

# PHP

# avanzado

Realizado por A.Garay (dpto. Informática)

# Resumen de contenidos

- [Conceptos](#)
- [El protocolo HTTP](#)
- [Solicitud de descargas: hipervínculos y formularios](#)
- [Envío de parámetros GET y POST](#)
- [Variables superglobales: recepción de parámetros en PHP](#)
- [Codificación URL's](#)
- Etiquetas básicas de formularios
  - [form](#)
  - [input](#)
  - [label](#)
  - select: [normal](#) y [múltiple](#)
  - [textarea](#)
  - [fieldset](#)
- [Envío de archivos al servidor](#)
- [isset\(\) y empty\(\)](#)
- [Trucos](#)

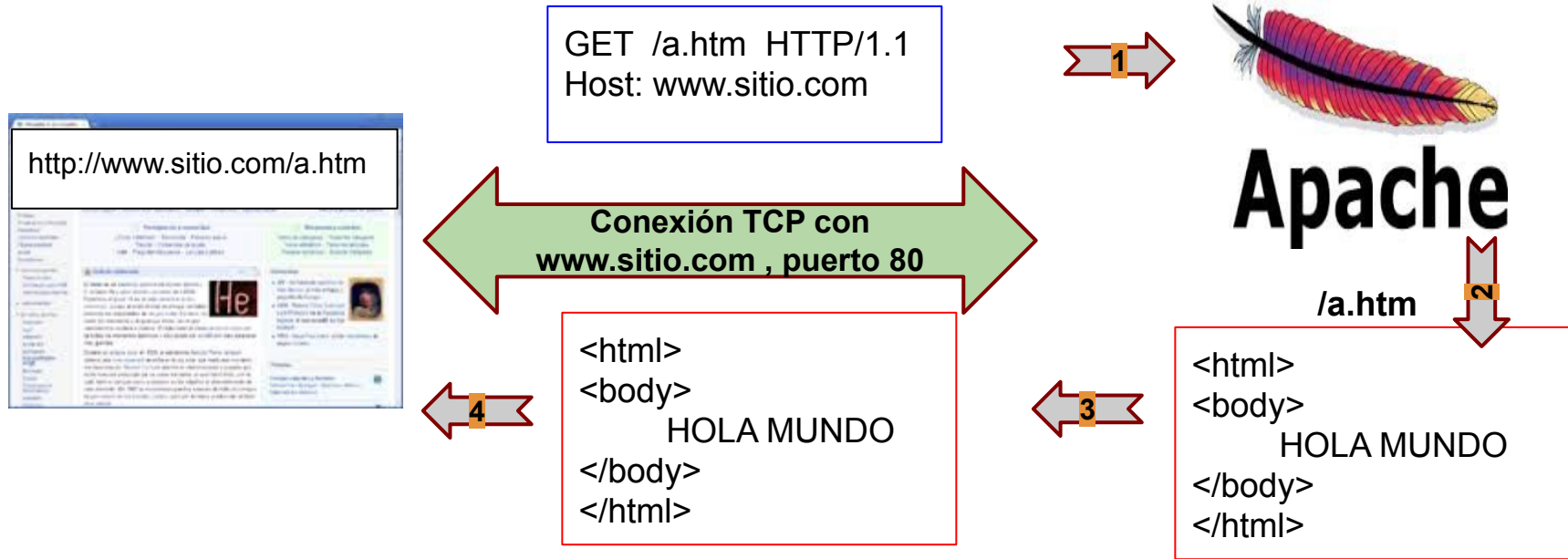
# Conceptos (1/4) Entorno servidor

- La programación web en entorno servidor se basa en el protocolo HTTP
- El protocolo HTTP sirve fundamentalmente para descargar archivos desde un servidor a un cliente (usualmente desde un servidor web a un navegador).
- La información fundamental que un cliente HTTP le comunica a un servidor es **qué archivo (o recurso) es el que quiere descargarse.**
- La ubicación de dicho recurso a descargar se indica utilizando un URI (uniform resource identifier)

