</option>
    <option value="saa">Saab</option>
  </optgroup>
  <optgroup label="Coches Alemanes">
    <option value="vol">Volkswagen</option>
    <option value="aud">Audi</option>
</select>
```



```
</optgroup>  
</select>
```

Etiqueta BUTTON

Aunque podemos añadir un botón a un formulario utilizando la etiqueta `<input>`, HTML5 incorpora una nueva y más versátil etiqueta: **`<button>`**. La principal diferencia entre crear un botón de una u otra manera es que, la etiqueta `<button>` permite cualquier contenido (excepto enlaces y formularios).

Además de los atributos globales, así como **autofocus**, **disabled**, **form** y **name** ya conocidos, la etiqueta `<button>` puede incluir además el atributo **type**, que podrá tomar los valores **button**, **reset** y **submit**.

Etiqueta METER

Incorporada en HTML5, la etiqueta **`<meter>`** nos ayudará a definir un indicador de valores en un rango conocido. En su caso puede ser utilizada para mostrar la temperatura de un dispositivo, el grado de uso de disco duro, las puntuaciones de los usuarios, etc. En la siguiente tabla se resumen los atributos que podremos utilizar con esta etiqueta. Debemos tener en cuenta que los valores de dichos atributos podrán ser de tipo **entero** o **real**.

Atributo	Descripción
high	Define el punto a partir del cual el indicador considera que el valor es alto, por lo que el navegador lo coloreará de forma diferente.
low	Realiza la función contraria al atributo anterior.
max	Valor máximo que puede alcanzar el indicador. Toma la unidad como valor por defecto.
min	Valor mínimo que puede alcanzar el indicador, siendo cero el valor por defecto.
optimum	Especifica el valor óptimo para el rango.
value	Define el valor actual del indicador.

Etiqueta FIELDSET

Esta etiqueta nos permitirá agrupar y organizar los campos de un formulario, dibujando una caja en torno a los campos que agrupa. Generalmente puede utilizarse conjuntamente con la etiqueta **`<legend>`** que facilitará la definición de una leyenda o título para el grupo de elementos.

Etiqueta PROGRESS

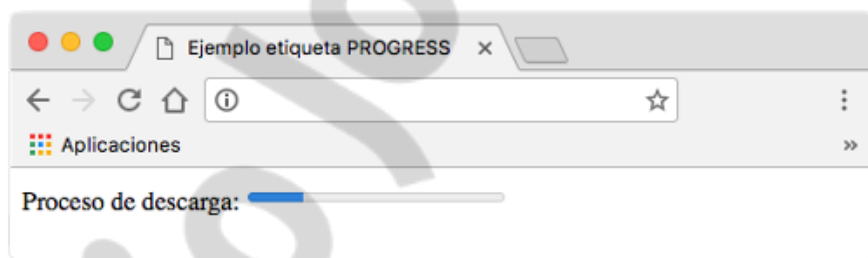
Utilizado junto con JavaScript podremos utilizar el control **<progress>** para indicar el progreso de una determinada tarea, como por ejemplo cuánto resta para completar un formulario, el progreso de una descarga o subida de un archivo, etc. Aunque suele utilizarse conjuntamente con formularios, podemos utilizarla fuera de estos para reseñar el progreso de cualquier otra tarea.

Atributo	Descripción
max	Permite especificar cuántos pasos requiere la tarea actual.
value	Define cuánto de la tarea se ha completado hasta el momento.

Veamos un ejemplo:

```
Proceso de descarga:  
<progress value="22" max="100"></progress>
```

Añadir la línea de código anterior a nuestro documento HTML provocará la siguiente salida en el navegador:



Recepción y proceso de datos

Cuando enviamos al servidor el formulario y su información asociada, podemos rescatar dicha información utilizando las variables predefinidas **\$_GET** y **\$_POST** en función del método empleado para el envío de los datos. Como se dijo anteriormente, ambas variables son un array asociativo, donde el índice de cada elemento es el nombre de cada una de las variables incluidas en la URL. Más adelante hablaremos detenidamente de este tipo de arrays.

```
<form action="proceso.php" method="GET">  
  <label for="nom">Nombre:</label>  
  <input type="text" name="nom" />  
  <br/>  
  <label for="eda">Edad:</label>
```

```
<input type="number" name="eda" />
<br/>
<input type="submit" name="btt" value="Enviar" />
</form>
```

Supongamos el fragmento de código anterior. Cuando el usuario pulse el botón **ENVIAR**, el contenido del formulario se enviará a la página **proceso.php**. Como estamos utilizando el método **GET**, el identificador y el valor de cada parámetro se adjuntará codificado en la URL tal y como se indicó en el apartado anterior.

`proceso.html?nom=valor1&eda=valor2&btt=Enviar`

Observamos que la información se envía codificada en formato texto. Obviamente, los campos del formulario podrían haberse definido todos de tipo texto, pero cada tipo impone ciertas limitaciones a la hora del envío. De esta manera, si introducimos un valor erróneo en un campo, el formulario no se enviará hasta que sea corregido.

Pero, ¿cómo accede el script `proceso.php` al valor de esos parámetros? Como la información del formulario se ha enviado utilizando el método **GET**, tendremos que usar el array **\$_GET** y el nombre de cada parámetro como índice para acceder a cada valor. De esta manera, el código del script PHP podría ser el siguiente:

