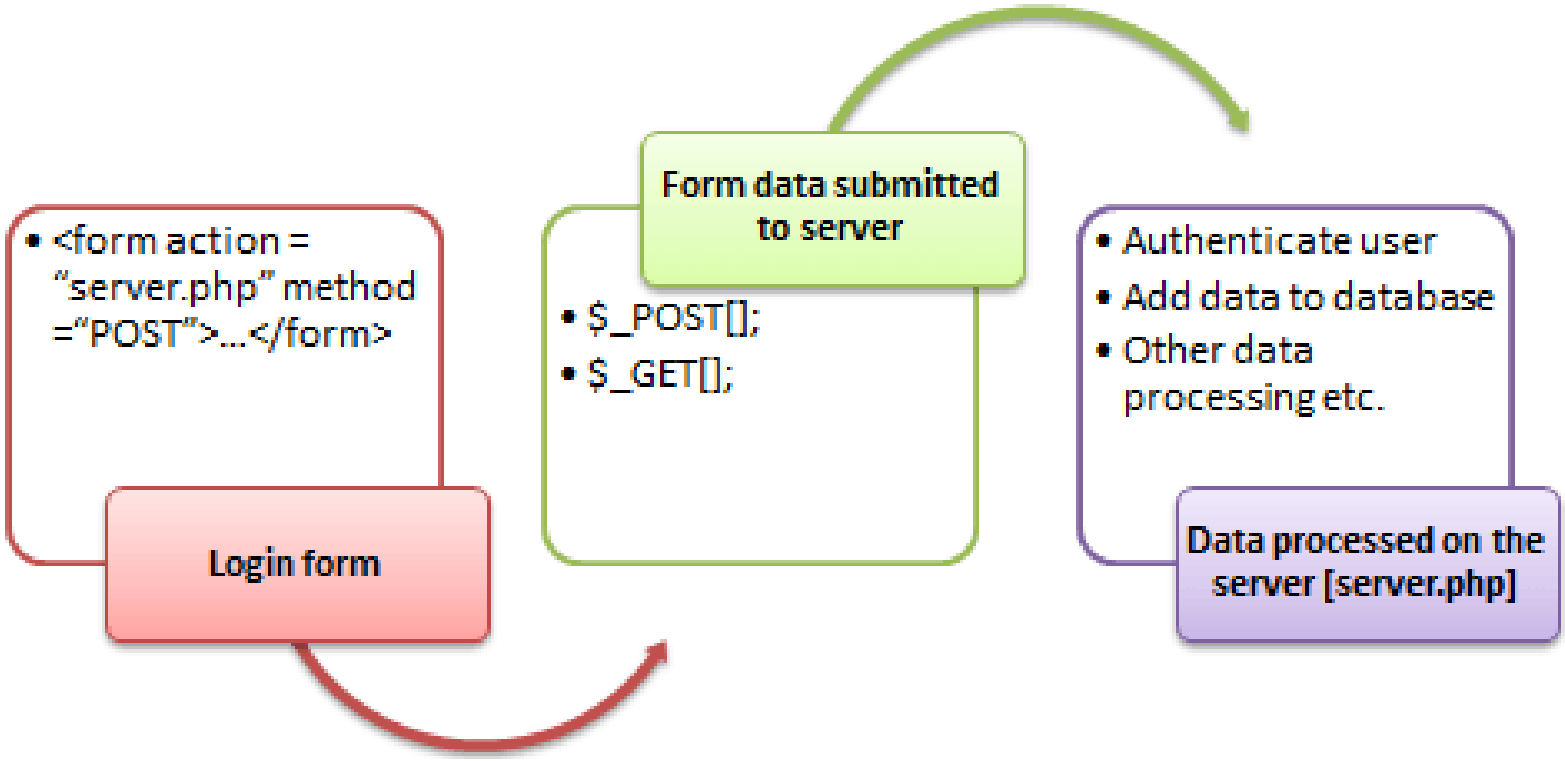
Tema 6: Formularios

1. Acceso a formularios HTML desde PHP
2. El formulario de PHP
3. Subida de archivos al servidor
4. Validación de los datos de un formulario


