Índice

Controles en Formularios

Introducción

Para que un control envíe información es necesario:

- Que el control esté incluido en un formulario (<**form**>).
- Que el formulario tenga establecido el atributo **action**, con la dirección absoluta o relativa del fichero php que procesa la información.
- Que el control tenga establecido el atributo **name**.
  Nota: el atributo **name** puede contener cualquier carácter (números, acentos, guiones, etc), pero si contiene espacios, PHP sustituye los espacios por guiones bajos (_) al procesar los datos enviados.
- Que el formulario contenga un botón de tipo submit.

Un ejemplo de un formulario válido es:

```
<form action="ejemplo.php">
  <p>Nombre: <input type="text" name="nombre"></p>
  <p><input type="submit" value="Enviar"></p>
</form>
```

El programa que recibe los datos, los guarda automáticamente en la matriz **$_REQUEST**. Mediante la orden **print_r($_REQUEST)** se puede mostrar el contenido de dicha matriz.
El siguiente ejemplo muestra lo que escribiría el programa PHP (ejemplo.php) si recibiera la información del formulario anterior (ejemplo.html).

```
<?php
  print "<pre>";
  print_r($_REQUEST);
  print "</pre>";
?>
```

```
Array
(
    [nombre] =>
)
```

El nombre del elemento de la matriz $_REQUEST coincide con el nombre del control (atributo **name**) en el formulario (excepto en el control de tipo imagen).

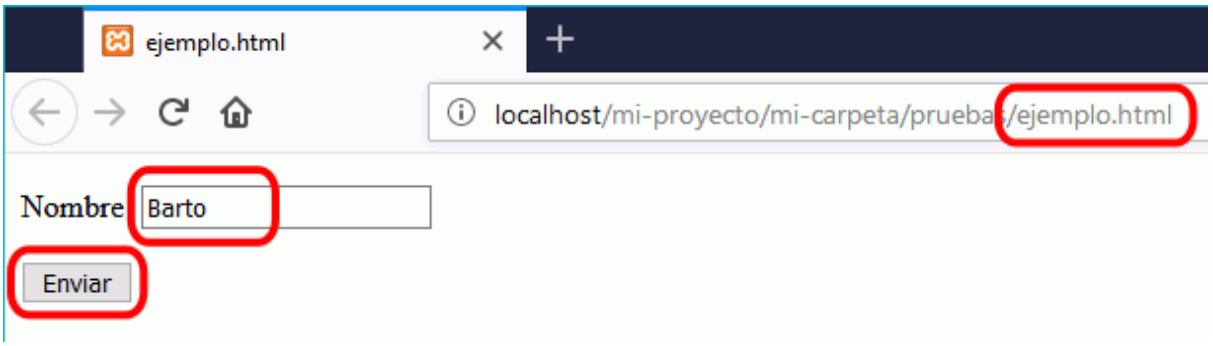
Atributo method

El atributo **method** de la etiqueta **<form>** permite elegir si la información de los controles se incluye en la llamada a la página (**method="get"**) o se proporciona posteriormente (**method="post"**). Si el atributo no está definido, la opción por defecto es get.

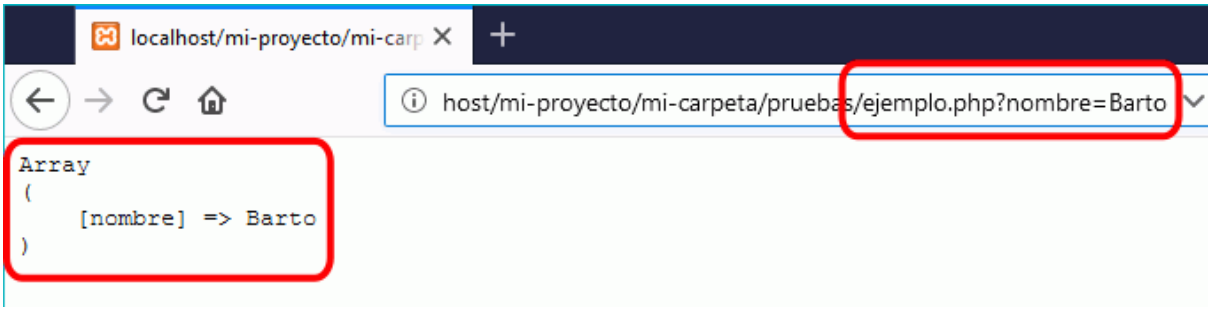
Con el valor get se pueden ver en la barra de dirección los nombres de los controles y los valores introducidos por el usuario, mientras que con el valor post no, pero los datos recibidos son idénticos.

- Formulario con **get**

```
<form action="ejemplo.php" method="get">
  <p>Nombre: <input type="text" name="nombre"></p>
  <p><input type="submit" value="Enviar"></p>
</form>
```




```
<?php
  print_r($_GET);          //  Ambos  array  tienen  la  misma
información
  print_r($_REQUEST);
?>
```

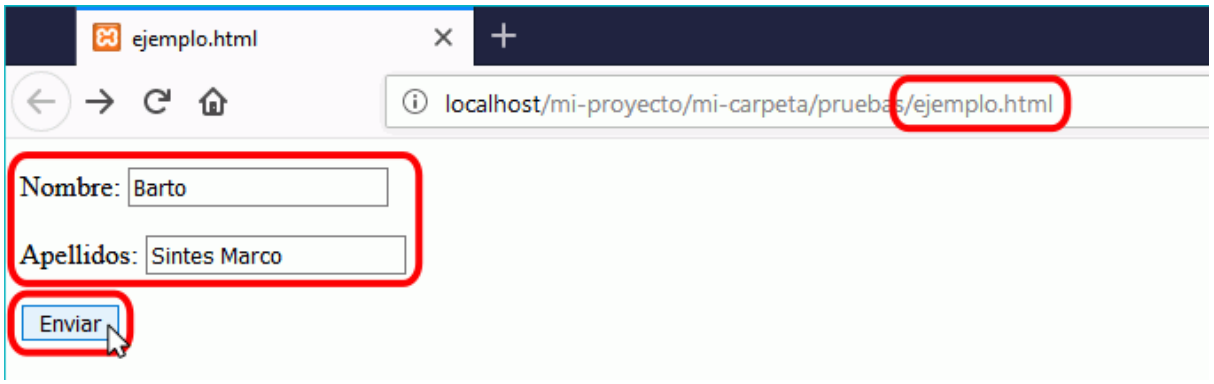


Las principales características del envío de formulario por el método **get** son:

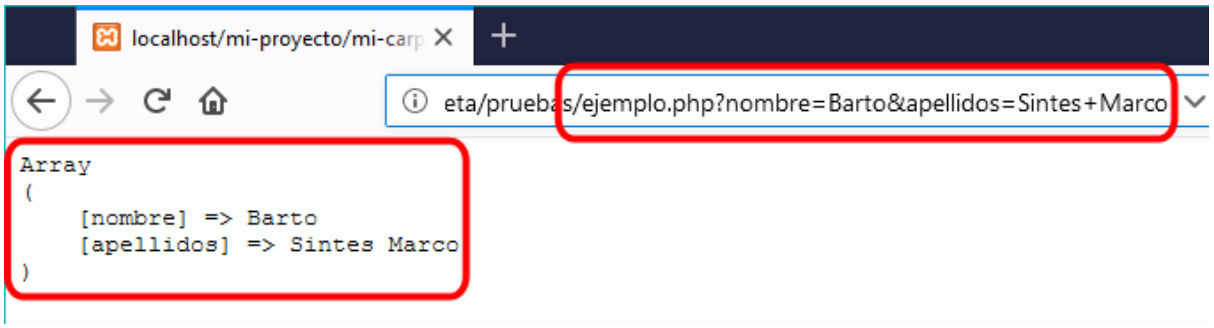
- Envía la información en la propia URL, estando limitada a 2000 caracteres.
- La información es visible por lo que con este método nunca se envía información sensible.
- No se pueden enviar datos binarios (archivos, imágenes...).
- En PHP los datos se pueden recibir con el array asociativo $_GET además del $_REQUEST.