```
<h3>Resultado del formulario</h3>
<?php
    $nombre = $_GET["nom"] ;
    $apellidos = $_GET["ape"] ;
    echo "Bienvenido/a, $nombre $apellidos<br/>" ;
?>
```

Obviamente, si el método utilizado para enviar la información fuese **POST**, utilizaríamos de igual manera la variable global **\$_POST**. En cualquier caso, cada uno de los métodos plantea una serie de cuestiones que hay que tener en cuenta. En tanto que el método **GET** no permite enviar información binaria (archivos, imágenes, etc.), el método **POST** rompe la funcionalidad del botón **ATRÁS** del navegador y el de actualizar repite la operación.

Capturar información de un campo CHECKBOX

La información transmitida a través de un formulario, se rescata desde PHP independientemente del tipo del campo. No obstante, existen situaciones deben tratarse de manera diferente. Una de estas situaciones la encontramos cuando decidimos utilizar el tipo **checkbox**.

Los campos de este tipo permiten seleccionar diferentes valores al mismo tiempo, lo cual nos lleva a una pregunta. Sabiendo que la información se envía en pares (identificador,

valor), si un campo de tipo *checkbox* nos permite seleccionar múltiples opciones, ¿cómo enviamos dicha información? Supongamos el siguiente formulario.

```
<form action="proceso.php" method="GET">
  <label for="afic">Aficiones: </label>
  <input type="checkbox" name="afic" value="1" />Fútbol</br>
  <input type="checkbox" name="afic" value="2" />Baloncesto</br>
  <input type="checkbox" name="afic" value="3" />Videojuegos</br>
  <input type="checkbox" name="afic" value="4" />Música</br>
  <input type="checkbox" name="afic" value="5" />Leer</br>
  <input type="submit" name="sub" value="Enviar" />
</form>
```

En primer lugar, observamos en el fragmento de código anterior, que todos los campos de tipo *checkbox* tienen el mismo valor de id. Esto es completamente lógico, dado que todas las opciones suponen una posible respuesta a una misma pregunta. Pero, si enviásemos ahora el formulario y capturáremos en el script PHP correspondiente el valor del campo `afi`, tendríamos que únicamente ha llegado el último valor seleccionado.

Para solucionar este problema tendremos que hacer uso de *arrays*... Sí. ¡Existen los arrays en HTML! Definimos el nombre de cada elemento **input** de tipo *checkbox* como `afi[]` con lo que estamos indicando al navegador que todos estos campos forman parte de una colección de elementos y cada uno de ellos tendrá un determinado valor.

```
<form action="proceso.php" method="GET">
  <label for="afic">Aficiones: </label>
  <input type="checkbox" name="afi[]" value="1" />Fútbol</br>
  <input type="checkbox" name="afi[]" value="2" />Baloncesto</br>
  <input type="checkbox" name="afi[]" value="3" />Videojuegos</br>
  <input type="checkbox" name="afi[]" value="4" />Música</br>
  <input type="checkbox" name="afi[]" value="5" />Leer</br>
  <input type="submit" name="sub" value="Enviar" />
</form>
```

Efectivamente, definimos cada campo del formulario como un array que, posteriormente será procesado del lado del servidor. Así, el valor de `$_GET["afi"]` será un array escalar, y la manera en que debemos trabajar con él será estudiada más adelante.

Capturar información desde un campo **SELECT** múltiple

Recordamos que la etiqueta `<select>` nos permite elegir una opción de entre un grupo predefinido de posibilidades. No obstante, utilizando el atributo **multiple** modificamos el comportamiento de dicha etiqueta, que ahora nos permitirá elegir más de una opción de entre las disponibles. Supongamos el siguiente fragmento de código.

```

<form action="proceso.php" method="GET">
  <label for="afis">Aficiones: </label>
  <select name="afis" size="5" multiple>
    <option value="1">Fútbol</option>
    <option value="2">Baloncesto</option>
    <option value="3">Videojuegos</option>
    <option value="4">Música</option>
    <option value="5">Leer</option>
  </select>
  <input type="submit" value="Enviar" />
</form>

```

Nos encontramos ante una situación similar a la que planteamos en el apartado anterior. Tal y como se ha definido el campo **select**, al hacer clic en el botón, se enviará el último valor seleccionado. Para solucionar este problema, optamos por utilizar también un *array*, ya que nuestra intención es que este campo envíe cero, uno o más de un valor.

Hacemos **name="afis[]"** en la etiqueta **<select>** y, una vez enviado el formulario nos encontraremos en el servidor con la misma situación que antes. El valor de la variable **\$_GET["afis"]** será también un *array* que deberemos tratar de forma apropiada. De todas formas, con los conocimientos que tenemos hasta el momento, podemos comprobar que efectivamente se envían los valores de manera correcta.