Web server
Client side validation script
Form submission data
Form processor script
- do validations
- send email
- save data
Email
Response page
Database


• <form action = "server.php" method ="POST">...</form>
Login form
Form data submitted to server
• $_POST[];
• $_GET[];
• Authenticate user
• Add data to database
• Other data processing etc.
Data processed on the server [server.php]

Acceso a formularios desde PHP

- Desde PHP se puede acceder fácilmente a los datos introducidos desde un formulario HTML
- Veámoslo con un ejemplo simple

Acceso a formularios desde PHP

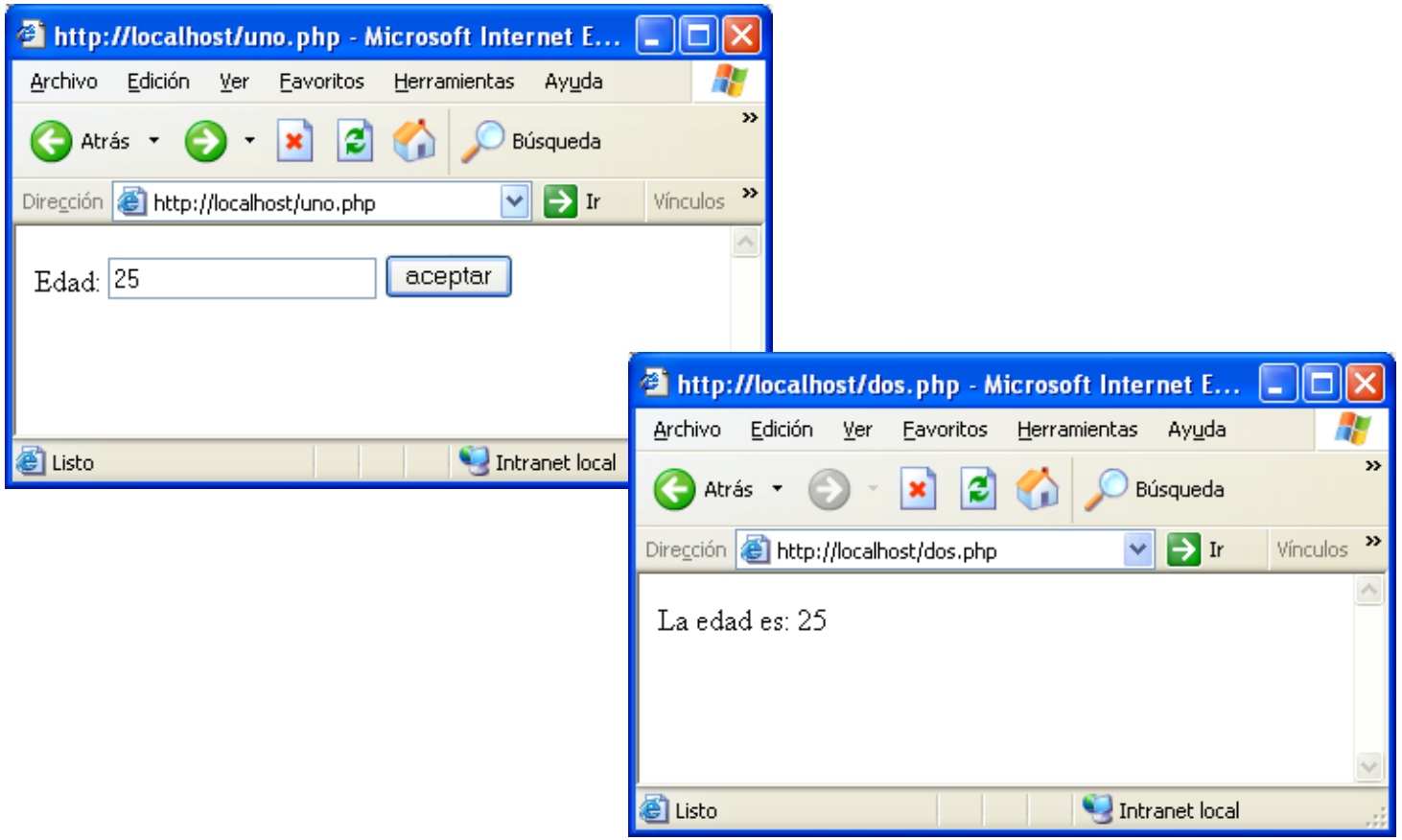
- archivo uno.php

```
<HTML>
<BODY>
<FORM ACTION=”dos.php” METHOD=”POST”>
   Edad: <INPUT TYPE=”text” NAME=”edad”>
   <INPUT TYPE=”submit” VALUE=”aceptar”>
</FORM>
</BODY>
</HTML>
```

