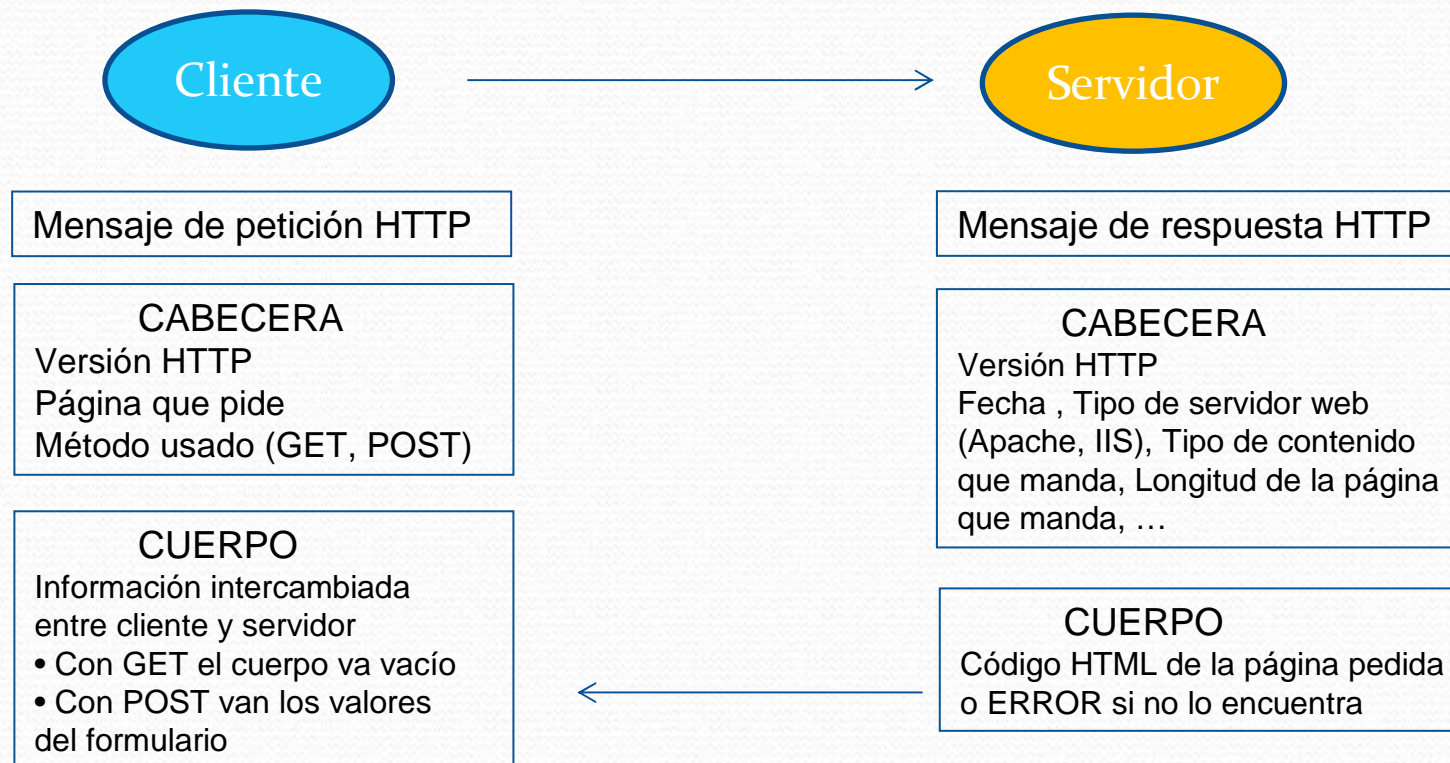


# U.T. 4: Recuperación y utilización de información proveniente del cliente web.

# FORMULARIOS: Protocolo HTTP

- HTTP es un protocolo sin estado. El cliente inicia la comunicación:



# FORMULARIOS: Protocolo HTTP

Ejemplo de mensaje de respuesta HTTP

Servidor

```
HTTP/1.1 200 OK
Date: Thu, 01 Jan 2001 06:32:56 GMT
Server: Apache/1.3.14 (Unix) PHP/3.0.14
Last-Modified: Tue, 04 Apr 2000 13:54:53 GMT
Content-Length: 465
Connection: close
Content-Type: text/html
```

CABECERA del  
mensaje

Línea vacía

```
<HTML>
<HEAD><TITLE>Dpto. O.E.I. - U.P.M.</TITLE>
</HEAD>
<BODY BGCOLOR="#000000" VLINK="#FFFFFF">
...
```

CUERPO del  
mensaje



# FORMULARIOS: Acceso desde PHP

- ❑ Los Formularios **no** forman parte de PHP, sino del lenguaje estándar de Internet, **HTML**.
  - ❑ Puesto que se utiliza el protocolo HTTP, estamos limitados por su interfaz: sólo se puede utilizar alguno de los comandos del protocolo para establecer la comunicación:
    - ❑ Se utilizan dos comandos del protocolo: GET o POST
  - ❑ Dos tipos diferentes de peticiones, según atributo **method** del **<FORM>**:
    - ❑ Peticiones GET (método GET de HTTP)
    - ❑ Peticiones POST (método POST de HTTP)
- `<FORM ACTION="lo_que_sea.php" METHOD="post/get">`
- ❑ Al pulsar el botón de envío el navegador construye la petición adecuada
-

# FORMULARIOS: Acceso desde PHP

## ☐ Peticiones GET (método GET de HTTP):

- ☐ Los parámetros se le añaden a la URL tras un signo de “?” y se concatenan con &.