```

<?php
echo "Aficiones con <strong>checkbox</strong>: " ;
print_r($_GET["afic"]);

echo "<br/>" ;
echo "Aficiones con <strong>select</strong>: " ;
print_r($_GET["afis"]);

```

Máquina de estados

Hasta el momento, cuando ha sido necesario solicitar información por teclado al usuario, hemos creado un documento web con un formulario que, posteriormente era enviado a un script PHP encargado de procesar los datos. En desarrollos sencillos puede verse como una solución factible y podríamos permitirnos trabajar de esta manera. No obstante, el número de archivos irá aumentando progresivamente en nuestro servidor, conforme nuestra aplicación vaya creciendo.

¿Cómo podríamos solucionar el problema anterior? Fácil. Haciendo que una página se envíe la información a sí misma. De esta manera implementamos lo que se conoce como **máquina de estados**. Al comienzo de la página comprobaremos si estamos recibiendo información o

no, realizando una acción u otra según convenga. Esto reduciría el número de ficheros de un proyecto, mejorando igualmente la posibilidad de fallo.

No obstante, aunque es una técnica aconsejable, no siempre será conveniente utilizarla, quedando en manos del desarrollador determinar cuándo es o no factible. Veamos un ejemplo. Como queremos solicitar al usuario su nombre y apellidos, construimos un script PHP con el formulario apropiado.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Programando con PHP</title>
    <meta charset="UTF-8" />
  </head>
  <body>
    <form method="GET">
      <input type="text" name="nom" /><br/>
      <input type="number" name="eda" min="18" value="18" required /><br/>
      <button type="submit">Enviar datos</button>
    </form>
  </body>
</html>
```

Como única novedad, hasta el momento, observamos que hemos omitido el atributo `action` en la etiqueta `<form>`. Hasta ahora habíamos dicho que era obligatorio su uso pero, a partir de HTML5 podemos omitirlo si deseamos que la información del formulario se envíe al mismo documento web.

Como la información será enviada al mismo script, tendremos que abrir un ámbito PHP donde recojamos dicha información. Recordemos utilizar la variable global correspondiente cuando el formulario se envía mediante el método **GET**. Añadimos en este caso las siguientes líneas.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Programando con PHP</title>
    <meta charset="UTF-8" />
  </head>
  <body>
    <?php
      $nombre = $_GET["nom"] ;
      $edad   = $_GET["eda"] ;
    ?>
    <form method="GET">
      <input type="text" name="nom" /><br/>
      <input type="number" name="eda" min="18" value="18" required /><br/>
```

```
<button type="submit">Enviar datos</button>
</form>
</body>
</html>
```

Las variables **\$nombre** y **\$edad** contendrán los valores enviados desde el formulario. Teclea el código anterior en tu máquina y realiza la petición al servidor. ¿Qué sucede? Efectivamente: no se encuentran los índices nom y eda en **\$_GET**, porque aún no se ha enviado el formulario. ¿Cómo podemos evitar que se produzca este error? Nos podemos sentir tentados de utilizar una sentencia if y realizar una sencilla comprobación. En este caso necesitaremos recurrir a la función **isset()** que comprobará si una variable está definida y no es NULL.

```
if (isset($_GET["nom"])) {
    $nombre = $_GET["nom"] ;
    $edad    = $_GET["eda"] ;
}
```

La primera vez que se carga el documento en el navegador se comprobará si se ha definido la variable **\$_GET["nom"]**. En caso afirmativo, se procederá a almacenar en las variables **\$nombre** y **\$edad** los valores correspondientes. Basta con comprobar si existe **\$_GET["nom"]**, ya que, tal y como hemos definido el formulario, siempre se enviarán valores para ambos campos. Obviamente, no tenemos en cuenta la posibilidad de que alguien modifique directamente en el navegador los parámetros de la URL. No obstante, existe la posibilidad de utilizar el operador ternario **?:** o el operador de **fusión de null** (null coalesce) incorporado en PHP 7. Optamos por esta última alternativa, que queda mucho más elegante.

```
$nombre = $_GET["nom"]??"" ;
$edad    = $_GET["eda"]??18 ;
```

De esta manera, incluso definimos un valor inicial por defecto para **\$nombre** y **\$edad**, si no se nos ha proporcionado alguno a través del formulario. Finalmente mostraremos un sencillo saludo al usuario indicando cuál es su edad.

```
$nombre = $_GET["nom"]??"" ;
$edad    = $_GET["eda"]??18 ;
if (!empty($nombre))
    echo "Bienvenido/a $nombre, tienes $edad años.<br/>" ;
```

La función **empty()** que hemos utilizado en la sentencia de selección se encargará de comprobar si la variable **\$nombre** está vacía. Nótese que en este caso la variable si está definida. Hemos de tener en cuenta que una variable se considerará vacía si contiene los valores **0**, **0.0**, **NULL**, **false**, **"0"**, **""** o si es un array vacío.

Bibliografía

PHP.net

Guía oficial del lenguaje

PHP and MySQL Web Development

Luke Welling, Laura Thompson

Developer's Library

Learning PHP7

Antonio López

Packt Publishing