

# Uso de formularios para leer datos del cliente

De WikiEducator

< Usuario:ManuelRomero

Formularios: Pasando información del cliente al servidor

## ¡Los formularios como entrada de datos a nuestros script

**PHP** Un lenguaje de script al lado del servidor



**Formularios** | Ejercicios | Práctica | Volver

### INDICE

## Contenido

- 1 Introducción
- 2 Cómo leer datos de usuario
- 3 Creando un formulario
- 4 Atributos de la etiqueta
  - 4.1 GET o POST
  - 4.2 Elementos dentro del formulario
- 5 Creando formularios
- 6 Obtener datos de un formulario
- 7 Redirigiendo páginas
- 8 Referenciando la propia página
- 9 Pasando información de una página a otra

- 10 Transfiriendo ficheros entre cliente y servidor
  - 10.1 Acciones en el Cliente
  - 10.2 Acciones en el Servidor: \$\_FILES
    - 10.2.1 Copiando el fichero a una carpeta
    - 10.2.2 Comprobando errores
    - 10.2.3 Ver tamaño del fichero y otras directivas en php.ini
    - 10.2.4 Tipo de fichero
- 11 Trabajar con directorios



Show presentation

## Introducción

- Tratamos de ver las instrucción que permitan aportar valores al programa.
- Todos los lenguajes de programación tienen primitivas o incluso instrucciones propias para este cometido
- Un programa necesita **interactuar** con el usuario.

Para ello debemos tener dos tipos de instrucciones como podemos ver en la imagen

1. Leer valores del teclado

## 2. Mostrar resultados en pantalla



- En el caso de PHP, hemos visto alguna primitiva para mostrar valores por pantalla (En realidad lo que hace es escribirlas al fichero html que entrega al cliente).
- Estas instrucciones son ***echo*** y ***print***.
- Ambos dos son instrucciones del lenguaje, y tienen una pequeña diferencia:



### Actividad

Completa el siguiente programa

```
<?php
$n1=1;
$n2=2;

//Usando echo con múltiples parámetros
//Visualiza la suma, la resta, y la multiplicación
//Al ser varios parámetros usa las comas para separar uno de otro
echo "Usando <b>echo</b> <br/>";
echo "el resultado de sumar $n1+$n2 es ",($n1+$n2),
    " la resta ", ($n1-$n2),
    " el producto", ($n1* $n2),
    " y la divisi&oacuten ",($n1/$n2), "<br />";

//????????
//Usando print, solo puedo usar un parámetro, así que tenemos que concatenar (operador .)
//Recupera el valor que retorna print y visualízalo
echo "<br/> usando <b>print</b><br/>";
print "el resultado de sumar $n1+$n2 es ". ($n1+$n2).
    " la resta ". ($n1-$n2).
    " el producto ". ($n1* $n2).
    " y la divisi&oacuten ".($n1/$n2). "<br />";

?>
```

- Vemos el siguiente resultado

Usando **echo**

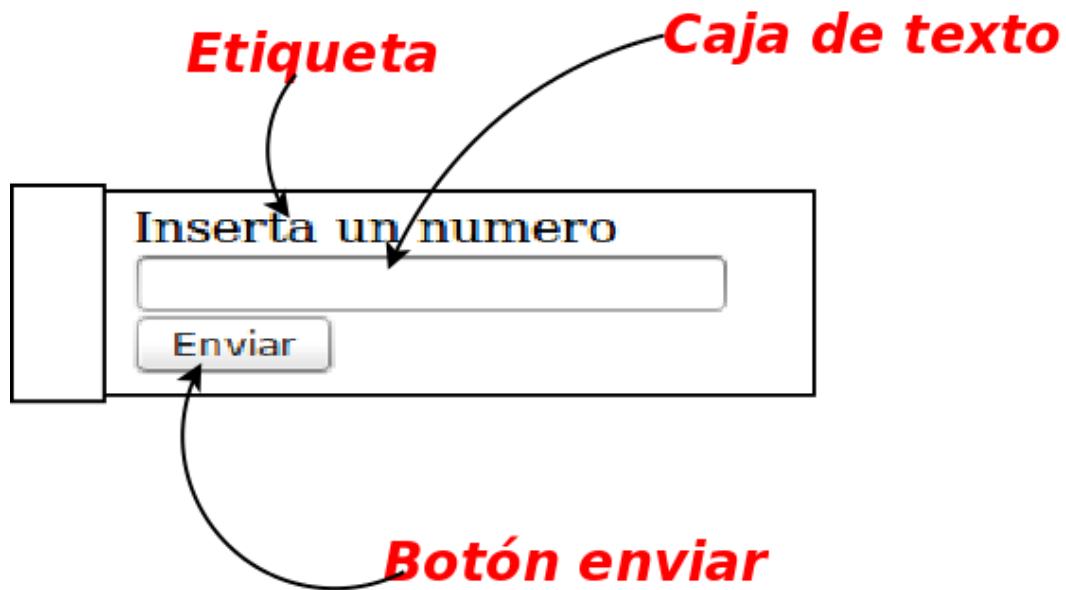
el resultado de sumar 1+2 es 3 la resta -1 el producto 2 y la división 0.5

usando **print**

el resultado de sumar 1+2 es 3 la resta -1 el producto 2 y la división 0.5

## Cómo leer datos de usuario

- Nos falta ver cómo podemos hacer que el cliente (a través del navegador) aporte valores al programa escribiéndolos por el teclado.
- Hay que partir de la situación. Realmente el script se ejecuta en el servidor cuando un navegador solicita una página.
- No puedo detener el programa esperando que el usuario aporte un valor. Esto no es posible en una programación web.
- Pero puedo enviar junto con la solicitud de la página, valores que aporte el usuario, mediante un formulario.
- El formulario será parte de la página del cliente, código **html**.



### Leyendo del usuario

- En el formulario tendremos cajas de texto (**input**) donde, en el navegador, el usuario podrá escribir contenido.
- En los diferentes elementos de entrada de un formulario , como una texto, el usuario podrá escribir valores.
- Al darle el botón enviar (**submit del formulario**), dichos valores irán al servidor para ser leídos y usados en un script; \*Porteriormetne veremos como leerlo en el servidor.
- Repasemos cómo crear formularios en el cliente y lo que más nos interesa, como leerlos en el servidor.

## Creando un formulario

- Esta parte la veis en el módulo de diseño de interfaces , no obstante comentaremos lo que aquí vamos a utilizar.
- Un formulario se establece con la etiqueta **form**.

### Etiqueta form



### Formularios vistos desde el servidor

Para la programación servidor, entendemos por formulario una sección del código html que va a poder contener,

además de otros elementos varios objetos gráficos con los que el usuario va a poder interactuar e insertar valores para que éstos lleguen al servidor

## Atributos de la etiqueta

- Etiqueta **form** con una serie de atributos, de los que **ahora** nos interesan dos principalmente:
  1. action especifica el fichero que se invocará al servidor. Este fichero contendrá el código php que queremos que se ejecute.
  2. method especifica el modo en el que se van a pasar los parámetros (valores introducidos a los diferentes objetos del formulario, o que tengan asignados por defecto).

## GET o POST

```
<form action="mifichero.php" method="POST"
.....
</form>
```

- Por defecto los valores son pasados por GET
- Este método es fácil de ver pues se visualiza en el URL, apareciendo como parte de él separado por el signo interrogación con parejas **variable=valor**.



