

# Formularios y PHP

# Introducción I

- Desde PHP se puede acceder fácilmente a los datos introducidos desde un formulario HTML.

Fichero uno.php

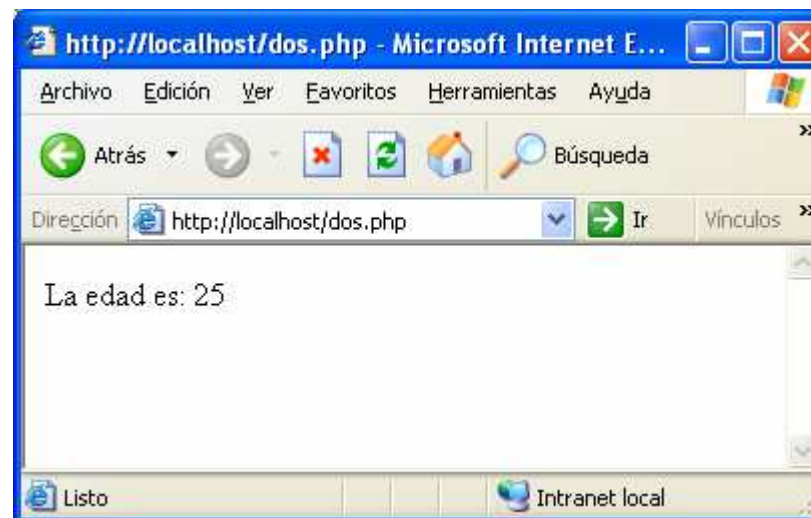
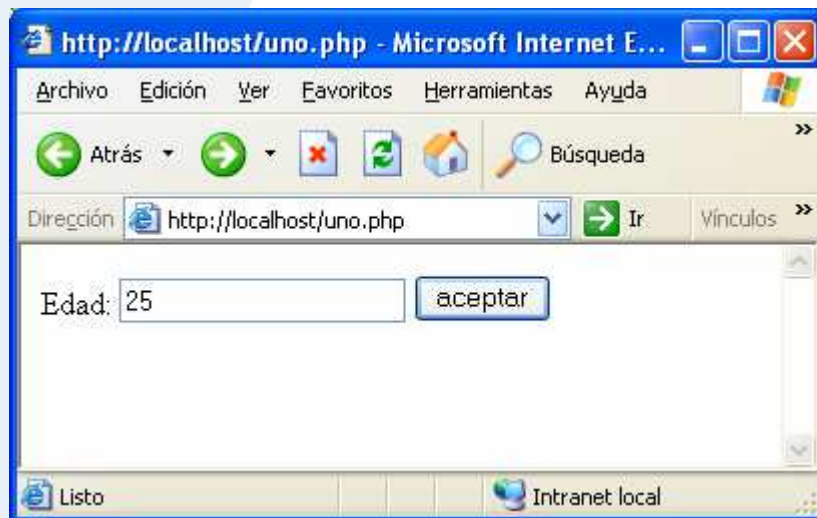
```
<HTML>
  <BODY>
    <FORM ACTION="dos.php" METHOD="POST">
      Edad: <INPUT TYPE="text" NAME="edad">
        <INPUT TYPE="submit" name="submit" VALUE="aceptar">
    </FORM>
  </BODY>
</HTML>
```

# Introducción II

Fichero dos.php

```
<HTML>
  <BODY>
    <?PHP
      $edad= $_POST['edad'];
      print("La edad es: $edad");
    ?>
  </BODY>
</HTML>
```