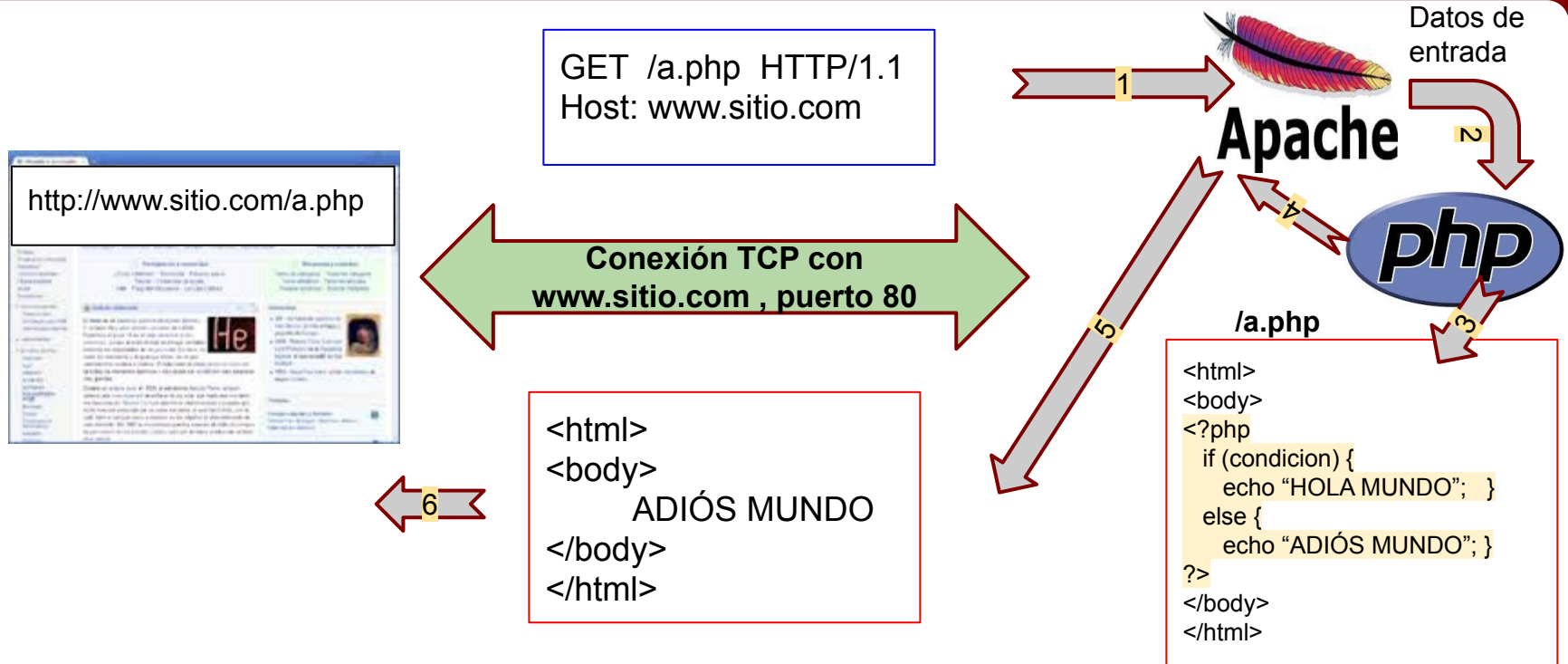
# Conceptos (2/4) URI básica

- Es una cadena de caracteres sirve para concretar en qué host y en qué lugar “dentro” de ese host se encuentra el recurso que queremos.
  - `protocolo://host/ruta/nombreArchivo`
  - `http://www.fp.org/novedades/mayo/novedades.html`
- A veces se habla de URL (Uniform resource locator) en lugar de URI. La diferencia es que una URL no incluye la sección **#marca** que indica en qué lugar dentro del archivo se encuentra la información que buscamos. Es decir, URI es un concepto más amplio que URL y por eso lo utilizaremos.

# Conceptos (3/4) La web estática



# Conceptos (4/4) La web dinámica



# HTTP (1/2) Sintaxis básica petición

**Comando** **RutaAlRecurso** **Vers.HTTP**

**Encabezado1:** Valor1

....

**EncabezadoN:** ValorN

**Dato1:** valor1

.....