### Ejemplo

#### Método GET

- HTML en el cliente:
- Vemos dos *input* de type text y con dos atributos asignados: **name** y **value**:
  1. **name** es el que vamos a utilizar para recuperar el contenido del **input** en el servidor.



**Tip:** El **name** es al servidor lo mismo que el **id** es al cliente, con id podéis acceder a los valores de los elementos con javascript, con el *name* lo haremos en php

1. **value** es el valor. Este valor se sustituye por el contenido del **input** del formulario.

```
<form action="mifichero.php" method="GET">
  Nombre
  <input type="text" name = 'nombre' value='maría'>
  Apellido
  <input type="text" name = 'apellido' value='Ruiz'>
  <br />
  <input type="submit" value="enviar">
</form>
```

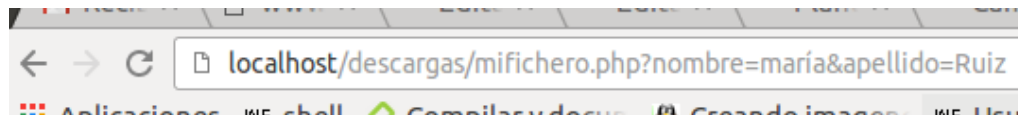
**Al visualizar en el navegador aparecerá la página siguiente**

Dos cajas de texto y el botón submit



**Al presionar submit si observamos el url va el signo ? y luego las parejas *variable=valor* separadas entre ellas por &.**

- Observamos como aparece el texto después de la url:



## Atributos

- En este caso estamos indicando que cuando se envíe el formulario, se intentará ejecutar un fichero llamado **mifichero.php**.

- La ubicación del fichero, como no se especifica, se busca en la misma ubicación donde está el fichero que actualmente está viendo el cliente.
- También se especifica que los valores enviados con el formulario; irán en el cuerpo de documento usando el protocolo http, y no en el URI con el signo ? como sucede si se especificara GET.



## Puntos clave

Tan inseguro es usar **GET** como **POST**. si queremos garantizar seguridad debemos usar **https** en lugar de **http**

- Por supuesto hay más atributos, el id es importante para poder acceder a ese elemento con javascript.
- Tanto **method** como **action** son necesarios para la programación web.
- Hay mas atributos, es importante el atributo **enctype** que permite usar algún tipo de cifrado para enmascarar la información que se envía, y poder especificar también si en el formulario se van a enviar grandes cantidades de bytes, como imágenes u otro tipo de ficheros.



**Tip:** Usar https si se quiere confidencialidad con un nivel aceptable de seguridad.

- Este atributo es importante cuando en lugar de input de tipo texto **<input type=text>** enviemos ficheros **<input type=file>** u otros contenidos diferentes.

## Elementos dentro del formulario

- Dentro del formulario debemos poder recoger información que el cliente nos facilite.
- Al menos deberíamos de conocer dos elementos **input** y **button** o bien **submit**.
- El input representa una caja de texto.
- El submit es un botón que tiene automatizada la acción de enviar el formulario al hacer click sobre él.

## Creando formularios

Es interesante ojear esta sencilla página que te informa de cómo hacer formularios.



[http://www.aulacltic.es/html/t\\_8\\_1.htm](http://www.aulacltic.es/html/t_8_1.htm)

## Elemento *input*

- Como ya hemos comentado, es un elemento de entrada de texto que se rellena en la página web que tiene el cliente, y se envía al servidor donde se puede recuperar esta información para el script a ejecutar (se recuperará durante la ejecución).

## Atributos importantes de un `<input type=text>`

### type

Indicaremos el tipo de elemento de entrada (text, password, email, checkbox...).

Aquí podemos ver una lista de posibles valores, tened en cuenta que con html5 se introdujeron 13 nuevos tipos.

[http://www.w3schools.com/tags/att\\_input\\_type.asp](http://www.w3schools.com/tags/att_input_type.asp)  
...)

### atributo `type=hidden`.

También es interesante el valor *hidden* para el **type** de un *input* (especialmente usado para pasar valores del cliente al servidor de forma transparente para el usuario).

### name

El valor de este atributo especifica el nombre asociado a este *input*.

Es este valor el que necesitamos para recuperar la información del *input* en el servidor.

### value

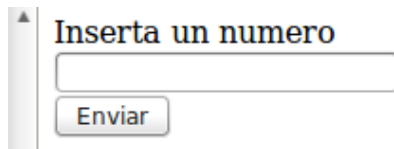
Es el valor que tiene el *input*. Si queremos que por defecto tenga un valor.

- Este valor es sustituido por el contenido del *input* cuando se envía al servidor.
- Dentro del *form* necesitaremos al menos un *input* y un **submit**.
- Veamos el siguiente ejemplo. En el cliente tenemos el siguiente formulario

```
<!DOCTYPE html>
<html>
  <head>
    <title>Tabla de multiplacar</title>
  </head>
  <body>
    <form action="tabla.php" method="GET">
      Inserta un numero <br>
      <input type="text" name="numero"/>
      <br/>
      <input TYPE="submit" VALUE="Enviar"/>
    </form>
  </body>
</html>
```

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

- Y obtenemos la siguiente imagen



1. Al presionar el botón de enviar se envía la página al servidor.
2. La página o script la especificamos en el atributo **action** del elemento **form**; la gestiona *tabla.php* en este caso.
3. En el servidor para recuperar el valor utilizaremos la variable **superglobal**(Lo veremos más adelante).
4. Esta superglobal puede ser **\$\_GET** **\$\_POST** o **\$\_REQUEST**.
5. Una tabla es una estructura indexada por índices.
6. Leeremos el índice nombre de variable de esta estructura superglobal.
7. **\$\_GET** o **\$\_POST** dependiendo de el método de envío, o **\$\_REQUEST** sirve para ambas (no recomendado).

Valor numérico introducido: `<?php echo $_GET['numero'] ?>`

## Obtener datos de un formulario

- Una vez que estamos en el servidor, los datos son pasados del cliente al servidor usando las variables superglobales o matrices **\$\_POST** **\$\_GET**, **\$\_REQUEST**.
- Dependerá del modo en el que pasemos los datos del formularios desde el cliente

atributo **method** del **form**

- Para leer el datos indexaremos la matriz por el valor del atributo **name** de **input** correspondiente.
- Por ejemplo en el cliente tenemos

```
<form method=POST action ="resuelve.php">
<input type=text name=nombre>
</form>
```

- En el servidor el servidor el fichero **resuelve.php**

```
.....
$nombre = $_POST['nombre'];
.....
//También podríamos $_REQUEST['nombre'];
```