En caso de que haya varios controles que envíen información en un formulario con **get**, los nombres y valores aparecen en la barra de dirección separados por el carácter ampersand (&), como nombre1=valor1&nombre2=valor2&...

```
<form action="ejemplo.php" method="get">
  <p>Nombre: <input type="text" name="nombre"></p>
  <p>Apellidos: <input type="text" name="apellidos"></p>
  <p><input type="submit" value="Enviar"></p>
</form>
```

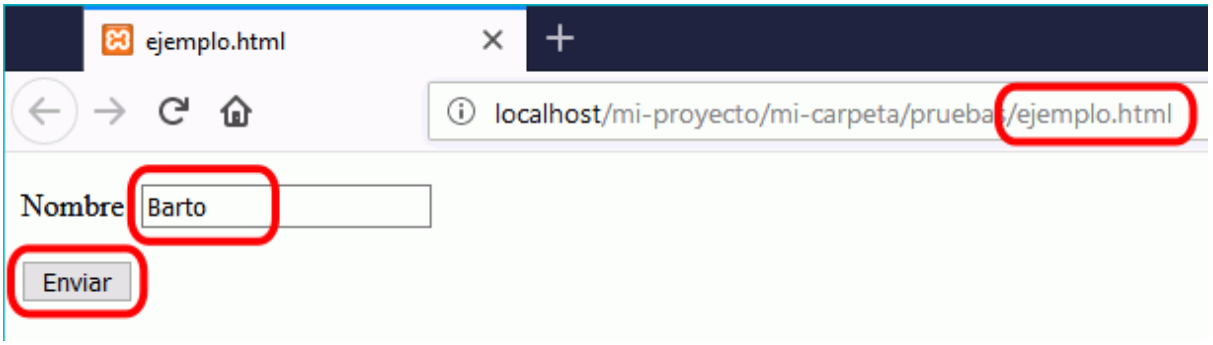



```
<?php
  print_r($_GET);          //  Ambos  array  tienen  la  misma
información
  print_r($_REQUEST);
?>
```



- Formulario con **post**

```
<form action="ejemplo.php" method="post">
  <p>Nombre: <input type="text" name="nombre"></p>
  <p><input type="submit" value="Enviar"></p>
</form>
```




```
<?php
  print_r($_POST);    // Ambos array tienen la misma información
  print_r($_REQUEST);
?>
```

Las principales características del envío de formulario por el método **post** son:

- No tiene límite de cantidad de información a enviar.
- La información proporcionada no es visible, por lo que se puede enviar información sensible.
- Se puede usar para enviar texto normal, así como datos binarios (archivos, imágenes...).
- En PHP los datos se pueden recibir con el array asociativo $_POST además del $_REQUEST.

Botón Enviar

Este control se puede crear con la etiqueta **<input>** o con la etiqueta **<button>**. En ambos casos, este control se envía siempre que esté definido el atributo **name** y el valor enviado es el valor del atributo **value** o el contenido de la etiqueta.

Este control se puede crear con la etiqueta <input> o con la etiqueta <button>. En ambos casos, este control se envía siempre que esté definido el atributo name y el valor enviado es el valor del atributo value o el contenido de la etiqueta.

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

```
<?php
  print_r($_REQUEST);
?>
```

```
Array
(
)
```

```
<input type="submit" value="Enviar" name="boton1">
```

```
<?php
  print_r($_REQUEST);
?>
```

```
Array
(
  [boton1] => Enviar
)
```

```
<button type="submit">Submit</button>
```

```
<?php
  print_r($_REQUEST);
?>
```

```
Array
(
)
```

```
<button type="submit" name="boton2">Enviar</button>
```

```
<?php
  print_r($_REQUEST);
?>
```

```
Array
(
  [boton2] => Enviar
)
```

```
<button type="submit" value="Enviar"
name="boton3">Submit</button>
```

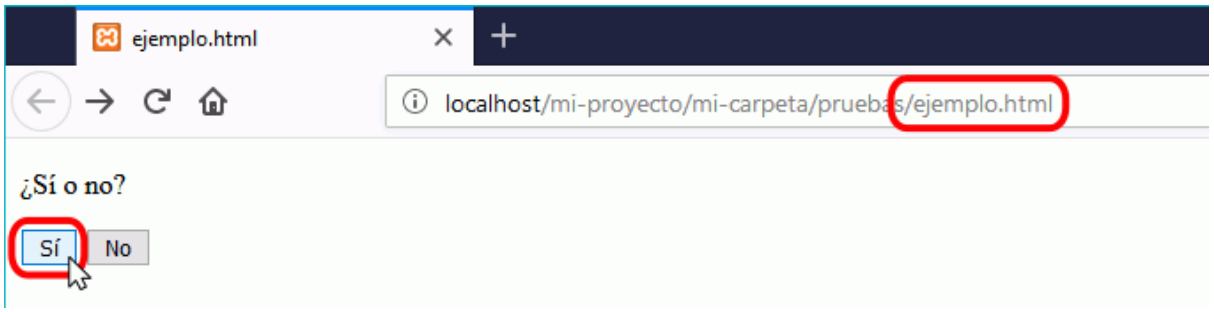
```
<?php
  print_r($_REQUEST);
?>
```

```
Array
```