- ☐ En el servidor, los valores se guardan en el array asociativo `$_GET`.

`http://site/procesa.php?name1=value1&name2=value2&name3=value3`

## ☐ Reglas de codificación URL

- ☐ RFC 1738

- ☐ Los caracteres especiales se traducen:

- ☐ Espacios en blanco se traducen a “+”, y después a %2B

- ☐ La # se traduce por %23

- ☐ Los caracteres especiales se envían con el formato %NN (NN: valor hexadecimal de carácter ).

## ☐ Son caracteres especiales:

- ☐ Ñ,ñ,á,.. (no tienen un carácter US ASCII asociado)

- ☐ Los peligrosos: <, >, ", #

- ☐ Los reservados con significado especial: /, @, ?, &
-



# FORMULARIOS: Acceso desde PHP

## ☐ Ejemplo:

```
echo "<a href=\"proces.php?user=$user&uid=$uid\">";
```

## ☐ Si \$user="Alvaro Gil" y \$uid="13&05" genera el enlace:

```
<a href="proces.php?user=Álvaro Gil&uid=12&57"> // INCORRECTO
```

## ☐ Usar las funciones php **urlencode** y **urldecode**

- ☐ **urldecode** en realidad no hace falta usarla , el intérprete de php realiza esta transformación cuando recibe la petición

```
echo "<a href=\"proces.php?user=".urlencode($user)."&uid=".urlencode($uid).\">";
```

## ☐ Esto genera:

```
<a href="proces.php?user=%C1lvaro+Gil&uid=12%2657"> // CORRECTO
```

# FORMULARIOS: Acceso desde PHP

- ❑ Peticiones **POST** (método POST de HTTP)
    - ❑ Los parámetros se envían en el **cuerpo del mensaje no en la** cadena de solicitud (query string).
    - ❑ En el servidor, los valores se guardan en el array asociativo **\$\_POST**.
    - ❑ Como con GET, los caracteres especiales se traducen a ASCII.
    - ❑ Es necesario indicar el tipo de codificación en el <form> con el atributo **enctype**
      - ❑ **application/x-www-form-urlencoded** (Por defecto).
        - ❑ NO PERMITE ENVIAR ARCHIVOS
      - ❑ **multipart/form-data**.
        - ❑ PERMITE ENVIAR ARCHIVOS
-



# FORMULARIOS: Acceso desde PHP

- ☐ GET vs POST
  - ☐ Problemas GET
    - ☐ No se puede enviar información binaria (archivos, imágenes, etc.) => necesario el método POST.
    - ☐ Los datos se ven en la URL del navegador.
  - ☐ Problemas POST
    - ☐ Rompe la funcionalidad del botón “Atrás” del navegador
    - ☐ El botón actualizar repite la operación
  - ☐ Principios generales
    - ☐ GET implica “obtener” información.
    - ☐ POST implica “realizar” una acción con un “efecto secundario”.
    - ☐ Mejor POST para procesar formularios
-



# FORMULARIOS: Acceso desde PHP

Por tanto:

☐ Acceso a los valores introducidos en los formularios a través de arrays globales:

- ☐ **`$_GET`**: parámetros enviados mediante GET o en la URL
  - ☐ **`$_POST`**: parámetros enviados mediante POST
  - ☐ **`$_REQUEST`**: la unión de **`$_GET`** y **`$_POST`** (también **`$_COOKIES`**)
  - ☐ Recuerda que la entrada en el `php.ini` que activaba el uso de variables globales ha quedado deprecated:  
*`register_globals=OFF` en `php.ini`*
  - ☐ Esto crea una variable global para cada parámetro recibido, pero se desaconseja por razones de seguridad
-

# FORMULARIOS: Acceso desde PHP

- ❑ Desde PHP se puede acceder a los datos de un formulario HTML.

- ❑ Fichero **uno.html**:

```
<html><body>  
<form action="dos.php" method="POST">  
    Edad: <input type="text"  
    name="edad">  
    <input type="submit" value="aceptar">  
</form>  
</body></html>
```

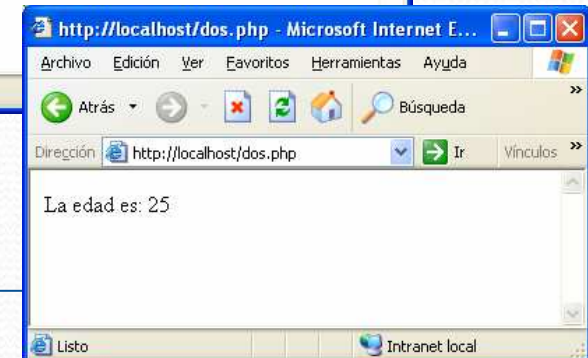
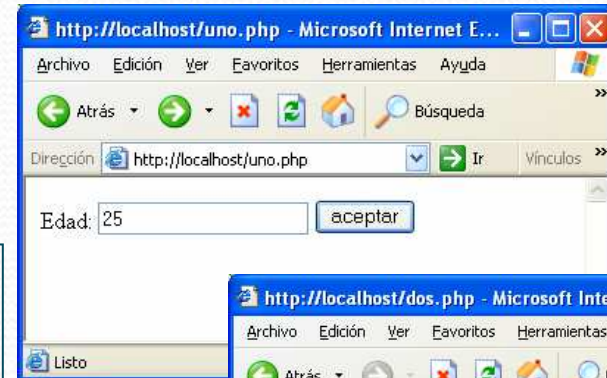
Página a la que se llamará al hacer click en submit

Estos valores deben ser iguales. Si no, no podremos recuperar la información.