- archivo dos.php

```
<HTML>
<BODY>
<?PHP
   print (“La edad es: $edad”);
?>
</BODY>
</HTML>
```

Acceso a formularios desde PHP



Acceso a formularios desde PHP

- A partir de PHP 4.2.0, el valor por defecto de la directiva de PHP **register_globals** es off
- Esto tiene una gran importancia sobre los formularios, ya que no es posible acceder a las variables enviadas de la manera anterior (como variables globales). En su lugar hay que utilizar la variable predefinida de PHP **$_REQUEST**, escribiendo $_REQUEST[‘edad’] en lugar de $edad
- Se puede poner register_globals = on en el archivo de configuración php.ini, pero no es recomendable por motivos de seguridad. Una alternativa que permite hacer mínimos cambios en el código ya existente es la siguiente:

```
$edad = $_REQUEST[‘edad’];
```

Acceso a formularios desde PHP

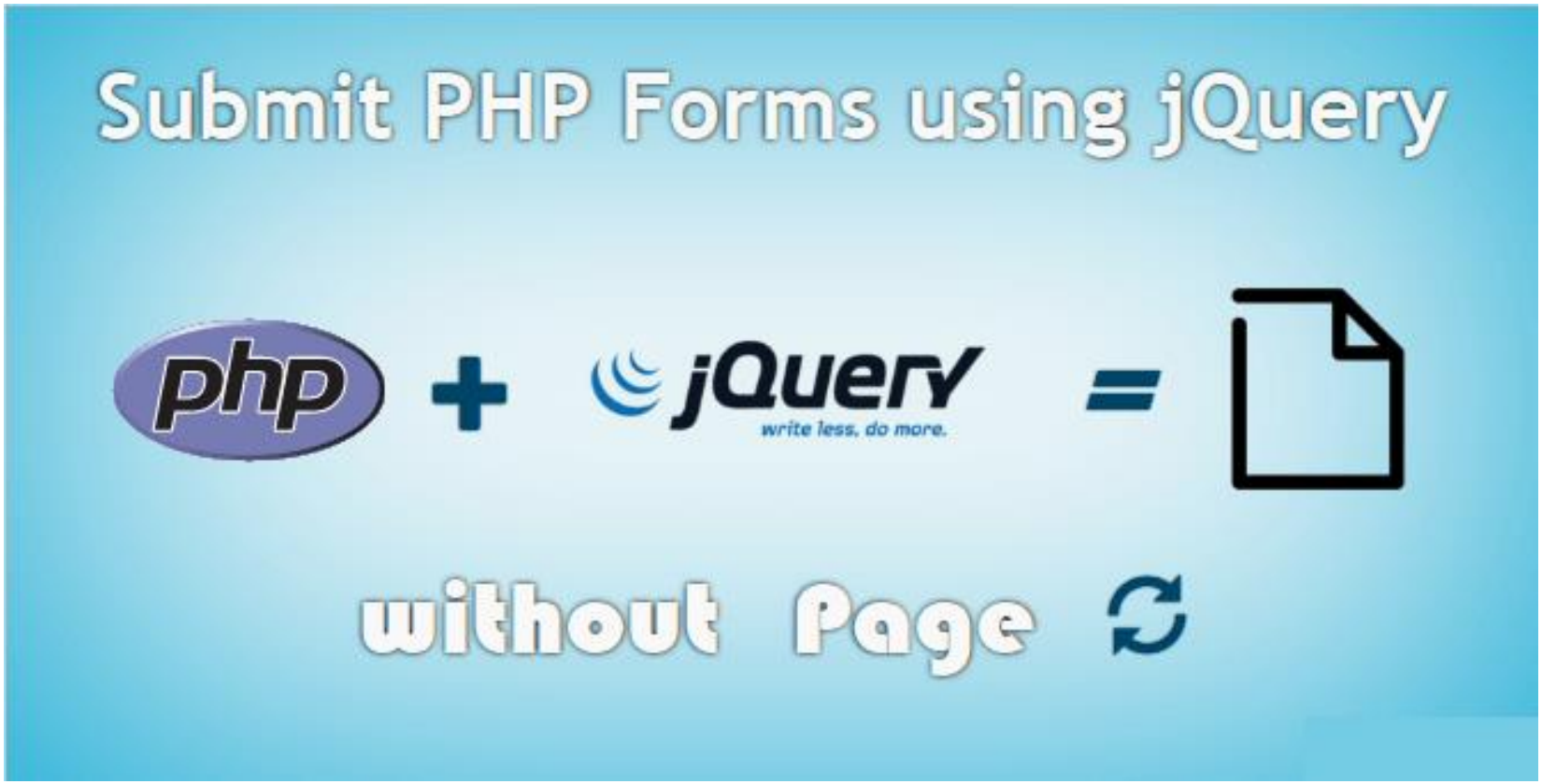
- archivo uno.php

```
<HTML>
<BODY>
<FORM ACTION=”dos.php” METHOD=”POST”>
   Edad: <INPUT TYPE=”text” NAME=”edad”>
   <INPUT TYPE=”submit” VALUE=”aceptar”>
</FORM>
</BODY>
</HTML>
```