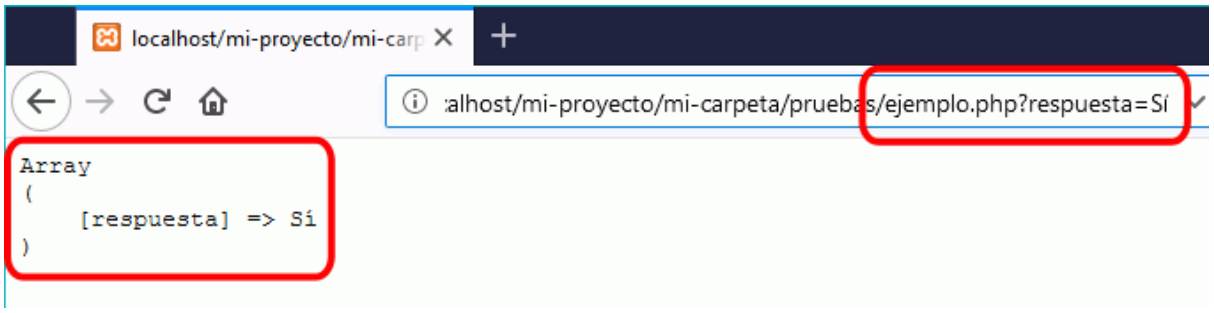
```
(
   [boton3] => Enviar
)
```

Normalmente no se suele dar el atributo **name** al botón enviar ya que la información se envía cuando se hace clic en el botón. En algunos casos, sí que puede ser conveniente, por ejemplo cuando el formulario contiene varios botones y queremos saber cuál se ha pulsado, como en el ejemplo siguiente, en el que los atributos **name** son iguales, pero los atributos value son distintos:

```
<form action="ejemplo.php" method="get">
  <p>¿Sí o no?</p>
  <p>
    <input type="submit" name="respuesta" value="Sí">
    <input type="submit" name="respuesta" value="No">
  </p>
</form>
```




```
<?php
  print_r($_REQUEST);
?>
```



Caja de texto, caja de contraseña y área de texto

Este control se envía siempre. El valor enviado es el contenido de la caja o área.

```
<input type="text" name="cajatexto1">
```

```
<?php
  print_r($_REQUEST);
?>
```

```
Array
(
    [cajatexto1] =>
)
```

```
<input type="text" name="cajatexto2" value="Cualquier valor">
```

```
<?php
  print_r($_REQUEST);
?>
```

```
Array
(
    [cajatexto2] => Cualquier valor
)
```

```
<input type="password" name="contrasena1">
```

```
<?php
  print_r($_REQUEST);
?>
```

```
Array
(
    [contrasena1] =>
)
```

```
<input type="password" name="contrasena2" value="123456">
```

```
<?php
  print_r($_REQUEST);
?>
```

```
Array
(
    [contrasena2] => 123456
```

```
)
```

```
<textarea rows="4" cols="20" name="area1"></textarea>
```

```
<?php
  print_r($_REQUEST);
?>
```

```
Array
(
    [area1] =>
)
```

```
<textarea rows="4" cols="20" name="area2"
value="123456">Contenido caja</textarea>
```

```
<?php
  print_r($_REQUEST);
?>
```

```
Array
(
    [area2] => Contenido caja
)
```

Casilla de verificación

Este control se envía solamente si se marca la casilla. El valor enviado es "on" si la casilla no tiene definido el atributo **value** o el valor del atributo **value** si ésta está definida.

```
<input type="checkbox" name="casilla1"> (sin marcar)
```

```
<?php
  print_r($_REQUEST);
?>
```

```
Array
(
)
```

```
<input type="checkbox" name="casilla1"> (marcada)
```

```
<?php
  print_r($_REQUEST);
?>
```

```
Array
(
    [casilla] => on
)
```

```
<input type="checkbox" name="casilla2" value="marcada"> (marcada)
```

```
<?php
  print_r($_REQUEST);
?>
```

```
Array
(
    [casilla2] => marcada
)
```

Botón radio

Este control se envía solamente si se marca alguno de los botones radio que forman el control. El valor enviado es "on" si el botón marcado no tiene definido el atributo value o el valor del atributo **value** si éste está definido.

```
<input type="radio" name="radio1"> (sin marcar)
<input type="radio" name="radio1">
```

```
<?php
  print_r($_REQUEST);
?>
```

```
Array
(
)
```

```
<input type="radio" name="radio1"> (marcada esta opción)
<input type="radio" name="radio1">
```

```
<?php
  print_r($_REQUEST);
?>
```

```
Array
(
    [radio1] => on
)
```

```
<input type="radio" name="radio1">
<input type="radio" name="radio1"> (marcada esta opción)
```

```
<?php
  print_r($_REQUEST);
?>
```

```
Array
(
    [radio1] => on
)
```

```
<input type="radio" name="radio2" value="uno"> (marcado uno)
<input type="radio" name="radio2" value="dos">
```

```
<?php
  print_r($_REQUEST);
?>
```

```
Array
(
    [radio2] => uno
)
```

```
<input type="radio" name="radio2" value="uno">
<input type="radio" name="radio2" value="dos"> (marcado dos)
```

```
<?php
  print_r($_REQUEST);
?>
```

```
Array
```

```
(
    [radio2] => dos
)
```

Menú (select)

Este control envía siempre la opción elegida. El valor enviado es el contenido de la etiqueta option elegida si la opción elegida no tiene definido el atributo value o el valor del atributo value si éste está definido.
Si el menú admite selección múltiple, entonces el nombre del menú debe acabar con corchetes ([]) y se envía como una matriz, de tantos elementos como opciones se hayan elegido.

```
<select name="menu1">
  <option></option>
</select>
```

```
<?php
  print_r($_REQUEST);
?>
```

```
Array
(
    [menu1] =>
)
```

```
<select name="menu1">
  <option>Opción 1</option> (seleccionamos opción 1)
  <option>Opción 2</option>
</select>
```

```
<?php
  print_r($_REQUEST);
?>
```

```
Array
(
    [menu1] => Opción 1
)
```

```
<select name="menu2">
  <option value="Uno">Opción 1</option> (seleccionamos opción 1)
  <option value="Dos">Opción 2</option>
```

```
</select>
```

```
<?php
  print_r($_REQUEST);
?>
```

```
Array
(
    [menu1] => Uno
)
```

```
<select name="menu3[]" size="3" multiple>
  <option>Opción 1</option> (seleccionamos la opción 1 y 3)
  <option>Opción 2</option>
  <option>Opción 3</option> (seleccionamos la opción 1 y 3)
  <option>Opción 4</option>
</select>
```

```
<?php
  print_r($_REQUEST);
?>
```

```
Array
(
    [menu3] => Array
    (
          [0] => Opción 1
          [1] => Opción 2
    )
)
```

Control oculto (input hidden)

Este control se envía siempre y el valor enviado es el valor del atributo **value**.

```
<input type="hidden" name="oculto1">
```

```
<?php
  print_r($_REQUEST);
?>
```

```
Array
```

```
(
    [oculto1] =>
)
```

```
<input type="hidden" name="oculto2" value="valor">
```

```
<?php
  print_r($_REQUEST);
?>
```

```
Array
(
    [oculto2] => Valor
)
```

Imagen (input image)

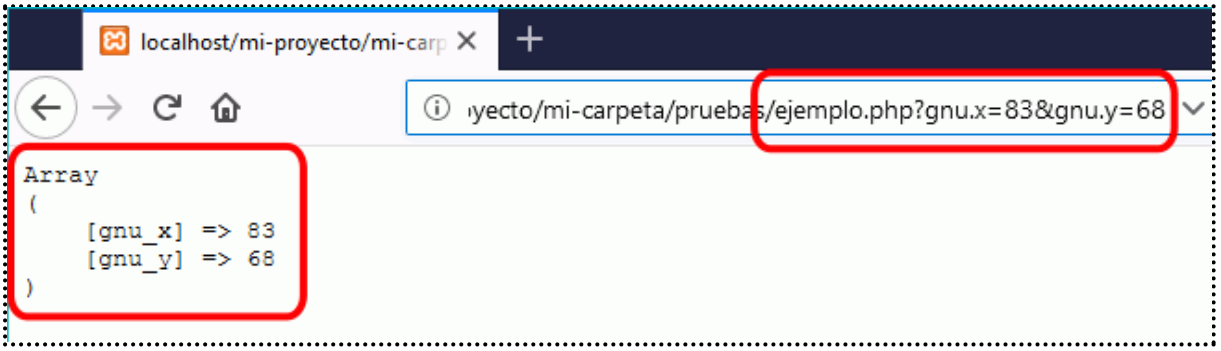
El control de tipo imagen inserta una imagen que funciona como un botón (aunque ni Firefox ni Internet Explorer le da relieve como a los botones). Al hacer clic en un punto de la imagen es como si se hubiera pulsado a un botón submit y se envían las coordenadas del punto en el que se ha hecho clic (junto con los valores de los otros controles del formulario).