## Verificando si una variable existe

- Este tema es muy interesante en php.
- Supongamos el siguiente código

```
<?php
<?php
echo "<h3>Probamos la función is_null</h3>";
echo "<hr />";

$a;
echo is_null($a)? "SI. <b>\$a </b>, es nulo <br>\n": "NO <b>\$a</b> no es nulo<br>\n"; //SI
$a=null;
echo is_null($a)? "SI. <b>\$a=null</b>, \$a es nulo <br>\n": "NO <b>\$a=null \$a</b> no es nulo<br>\n"; //SI
$a=5;
echo is_null($a)? "SI. <b>\$a=5</b>, \$a es nulo <br>\n": "NO <b>\$a=5</b> \$a no es nulo<br>\n"; //NO
$a="";
echo is_null($a)? "SI. <b>\$a=\"\"</b>, \$a es nulo <br>\n": "NO <b>\$a=\"\"</b> \$a no es nulo<br>\n"; //NO
$a=false;
echo is_null($a)? "SI. <b>\$a=false</b>, \$a es nulo <br>\n": "NO <b>\$a=false</b> \$a no es nulo<br>\n"; //NO
$a=0;
echo is_null($a)? "SI. <b>\$a=0</b>, \$a es nulo <br>\n": "NO <b>\$a=0</b> \$a no es nulo<br>\n"; //NO
unset($a); //Eliminamos la variable

echo "<h3>Probamos la función isset</h3>";
echo "<hr />";

$a;
echo isset($a)? "SI <b>\$a</b> está definido <br>\n": "NO <b>\$a</b> no está definido<br>\n"; //NO
$a=null;
echo isset($a)? "SI <b>\$a=null</b> \$a está definido<br>\n": "NO <b>\$a=null</b> \$a no está definido<br>\n"; //NO
$a=5;
echo isset($a)? "SI <b>\$a=5</b> \$a está definido<br>\n": "NO <b>\$a=5</b> \$a no está definido<br>\n"; //SI
$a="";
echo isset($a)? "SI <b>\$a=\"\"</b> \$a está definido<br>\n": "NO <b>\$a=\"\"</b> \$a no está definido<br>\n"; //SI
$a=false;
echo isset($a)? "SI <b>\$a=false</b> \$a está definido<br>\n": "NO <b>\$a=false</b> \$a no está definido<br>\n"; //SI
$a=0;
echo isset($a)? "SI <b>\$a=0</b> \$a está definido <br>\n": "NO <b>\$a=0</b> \$a no está definido<br>\n"; //SI
unset($a); //Eliminamos la variable

echo "<h3>Probamos la función empty</h3>";
echo "<hr />";

$a;
echo empty($a)? "SI <b>\$a</b> está vacío <br>\n": "NO \$a</b> no es nulo<br>\n"; //SI
$a=null;
echo empty($a)? "SI <b>\$a=null</b> \$a está vacío<br>\n": "NO <b> \$a=null</b> \$a no está vacío<br>\n"; //SI
```

```
$a=5;  
echo empty($a)? "SI <b>\$a=5</b> \$a está vacío<br>\n": "NO <b>\$a=5</b> \$a no está vacío<br>\n";//NO  
$a="";  
echo empty($a)? "SI <b>\$a=\"\"</b> \$a está vacío<br>\n": "NO <b>\$a=\"\"</b> \$a no está vacío<br>\n";//SI  
$a=false;  
echo empty($a)? "SI <b>\$a=false</b> \$a está vacío<br>\n": "NO <b>\$a=false</b> \$a no está vacío<br>\n";//SI  
$a=0;  
echo empty($a)? "SI <b>\$a=0</b> \$a está vacío<br>\n": "NO <b>\$a=0</b> \$a no está vacío<br>\n";//SI
```

- Podemos observar la siguiente salida

### Probamos la función `is_null`

---

SI `$a` , es nulo  
SI `$a=null`, `$a` es nulo  
NO `$a=5` `$a` no es nulo  
NO `$a=""` `$a` no es nulo  
NO `$a=false` `$a` no es nulo  
NO `$a=0` `$a` no es nulo

### Probamos la función `isset`

---

NO `$a` no está definido  
NO `$a=null` `$a` no está definido  
SI `$a=5` `$a` está definido  
SI `$a=""` `$a` está definido  
SI `$a=false` `$a` está definido  
SI `$a=0` `$a` está definido

### Probamos la función `empty`

---

SI `$a` está vacío  
SI `$a=null` `$a` está vacío  
NO `$a=5` `$a` no está vacío  
SI `$a=""` `$a` está vacío  
SI `$a=false` `$a` está vacío  
SI `$a=0` `$a` está vacío

- Observar que el hecho de que aparezca una variable no implica que la variable exista.

Estas funciones se vuelven a estudiar más adelante, pero debemos observar las siguientes características



## **isset(\$variable) empty(\$variable) is\_null(\$variable)**

`isset($variable)`  
Esta función devuelve **true** si `$variable` existe y no tiene valor **null**

`is_null($variable)`  
Esta función devuelve **true** si `$variable` tiene valor **null** o no existe



**Tip: *isset* y *is\_null* son complementarias.**

`empty($variable)`  
Esta función devuelve **true** si `$variable` tiene un valor vacío



**Tip: *Cuidado con los valores siguientes.***

1. El *string* "",
2. El *entero* **0**
3. El *booleano* **false**

**se consideran valores vacíos en esta función.**

- Es muy importante en muchas ocasiones, ver si una variable tiene o no valor.
- Sirve para ver si el usuario ha insertado o no valor en un campo de texto.
- Sirve para saber si estoy cargando esta página porque he dado un click en el botón submit, o no.
- Para ello usaremos la función ya conocida ***isset(\$variable)***, donde `$variable` es la variable que queremos ver si tiene valor.

</source>

- A continuación vamos a realizar una serie de prácticas con formularios



## Actividad

Haz un formulario en el que insertemos un número y el servidor web nos visualice la tabla de multiplicar

- Comprobaremos previamente que la variable exista y tenga un valor numérico
1. ***isset(\$variable)*** , Para ver que exista la variable y no tiene un valor nul
  2. ***is\_null(\$variable)*** Puedes usarla en lugar de la anterior, recuerda que son complementarias

```
isset($variable) es igual a !is_null($variable)  
is_null($variable) es igual a !isset($variable)
```

1. ***is\_numeric(\$variable)*** me dice si el valor de la variable es numérico

```
$nombre = ""; //nombre tendrá el valor nulo pero es de tipo null  
if ($nombre==null) //Me dará verdad  
.....  
if (is_null($nombre)) //Me dará falso
```

- A continuación vamos a ver como usar y leer datos de un formulario.



## Formulario

Realiza un formulario donde pidamos al usuario datos para confeccionar una ficha

- Nombre
- Apellidos
- Dirección

- Fecha de nacimiento
- Edad
- Idiomas que habla de entre 4 idiomas (Checkbox)
- Si es hombre, mujer o no quiere informar de ello (radio)
- Dirección de correo electrónico.
- Estudios realizados entre ESO, BACHILLER, CICLO FORMATIVO, GRADO UNIVERSITARIO (select)

**Posible solución index.php**

[▼]

**Posible solución: datos.php**

[▼]

## Filtrando valores

- Independientemente de que el se validen/verifiquen valores en el cliente, ***es obligatorio*** verificarlo siempre en el servidor.
- En otro capítulo veremos temas de seguridad en los datos, pero para ver lo peligroso que puede ser, podría ser que el usuario escribiera un script en una caja de texto. Esto en principio no deberíamos de permitirlo.
- miramos el siguiente código sencillo

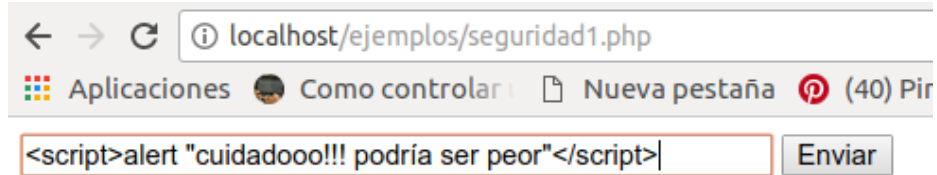
```
<?php
$nombre =$_GET['nombre'];
?>
<!doctype html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport"
    content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Document</title>
</head>
<body>
<form action="seguridad1.php">
  <input type="text" name="nombre" id="">
  <input type="submit" name="enviar" id="">
  <?php echo $nombre ?>
</form>
</body>
</html>
```