- ❑ Fichero **dos.php**

```
<?php  
echo "La edad es:". $_REQUEST['edad'];  
?>
```

Podría ser \$\_POST





# FORMULARIOS: Acceso desde PHP

## ❑ Formulario:

```
<form action="respuesta.php?valor=10" method="post">  
<input type="text" name="nombre">  
</form>
```

## ❑ PHP (¿todos correctos?)

```
<?  
echo 'valor: '.$_GET['valor'].'<br>';  
echo 'valor: '.$_POST['valor'].'<br>'; → vacío  
echo 'valor: '.$_REQUEST['valor'].'<br>';  
echo 'nombre: '.$_GET['nombre'].'<br>'; → vacío  
echo 'nombre: '.$_POST['nombre'].'<br>';  
echo 'nombre: '.$_REQUEST['nombre'].'<br>';  
?>
```

---

# FORMULARIOS: Acceso desde PHP

- ❑ Acceso a los diferentes tipos de elementos de entrada de formulario
    - ❑ Elementos de tipo **INPUT**
      - ❑ TEXT
      - ❑ RADIO
      - ❑ CHECKBOX
      - ❑ BUTTON
      - ❑ FILE
      - ❑ HIDDEN
      - ❑ PASSWORD
      - ❑ SUBMIT / RESET
    - ❑ Elemento **SELECT**
      - ❑ Simple / múltiple
    - ❑ Elemento **TEXTAREA**
-



# FORMULARIOS: Acceso desde PHP

## □ TEXT

Introduzca la cadena a buscar:

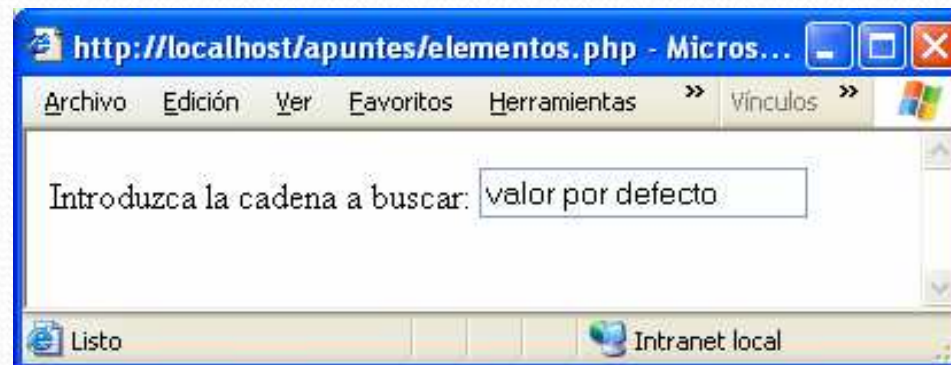
```
<INPUT TYPE="text" NAME="cadena" VALUE="valor por defecto" SIZE="20">
```

```
<?PHP
```

```
    $cadena = $_REQUEST['cadena'];
```

```
    echo $cadena;
```

```
?>
```



# FORMULARIOS: Acceso desde PHP

## ☐ RADIO

Sexo:

```
<INPUT TYPE="radio" NAME="sexo" VALUE="M" CHECKED>Mujer
```

```
<INPUT TYPE="radio" NAME="sexo" VALUE="H">Hombre
```

```
<?PHP
```

```
    $sexo = $_REQUEST['sexo'];
```

```
    echo ($sexo);
```

```
?>
```



- ☐ Los botones radio se llaman igual para que si se elije uno se desmarquen los otros.
-



# FORMULARIOS: Acceso desde PHP

## ☐ BUTTON

```
<INPUT TYPE="button" NAME="actualizar" VALUE="Actualizar datos">
```

```
<?PHP
```

```
    $actualizar = $_REQUEST['actualizar'];
```

```
    if ($actualizar)
```

```
        print ("Se han actualizado los datos");
```

```
?>
```



# FORMULARIOS: Acceso desde PHP

## ☐ HIDDEN

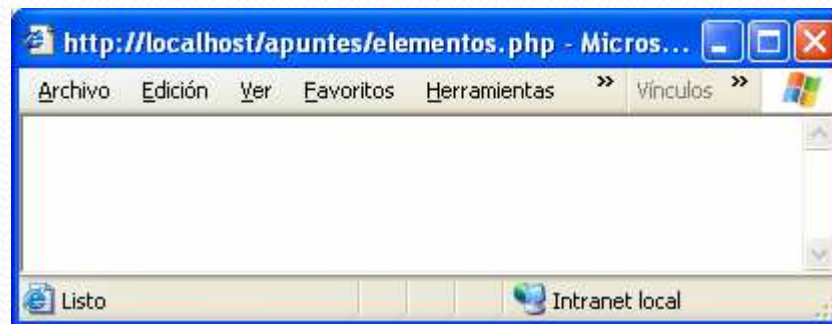
```
<INPUT TYPE='hidden' NAME='username' VALUE="$usuario">
```

```
<?PHP
```

```
$username = $_REQUEST['username'];
```

```
echo $username;
```

```
?>
```