**DatoN:** valorN

<línea\_en\_blanco>

(tan sólo el encabezado “Host” es obligatorio en HTTP/1.1)

**GET** / **HTTP/1.1**

**Host:** www.sitio.com

**Connection:** keep-alive

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

**User-Agent:** Mozilla/5.0 (X11; Linux i686)  
AppleWebKit/537.36 (KHTML, like Gecko)

Chrome/37.0.2062.120 Safari/537.36

**Accept-Encoding:** gzip,deflate,sdch

**Accept-Language:** es,en-US;q=0.8,en;q=0.6

Más información en <http://code.tutsplus.com/tutorials/http-headers-for-dummies--net-8039>

# HTTP (2/2) Sintaxis básica respuesta

**Protocolo** **CódigoDeEstado** **Explicación**

**Encabezado1:** Valor1

....

**EncabezadoN:** ValorN

{línea en blanco}

**ContenidoDelEnvíoLínea1**

**ContenidoDelEnvíoLínea1**

...

**ContenidoDelEnvíoLíneaM**

**HTTP/1.1 200 OK**

**Date:** Sat, 27 Sep 2014 16:07:32 GMT

**Server:** Apache/2.4.10 (Unix) OpenSSL/1.0.1i

PHP/5.5.15 mod\_perl/2.0.8-dev Perl/v5.16.3

**Last-Modified:** Sat, 27 Sep 2014 15:55:17 GMT

**ETag:** "31-5040e0f588d87"

**Accept-Ranges:** bytes

**Content-Length:** 49

**Content-Type:** text/html

**<html><body><h1>**

**HOLA**

**</h1></body></html>**

Más información de headers en <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html>



# Solicitud de descargas: hipervínculos

- La forma más habitual mediante la que un navegador web solicita a un servidor la descarga de un recurso es la que se “dispara” cuando un usuario “pincha” un hipervínculo de una página web que el navegador ha dibujado y cuyo código es algo parecido a:
  - `<a href="http://host/ruta/archivo"> Texto hipervínculo </a>`
- Esto provocaría una conexión vía http a la máquina “host” en su puerto 80 (por defecto), para descargarse el “archivo” que se encuentra ubicado en la “ruta” indicada por debajo de su “document root”

# Contenido dinámico y estático

- El archivo del ejemplo anterior, suele ser un archivo “html”, en cuyo caso el navegador lo descargaría e inmediatamente lo dibujaría, en función del código contenido.
- Si queremos que la página web descargada varíe su contenido dependiendo de ciertas circunstancias necesitamos que su contenido no sea estático (HTML), sino que se regenere cada vez que se invoca su descarga (PHP, ASP, JSP, etc...)
- Sin embargo, si queremos que estas circunstancias cambien en función de aspectos que ocurren en el lado cliente y no en el lado servidor sólo, el cliente necesita algún mecanismo para poder pasarle información al servidor justo en el momento que hace la petición de descarga del recurso.
- La forma más habitual de conseguir esto es mediante el uso de formularios

# Formulario básico

```
<form action="http://host/rutaArch/" method="get">
  <input type="text" name="PARAM1" >
  .....
  <input type="text" name="PARAMn" >
  <input type="submit" value="Enviar" />
</form>
```

- Pintaría un formulario con “n” cuadros de texto donde el usuario puede escribir lo que quiera utilizando el teclado.



Diagrama de un formulario web. Hay dos campos de texto. El primer campo tiene el texto "dato1" escrito en él. El segundo campo tiene el texto "datoN" escrito en él. A la derecha de los campos hay un botón con el texto "Enviar".

- Cuando se pulsa el botón “Enviar” el navegador construye y descarga la siguiente URI.

`http://host/rutaArch/?PARAM1=dato1 & ... & PARAMn = datoN`

- Donde dato1 ... datoN son los datos reales introducidos por el cliente en el formulario web

# Envío de parámetros GET (HTTP)

```
<form
  name="miformulario"
  action="http://www.sitio.com/prueba.php"
  method="get"
>
    <input type="text" name="nombre" >
    <input type="submit" value="Enviar" />
</form>
```