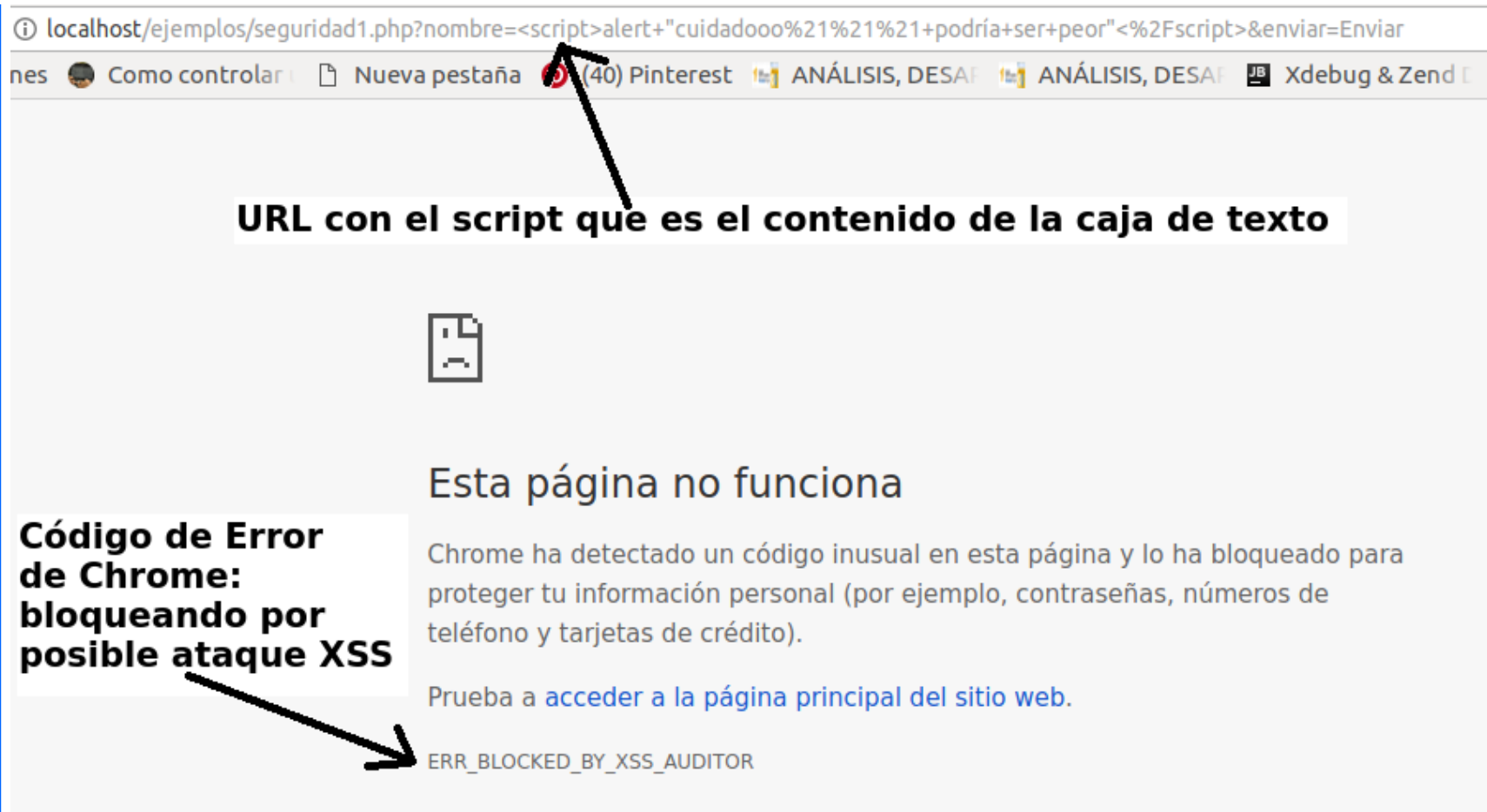
- Ahora observamos lo que ocurre ejecutando esto en chrome y firefox escribiendo en la caja de texto

```
<script>alert "Cudado!!! esto podría ser peor!!!</source>
```

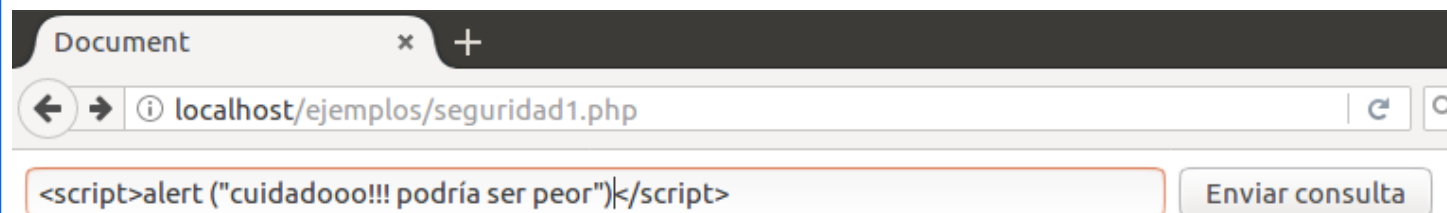
## Escribimos en chrome



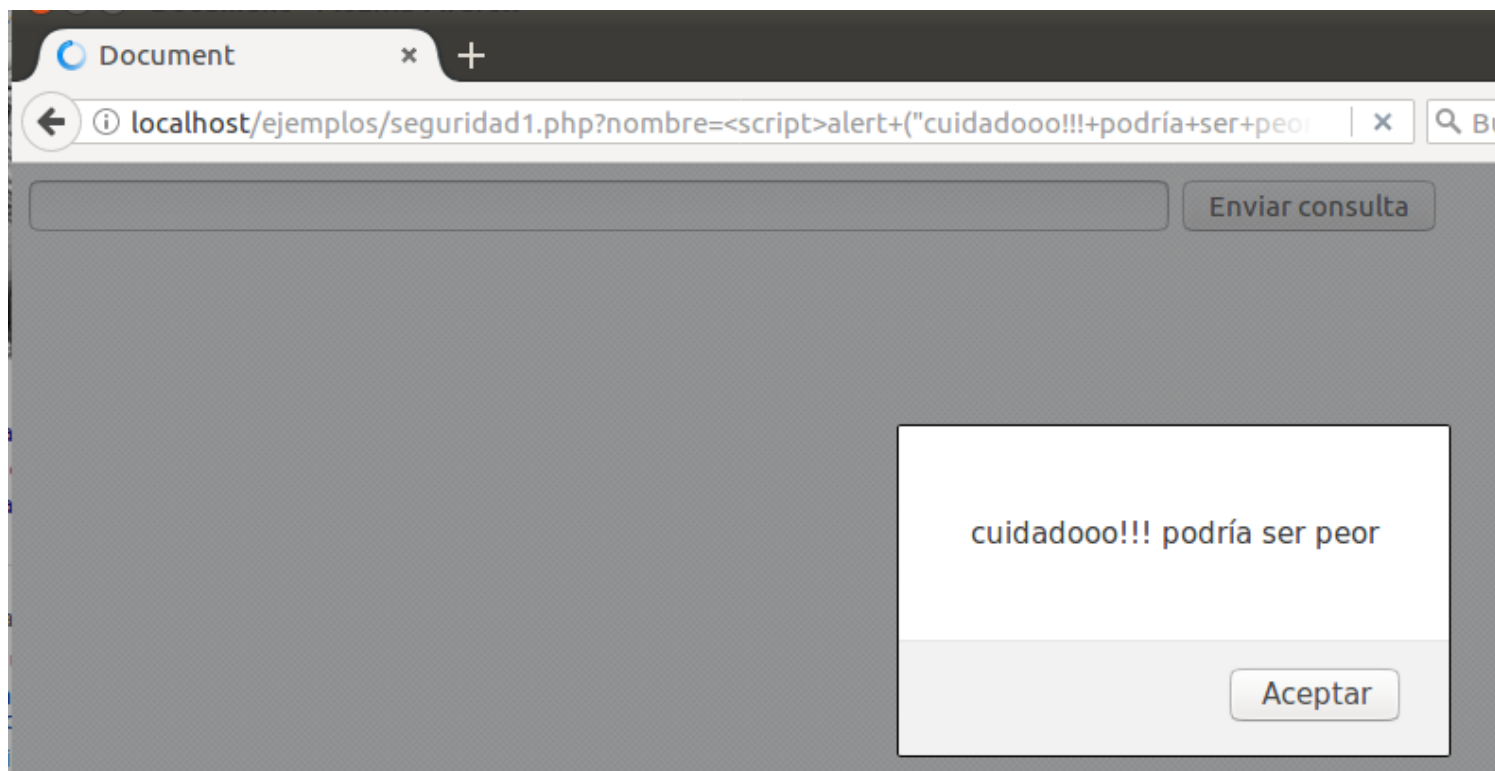
Vemos como *chrome* corta lo que considera un posible ataque xss