# FORMULARIOS: Acceso desde PHP

## ☐ PASSWORD

Contraseña: <INPUT TYPE="password" NAME="clave">

```
<?PHP
```

```
$clave = $_REQUEST['clave'];
```

```
echo $clave;
```

```
?>
```



# FORMULARIOS: Acceso desde PHP

## ❑ SUBMIT

```
<INPUT TYPE=submit NAME="enviar" VALUE="Enviar datos">
```

```
<?PHP
```

```
    $enviar = $_REQUEST['enviar'];
```

```
    if ($enviar)
```

```
        echo "Se ha pulsado el botón de enviar";
```

```
?>
```





# FORMULARIOS: Acceso desde PHP

## ❑ RESET

```
<INPUT TYPE=reset NAME="borrar" VALUE="Limpiar datos">
```

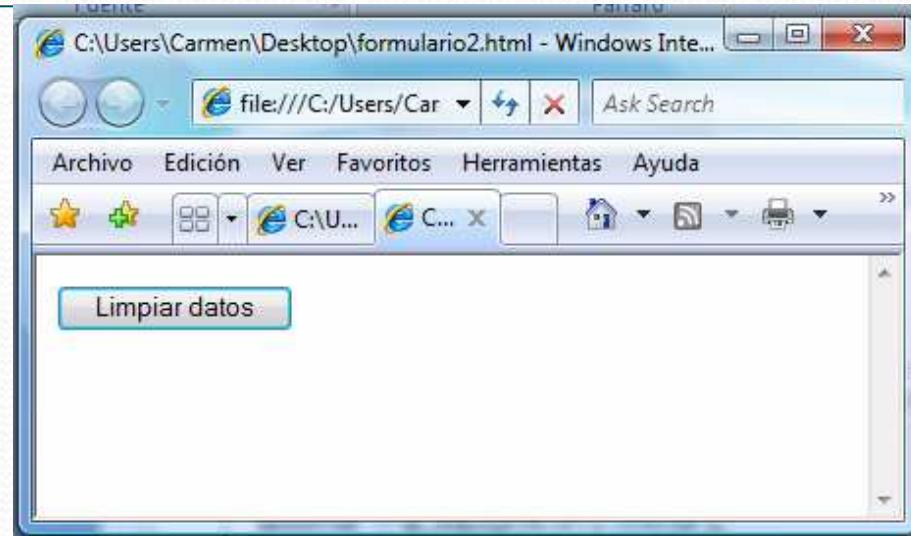
```
<?PHP
```

```
    $borrar = $_REQUEST['borrar'];
```

```
    if ($borrar)
```

```
        echo "Se ha pulsado el botón de limpiar datos";
```

```
?>
```



# FORMULARIOS: Acceso desde PHP

## □ SELECT Simple

Color:

```
<SELECT NAME="color">
<OPTION VALUE="rojo" SELECTED>Rojo</OPTION>
<OPTION VALUE="verde">Verde</OPTION>
<OPTION VALUE="azul">Azul</OPTION>
</SELECT>
```

```
<?PHP
```

```
    $color = $_REQUEST['color'];
```

```
    echo $color;
```

```
?>
```





# FORMULARIOS: Acceso desde PHP

## ☐ SELECT Múltiple

### ☐ Valores vectoriales de un formulario:

- ☐ Select múltiples o botones de comprobación con el mismo nombre
- ☐ Si no se indica nada, sólo se tiene acceso a un valor
- ☐ En el código HTML hay que añadir "[]" al nombre del control
- ☐ Devuelve un array, con count() podemos conocer su tamaño

Idiomas:

```
<SELECT MULTIPLE SIZE="3" NAME="idiomas[]">
<OPTION VALUE="ingles" SELECTED>Inglés</OPTION>
<OPTION VALUE="frances">Francés</OPTION>
<OPTION VALUE="aleman">Alemán</OPTION>
<OPTION VALUE="holandes">Holandés</OPTION>
</SELECT>
```

```
<?PHP
```

```
    $idiomas = $_REQUEST['idiomas'];
    foreach ($idiomas as $idioma)
        echo "$idioma<BR>\n";?>
```



# FORMULARIOS: Acceso desde PHP

## ☐ CHECKBOX

- ☐ botones de comprobación con el mismo nombre → usar **arrays [ ]**

```
<INPUT TYPE="checkbox" NAME="extras[]" VALUE="garaje" CHECKED>Garaje  
<INPUT TYPE="checkbox" NAME="extras[]" VALUE="piscina">Piscina  
<INPUT TYPE="checkbox" NAME="extras[]" VALUE="jardin">Jardín
```

```
<?PHP  
$extras = $_REQUEST['extras'];  
foreach ($extras as $extra)  
    echo "$extra<BR>\n";  
?>
```





# FORMULARIOS: Acceso desde PHP

## □ TEXTAREA

Comentario:

```
<TEXTAREA COLS="50" ROWS="4" NAME="comentario">
```

Este libro me parece ...