- archivo dos.php

```
<HTML>
<BODY>
<?PHP
   $edad = $_REQUEST[‘edad’];
   print (“La edad es: $edad”);
?>
</BODY>
</HTML>
```

¿Y sin submit? (sin PostBack)



Acceso a formularios desde PHP

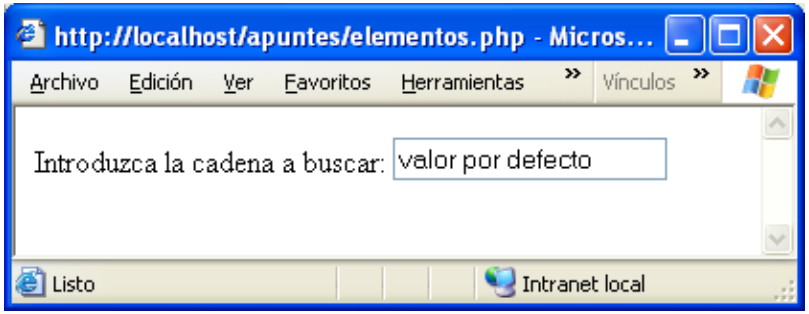
- Acceso a los diferentes tipos de elementos de entrada de formulario
  - Elementos de tipo INPUT
    - TEXT
    - RADIO
    - CHECKBOX
    - BUTTON
    - FILE
    - HIDDEN
    - PASSWORD
    - SUBMIT
  - Elemento SELECT
    - Simple / múltiple
  - Elemento TEXTAREA

Acceso a formularios desde PHP

- TEXT

```
Introduzca la cadena a buscar:
<INPUT TYPE="text" NAME="cadena" VALUE="valor por defecto" SIZE="20">


<?PHP
   $cadena = $_REQUEST['cadena'];
   print ($cadena);
?>
```