**Escribimos el código en firefox**



**Vemos como en firefox sí que se puede ejecutar XSS**



#### Tip:

**Cross Site-Scripting XSS** Ataques que consisten en ejecutar código de script en el cliente.

- El Objetivo es obtener valores de cookies, variables de sesión o redireccionar a otras url
- Hay muchas formas de evitar esto, como [htmlpurifier](http://htmlpurifier.org/) <http://htmlpurifier.org/> que analizaremos cuando veamos seguridad en la web.
- De momento nos limitaremos (que ya es una buena forma de garantizar contenido correcto y evitar problemas) a usar funciones del tipo filter <http://php.net/manual/es/function.filter-var.php>.

Tenemos la opción de `filter_var()` y `filter_input()`.

#### **filter\_var(\$variable, \$filtro)**

1. `$variable` . Es la variable a filtrar. Correspondería al valor del **name** del **input** que queremos recuperar.

2. \$filtro. Es el tipo de filtro que se quiere aplicar. Para ver los tipos de filtros, consultamos a la página web <http://php.net/manual/es/filter.filters.validate.php>.

### **filter\_input(\$tipo\_entrada, \$variable, \$filtro)**

1. \$tipo\_entrada: Uno de los siguientes: INPUT\_GET, INPUT\_POST, INPUT\_COOKIE, INPUT\_SERVER o INPUT\_ENV.
  2. \$variable: como en el caso anterior.
  3. \$filtro: como en el caso anterior.
- Ambas funciones retornan el valor de la variable requerida, o false si el filtro falla o null, si la variable no tenía valor.



### **Actividad**

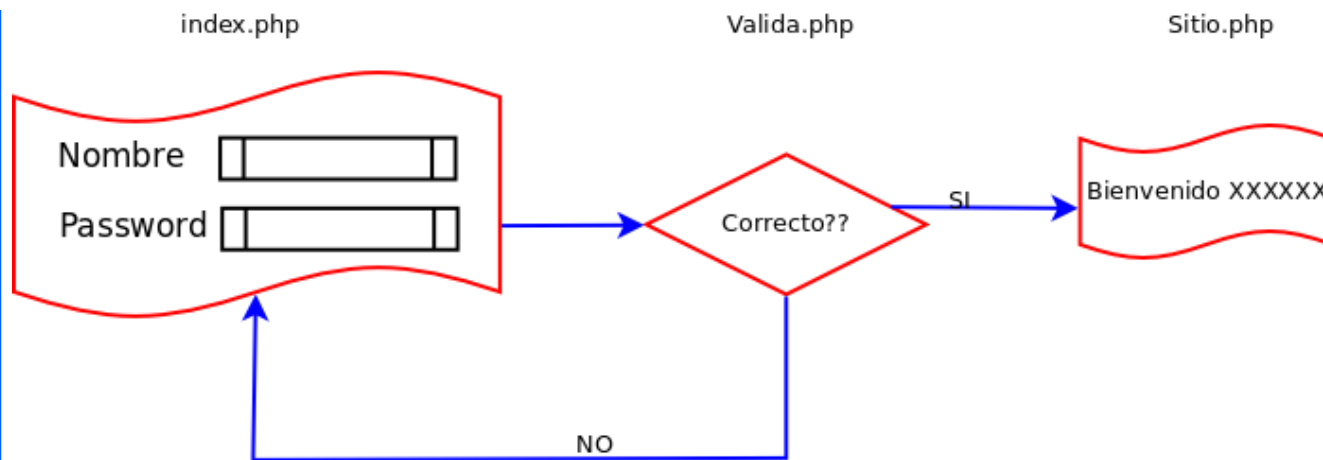
Filtra los valores en un formulario y verifica el tipo de la variable introducido

- Podemos buscar ejercicios de formularios en la lista ya dada en el tema anterior:

<http://www.tecn.upf.es/~ocelma/cpom/practicas/>

## **Redirigiendo páginas**

- Imaginemos que queremos hacer una página donde pidamos al usuario nombre y password.
- El password va a ser 12345. Si el password es correcto iremos a otra página en la que le queremos dar la bienvenida con el nombre que introdujo.
- Pensemos en cómo podemos pasar ese nombre a la página.



- Analicemos las maneras de hacerlo, pero previamente veamos una función muy interesante.
- La usaremos en muchas ocasiones, y sirve para invocar a otras páginas en un momento dado

### **header(...);**

- header() se usa enviar encabezados HTTP sin formato.
- Con esta cabecera http, podemos invocar una determinada **url** que queremos cargar, así que es ahí donde podemos hacer referencia a la página que queremos ver.
- Es muy importante saber que **header()** debe ser llamado antes de mostrar nada por pantalla
- Aquí se puede acceder a la referencia oficial.

```
http://es.php.net/manual/es/function.header.php
```

- Por ahora la usaremos de dos maneras para un mismo cometido

### **Cargar una página inmediatamente**

```
header("Location:URL_de_la_página");
```

### **Cargar una página con un tiempo de demora (por ejemplo para leer un mensaje)**

```
header ("Refresh:5; url=URL_de_la_pagina");
```

- Ahora estamos en condiciones de probarlas con el ejemplo anterior.
- Como siempre recuerdo que es muy importante entender las acciones que se van haciendo.
- Después de ejecutar la función header con redirección, no tiene sentido que se ejecute el resto del script, por lo que es aconsejable y útil poner la función de finalización de script ***exit()***.