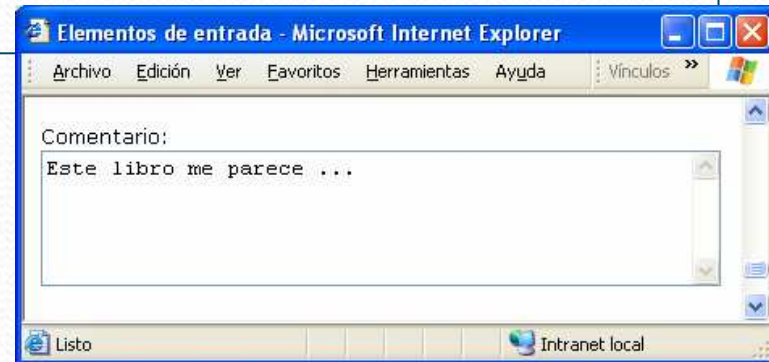
```
</TEXTAREA>
```

```
<?PHP
```

```
    $comentario = $_REQUEST['comentario'];
```

```
    echo $comentario;
```

```
?>
```

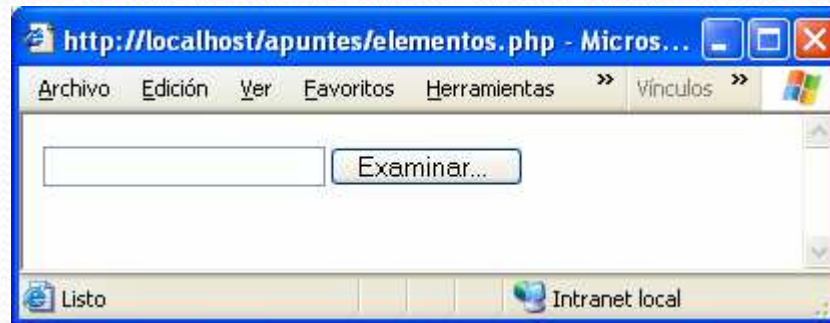


- Ejercicio 1: formulario simple que incluya todos los controles excepto FILE y BUTTON
-

# FORMULARIOS: Acceso desde PHP

## □ FILE

```
<FORM ACTION="procesa.php" METHOD="post" ENCTYPE="multipart/form-data">  
<INPUT TYPE="file" NAME="fichero">  
</FORM>
```



- Ejercicio 2: página php que muestra los datos introducidos desde el formulario del ejercicio 1.
    - Ilustra cómo acceder a los valores introducidos desde todos los tipos de elementos de entrada de un formulario, con excepción de los tipos BUTTON y FILE, que se tratan en ejercicios posteriores
-



# FORMULARIOS: Subida de ficheros al servidor

- ☐ Para subir un fichero al servidor se utiliza el elemento de entrada FILE
  - ☐ Hay que tener en cuenta una serie de consideraciones importantes:
    - ☐ El elemento FORM debe tener el atributo **ENCTYPE="multipart/form-data"**
    - ☐ El fichero tiene un límite en cuanto a su tamaño. Este límite se fija de dos formas diferentes y complementarias:
      - ☐ En el fichero de configuración php.ini
      - ☐ En el propio formulario
-

# FORMULARIOS: Subida de ficheros al servidor

## ❑ php.ini

```
.....  
; File Uploads ;  
.....  
; Si se permite o no subir archivos mediante HTTP  
file_uploads = On  
;  
; Tamaño máximo de cada archivo subido.  
upload_max_filesize = 2M  
; Tamaño máximo de los datos mandados por POST  
;(incluidos los que no sean archivos)  
post_max_size = 8M
```

## ❑ formulario

se tiene que llamar así y es un entero con el valor en bytes

```
<INPUT TYPE="HIDDEN" NAME="MAX_FILE_SIZE" VALUE='102400'>  
<INPUT TYPE="FILE" NAME="fichero">
```



# FORMULARIOS: Subida de ficheros al servidor

```
<INPUT TYPE="FILE" SIZE="44" NAME="imagen">
```

- ❑ La variable `$_FILES` contiene toda la información del fichero subido:
    - ❑ `$_FILES['imagen']['name']`
      - ❑ Nombre original del fichero en el cliente
    - ❑ `$_FILES['imagen']['type']`
      - ❑ Tipo MIME del fichero. Por ejemplo, "image/gif"
    - ❑ `$_FILES['imagen']['size']`
      - ❑ Tamaño en bytes del fichero subido
    - ❑ `$_FILES['imagen']['tmp_name']`
      - ❑ Nombre temporal del fichero que se genera para guardar el fichero subido
    - ❑ `$_FILES['imagen']['error']`
      - ❑ Código de error asociado a la subida del fichero
-

# FORMULARIOS: Subida de ficheros al servidor

## ❑ Consideraciones (cont)

- ❑ Debe darse al fichero un **nombre único**. Por ello, y como norma general, debe descartarse el nombre original del fichero y crear uno nuevo que sea único, p.e añadiéndole la fecha y hora
- ❑ El fichero subido se almacena en un directorio temporal y hemos de moverlo al directorio de destino usando la función

**move\_upload\_file()**

## ❑ Procedimiento:

si se ha subido correctamente el fichero

Asignar un nombre al fichero

Mover el fichero a su ubicación definitiva

si no

Mostrar un mensaje de error

finsi

---



# FORMULARIOS: Subida de ficheros al servidor

## ❑ Procedimiento:

si se ha subido correctamente el fichero

- ❑ Lo comprobamos con **is\_uploaded\_file("nombre temporal de \$\_FILES")**
- ❑ Devuelve **true** si el archivo que se le pasa se ha subido por HTTP POST.  
Evita que el usuario intente usar archivos del servidor /etc/passwd