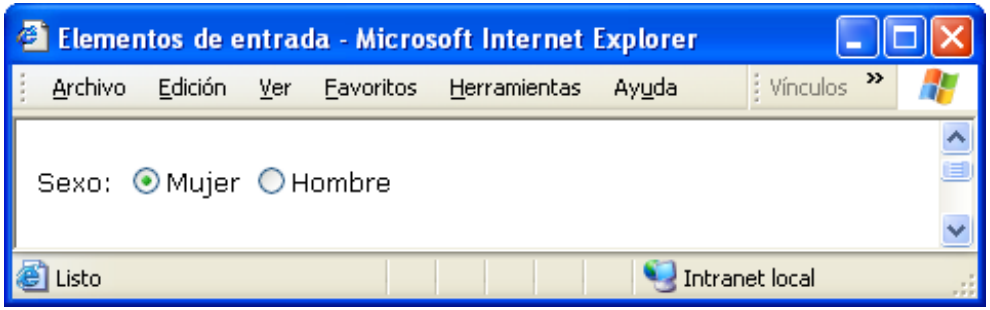


Acceso a formularios desde PHP

- RADIO

```
Sexo:
<INPUT TYPE="radio" NAME="sexo" VALUE="M" CHECKED>Mujer
<INPUT TYPE="radio" NAME="sexo" VALUE="H">Hombre


<?PHP
   $sexo = $_REQUEST['sexo'];
   print ($sexo);
?>
```



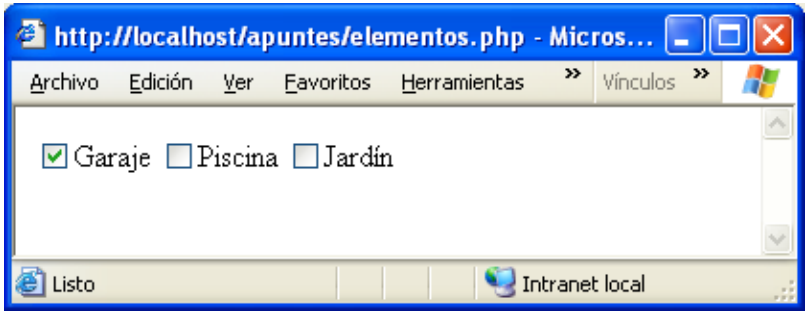
Acceso a formularios desde PHP

- CHECKBOX

```
<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 $extra)
      print ("$extra<BR>\n");
 ?>
```

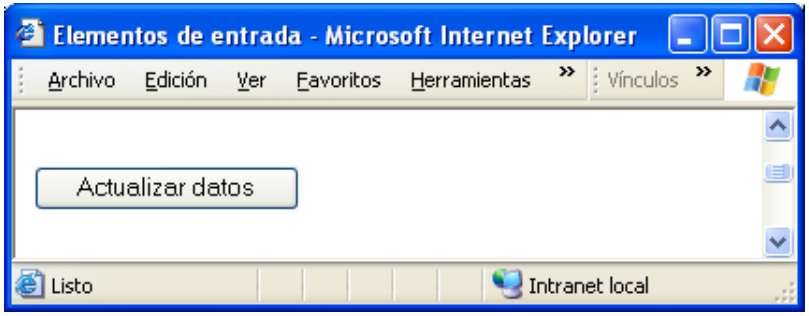


Acceso a formularios desde PHP

- BUTTON

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

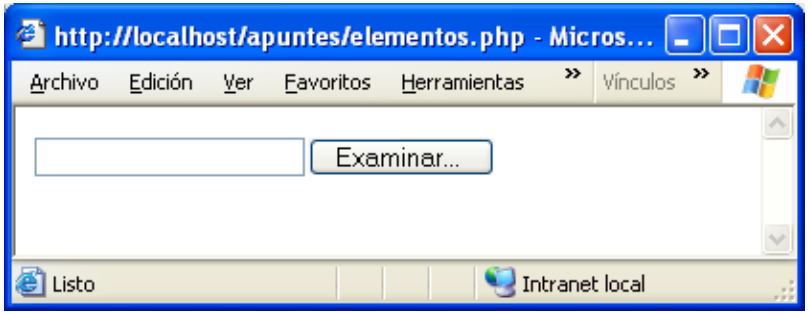
```
<?PHP
   $actualizar = $_REQUEST['actualizar'];
   if ($actualizar)
      print ("Se han actualizado los datos");
?>
```