El origen de las coordenadas es el extremo superior izquierdo de la imagen. El valor X aumenta a medida que nos desplazamos a la derecha y el valor Y aumenta a medida que nos desplazamos hacia abajo.

```
<input type="image" name="gnu" src="gnu.png" alt="Logotipo GNU">
```

```
Array
(
  [gnu_x] => 83
  [gnu_y] => 68
)
```

Las coordenadas se reciben en $_REQUEST en dos elementos que añaden al nombre del control el sufijo _x e _y. En la barra de dirección el nombre aparece con el sufijo .x e .y.



Si se define el atributo **value**, el formulario debe enviar tanto las coordenadas como el nombre del control con el valor del atributo **value**. Dependiendo de la versión del navegador es posible que el value de la imagen no se envíe.

Archivo (input file)

El selector de archivo permite enviar un archivo desde el ordenador del cliente al servidor. En un formulario "normal", este control se envía siempre y el valor enviado es el nombre del archivo elegido.

```
<input type="file" name="archivo">
```

```
<?php
  print_r($_REQUEST);
?>
```

```
Array
(
  [archivo] => texto.txt
)
```

Para que este control envíe toda la información, el formulario debe tener el atributo enctype con el valor **multipart/form-data** y ser enviado con el método **POST**. La información se almacena entonces en la matriz **$_FILES** (pero no en la variable $_REQUEST).

```
<form enctype="multipart/form-data" action="ejemplo.php"
method="post">
  <input type="file" name="archivo2">
</form>
```

```
<?php
```

```
  print_r($_FILES);
?>
```

```
Array
(
    [archivo1] => Array
    (
          [name] => loquesea.txt
          [type] => text/plain
          [tmp_name] => C:\ejemplos\texto.txt
          [error] => 0
          [size] => 27890
    )
)
```

Antes de utilizar este control, hay que configurar en el archivo php.ini el tamaño máximo de los archivos que se pueden enviar (mediante la directiva post_max_size).

Recogida de datos

Matriz $_REQUEST

Cuando se envía un formulario, PHP almacena la información recibida en una matriz llamada **$_REQUEST**. El número de valores recibidos y los valores recibidos dependen tanto del formulario como de la acción del usuario.

Cualquier control se envía solamente si está establecido su atributo name. El atributo name del control puede contener cualquier carácter (números, acentos, guiones, etc), pero si contiene espacios, los espacios se sustituyen por guiones bajos (_). Cada control crea un elemento de la matriz **$_REQUEST**, que se identifica como $_REQUEST[valor_del_atributo_name] y que contiene el valor entregado por el formulario (en su caso).

Mientras se está programando, para comprobar que el fichero php está recibiendo la información enviada por el control, lo más fácil es utilizar la función print_r($matriz) para mostrar el contenido de la matriz **$_REQUEST**. Una vez se ha comprobado que la información llega correctamente, la línea se debe comentar o eliminar.

Referencia a $_REQUEST dentro y fuera de cadenas

Al hacer referencia a los elementos de la matriz $_REQUEST, hay que tener en cuenta si la referencia se encuentra dentro de una cadena o fuera de ella.

- Si se hace referencia dentro de una cadena, el índice se debe escribir sin comillas. Si se escribe cualquier tipo de comillas (simples o dobles) se produce un error.

```
<form action="ejemplo.php" method="post">
          <p>Nombre:     <input     type="text"     name="nombre"
value="Antonio"></p>
  <p><input type="submit" value="Enviar"></p>
</form>
```

```
<?php
  print "<p>Su nombre es $_REQUEST[nombre]</p>";
?>
```

```
<p>Su nombre es Antonio</p>
```

```
<?php
  print "<p>Su nombre es $_REQUEST["nombre"]</p>";
?>
```

```
Parse error: syntax error, unexpected '"', expecting '-' or
identifier (T_STRING) or variable (T_VARIABLE) or number
(T_NUM_STRING) in ejemplo.php on line 2
```

```
<?php
  print "<p>Su nombre es $_REQUEST['nombre']</p>";;
?>
```

```
Parse error: syntax error, unexpected ''
(T_ENCAPSED_AND_WHITESPACE), expecting '-' or identifier
(T_STRING) or variable (T_VARIABLE) or number (T_NUM_STRING) in
ejemplo.php on line 2
```

- Si se hace referencia fuera de una cadena, el índice se debe escribir con comillas dobles o simples. Si se escribe sin comillas, se produce un aviso

```
<?php
  print "<p>Su nombre es " . $_REQUEST["nombre"] . "</p>";
?>
```

```
<p>Su nombre es Antonio</p>
```

```
<?php
  print "<p>Su nombre es " . $_REQUEST['nombre'] . "</p>";
?>
```

```
<p>Su nombre es Antonio</p>
```

```
<?php
  print "<p>Su nombre es " . $_REQUEST[nombre] . "</p>";
?>
```

```
Warning: Use of undefined constant nombre - assumed 'nombre' in
ejemplo.php on line 2

<p>Su nombre es Antonio</p>
```

En ningún caso se pueden utilizar los caracteres especiales \" o \', ni dentro ni fuera de las cadenas.

```
<?php
  print "<p>Su nombre es $_REQUEST[\"nombre\"]</p>";
?>
```

```
Parse error: syntax error, unexpected '' (T_ENCAPSED_AND_WHITESPACE), expecting '-' or identifier (T_STRING) or variable (T_VARIABLE) or number (T_NUM_STRING) in ejemplo.php on line 2
```

```
<?php
  print "<p>Su nombre es $_REQUEST[\'nombre\']</p>";
?>
```

```
Parse error: syntax error, unexpected '' (T_ENCAPSED_AND_WHITESPACE), expecting '-' or identifier (T_STRING) or variable (T_VARIABLE) or number (T_NUM_STRING) in ejemplo.php on line 2
```

Comprobación de existencia

Un programa PHP no debe suponer que los controles le van a llegar siempre porque cuando eso no ocurra, el programa no funcionará correctamente.

Controles vacíos

La primera situación que los programas deben tener en cuenta es que los controles no contengan ningún valor.