Asignar un nombre al fichero

- ❑ Añadir marca de tiempo

Mover el fichero a su ubicación definitiva

- ❑ **move\_uploaded\_file ( \$\_FILES['archivo'] ['tmp\_name'], \$destino)**
- ❑ Lo mueve y si no puede da error

si no

Mostrar un mensaje de error

finsi

---

# FORMULARIOS: Subida de ficheros al servidor

## ❑ Ejemplo(I)

```
<html><body>  
Inserción de la fotografía del usuario:  
<form action="inserta.php" method="post" enctype="multipart/form-data">  
<?php  
echo "Nombre usuario:<input type='text' name='usuario'/><br/>";  
echo "Fichero con su fotografía:<input type='file' name='imagen'/><br/>";  
?>  
<input type="submit" value="Enviar">  
</form></body></html>
```

Inserción de la fotografía del usuario:

Nombre usuario:

Fichero con su fotografía:



# FORMULARIOS: Subida de ficheros al servidor

## ❑ Ejemplo(II) - inserta.php

```
<html><body><?php
    echo "name:".$_FILES['imagen']['name']."\n";
    echo "tmp_name:".$_FILES['imagen']['tmp_name']."\n";
    echo "size:".$_FILES['imagen']['size']."\n";
    echo "type:".$_FILES['imagen']['type']."\n";
    if (is_uploaded_file($_FILES['imagen']['tmp_name'])) {
        $nombreDirectorio = "img/";
        $nombreFichero = $_FILES['imagen']['name'];
        $nombreCompleto = $nombreDirectorio.$nombreFichero;
        if (is_dir($nombreDirectorio)) { // es un directorio existente
            $idUnico = time();
            $nombreFichero = $idUnico."-".$nombreFichero;
            $nombreCompleto = $nombreDirectorio.$nombreFichero;
            move_uploaded_file($_FILES['imagen']['tmp_name'],$nombreCompleto);
            echo "Fichero subido con el nombre: $nombreFichero<br>";
        }
        else echo 'Directorio definitivo inválido';
    }
    else
        print ("No se ha podido subir el fichero\n");
?></body></html>
```

```
name:Foto.png
tmp_name:C:\xampp\tmp\php7EE.tmp
size:15811
type:image/x-png
Fichero subido con el nombre: 1241894493-Foto.png
```

# FORMULARIOS: Subida de ficheros al servidor

## ☐ **is\_uploaded\_file (\$\_FILES['imagen']['tmp\_name'])**

- ☐ Devuelve TRUE si el archivo que se pasa fue cargado a través de HTTP POST. Evita que un usuario intente que se manejen ficheros no cargados por POST. p.e /etc/passwd
- ☐ Necesita como argumento \$\_FILES['archivo\_usuario']['tmp\_name']
- ☐ Si se le pasa \$\_FILES['archivo\_usuario']['name'] **no funciona**.

```
<?php
if (is_uploaded_file($_FILES['archivo_usuario']['tmp_name'])) {
    echo "El archivo ". $_FILES['archivo_usuario']['name'] ." fue cargado correctamente.\n";
    echo "Mostrando su contenido\n";
    readfile($_FILES['archivo_usuario']['tmp_name']);
} else {
    echo "Posible ataque de carga de archivo: ";
    echo "nombre de archivo ". $_FILES['archivo_usuario']['tmp_name'] . " . " .";}
?>
```



# FORMULARIOS: Subida de ficheros al servidor

❑ **move\_uploaded\_file (\$\_FILES['imagen']['tmp\_name'],\$destino)**



A red bracket connects the `['tmp_name']` part of the function signature to a box containing the text `nombre_temporal_archivo`. A downward arrow points from the bracket to the box.

`nombre_temporal_archivo`

- ❑ Esta función realiza un chequeo para asegurar que el archivo indicado por el primer parámetro sea un archivo cargado a través de HTTP POST.
  - ❑ Si el archivo es válido, será movido al nombre de archivo dado por **destino**.
  - ❑ Si *nombre\_temporal\_archivo* no es un archivo cargado válido, no hará nada, y devolverá FALSE.
  - ❑ Si *nombre\_temporal\_archivo* es un archivo cargado válido, pero no puede ser movido por alguna razón, no hará nada, devolverá FALSE y dará una advertencia.
-

# FORMULARIOS: Subida de ficheros al servidor

## ☐ Variable predefinida \$\_FILES

### ☐ Precauciones:

- ☐ Permisos de escritura en el directorio temporal
- ☐ Permisos de escritura en el directorio de destino
- ☐ Atención con los ficheros que puedan subir los usuarios
- ☐ Troyanos, scripts, ejecutables, etc.

## ☐ Ejercicio 4: subida de un fichero al servidor

- ☐ Ilustra cómo subir ficheros a un servidor, cómo controlar su tamaño, cómo crear un nombre único para el fichero y cómo almacenarlo en el lugar deseado.
-