Acceso a formularios desde PHP

- FILE

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

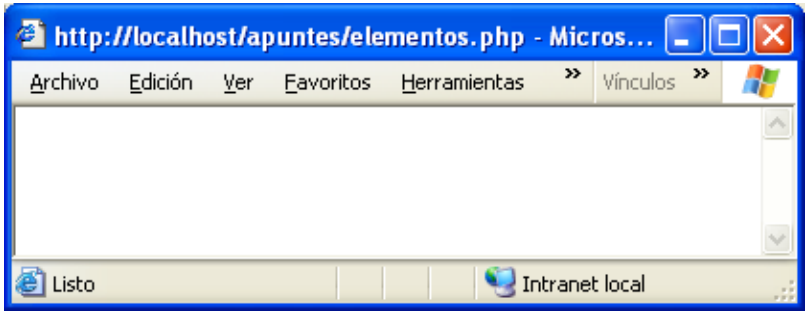


Acceso a formularios desde PHP

- HIDDEN

```
<?PHP
   print("<INPUT TYPE='hidden' NAME='username' VALUE='$usuario'>\n");
?>
```

```
<?PHP
   $username = $_REQUEST['username'];
   print ($username);
?>
```

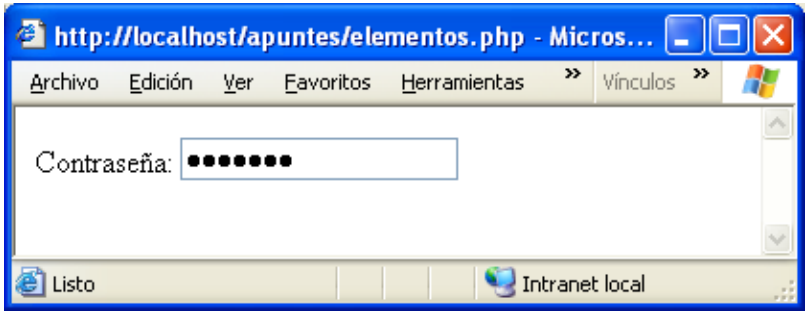


Acceso a formularios desde PHP

- PASSWORD

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

```
<?PHP
   $clave = $_REQUEST['clave'];
   print ($clave);
?>
```

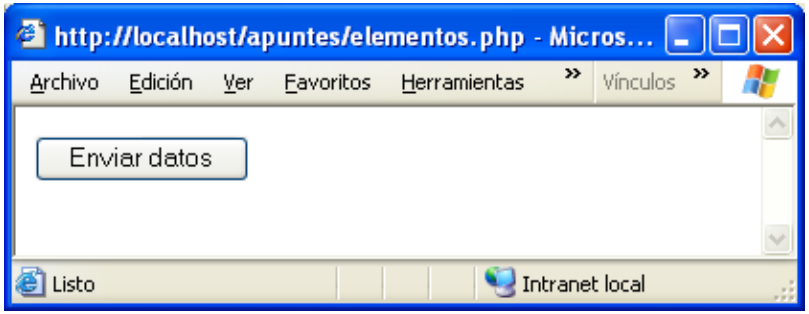


Acceso a formularios desde PHP

- SUBMIT

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

```
<?PHP
   $enviar = $_REQUEST['enviar'];
   if ($enviar)
      print ("Se ha pulsado el botón de enviar");
?>
```



Acceso a formularios desde PHP

- SELECT simple

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

```
<?PHP
   $color = $_REQUEST['color'];
   print ($color);
?>
```

Acceso a formularios desde PHP

- SELECT múltiple

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

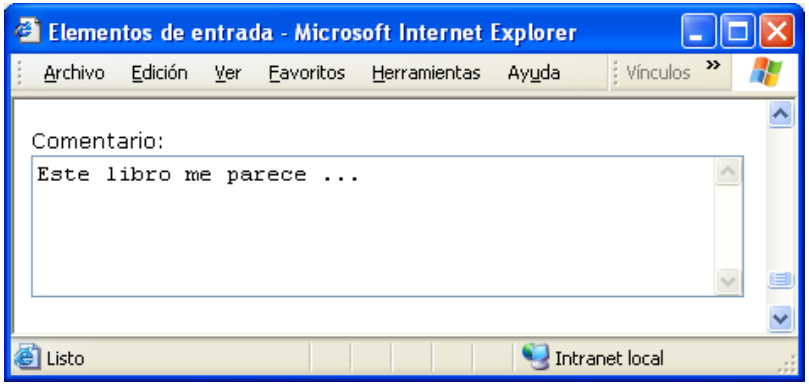
```
<?PHP
   $idiomas = $_REQUEST['idiomas'];
   foreach ($idiomas as $idioma)
      print ("$idioma<BR>\n");
?>
```