En el programa de ejemplo, si el usuario ha utilizado el formulario y ha escrito un dato, el programa PHP recibe el control y funciona correctamente:

```
<form action="ejemplo.php" method="post">
          <p>Nombre:     <input     type="text"     name="nombre"
value="Antonio"></p>
  <p><input type="submit" value="Enviar"></p>
</form>
```

```
<?php
  print "<pre>";
  print_r($_REQUEST);
  print "</pre>";
  print "<p>Su nombre es $_REQUEST[nombre]</p>";;
?>
```

```
<pre>
Array
(
    [nombre] => Antonio
)
</pre>
<p>Su nombre es Antonio</p>
```

Pero si el usuario no escribe nada en el formulario, aunque el programa funcione correctamente, la respuesta del programa puede confundir al usuario:

```
<form action="ejemplo.php" method="post">
  <p>Nombre: <input type="text" name="nombre" value=""></p>
  <p><input type="submit" value="Enviar"></p>
</form>
```

```
<?php
  print "<pre>";
  print_r($_REQUEST);
  print "</pre>";
  print "<p>Su nombre es $_REQUEST[nombre]</p>";
?>
```

```
<pre>
Array
(
    [nombre] =>
)
</pre>
<p>Su nombre es</p>
```

Este problema se puede resolver incluyendo una estructura if ... else ... que considere la posibilidad de que no se haya escrito nada en el formulario:

```
<form action="ejemplo.php" method="post">
  <p>Nombre: <input type="text" name="nombre" value=""></p>
  <p><input type="submit" value="Enviar"></p>
</form>
```

```
<?php
  print "<pre>";
  print_r($_REQUEST);
  print "</pre>";
  if ($_REQUEST["nombre"] == ""){
    print "<p>No ha escrito ningún nombre</p>";
  }else{
    print "<p>Su nombre es $_REQUEST[nombre]</p>";
  }
?>
```

```
<pre>
Array
(
    [nombre] =>
)
</pre>
<p>No ha escrito ningún nombre</p>
```

Controles inexistentes: isset()

Un problema más grave es que el programa suponga que existe un control que en realidad no le ha llegado.

Eso puede ocurrir, por ejemplo, en el caso de las casillas de verificación y los botones radio, ya que los formularios envían el control solamente cuando se han marcado.

```
<form action="ejemplo.php" method="post">
  <p>Deseo recibir información:
    <input type="checkbox" name="acepto"> (marcada la casilla)
  </p>
  <p><input type="submit" value="Enviar"></p>
</form>
```

```
<?php
  print "<pre>";
```

```
  print_r($_REQUEST);
  print "</pre>";
  if ($_REQUEST["acepto"] == "on"){
    print "<p>Desea recibir información</p>";
  }else{
    print "<p>No desea recibir información</p>";
  }
?>
```

```
<pre>
Array
(
    [acepto] => on
)
</pre>
<p>Desea recibir información</p>
```

Pero si el usuario no marca la casilla, la matriz $_REQUEST no contiene el dato y el programa genera un aviso por hacer referencia a un índice no definido, aunque se haya incluido la estructura if ... else ... :

```
<form action="ejemplo.php" method="post">
  <p>Deseo recibir información:
    <input type="checkbox" name="acepto"> (sin marcar la casilla)
  </p>
  <p><input type="submit" value="Enviar"></p>
</form>
```

```
<?php
  print "<pre>";
  print_r($_REQUEST);
  print "</pre>";
  if ($_REQUEST["acepto"] == "on"){
    print "<p>Desea recibir información</p>";
  }else{
    print "<p>No desea recibir información</p>";
  }
?>
```

```
<pre>
Array
(
)
</pre>

Notice: Undefined index: acepto in ejemplo.php on line 4
```

```
<p>No desea recibir información</p>
```

Esto puede ocurrir en realidad en cualquier formulario (incluso en formularios con cajas de texto que en principio envían siempre el control) ya que el usuario puede escribir directamente la dirección del programa PHP en el navegador y ejecutar el programa PHP sin pasar por el formulario. En ese caso, el programa no recibe ningún control y el programa genera un aviso por hacer referencia a un índice no definido:

```
<form action="ejemplo.php" method="post">
          <p>Nombre:     <input     type="text"     name="nombre"
value="Antonio"></p>
  <p><input type="submit" value="Enviar"></p>
</form>
```

```
<?php
  print "<pre>";
  print_r($_REQUEST);
  print "</pre>";
  if ($_REQUEST["nombre"] == ""){
    print "<p>No ha escrito ningún nombre</p>";
  }else{
    print "<p>Su nombre es $_REQUEST[nombre]</p>";
  }
?>
```

```
<pre>
Array
(
)
</pre>
Notice: Undefined index: nombre in ejemplo.php on line 4

<p>No ha escrito ningún nombre</p>
```

Estos problemas se pueden resolver comprobando que el índice está definido antes de hacer referencia a él, utilizando la función **isset($variable)**, que admite como argumento una variable y devuelve **true** si existe y **false** si no existe.

```
<form action="ejemplo.php" method="post">
  <p>Deseo recibir información:
    <input type="checkbox" name="acepto"> (marcada la casilla)
  </p>
  <p><input type="submit" value="Enviar"></p>
</form>
```

```
<?php
  if (isset($_REQUEST["acepto"])){
    print "<p>Desea recibir información</p>";
  }else{
    print "<p>No desea recibir información</p>";
  }
?>
```

```
<p>Desea recibir información</p>
```

```
<form action="ejemplo.php" method="post">
  <p>Deseo recibir información:
    <input type="checkbox" name="acepto"> (sin marcar la casilla)
  </p>
  <p><input type="submit" value="Enviar"></p>
</form>
```

```
<?php
  if (isset($_REQUEST["acepto"])){
    print "<p>Desea recibir información</p>";
  }else{
    print "<p>No desea recibir información</p>";
  }
?>
```

```
<p>No desea recibir información</p>
```

Es conveniente efectuar siempre la verificación de existencia para prevenir los casos en que un usuario intente acceder a la página PHP sin pasar por el formulario.

Seguridad en las entradas

Un usuario puede insertar código html en la entrada de un formulario, lo que puede acarrear comportamientos inesperados y riesgos de seguridad. El siguiente ejemplo solamente perjudica al aspecto de la página:

```
<form action="ejemplo.php" method="post">
  <p>Nombre: <input type="text" name="nombre"
value="<strong>Antonio</strong>"></p>
  <p><input type="submit" value="Enviar"></p>
</form>
```

```
<?php
```

```
  print "<pre>";
  print_r($_REQUEST);
  print "</pre>";
  if ($_REQUEST["nombre"] == ""){
    print "<p>No ha escrito ningún nombre</p>";
  }else{
    print "<p>Su nombre es $_REQUEST[nombre]</p>";
  }
?>
```

```
<pre>
Array
(
    [nombre] => <strong>Antonio</strong>
)
</pre>
<p>Su nombre es <strong>Antonio</strong></p>
```

Eliminar etiquetas: strip_tags($cadena)