# FORMULARIOS: Validación de los datos de un formulario

- ❑ Toda la información proveniente de un formulario debe considerarse por norma como contaminada, y hay que validarla antes de darla por buena y procesarla
- ❑ Lo más eficiente es mostrar los errores sobre el propio formulario para facilitar su corrección. Procedimiento:

si se ha enviado el formulario

si hay errores

Mostrar formulario con errores

si no

Procesar formulario

finsi

si no

Mostrar formulario

finsi

---

# FORMULARIOS: Validación de los datos de un formulario

- ❑ Este procedimiento se puede resumir para que sólo haya que mostrar una vez el formulario, bien con los valores por defecto o con los valores introducidos, y con los errores en su caso:

si se ha enviado el formulario

validar datos

finsi

si se ha enviado el formulario y no hay errores

Procesar formulario

si no

Mostrar formulario con valores por defecto o ya enviados

finsi

---



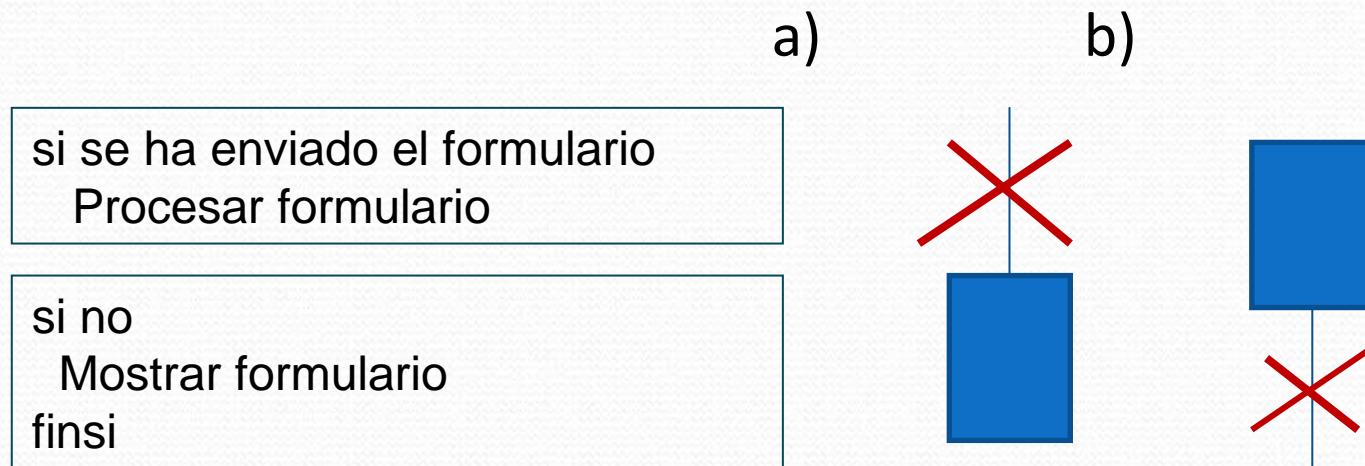
# FORMULARIOS: Procesar formularios desde PHP

- ☐ Una forma de trabajar con formularios en PHP es utilizar un **único programa** que procese el formulario o lo muestre según haya sido o no enviado, respectivamente
- ☐ Ventajas:
  - ☐ Disminuye el número de ficheros
  - ☐ Permite validar los datos del formulario en el propio formulario
- ☐ Procedimiento:

```
si se ha enviado el formulario
    Procesar formulario
si no
    Mostrar formulario
finsi
```

# FORMULARIOS: Procesar formularios desde PHP

## ❑ Esquema de funcionamiento



- ❑ La 1ª vez que se carga la página se muestra el formulario (a)
- ❑ La 2ª vez se procesa el formulario (b)



# FORMULARIOS: Procesar formularios desde PHP

- ❑ Para saber si se ha enviado el formulario se acude a la variable correspondiente al botón de envío. Si este botón aparece de la siguiente forma en el formulario HTML:

```
<INPUT TYPE=SUBMIT NAME="enviar" VALUE="procesar">
```

- ❑ entonces la condición anterior se transforma en:

```
if (isset($_POST['enviar']))
```

- ❑ o bien

```
if ($_POST['enviar'] == "procesar")
```

- ❑ Ejercicio 3: formulario de PHP
    - ❑ Ilustra cómo crear páginas multipropósito que sirven tanto para mostrar un formulario como para procesarlo
-

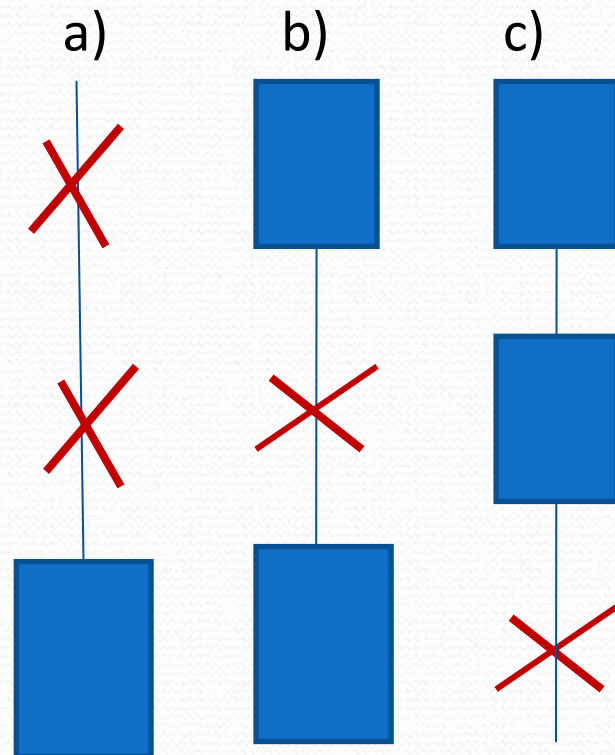
# FORMULARIOS: Validación de los datos de un formulario