Pepe

Enviar



**GET** /prueba.php?nombre=Pepe HTTP/1.1

**Host:** www.sitio.com

**Connection:** keep-alive

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

**User-Agent:** Mozilla/5.0 (X11; Linux i686)  
AppleWebKit/537.36 (KHTML, like Gecko)

Chrome/37.0.2062.120 Safari/537.36

**Accept-Encoding:** gzip, deflate, sdch

**Accept-Language:** es,en-US;q=0.8,en;q=0.6

Los parámetros enviados por método GET son visibles en la barra de navegación del navegador.  
Es el método por defecto en un formulario y el único posible en hipervínculos <a>

# Envío de parámetros POST (HTTP)

```
<form
  name="miformulario"
  action="http://www.sitio.com/prueba.php"
  method="post"
>
    <input type="text" name="nombre" >
    <input type="submit" value="Enviar" />
</form>
```

Pepe

Enviar



**POST** /prueba.php HTTP/1.1

**Host:** www.sitio.com

**Connection:** keep-alive

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

**User-Agent:** Mozilla/5.0 (X11; Linux i686)

AppleWebKit/537.36 (KHTML, like Gecko)

Chrome/37.0.2062.120 Safari/537.36

**Accept-Encoding:** gzip, deflate, sdch

**Accept-Language:** es,en-US;q=0.8,en;q=0.6

**Content-Type:** application/x-www-form-urlencoded

**Content-Length:** 11

nombre=Pepe

# Variables superglobales en PHP

Se trata de un conjunto de variables predefinidas y accesibles desde cualquier ámbito (funciones, clases o archivos).

- **\$GLOBALS**: contiene todas las variables globales que vienen a continuación.
- **\$\_SERVER**: contiene las variables del servidor Web (cabeceras, rutas, etc.)
- **\$\_POST**: contiene los datos enviados en un formulario HTML con method="post"
- **\$\_GET**: contiene los datos enviados en un formulario HTML con method="get"
- **\$\_COOKIE**: contiene las variables proporcionadas por cookies
- **\$\_REQUEST**: contiene todos los datos enviados (GET + POST + COOKIES)
- **\$\_FILES**: contiene variables proporcionadas por medio de ficheros
- **\$\_ENV**: contiene las variables proporcionadas por el entorno
- **\$\_SESSION**: contiene las variables registradas en la sesión del script

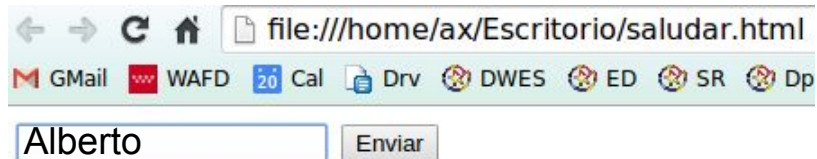
# Contenido de \$\_SERVER

MIBDIRS	C:/xampp/php/extras/mibs
MYSQL_HOME	\\xampp\\mysql\\bin
OPENSSL_CONF	C:/xampp/apache/bin/openssl.cnf
PHP_PEAR_SYSCONF_DIR	\\xampp\\php
PHPRC	\\xampp\\php
TMP	\\xampp\\tmp
HTTP_HOST	localhost
HTTP_CONNECTION	keep-alive
HTTP_ACCEPT	text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
HTTP_UPGRADE_INSECURE_REQUESTS	1
HTTP_USER_AGENT	Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/45.0.2454.101 Safari/537.36
HTTP_ACCEPT_ENCODING	gzip, deflate, sdch
HTTP_ACCEPT_LANGUAGE	es,en-US;q=0.8,en;q=0.6
PATH	C:\ProgramData\Oracle\Java\javapath;C:\GTK\bin;C:\oracle\app\oracle\product\11.2.
SystemRoot	C:\Windows
COMSPEC	C:\Windows\system32\cmd.exe
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
WINDIR	C:\Windows
SERVER_SIGNATURE	Apache/2.4.10 (Win32) OpenSSL/1.0.1h PHP/5.4.31 Server at localhost Port 80
SERVER_SOFTWARE	Apache/2.4.10 (Win32) OpenSSL/1.0.1h PHP/5.4.31
SERVER_NAME	localhost
SERVER_ADDR	:::1
SERVER_PORT	80
REMOTE_ADDR	:::1
DOCUMENT_ROOT	C:/xampp/htdocs
REQUEST_SCHEME	http
CONTEXT_PREFIX	
CONTEXT_DOCUMENT_ROOT	C:/xampp/htdocs
SERVER_ADMIN	postmaster@localhost
SCRIPT_FILENAME	C:/xampp/htdocs/PruebasPHP/kl/variablesServer.php
REMOTE_PORT	52215
GATEWAY_INTERFACE	CGI/1.1
SERVER_PROTOCOL	HTTP/1.1
REQUEST_METHOD	GET
QUERY_STRING	
REQUEST_URI	/PruebasPHP/kl/variablesServer.php
SCRIPT_NAME	/PruebasPHP/kl/variablesServer.php
PHP_SELF	/PruebasPHP/kl/variablesServer.php
REQUEST_TIME_FLOAT	1443606456.796
REQUEST_TIME	1443606456