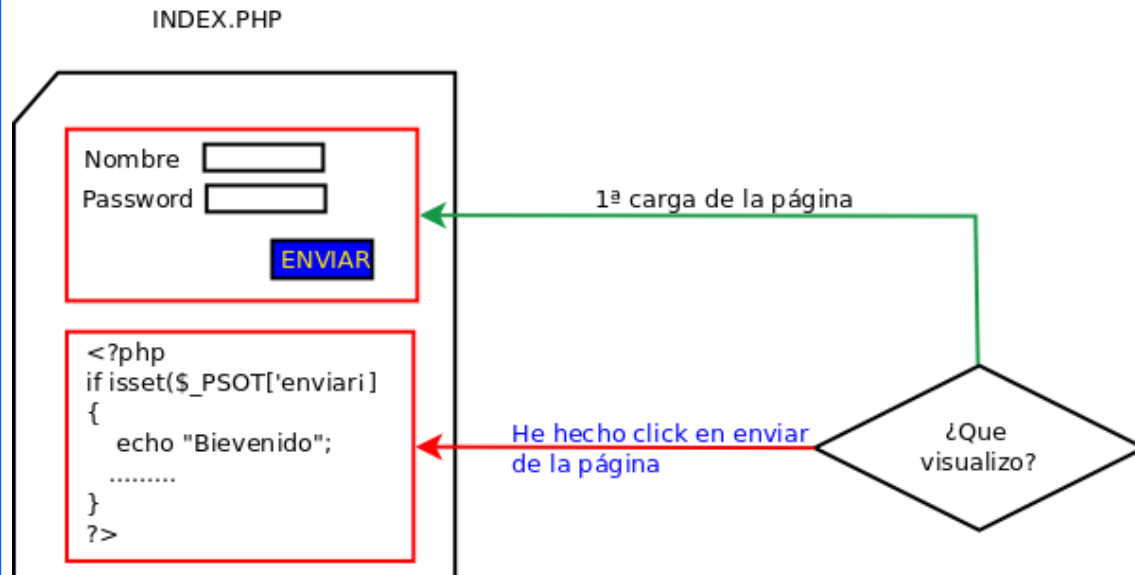
```
header (.....);
exit();
```

## Referenciando la propia página

- A veces puede ser que en la propia página tengamos el código que queremos que se ejecute cuando hacemos un click en el botón submit.
- Esto simplifica el número de páginas que tenemos en nuestro desarrollo.
- En el desarrollo web, hay una tendencia llamada **SPA** *Simple Page Application*

[https://juanda.gitbooks.io/webapps/content/spa/arquitectura\\_de\\_un\\_spa.html](https://juanda.gitbooks.io/webapps/content/spa/arquitectura_de_un_spa.html)

- Esta sería la forma de conseguirlo desde el desarrollo php.
- En este caso tenemos la siguiente situación:



- En la imagen vemos una forma de proceder

- Creamos una página web con un formulario, donde el atributo **action** tiene como valor el nombre de la página actual. Este valor lo podemos escribir explícitamente o bien usar el valor de `$_SERVER['PHP_SELF']` que contiene el nombre del archivo de script ejecutándose actualmente, relativa al directorio raíz de documentos del servidor.

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport"
    content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Document</title>
</head>
<body>
  <form action="."<?PHP echo $_SERVER['PHP_SELF']?> method="POST">
    <input type="submit" value="enviar" name="enviar">
  </form>
</body>
</html>
```

o bien poner el nombre del fichero o script, en este caso suponemos *index.php*.

```
<form action="index.php" method="POST">
```

- En el caso de **index.php** se puede poner un punto, por lo que se podría sustituir la etiqueta del **form**.

```
<form action="." method="POST">
<source>
```

\*Dentro de la ejecución del script, tenemos que ver si es la primera vez que se carga la página o no.

\*Es decir puede ser que esté cagando la página por que he escrito en la url la página o porque he realizado un click sobre el botón submit del formulario correspondiente.

\*Dentro del código, esto lo podemos saber interrogando si existe la variable del formulario que corresponde al submit `''''$_POST['enviar']''''`

```
<source lang=php>
<?php
if isset($_POST['enviar']){
  //En este caso estamos cargando este fichero
  //por que hemos hecho click en el botón submit
}else{
  //Lo que queramos que se ejecute si no hemos hecho click
  //o nada si no queremos contemplar esta situación
}

*/?>
```

```
<!doctype html>
```

.....



## Actividad

Haz una página de bienvenida que muestre los datos de usuario y pass al acceder al sistema

- Se tiene que hacer en una única página

**Posible solución datos\_acceso.php**

[▼]

### Formulario de acceso

← → ↻ ⓘ localhost/ejemplos/datos\_acceso.php

Aplicaciones Como controlar Nueva pestaña

Nombre

Password

### Una vez dado el botón enviar

← → ↻ ⓘ localhost/ejemplos/datos\_acceso.php

Aplicaciones Como controlar Nueva pestaña

**Bienvenido a este sitio web**

**Nombre Manuel Password Romero**



## Pasando información de una página a otra

- Partimos de dos puntos básicos e importantes:



### Puntos clave

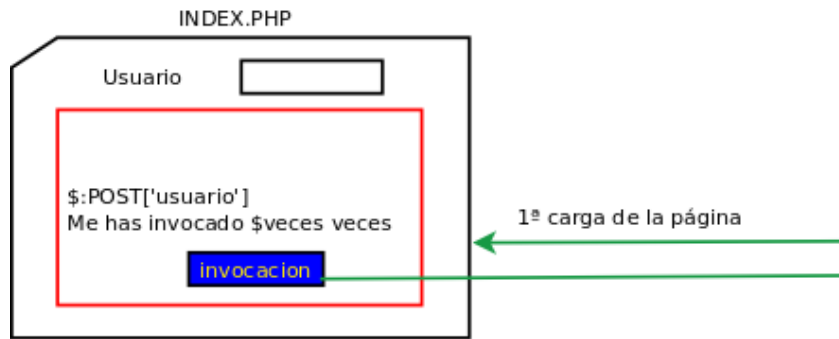
1. La programación web está basada en el protocolo **http**
2. El protocolo **http** es un protocolo sin estado

- Cada vez que cargamos una página web, hay una solicitud ante la cual, el servidor web si se cumplen todas las condiciones, entrega al cliente la página solicitada.
- En caso de que la página ejecutada tenga código php (extensión del fichero), el servidor web ejecuta dicho código y entrega como parte de la página el resultado de la ejecución.
- El servidor no tiene en cuenta a quién entrega la página, no sabe si es la primera vez que te entrega la página o la enésima vez que lo hace.
- Si quiere mantener información entre diferentes páginas, se ha de gestionar por programación.
- Una manera ya le hemos visto usando la función de cabecera **header**

```
header (Location:url?variable1=valor&variable2=valor)
```

### Usando campos ocultos

- Otra manera es usando campos ocultos
- Veamos su funcionamiento
- Hay situaciones donde queremos recopilar además de la información que el usuario rellena, algún dato más.
- Supongamos que queremos saber las veces que una página se invoca a sí mismo
- Cualquiera pensaría en crear una variable, y cada vez que llamemos a la página incrementar en una unidad.



- Cada vez que llamemos a la página siempre que el usuario se haya identificado vamos a especificar las veces que ha invocado a la página.
- Para ello necesitamos enviar a la página del servidor la información de las veces que se ha invocado a la página.
- La idea es que en campo oculto contenga ese valor, el servidor lea este valor, lo incremente y lo vuelva almacenar en el campo oculto.

```
<input type="hidden" name="valorOcultoRescatar" value="$variable">
```



## Actividad

Implementa el programa anterior y verifica su funcionamiento

**Possible solución**



## Transfiriendo ficheros entre cliente y servidor

- Es muy sencillo y frecuente subir ficheros entre cliente y servidor

Subiendo al servidor



- Cuando vamos a subir ficheros hay que conocer acciones a indicar tanto en la parte de cliente como en la de servidor.

## Acciones en el Cliente

### input type=file

- Debemos especificar un elemento **input** con de **type file** en un formulario.
- Como todo input debe tener asignado un **name** para acceder a él en el servidor.