Una manera de resolver el problema anterior es utilizar la función **strip_tags($cadena)**, que devuelve la cadena sin etiquetas, como muestra el siguiente ejemplo.

```
<?php
  print "<pre>";
  print_r($_REQUEST);
  print "</pre>";
  if ($_REQUEST["nombre"] == ""){
    print "<p>No ha escrito ningún nombre</p>";
  }else{
    print "<p>Su nombre es $_REQUEST[nombre]</p>";
   print "<p>Su nombre es " . strip_tags($_REQUEST["nombre"]).
"</p>";
  }
?>
```

```
<pre>
Array
(
    [nombre] => <strong>Antonio</strong>
)
</pre>
<p>Su nombre es Antonio</p>
<p>Su nombre es <strong>Antonio</strong></p>
```

Aunque hay que tener en cuenta que esta función elimina cualquier cosa que se interprete como una etiqueta, es decir, que empiece por "<".

```
<form action="ejemplo.php" method="post">
  <p>Nombre: <input type="text" name="nombre" value="<pepe>"></p>
  <p><input type="submit" value="Enviar"></p>
</form>
```

```
<?php
  print "<pre>";
  print_r($_REQUEST);
  print "</pre>";
  if ($_REQUEST["nombre"] == ""){
    print "<p>No ha escrito ningún nombre</p>";
  }else{
    print "<p>Su nombre es " . strip_tags($_REQUEST["nombre"]).
"</p>";
  }
?>
```

```
<pre>
Array
(
    [nombre] => <pepe>
)
</pre>
<p>Su nombre es </p>
```

Eliminar espacios en blanco iniciales y finales: trim($cadena)

A veces, al rellenar un formulario, los usuarios escriben por error espacios en blanco al principio o al final de las cajas de texto y si el programa PHP no tiene en cuenta esa posibilidad se pueden obtener resultados inesperados.

En el programa de ejemplo del apartado anterior en el que se escribía una estructura if ... else ... para avisar al usuario si no escribía el nombre, si el usuario escribe únicamente espacios en blanco, el programa funciona, pero no da la respuesta adecuada ya que la cadena con espacios en blanco no es una cadena vacía.

```
<form action="ejemplo.php" method="post">
  <p>Nombre: <input type="text" name="nombre" value="    ">
  <p><input type="submit" value="Enviar"></p>
</form>
```

```
<?php
```

```
  print "<pre>";
  print_r($_REQUEST);
  print "</pre>";
  if ($_REQUEST["nombre"] == ""){
    print "<p>No ha escrito ningún nombre</p>";
  }else{
    print "<p>Su nombre es $_REQUEST[nombre]</p>";
  }
?>
```

```
<pre>
Array
(
    [nombre] =>
)
</pre>
<p>Su nombre es </p>
```

Este problema se puede resolver utilizando la función **trim($cadena)**, que elimina los espacios en blanco iniciales y finales y devuelve la cadena sin esos espacios. Modificando el ejemplo anterior, la cadena introducida queda reducida a la cadena vacía y la comprobación la detecta:

```
<form action="ejemplo.php" method="post">
  <p>Nombre: <input type="text" name="nombre" value="   ">
  <p><input type="submit" value="Enviar"></p>
</form>
```

```
<?php
  print "<pre>";
  print_r($_REQUEST);
  print "</pre>";
  if (trim($_REQUEST["nombre"]) == ""){
    print "<p>No ha escrito ningún nombre</p>";
  }else{
    print "<p>Su nombre es $_REQUEST[nombre]</p>";
  }
?>
```

```
<pre>
Array
(
    [nombre] =>
)
```

```
</pre>
<p>No ha escrito ningún nombre</p>
```

Utilización de variables

Hemos visto que para resolver los problemas comentados en los apartados anteriores, al incluir cualquier referencia a $_REQUEST[control] habría que:

- comprobar que el control esté definido con la función **isset()**
- eliminar las etiquetas con la función **strip_tags()**
- eliminar los espacios en blanco iniciales y finales con la función **trim()**

Una manera de hacerlo sin complicar excesivamente el programa es guardar los valores de la matriz $_REQUEST en variables y realizar todas las comprobaciones al definir esas variables. En el resto del código basta con utilizar la variable en vez del elemento de la matriz $_REQUEST. Se puede ver cómo el programa no genera avisos y muestra el mensaje correspondiente al dato recibido:

```
<form action="ejemplo.php" method="post">
  <p>Nombre: <input type="text" name="nombre" value="">
  <p><input type="submit" value="Enviar"></p>
</form>
```

```
<?php
  print "<pre>";
  print_r($_REQUEST);
  print "</pre>";
  if (isset($_REQUEST["nombre"])){
    $nombre = trim(strip_tags($_REQUEST["nombre"]));
  }else{
    $nombre = "";
  }

  if ($nombre == ""){
    print "<p>No ha escrito ningún nombre</p>";
  }else{
    print "<p>Su nombre es$nombre</p>";
  }
?>
```

```
<pre>
Array
(
    [nombre] =>
)
</pre>
<p>No ha escrito ningún nombre</p>
```

```
<form action="ejemplo.php" method="post">
  <p>Nombre: <input type="text" name="nombre"
value="<strong>Antonio</strong>"></p>
  <p><input type="submit" value="Enviar"></p>
</form>
```

```
<?php
  print "<pre>";
  print_r($_REQUEST);
  print "</pre>";
  if (isset($_REQUEST["nombre"])){
    $nombre = trim(strip_tags($_REQUEST["nombre"]));
  }else{
    $nombre = "";
  }

  if ($nombre == ""){
    print "<p>No ha escrito ningún nombre</p>";
  }else{
    print "<p>Su nombre es $nombre</p>";
  }
?>
```

```
<pre>
Array
(
    [nombre] =>  <strong>Antonio</strong>
)
</pre>
<p>Su nombre es Antonio</p>
```

La asignación de la variable se puede realizar en una sola instrucción, utilizando la notación abreviada: (condición) ? verdadero : falso;

```
<form action="ejemplo.php" method="post">
  <p>Nombre: <input type="text" name="nombre"
value="<strong>Antonio</strong>"></p>
  <p><input type="submit" value="Enviar"></p>
</form>
```

```
<?php
  print "<pre>";
  print_r($_REQUEST);
  print "</pre>";
```

```
  $nombre = isset($_REQUEST["nombre"]) ?
trim(strip_tags($_REQUEST["nombre"])): "";

  if ($nombre == ""){
    print "<p>No ha escrito ningún nombre</p>";
  }else{
    print "<p>Su nombre es $nombre</p>";
  }
?>
```

```
<pre>
Array
(
    [nombre] =>  <strong>Antonio</strong>
)
</pre>
<p>Su nombre es Antonio</p>
```

htmlspecialchars()

Ciertos caracteres tienen un significado especial en HTML y deben ser representados por entidades HTML si se desea preservar su significado.