# Recepción de parámetros en PHP (GET)

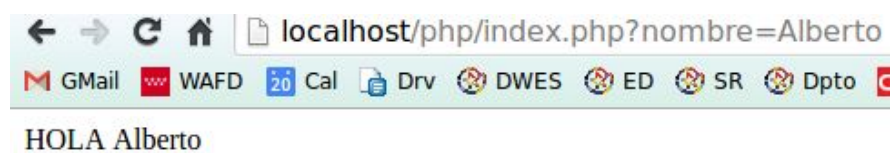
## saludar.html

```
<form action="http://localhost/php/index.php">
<input type="text" name="nombre">
<input type="submit">
</form>
```



## php / index.php

```
<?php
    $nombre = $_GET["nombre"];
    echo "HOLA $nombre";
?>
```

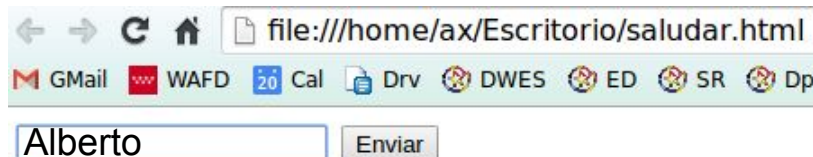




# Recepción de parámetros en PHP (POST)

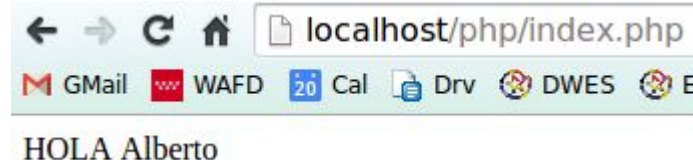
## saludar.html

```
<form action="http://localhost/php/index.php"  
method="post">  
<input type="text" name="nombre">  
<input type="submit">  
</form>
```



## php / index.php

```
<?php  
    $nombre = $_POST["nombre"];  
    echo "HOLA $nombre";  
?>
```



# Codificación URL

- Cuando el usuario introduce datos en un campo de texto o similar, puede introducir cualquier tipo de caracteres: eñes, tildes, espacios, etc...
- Estos caracteres se tienen que enviar posteriormente a través de una URL o URI y muchos de ellos pueden dar problemas, por eso antes de ser enviados se codifican de una forma especial (UTF-8), utilizando sólo caracteres ASCII. A continuación algunos ejemplos:
  - **Espacio** (+)
  - **á** (%C3%A1) **é** (%C3%A9) **í** (%C3%AD) **ó** (%C3%B3) **ú** (%C3%BA)
  - **Á** (%C3%81) **É** (%C3%89) **Í** (%C3%8D) **Ó** (%C3%93) **Ú** (%C3%9A)
  - **ñ** (%C3%B1) **Ñ** (%C3%91)
  - **[** (%5B) **]** (%5D) **{** (%7B) **}** (%7D) **¿** (%BF) **?** (%3F)
  - **/** (%2F) **%** (%25) **=** (%3D) **+** (%2B) **&** (%26)

# Etiqueta <form>

- Declara la existencia de un formulario
- No se enviará ningún dato de ningún campo (input, select, etc.) al servidor, que no se encuentre entre las etiquetas de apertura y cierre de un formulario.
- Atributos:
  - **method**: el método de envío; GET por defecto, o POST
  - **action**: el programa en el servidor que recibirá los datos
  - **enctype**: el tipo de codificación utilizado:
    - “application/x-www-form-urlencoded” por defecto.
    - “multipart/form-data” cuando se quieren enviar ficheros al servidor.