Acceso a formularios desde PHP

- TEXTAREA

```
Comentario:
<TEXTAREA COLS="50" ROWS="4" NAME="comentario">
Este libro me parece ...
</TEXTAREA>
```

```
<?PHP
   $comentario = $_REQUEST['comentario'];
   print ($comentario);
?>
```



El formulario de PHP

- La forma habitual 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 archivos
  - Permite validar los datos del formulario en el propio formulario
- Procedimiento:

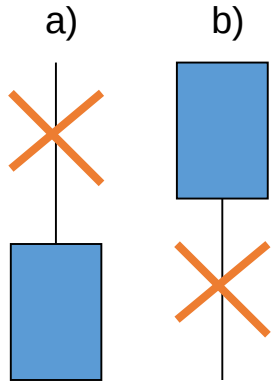
```
si se ha enviado el formulario:
   Procesar formulario
si no:
   Mostrar formulario
fsi
```

El formulario de PHP

- Esquema de funcionamiento:

```
si se ha enviado el formulario:
   Procesar formulario
```

```
si no:
   Mostrar formulario
fsi
```



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

El formulario de 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($enviar))
```

o bien

```
if(array_key_exists('enviar', $_POST)){
```

Subida de archivos al servidor

- Para subir un archivo 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 archivo tiene un límite en cuanto a su tamaño. Este límite se fija de dos formas diferentes:
    - En el archivo de configuración php.ini
    - En el propio formulario

Subida de archivos al servidor

php.ini

```
;;;;;;;;;;;;;;;;
; File Uploads ;
;;;;;;;;;;;;;;;;
; Whether to allow HTTP file uploads.
file_uploads = On

; Temporary directory for HTTP uploaded files (will use
; system default if not specified).
;upload_tmp_dir =

; Maximum allowed size for uploaded files.
upload_max_filesize = 2M
```

formulario

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

Subida de archivos al servidor

- Consideraciones (cont)
  - Debe darse al archivo un nombre que evite coincidencias con archivos ya subidos. Por ello, y como norma general, debe **descartarse el nombre original** del archivo y crear uno nuevo que sea único
  - El archivo 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 archivo:
   Asignar un nombre al archivo
   Mover el archivo a su ubicación definitiva
si no:
   Mostrar un mensaje de error
fsi
```

Subida de archivos al servidor

HTML

```
<INPUT TYPE="HIDDEN" NAME="MAX_FILE_SIZE" VALUE="102400">
<INPUT TYPE="FILE" SIZE="44" NAME="imagen">
```

- La variable $_FILES contiene toda la información del archivo subido:
  - $_FILES['imagen']['name']
    - Nombre original del archivo en la máquina cliente
  - $_FILES['imagen']['type']
    - Tipo mime del archivo. Por ejemplo, "image/gif"
  - $_FILES['imagen']['size']
    - Tamaño en bytes del archivo subido
  - $_FILES['imagen']['tmp_name']
    - Nombre del archivo temporal en el que se almacena el archivo subido en el servidor
  - $_FILES['imagen']['error']
    - Código de error asociado al archivo subido

Subida de archivos al servidor

PHP

```php
if (is_uploaded_file ($_FILES['imagen']['tmp_name']))
{
   $nombreDirectorio = "img/";
   $idUnico = time();
   $nombrearchivo = $idUnico . "-" . $_FILES['imagen']['name'];

   move_uploaded_file ($_FILES['imagen']['tmp_name'],
      $nombreDirectorio . $nombrearchivo);
}
else
   print ("No se ha podido subir el archivo\n");
```