La función **htmlspecialchars()** de PHP realiza la siguiente sustitución de caracteres por su correspondiente entidad HTML:

Carácter	Sustitución
&	&amp;
" (comilla doble)	&quot;
‘ (comilla simple)	&#039;
< (menor que)	&lt;
> (mayor que)	&gt;

La función **htmlspecialchars()** además de la cadena a convertir admite otros parámetros entre el que hay que destacar el segundo que es un entero ($flags). Con este segundo parámetros se especifica cómo manejar las comillas, las secuencias de unidad de código inválidas y el tipo de documento utilizado. Por defecto el valor que tiene es (ENT_COMPAT | ENT_HTML401) que convierte las comillas dobles y deja solo las comillas simples y maneja el código como HTML 4.01).

```
<form action="ejemplo.php" method="post">
  <p>Nombre: <input type="text" name="nombre"
value='"Antonio"'></p>
  <p><input type="submit" value="Enviar"></p>
</form>
```

```
<?php
  print "<pre>";
  print_r($_REQUEST);
  print "</pre>";
  if (isset($_REQUEST["nombre"])){
    $nombre =
htmlspecialchars(trim(strip_tags($_REQUEST["nombre"])));
  }else{
    $nombre = "";
  }

  if ($nombre == ""){
    print "<p>No ha escrito ningún nombre</p>";
  }else{
    print "<p>Su nombre es $nombre</p>";
  }
?>
```

```
<pre>
Array
(
    [nombre] => "Antonio"
)
</pre>
<p>Su nombre es &amp;quot;Antonio&amp;quot;</p>
```

El ejemplo anterior tiene el inconveniente de que la primera función que se aplica es strip_tags() por lo que se eliminará todo lo que esté entre marcas (<>).

```
<form action="ejemplo.php" method="post">
  <p>Nombre: <input type="text" name="nombre"
value="<strong>Antonio</strong>"></p>
  <p><input type="submit" value="Enviar"></p>
</form>
```

```
<?php
  print "<pre>";
  print_r($_REQUEST);
  print "</pre>";
```

```
  if (isset($_REQUEST["nombre"])){
    $nombre =
htmlspecialchars(trim(strip_tags($_REQUEST["nombre"])));
  }else{
    $nombre = "";
  }

  if ($nombre == ""){
    print "<p>No ha escrito ningún nombre</p>";
  }else{
    print "<p>Su nombre es $nombre</p>";
  }
?>
```

```
<pre>
Array
(
    [nombre] => <strong>Antonio</strong>
)
</pre>
<p>Su nombre es Antonio</p>
```

Si no queremos quitar etiquetas, pero no queremos que se consideren como etiquetas sino como texto, habría que aplicar primero la función htmlspecialchars(), como en el ejemplo siguiente.

```
<form action="ejemplo.php" method="post">
  <p>Nombre: <input type="text" name="nombre"
value="<strong>Antonio</strong>"></p>
  <p><input type="submit" value="Enviar"></p>
</form>
```

```
<?php
  print "<pre>";
  print_r($_REQUEST);
  print "</pre>";
  if (isset($_REQUEST["nombre"])){
    $nombre =
strip_tags(trim(htmlspecialchars($_REQUEST["nombre"])));
  }else{
    $nombre = "";
  }

  if ($nombre == ""){
    print "<p>No ha escrito ningún nombre</p>";
```

```
  }else{
    print "<p>Su nombre es $nombre</p>";
  }
?>
```

```
<pre>
Array
(
    [nombre] => <strong>Antonio</strong>
)
</pre>
<p>Su nombre es <strong>Antonio</strong></p>
```

En el ejemplo anterior la función **strip_tags()** no hace nada puesto que las marcas < y > se han cambiado antes por las entidades de carácter < y > así que se podría simplificar el programa:

```
<form action="ejemplo.php" method="post">
  <p>Nombre: <input type="text" name="nombre"
value="<strong>Antonio</strong>"></p>
  <p><input type="submit" value="Enviar"></p>
</form>
```

```
<?php
  print "<pre>";
  print_r($_REQUEST);
  print "</pre>";
  if (isset($_REQUEST["nombre"])){
    $nombre = trim(htmlspecialchars($_REQUEST["nombre"]));
  }else{
    $nombre = "";
  }

  if ($nombre == ""){
    print "<p>No ha escrito ningún nombre</p>";
  }else{
    print "<p>Su nombre es $nombre</p>";
  }
?>
```

```
<pre>
Array
(
    [nombre] => <strong>Antonio</strong>
```

```
)
</pre>
<p>Su nombre es <strong>Antonio</strong></p>
```

NOTA: Existe otra función de PHP **htmlentities()** muy parecida a **htmlspecialchars()**. A diferencia de htmlspecialchars, htmlentities convierte no sólo los caracteres especiales de la cadena en entidades HTML, sino todos los caracteres aplicables en entidades HTML.

```
<?php
  print "<h3>htmlspecialchars</h3>";
  print "<p>";
  print htmlspecialchars("Ejemplo <br>");
  print "</p>";
  print "<p>";
  print htmlspecialchars("µ †");
  print "</p>";

  print "<h3>htmlentities</h3>";
  print "<p>";
  print htmlentities("Ejemplo <br>");
  print "</p>";
  print "<p>";
  print htmlentities("µ †");
  print "</p>";
?>
```

La salida HTML es la misma en ambos casos:

```
<h3>htmlspecialchars</h3>
<p>Ejemplo <br></p>
<p>µ †</p>
<h3>htmlentities</h3>
<p>Ejemplo <br></p>
<p>>µ †</p>
```

Pero el código fuente de la página es diferente:

```
<h3>htmlspecialchars</h3>
<p>Ejemplo &lt;br&gt;</p>
<p>µ †</p>
<h3>htmlentities</h3>
<p>Ejemplo &lt;br&gt;</p>
<p>&micro; &dagger;</p>
```

Funciones de recogida de datos

La forma más cómoda de tener en cuenta todos los aspectos comentados en los puntos anteriores es definir una función:

- La función tendrá como argumento el nombre del control que se quiere recibir y devolverá el valor recibido (o una cadena vacía si el control no se ha recibido).
- Para tratar los datos recibidos se aplicarán únicamente las funciones htmlspecialchars() y trim():

```
trim(htmlspecialchars($_REQUEST[$var], ENT_QUOTES, "UTF-8"));
```

De esta manera, si se reciben etiquetas (<>), las etiquetas no se borrarán, sino que se conservarán como texto.
Si se quisieran borrar las etiquetas, habría que aplicar además la función strip_tags():

```
trim(htmlspecialchars(strip_tags($_REQUEST[$var])),   ENT_QUOTES,
"UTF-8");
```

- Una vez definida la función, al comienzo del programa se almacenarán en variables los datos devueltos por la función.
- En el resto del programa se trabaja con las variables.

Comprobación de datos

Antes de utilizar en un programa los datos recibidos a través de un formulario, es necesario comprobar si el dato recibido corresponde a lo esperado. Algunas comprobaciones se pueden hacer con comparaciones (si no es vacío, si es un valor comprendido entre unos valores determinado, etc.), pero antes de hacer esas comparaciones hay que comprobar que es del tipo esperado (número entero o decimal, texto, etc.) para procesarlo sin error.

En esta lección se comentan una serie de familias de funciones que permiten comprobar la existencia, el tipo o el contenido de un dato:

- funciones **is_**
- funciones **ctype_**
- funciones **filter_**
- funciones **_exists**

Pero antes se comenta un caso especial, la comprobación de números, para la que se aconseja utilizar las funciones **is_numeric()** y **ctype_digit()**.