# Etiqueta <input>

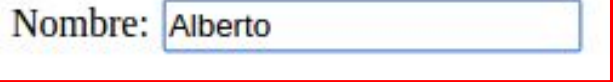
- Un elemento que sirve para que el usuario introduzca información que será enviada al servidor posteriormente (cuando se pulse el botón “submit” del formulario)
- Atributos
  - **type**: el tipo de control. Pueden ser:
    - **text**: una caja de texto
    - **password**: una caja de texto con asteriscos
    - **hidden**: una caja de texto no visible
    - **radio**: un botón de radio
    - **checkbox**: una caja de chequeo
    - **file**: un archivo
    - **reset**: un botón de reseteo. Se borran todos los datos del formulario.
    - **submit**: un botón de envío
    - **image**: una imagen (de envío)
    - OTROS: **color**, **date**, **datetime-local**, **email**, **month**, **number**, **range**, **search**, **tel**, **time**, **url**, **week**
  - **name**: nombre del control, imprescindible para que se envíe el dato al formulario
  - **value**: el valor por defecto del campo. En el caso de “check” o “radio” el valor sustituto de “on”
  - **readonly**: el control es de sólo lectura, se envía al servidor
  - **disabled**: el control está deshabilitado, no se envía al servidor
  - OTROS: **size**, **maxlength**, **checked**, **src**, **alt**, **required**

# Etiqueta <label>

- Asocia un texto a un elemento del formulario
- El vínculo entre el “label” y el elemento al que etiqueta se hace a través del atributo “for” del “label” cuyo valor ha de coincidir con el valor del “id” del elemento.
- Al estar vinculados, basta con señalar el label para editar el campo de texto (en el siguiente ejemplo), o bien para seleccionar un checkbox o un radio sin tener que “pinchar” sobre ellos.

```
<label for="nombreDePila">Nombre: </label>
```

```
<input type="text" id="nombreDePila" name="nombre" value="Alberto"/>
```



Nombre:

# Etiqueta <select>

- Control de selección (desplegable)
- Contiene elementos <option>, que tienen un atributo “value” y pueden aparecer pre-seleccionados con el atributo “selected”.
- Atributos destacados
  - **name**: nombre del control, imprescindible para que se envíe el dato al formulario. Si el select es múltiple, se pueden “empaquetar” en un solo parámetro si éste lleva un nombre terminado en corchetes [] (igual que en otros controles de selección múltiple como los checkbox)
  - **multiple**: indica que se podrán seleccionar varias opciones (siguiente diapositiva)
  - **size**: número de opciones visibles cuando el control es múltiple (siguiente diapositiva)

```
<form name="f1">
```

```
  <label for="ids">Aficiones</label> <select id="ids" name="aficiones">
```

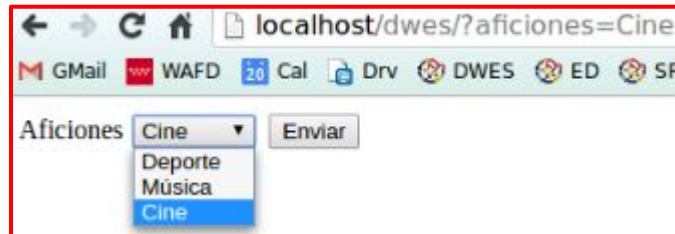
```
    <option value="Deporte">Deporte</option>
```

```
    <option value="Musica">Música</option>
```

```
    <option value="Cine" selected="selected">
```

```
      Cine</option>
```

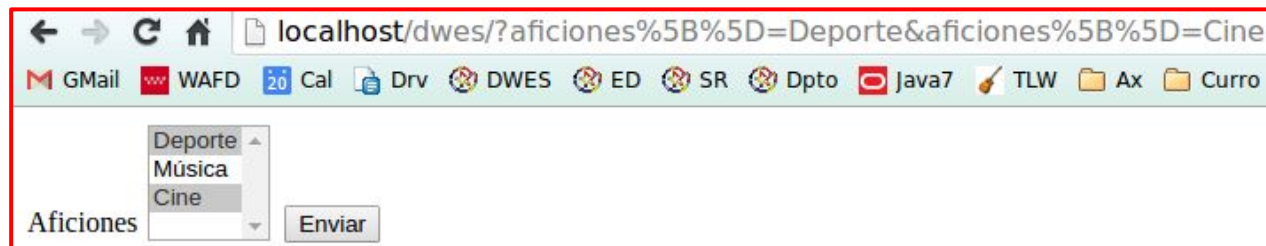
```
</select> <input type="submit" /> </form>
```