Subida de archivos al servidor

PHP

```php
if (is_uploaded_file ($_FILES['imagen']['tmp_name']))
{
   $nombreDirectorio = "img/";
   $nombrearchivo = $_FILES['imagen']['name'];

   $nombreCompleto = $nombreDirectorio . $nombrearchivo;
   if (is_file($nombreCompleto))
   {
      $idUnico = time();
      $nombrearchivo = $idUnico . "-" . $nombrearchivo;
   }

   move_uploaded_file ($_FILES['imagen']['tmp_name'],
      $nombreDirectorio . $nombrearchivo);
}
else
   print ("No se ha podido subir el archivo\n");
```

Validación de formularios

- 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
   fsi
si no:
   Mostrar formulario
fsi
```

Validación de formularios

- 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
fsi
si se ha enviado el formulario y no hay errores:
   Procesar formulario
si no:
   Mostrar formulario con valores por defecto o ya
   enviados
fsi
```

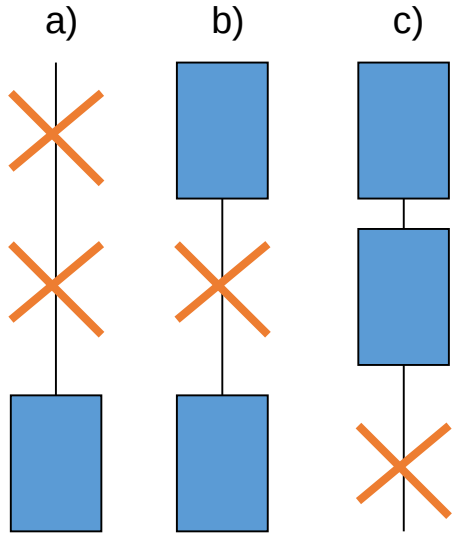
Validación de formularios

- Esquema de funcionamiento:

```
si se ha enviado el formulario:
   validar datos
fsi
```

```
si se ha enviado y no hay errores:
   Procesar formulario
```

```
si no:
   Mostrar formulario
fsi
```



  - 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)

Variables

- $_GET
- $_POST
- $_REQUEST

- Es muy común en los usuarios confundir estas dos variables ($_GET $_POST) y por ende darles un mal funcionamiento u utilización.
  ¿El porque de esta confusión? Pues porque con las dos obtienen el resultado, pero no tal vez se realice de la forma errónea.

  Las dos formas se utilizan como método de paso de datos en formularios y esta es una de las causas de la confusión.

- A continuación, sus diferencias.

$_GET

- Al utilizar el método GET la información puesta en el formulario es pasada al servidor a través de la url.
- Con lo cual toda la información queda visible ante el usuario.
- También puede ser guardada la url en marcadores con los datos enviados.
- Otra forma de pasar información al servidor mediante GET es directamente en un link.

```
<form method="get" action="get.php">
        <input type="text" value="" name="nombre" />
        <input type="submit" value="pasar datos" />
</form>
```

la_web/get.php?nombre=Enik

$_POST

- Al utilizar el método POST la información puesta en el formulario es pasada al servidor por debajo.
- Con lo cual toda la información no queda visible ante el usuario.
- Tampoco puede ser guardada la url en marcadores con los datos enviados.

```
<form method="post" action="post.php">
        <input type="text" value="" name="nombre" />
        <input type="submit" name="enviar" value="enviar datos" />
</form>
```

$_GET vs $_POST

- GET
  - Tiene un limite de caracteres al pasar datos.
  - Los datos son visibles.
  - Solicita información.

- POST
  - Se puede pasar grandes cantidades de datos.
  - Los datos no son visibles.
  - Envía información.

$_REQUEST

- La variable $_REQUEST es un array asociativo el cual contiene todos los datos pasados por GET POST y COOKIE.

- ¿Y para que usar $_REQUEST si podemos usar las variables $_GET $_POST Y $_COOKIE?

- Esta variable es muy útil en formularios multi-métodos

```
setcookie("mes", "octubre");
<form method="post" action="request.php?seccion=nombre">
        <input type="text" value="" name="nombre" />
        <input type="submit" name="enviar" value="enviar datos" />
</form>
```

```
$_REQUEST['seccion']
$_REQUEST['nombre']
$_REQUEST['mes']
```