## Esquema de funcionamiento:

si se ha enviado el formulario  
validar datos  
finsi

si se ha enviado y no hay errores  
Procesar formulario

si no  
Mostrar formulario  
finsi





# FORMULARIOS: Validación de los datos de un formulario

## ☐ Esquema de funcionamiento:

- ☐ La 1ª vez que se carga la página se muestra el formulario (a)
  - ☐ La 2ª y sucesivas veces se validan los datos
    - ☐ Si hay errores, se muestra de nuevo el formulario con los errores (b)
    - ☐ Si no hay, se procesa el formulario (c)
  - ☐ Ejercicio 5: validación de los datos introducidos en un formulario
    - ☐ Ilustra cómo realizar una validación de los datos introducidos en un formulario en el propio formulario
-

# Cabeceras HTTP: Introducción

[Plan de formación 2011: Familia profesional de Informática y Comunicaciones]

## □ Cabeceras HTTP

- Aportan información en peticiones y respuestas a servidores

- Función para enviar cabeceras HTTP:

`header("cabecera: valor");`

- Ejemplos:

- `header("location: http://www.upm.es");`

- `header("HTTP/1.0 404 Not Found");`

- `header("Pragma: no-cache");`

- Otras: Cache-Control, Expires, Last-Modified, etc.

- Es importante que esta orden se encuentre antes de la etiqueta `<html>` inicial.

- Las cabeceras HTTP pueden también modificar el comportamiento del navegador que recibe la respuesta

---



# Cabeceras HTTP: Introducción

## ❑ Petición

**GET** / HTTP/1.1

**Host:** www.google.com

**User-Agent:** Mozilla/5.0 (Windows NT 6.0; rv:5.0) Gecko/20100101 Firefox/5.0

**Accept:** text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8

**Accept-Language:** es-es,es;q=0.8,en-us;q=0.5,en;q=0.3

**Accept-Encoding:** gzip, deflate

**Accept-Charset:** ISO-8859-1,utf-8;q=0.7,\*;q=0.7

**Connection:** keep-alive

**Cookie:**

PREF=ID=0c4479bc48cc54b9:U=2153a747ac5d1320:FF=0:TM=1234112738:L  
M=1296123415:GM=1:S=wSoNqwpmFJGDC\_wn; NID=42=qaDR0eGhuO5H-  
sGvS67GmQvMsKCxQ8N9U9\_La6rq2m3nWwktAFUTrHF3P9d-jzAi-  
X5j5SrbFFD\_dIgLSW60CnMCxiupnBhapGUpLvIFdq3XsJzpOT4xHLtVF4t7zdy\_

---

# Cabeceras HTTP: Introducción

## ❑ Respuesta

**HTTP/1.1** 302 Found

**Location:** <http://www.google.es/>

**Cache-Control:** private

**Content-Type:** text/html; charset=UTF-8

**Date:** Sun, 10 Jul 2011 16:35:32 GMT

**Server:** gws

**Content-Length:** 218

**X-XSS-Protection:** 1; mode=block

---



# Cabeceras: Introducción

- ❑ Live HTTP Headers

- ❑ Es una extensión para los navegadores (firefox) que nos permite visualizar de manera cómoda las cabeceras HTTP navegador/servidor.

- ❑ Especificación completa de cabeceras HTTP:

<http://www.faqs.org/rfcs/rfc2616.html>

# Cabeceras: Introducción

- ❑ Enviar cabeceras HTTP en PHP: **header()**
  - ❑ Debe aparecer antes de enviar cualquier otra cosa al cliente (antes de cualquier hecho).
  - ❑ `header('Content-Type: text/html; charset=UTF-8');`

- ❑ Extraer cabeceras HTTP del cliente en PHP:

**apache\_request\_headers()**

```
<?php
```

```
var_dump(apache_request_headers());
```

```
?>
```

---



# Cabeceras HTTP en PHP

- ❑ Redirigir al cliente a otra dirección.

```
001 header ( 'Location: acceso_no_autorizado.php' );
```

# Cabeceras HTTP en PHP

- ❑ Mostrar un mensaje y redirigir al cliente a otra dirección.

```
001 header ('Refresh: 5; url=http://www.google.es');  
002 echo 'Lo que busca no existe, le redirigiremos a Google en 5 segundos'
```



# Cabeceras HTTP en PHP

- ❑ Ocultar la versión de nuestro intérprete PHP.

```
001 header( 'X-Powered-By: adivina-adivinanza' );
```

- ❑ Alternativamente también podríamos ocultar nuestra versión de PHP asignando el valor 0 a la directiva *expose\_PHP* del *php.ini*.
-

# Cabeceras HTTP en PHP

- ❑ Ofrecer la descarga de un archivo desde PHP.

```
001 header('Content-type: application/pdf');  
002 header('Content-Disposition: attachment;  
    filename="downloaded.pdf");  
003 readfile('original.pdf');
```

Vamos a mostrar un pdf

La fuente del PDF se encuentra  
en *original.pdf*

Proporciona un nombre de  
fichero recomendado y fuerza al  
navegador el mostrar el diálogo  
para guardar el fichero.



# Cabeceras HTTP en PHP

- ❑ Generar contenidos diferentes a páginas HTML con PHP.
  - ❑ Imágenes, documentos PDF, películas SWF de Flash en tiempo real (sin leerlos de un archivo)
  - ❑ Es posible gracias a librerías de PHP como GD, PDFlib, Ming, etc.

```
025     header( "Content-type:image/jpeg" );  
026     header( "Content-Disposition:inline ; filename=captcha.jpg" );
```

El fichero se abre en el navegador en lugar de descargarse (attachment)