# Etiqueta <select> múltiple

- Cuando se insertan controles con selección múltiple es fundamental que el atributo “name” incluya [] para que se interprete como un array de valores al enviarse.

```
<form name="f1">
  <label for="ids">Aficiones</label>
  <select multiple="multiple" id="ids" name="aficiones[]">
    <option value="Deporte" selected="selected">Deporte</option>
    <option value="Musica">Música</option>
    <option value="Cine" selected="selected">Cine</option>
  </select>
  <input type="submit" />
</form>
```



# Etiqueta <textarea>

- Declara la existencia de un control de área de texto. Es como un input de tipo “text” pero más grande.
- Atributos:
  - **name**: nombre del control, imprescindible para que se envíe el dato al formulario
  - **cols**: número de columnas del área de texto
  - **rows**: número de filas del área de texto
  - **readonly**: el control es de sólo lectura, se envía al servidor
  - **disabled**: el control está deshabilitado, no se envía al servidor

```
<form name="f1">  
<input type="text" value="peque&ntilde;o" />  
<textarea cols="15" rows="3"></textarea>  
</form>
```



pequeño

aquí  
cabe  
más información



# Etiqueta <fieldset>

- Agrupa visualmente algunos elementos de un formulario, dibujando un recuadro alrededor de todos ellos.
- Se complementa con la etiqueta <legend>título</legend> que le da título al recuadro.

```
<form name="f1">
```

```
<fieldset>
```

```
  <legend>Campos dentro del fieldset</legend>
```

```
    <input type="text" value="dentro1"/>
```

```
    <input type="text" value="dentro2"/>
```

```
</fieldset>
```

```
<label for="idfuera">Campo fuera del fieldset</label>
```

```
<input type="text" id="idfuera" value="fuera"/>
```

```
<input type="submit"/>
```

```
</form>
```

Campos dentro del fieldset

dentro1 dentro2

Campo fuera del fieldset fuera Enviar

# Envío de ficheros al servidor (1/2)

Se utiliza la variable superglobal `$_FILES['nombre_Param_archivo']`, que contiene información fija para cada archivo subido, en un array asociativo bidimensional, donde “info” puede valer:

- *name*: nombre del archivo
- *size*: tamaño del archivo
- *type*: tipo del archivo
- *tmp\_name*: nombre que tendrá el archivo temporal
- *error*: estado de la subida (0: OK; > 0: error)

Los archivos inicialmente son subidos a una carpeta temporal (definida en `upload_tmp_dir` del `php.ini`), con un nombre del tipo `phpXXX.tmp`.

El archivo será eliminado posteriormente, por eso es preciso copiarlos en otra carpeta para poder conservarlos.

Esto se hace con la función `copy(“ruta_archivo_origen”, “ruta_archivo_destino”)`.

Se puede limitar el tamaño máximo a enviar desde el servidor mediante la directiva `upload_max_filesize = 5M` escrita en el archivo de configuración `php.ini`, o bien en el cliente con un campo oculto del tipo `<input type="hidden" name="MAX_FILE_SIZE" value="5000000" />`

# Envío de ficheros al servidor (2/2)

- Código en cliente (subir.html)

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

- Código en servidor (subirF.php)

```
$nombre = $_FILES ['nomF'] ['name'];  
$carpeta = "/tmp/"; //Ruta REAL en el disco. Debe tener "apache" permiso de escritura en ella  
copy ( $_FILES ['nomF'] ['tmp_name'], $carpeta . $nombre );  
echo "El fichero $nombre se almacenó en $carpeta";
```

# Funciones `isset()` y `empty()`

- Si no queremos errores, warnings o notices en el lado servidor al intentar procesar parámetros que no se han recibido o que no tienen ningún valor, es útil manejar estas dos funciones.
  - `isset($_REQUEST['nombreVar'])` devuelve true, si se ha enviado algún parámetro de nombre “nombreVar”, y éste no es null. false en caso contrario.
    - P.ej `script.php?otroNombreVar=Pepe` devolvería false
  - `empty($_REQUEST['nombreVar'])` devuelve true, si no se ha enviado ningún valor para el parámetro de nombre “nombreVar”. false en caso contrario.
    - P.ej `script.php?nombreVar=` devolvería true.