```
<input type=file name=fichero>
```

### form method=POST enctype="multipart/form-data

- El formulario donde esté el **input** ha de tener especificado el atributo **enctype** establecido con el valor **multipart/form-data**.
- Cuando no especificamos valor a este atributo, se asume por defecto el valor **application/x-www-form-urlencoded**.
- Este valor implica que enviamos texto plano y lo podremos enviar tanto por GET como por POST.
- No obstante si vamos a transferir un fichero no necesariamente de texto **debemos** especificarlo estableciendo el valor de **enctype** a **multipart/form-data**.
- Este valor se emplea para transferir gran cantidad de texto u otros formatos de fichero entre cliente y servidor.

## Recursos de la Web



**enctype** es un atributo necesario para especificar el tipo de contenido usado para enviar la información del formulario al servidor.

- Necesariamente hemos de usar el método POST para este cometido.

```
<form action="index.php" method="POST" enctype='multipart/form-data'>
...
</form>
```

## Establecer tamaño en el cliente

- El tamaño de bytes que vamos a enviar también puede quedar establecido en el cliente, de modo que si el fichero tiene un tamaño mayor, no se envía.
- Para esto se establece antes del input file, un input hidden con **name** a MAX\_SIZE\_FILE y **value** el valor del tamaño máximo en bytes.

```
<form action="index.php" method="POST" enctype='multipart/form-data'>
...
<input type="hidden" name="MAX_SIZE_FILE" value="10000000">
<input type="file" name="fichero">
...
</form>
```

- Este mecanismo no envía nada al servidor, dejará de enviar el fichero al servidor .
- En el servidor se recibirá un error de valor **2** o constante UPLOAD\_ERR\_FORM\_SIZE, (Ver código de errores más abajo o en <http://php.net/manual/es/features.file-upload.errors.php>)
- Con todo lo dicho, la especificación en el cliente quedaría

```
<form action="descarga.php" method="POST" enctype="multipart/form-data">
  <input type="hidden" name="MAX_FILE_SIZE" value="10000000">
  <h3>Selecciona fichero </h3>
  <input type="file" name="fichero" id="" >
  <br />
  <input type="submit" value="Acceder" name="descarga">
</form>
```

## Acciones en el Servidor: \$\_FILES

- La forma de acceder al input del tipo file que viene del cliente en la solicitud al servidor es a través de la superglobal **\$\_FILES**.

- Lo primero que deberemos hacer es acceder a este elemento con el nombre del input.

```
$fichero = $_FILES['nombre_input_file']
```

- `$_FILES` es un array asociativo con tantos elementos con input de tipo file como vengan del formulario cuyo submit ha generado una solicitud al servidor.
- Cada posición a su vez contiene un array asociativo con información de ese fichero almacenada en 5 componentes:
  1. **name** Nombre del fichero en el cliente
  2. **type** Tipo de fichero subido
  3. **size** Tamaño en bytes del fichero
  4. **tmp\_name** Nombre asignado de forma temporal en el servidor
  5. **error** Error que se haya podido producir o 0 si no ha habido ninguno (Ver tabla más abajo)

Con el fichero que viene del cliente, en el servidor podemos hacer una serie de acciones:

1. Capturar el fichero y dejarlo en un directorio concreto.
2. Ver si se ha producido algún error especificando el código de error mediante una constante numérica.
3. Ver el tamaño del fichero.
4. Analizar el tipo de fichero para poder, por ejemplo, aceptarlo o descartarlo, o decidir en qué carpeta dejarlo en función del tipo. (Tener en cuenta que haya permisos de escritura para el usuario de apache (www-data) en la carpeta donde queramos dejar el fichero)

## Copiando el fichero a una carpeta

- La primera acción será copiarnos el fichero en una ubicación concreta dentro de nuestro servidor
- Lógicamente primero deberemos crear esa carpeta y asegurarnos que tenga permisos de escritura en ella el usuario apache (normalmente www-data)



### Actividad

```
chmod +w ....
```

- En el servidor tenemos el fichero disponible de forma temporal en la capeta /tmp. Podemos acceder a esta información en el elemento **`$_FILES['tmp_name']`**
- Para copiarlo usaremos la función **`move_uploaded_file($origen, $destino);`**, donde **`$origen`** es el fichero que queremos copiar con ubicación y **`$destino`** es la ubicación y nombre de fichero donde queremos dejarlo.
- Lo más habitual es dejar el fichero con el mismo nombre que tenía en el cliente, esta información la tenemos disponible en el atributo **`$_FILES['name']`**
- La función **`move_upload(..)`** *retorna un booleano que indica el éxito o fracaso de la acción.* <http://php.net/manual/es/function.move-uploaded-file.php>.
- A continuación un resumen de estas acciones

```
//Suponemos en el cliente
//... <input type=file name= fichero>
//
//Accedemos al fichero que está de forma temporal en el servidor
$origen = $_FILES['fichero']['tmp_name'];
//Accedemos al nombre del fichero con el que el cliente lo subió
$nombreFichero = $_FILES['fichero']['name'];
//Establecemos la ruta donde queremos dejar el fichero
//En este caso en la carpeta del proyecto tenemos una carpeta llamada descargas con permiso de escritura para www-data
$destino = "./descargas/".$nombreFichero
//Ahora procedemos a copiar y ver el éxito o fracaso
if (move_uploaded_file($origen, $destino))
    echo ("El fichero $nombreFichero se ha subido correctamente");
else
    echo ("Error subiendo el fichero $nombreFichero");
```



## Actividad

- Vamos a realizar una práctica dónde verifiquemos el funcionamiento de todo lo visto anteriormente.
- Sé curiosa y comprueba los atributos comentados.

- **`$_FILES[error]`** contiene información del error que se ha podido producir al subir el fichero
- La siguiente tabla es la lista de los posibles valores que va a haber en este elemento del array superglobal `$_FILES`

FICHERO 1					FICHERO 2					...	FICHERO N				
name	type	size	tmp_name	error	name	type	size	tmp_name	error		name	type	size	tmp_name	error

### CÓDIGOS DE ERROR SUBIENDO FICHEROS

Valor entero	Constante	Descripción
0	<b><code>UPLOAD_ERR_OK</code></b>	Fichero subido exitosamente
1	<b><code>UPLOAD_ERR_INI_SIZE</code></b>	<i>Tamaño excedido según directiva <b><code>upload_max_filesize</code></b> de <code>php.ini</code>.</i>
2	<b><code>UPLOAD_ERR_FORM_SIZE</code></b>	<i>El fichero subido excede la directiva <code>MAX_FILE_SIZE</code> especificada en el formulario HTML.</i>
3	<b><code>UPLOAD_ERR_PARTIAL</code></b>	<i>El fichero fue sólo parcialmente subido.</i>
4	<b><code>UPLOAD_ERR_NO_FILE</code></b>	<i>No se subió ningún fichero.</i>
6	<b><code>UPLOAD_ERR_NO_TMP_DIR</code></b>	<i>Falta la carpeta temporal.</i>
7	<b><code>UPLOAD_ERR_CANT_WRITE</code></b>	<i>No se pudo escribir el fichero en el disco.</i>
8	<b><code>UPLOAD_ERR_EXTENSION</code></b>	<i>Una extensión de PHP detuvo la subida de ficheros.</i>



### Recursos de la Web

<http://php.net/manual/es/features.file-upload.errors.php>

- Un posible código para obtener esta información