Comprobación de números con is_numeric() y ctype_digit()

Cuando se quiere comprobar que un dato recibido a través de un formulario es un número, en el caso de las funciones **is_** o **ctype_**, sólo tiene sentido utilizar dos de ellas.

Comprobación de números con is_numeric()

Para comprobar si el dato que se ha recibido es un número (o se puede interpretar como número) se debe utilizar la función **is_numeric($valor)**. Esta función lógica devuelve true si el argumento se puede interpretar como número y false si no lo es.

El motivo por el que se debe utilizar la función **is_numeric($valor)**, es que lo que se recibe a través de un formulario se guarda siempre como cadena (aunque sea un número). La función **is_numeric($valor**) es la única que comprueba si el argumento se puede interpretar como número (aunque el argumento sea de tipo cadena), la función i**s_int($valor)** o **is_float($valor)** hacen solamente comprobaciones sobre el tipo de los datos y devolverán siempre **false**.

Comprobación de números enteros positivos con ctype_digit()

Para comprobar si el dato que se ha recibido es un número entero positivo (sin decimales) se puede utilizar la función **ctype_digit($valor)**. Esta función lógica devuelve true si todos los caracteres del argumento son dígitos (0, 1, ..., 9) y false si no lo son.

Funciones is_

Las funciones **is_** son un conjunto de funciones booleanas que devuelven **true** si el argumento es de un tipo de datos determinado y **false** si no lo son.

	Función	**Tipo de datos**	**alias (funciones equivalentes)**
existencia	**isset($valor)**	devuelve si el dato está definido o no	
	is_null($valor)	null	
números	**is_bool($valor)**	booleano	
	is_numeric($valor)	número (puede tener signo, parte decimal y estar expresado en notación decimal, exponencial o hexadecimal).	
	is_int($valor)	entero	**is_integer($valor), is_long($valor)**
	is_float($valor)	float	**is_double($valor), is_real($valor)**
cadenas	**is_string($valor)**	cadena	

otros	**is_scalar($valor)**	escalar (entero, float, cadena o booleano)	
	is_array($valor)	matriz	
	is_callable($valor)	función	
	is_object($valor)	object	
	is_resource($valor)	recurso	
	is_countable($valor)	contable (matriz u objeto que implementa Countable)	

Si un dato es demasiado grande para el tipo de variable, las funciones devolverán **false**. Por ejemplo, si se usa como argumento un entero mayor que **PHP_INT_MAX** (2147483647 en Windows), la función **is_int()** devolverá false.

Nota: El alias **is_real()** está declarado obsoleto (*deprecated*) desde PHP 7.4 (noviembre 2019).

Estas funciones son en general de poca utilidad con datos provenientes de un formulario, ya que estas funciones lo que comprueban es el tipo de los datos y la información que llega de un formulario es siempre del tipo cadena. Las dos excepciones serían:

- la función **is_numeric(),** que evalúa si el argumento se puede interpretar como número (aunque el tipo sea cadena). Por tanto esta función se puede utilizar para comprobar si un dato recibido es un número.
- la función **isset()**, que evalúa si el argumento está o no definido, independientemente de su tipo.

El ejemplo siguiente muestra el uso de la función **is_numeric()**.

```
<form action="ejemplo.php" method="post">
  <p>Número: <input type="text" name="numero" value=""></p>
  <p><input type="submit" value="Enviar"></p>
</form>
```

```
<?php
  print "<pre>";
  print_r($_REQUEST);
  print "</pre>";
  $numero = $_REQUEST["numero"];
```

```
  if (is_numeric($numero)){
    print "<p>Ha escrito un número: $numero</p>";
  }else{
    print "<p>No ha escrito un número: $numero</p>";
  }
?>
```

Funciones ctype_

Las funciones **ctype_** son un conjunto de funciones booleanas que devuelven si todos los caracteres de una cadena son de un tipo determinado, de acuerdo con el juego de caracteres local. Estas funciones son las mismas que las que proporciona la biblioteca estándar de C ctype.h.

Función	Tipo de datos
ctype_alnum($valor)	alfanuméricos
ctype_alpha($valor)	alfabéticos (mayúsculas o minúsculas, con acentos, ñ, ç, etc)
ctype_cntrl($valor)	caracteres de control (salto de línea, tabulador, etc)
ctype_digit($valor)	dígitos
ctype_graph($valor)	caracteres imprimibles (excepto espacios)
ctype_lower($valor)	minúsculas
ctype_print($valor)	caracteres imprimibles
ctype_punct($valor)	signos de puntuación (caracteres imprimibles que no son alfanuméricos ni espacios en blanco)
ctype_space($valor)	espacios en blanco (espacios, tabuladores, saltos de línea, etc)
ctype_upper($valor)	mayúsculas
ctype_xdigit($valor)	dígitos hexadecimales

Estas funciones se pueden aplicar a los datos recibidos de un formulario ya que hacen comprobaciones de todos los caracteres de una cadena. Quizás la más útil es la función **ctype_digit()**, que evalúa si todos los caracteres son dígitos, por lo que permite identificar los números enteros positivos (sin punto decimal ni signo negativo)

Funciones filter_

La función filter más simple es la función **filter_var($valor [, $filtro [, $opciones]]**), que devuelve los datos filtrados o false si el filtro falla.

Los filtros predefinidos de validación son los siguientes:

Filtro	Tipo de datos
FILTER_VALIDATE_INT	entero
FILTER_VALIDATE_BOOLEAN	booleano
FILTER_VALIDATE_FLOAT	float
FILTER_VALIDATE_REGEXP	expresión regular
FILTER_VALIDATE_DOMAIN	dominio web
FILTER_VALIDATE_URL	URL
FILTER_VALIDATE_EMAIL	dirección de correo
FILTER_VALIDATE_IP	dirección IP
FILTER_VALIDATE_MAC	dirección MAC física

NOTA: El problema de los filtros FILTER_VALIDATE_INT y FILTER_VALIDATE_FLOAT es que dan false si el argumento es 0, lo que complica su uso para detectar números porque el caso del 0 debe considerarse aparte al hacer la validación.

El ejemplo siguiente muestra el uso de la función **filter_var()** con el argumento **FILTER_VALIDATE_INT**.

```
<form action="ejemplo.php" method="post">
  <p>Número: <input type="text" name="numero" value=""></p>
  <p><input type="submit" value="Enviar"></p>
</form>
```

```
<?php
  print "<pre>";
  print_r($_REQUEST);
  print "</pre>";
  $numero = $_REQUEST["numero"];

  if (filter_var($numero, FILTER_VALIDATE_INT)){
```

```
    print "<p>Ha escrito un número entero: $numero</p>";
  }else{
    print "<p>No ha escrito un número entero: $numero</p>";
  }
?>
```

Funciones xxx_exists()

Función function_exists()

La función booleana function_exists() devuelve si la función existe o no.

Función array_key_exists()

La función booleana **array_key_exists($indice, $matriz)** devuelve si un elemento determinado de una matriz existe o no.
Para comprobar si existe un elemento de una matriz también se puede utilizar la función **isset()** comentada anteriormente.