# Trucos (1/2) trabajar con formularios

- Asegurarse de que cada elemento que envíe datos simples tenga un **“name”** **distinto**
- Recordar que el campo **value** y el **valor** de la etiqueta no tienen por qué coincidir (p.ej. `<radio name="r" value="uno"> ONE </radio>`), enviaría al servidor el valor `r=uno`, pero mostraría en el navegador la etiqueta **“ONE”** al lado del radio button.
- Asegurarse de que los elementos `<radio>` que queremos que sean **exclusivos** tengan el mismo “name”, y que uno de ellos esté `checked="checked"` si no queremos enviar un campo vacío por error.
- Asegurarse de que los `<select multiple>` y los `<checkbox>` tengan un name del estilo “nombre[]” para que envíen todos sus valores y en el servidor los podamos recorrer con un “foreach”. Si no, tan sólo se reconocerá el último, y además un “foreach” en el lado servidor fallará porque la variable no se corresponderá con un array.
- `isset($_REQUEST['dato'])` es la expresión correcta para preguntar si se nos ha enviado algún dato múltiple en el parámetro cuyo nombre es `name="dato[]"` en el formulario
- Es bueno habituarse a utilizar el **atributo “id”** en todos los elementos, aunque sólo son imprescindibles en aquéllos en los que se necesite acceder vía `getElementById`, o se quieran formatear vía CSS de forma especial, o cuando se quiera asociar un `<label>` a alguno de ellos.

# Trucos (2/2) Trabajando con URI's

- La función **url\_encode(\$cadena)** convierte una cadena con caracteres especiales (tildes, espacios, eñes) en una cadena con caracteres ASCII estándar, codificados en UTF-8 (como los que van en las URI)
- La función **url\_decode(\$cadena)** interpreta los códigos que van en las URI y los convierte en sus caracteres especiales correspondientes.
- Si el método de envío es GET, la variable **\$\_SERVER['QUERY\_STRING']** contiene la cadena de caracteres de la URI a partir del signo de interrogación
  - ¡¡Ojo!! El contenido de esta cadena viene sin decodificar, no así como los datos de los `$_REQUEST`, `$_GET`, `$_POST`, etc...
- La función **parse\_str(\$cadena)** interpreta una cadena con formato de query string y crea variables para cada uno de los parámetros indicados en la cadena con sus datos correspondientes.

# Utilizando XDebug

- Para poder depurar un programa PHP podemos utilizar el módulo XDebug de la siguiente manera.
- Descargando la última versión desde <http://xdebug.org/wizard.php>
- Ubicando la dll descargada (si Windows. Si Linux hay que compilar el módulo) en el directorio c:\xampp\php\ext o equivalente
- Añadiendo o cambiando las siguientes líneas del "php.ini, y rearrancar Apache

```
zend_extension = "C:\xampp\php\ext\php_xdebug-<VERSION_ACTUAL>.dll"
xdebug.remote_enable = On
xdebug.remote_handler = "dbgp"
xdebug.remote_host = "127.0.0.1"
xdebug.remote_port = 9000
```
- Indicando a Eclipse que utilice XDebug (tanto en ejecución servidor como CLI)
  - Window => Preferences => PHP => Servers =>Default (edit) escogiendo XDebug
  - Window => Preferences => PHP => Installed PHPs => CLI (edit) escogiendo XDebug

# Referencias

- [Referencia y tutoriales HTML \(W3schools\)](#)
- [Guía de referencia HTML \(W3C\)](#)
- [Referencia y tutoriales CSS \(W3schools\)](#)
- [Referencia y tutoriales PHP W3schools](#)
- [Referencia y tutoriales PHP.net](#)