```
//Suponemos en el cliente
//... <input type=file name= fichero>

$fichero = $_FILES['fichero'];
.....
$error = $fichero['error'];
//Esto es igual que hacer $error = $_FILES['fichero']['error']

$error = $_FILES['error'];
switch ($error){
    case 0:
        echo "ERROR. Fichero subido de forma correcta. <br />";
        break;
    case 1:
        echo "ERROR. Tamaño de fichero superior al establecido en el servidor <br />";

        break;
    case 2:
        echo "ERROR. Tamaño de fichero superior al establecido en cliente<br />";
        echo "El tamaño se estableció en el input MAX_FILE_SIZE<br/>";
        echo "Tamaño establecido " . $_POST['MAX_FILE_SIZE'] . "<br/>";
        break;
    case 3:
        echo "ERROR. EL fichero sólo se subió parcialmente <br/>";
        break;
    case 4:
        echo "ERROR. No se subió ningún fichero <br/>";
        break;
    case 6:
        echo "ERROR. No se encuentra la carpeta temporal <br/>";
        break;
    case 7:
        echo "ERROR. No se pudo escribir en disco. revisa permisos <br/>";
        break;
    case 8:
        echo "ERROR. Una extensión de php detuvo la subida del fichero <br/>";
        break;
    default:
        echo "Valor de error desconocido";
}
```

## Ver tamaño del fichero y otras directivas en php.ini

- El tamaño de fichero queda definido en el servidor por la directiva de ***php.ini***

```
upload_max_filesize=
```

- Otras directivas relacionadas con la descargas de ficheros están establecidas en php.ini



- A continuación se detallan con sus valores por defecto. (Ver el fichero php.ini)

```

;;;;;;;;;;;;;
; File Uploads ;
;;;;;;;;;;;;;

; Whether to allow HTTP file uploads.
; http://php.net/file-uploads
;Comentario: Permite la descarga de ficheros
file_uploads = 0n

; Temporary directory for HTTP uploaded files (will use system default if not
; specified).
; http://php.net/upload-tmp-dir
;;Comentario : Establece el directorio temporal en el servidor donde se deja temporalmente el fichero subido
;;Si no se especifica se tomará /tmp en linux, o el directorio por defecto que use el SO de forma temporal
upload_tmp_dir =

; Maximum allowed size for uploaded files.
; http://php.net/upload-max-filesize
;;Comentario : Tamaño máximo del fichero permitido en el servidor
;;Se puede usar los múltiplos K M G T
upload_max_filesize = 20M

; Maximum number of files that can be uploaded via a single request
;;Número máximo de ficheros que se pueden descargar en una sola solicitud http
max_file_uploads = 20

```

- Podemos ver el tamaño exacto del fichero subido mediante el elemento size del array

```

//Suponemos en el cliente
//... <input type=file name= fichero>
....
$size = $_FILES['fichero']['size'];
....

```

- Entre otras cosas puede servir para descartar un fichero de menos de un tamaño concreto.

## Tipo de fichero

- Este es un atributo importante

1. Analizar el tipo de fichero para poder por ejemplo aceptarlo o descartarlo o decidir en qué carpeta dejarlo en función del tipo

- Para ver el tipo podemos observar la extensión del fichero.
- O bien analizar el tipo MIME que nos viene en **`$_FILES['type']`**
- Por ejemplo suponemos que queremos distribuir los ficheros en tres carpetas

1. Los ficheros que contengan imágenes a la carpeta ***./descargas/imagenes/***
2. Los ficheros que contengan música a la carpeta ***./descargas/musica/***
3. El resto de ficheros a la carpeta ***./descargas/otros/***

- En el tipo mime separa el tipo general del fichero con una barra.
- Así los de tipo música o audio sería ***audio/....'***
- Así los de tipo imagen ***image/....'***

.....

- Un posible código sería

```
//Suponemos en el cliente
//... <input type=file name= fichero>

.....
$origen = $_FILES['fichero']['tmp_name'];
$nombreFichero = $_FILES['fichero']['name'];
$tipo = $_FILES['fichero']['type'];
$tipo_fichero = explode('/', $tipo);
switch ($tipo_fichero[0]) {
    case 'audio':
        $dir_destino = "/var/www/descargas/uploads/musica";
        break;
    case 'image':
        $dir_destino = "/var/www/descargas/uploads/imagenes";
        break;
    default:
        $dir_destino = "/var/www/descargas/uploads/otros";
}
$destino = $dir_destino . '/' . basename($nombreFichero);
move_uploaded_file($origen, $destino);
```



**Tip:** la función ***explode***

Esta función rompe una cadena de caracteres en diferentes campos de un array indexado cada vez que encuentre un determinado carácter. Tango el carácter, como la cadena son argumentos pasados a la función. <http://php.net/manual/es/function.explode.php>

## Trabajar con directorios

- Una vez subidos los ficheros es habitual que se quieran mostrar en la página para que el usuario los pueda ver o acceder a ellos
- Para ello, php tiene una serie de funciones que nos permite interactuar con un directorio y sus ficheros igual que si trabajáramos en un terminal.



## PHP: Trabajando con ficheros y directorios

Funciones de directorio

<http://php.net/manual/es/ref.dir.php>

Funciones sobre ficheros

<http://php.net/manual/es/ref.filesystem.php>



**Tip:** No olvidar que es **www-data** o el propietario del proceso de **Apache** quien ha de tener los permisos necesarios para interactuar con los ficheros o directorios

### Ejemplo de uso de la clase *Directory*

- El tema de programación orientado a objetos lo veremos mas tarde, pero sabemos que para invocar a un método de un objeto se usa el operador de indirección ->
- Los métodos los vemos como funciones que son de una clase y los puede invocar un objeto. (Esta idea imprecisa de momento es suficiente).
- **Directory** Es una clase especializada en gestionar directorios (los directorios son ficheros igualmente)
- Para inicializarlo podemos usar un alias de **new Directory()** llamada **dir(...)**
- En su invocación pasamos el directorio que queremos ver

```
$directorio = dir("/var/www/musica/subidas/");
```

- Para leer el contenido de un fichero o un directorio (los archivos que contiene), se usa el método **read()**
- Una forma de hacerlo es

```
while ($archivo = $directorio -> read()){  
    .....  
}
```

- Es importante cerrar el fichero (en este caso de tipo directorio), con el método close

```
$directorio->close();
```



## Recursos de la Web

- Página oficial de php para la descarga de ficheros <http://php.net/manual/es/features.file-upload.php>
- Listado completo de todos los tipos MIME:

```
https://www.sitepoint.com/web-foundations/mime-types-complete-list/  
http://www.iana.org/assignments/media-types/media-types.xhtml
```

- Funciones de php para trabajar con ficheros [http://www.w3schools.com/php/php\\_ref\\_filesystem.asp](http://www.w3schools.com/php/php_ref_filesystem.asp)

Obtenido de «<http://es.wikieducator.org/index.php?title=Usuario:ManuelRomero/php/NewPHP/B2T1/formularios&oldid=22436>»

- Esta página fue modificada por última vez el 2 nov 2017, a las 12:18.
- Esta página se ha visitado 756 veces.
- El contenido está disponible bajo Creative Commons Attribution Share Alike License a menos que se indique lo contrario.