# Introduccion III



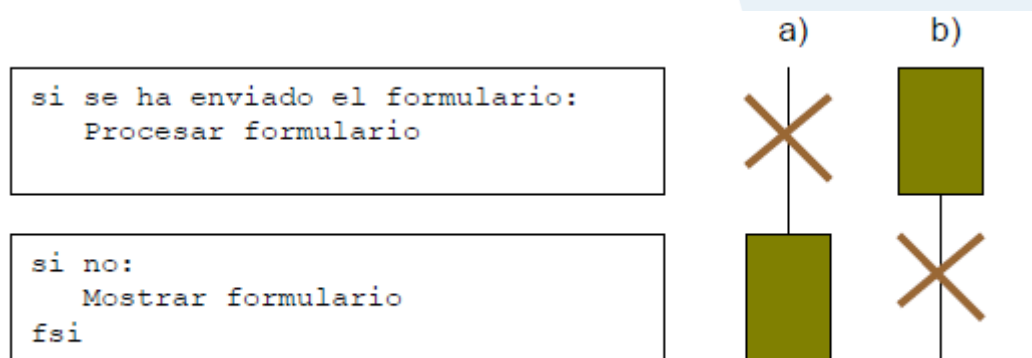
# Funcionamiento

- El fichero uno.php tiene un formulario, fijémonos en su ACTION:  
`<FORM ACTION="dos.php" METHOD="POST">`
- En ACTION se apunta al fichero que se redirecciona al pulsar el botón SUBMIT, o cualquier acción que ejecute un SUBMIT.  
P.E: mediante código javascript podemos hacer submit.
- El fichero dos.php recoge el valor mediante la propiedad POST:  
`$edad= $_POST['edad'];`

# Funcionamiento

- La forma habitual de trabajar con formularios en PHP es utilitzar un únic 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
  - Sino:  
Mostrar formulario

# Funcionamiento



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

# Funcionamiento

- Para saber si se ha enviado el formulario se acude a la variable correspondiente al botón de envío, normalmente submit:

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

entonces la condición anterior se transforma en:  
if(isset(\$enviar))

- Se pueden tener varios tipos SUBMIT con diferentes nombres y valores



# Ejemplo

- Transformar el ejemplo mostrado anteriormente a un solo fichero.
  - Uno.php y dos.php pasan a ser un único archivo.
  - si se han enviado los datos se procesan,
  - sino se muestra el formulario.

# OBTENER VALORES

- **`$_POST['nombre del campo']`**

Sirve para obtener el valor de los campos de la mayoría de elementos de un formulario: campos de texto, checkbox, radio button...

- **`$_GET['nombre del campo']`**

Sirve para obtener el valor de los campos que se pasan por referencia mediante url:

Ejemplo: <https://www.google.es/campo=perros>

# EJERCICIO 1

- Formulario con campo de texto que:
  - si se han enviado los datos se procesan,
  - sino se muestra el formulario.

## EJERCICIO 1.5

- Con el ejercicio anterior:
  - muestra la cadena que se envía (como antes).
  - Nos dice si todas las palabras de la cadena están encadenadas:

Bicicleta **tambor oruga** - > SI

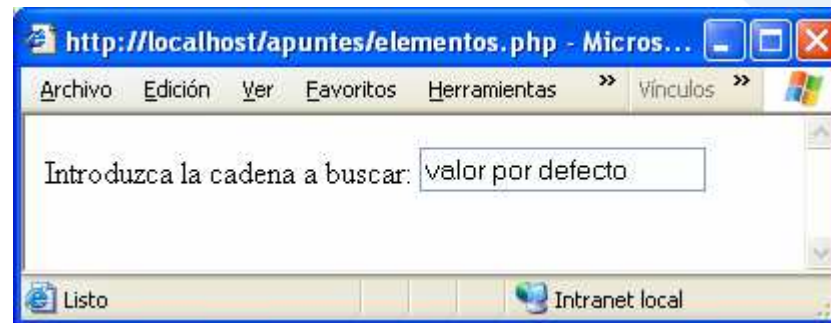
Hola que tal -> NO

# POST

## TEXT o NUMBER

Introduzca la cadena a buscar:

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



```
<?PHP
```

```
$cadena= $_POST['cadena'];  
print($cadena);
```

```
?>
```

# POST

## SUBMIT

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



```
<?PHP
```

```
$enviar= $_POST['enviar'];  
if($enviar)  
print("Se ha pulsado el botón de enviar");
```

```
?>
```

## EJERCICIO 2

### Calculadora1:

- Aprovecharemos que tenemos funciones definidas para hacer una calculadora que haga la suma de dos valores
- Formulario con dos campos de texto.
- 1 botón de envío de datos

# POST

## RADIO

Sexo:

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

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



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



# POST

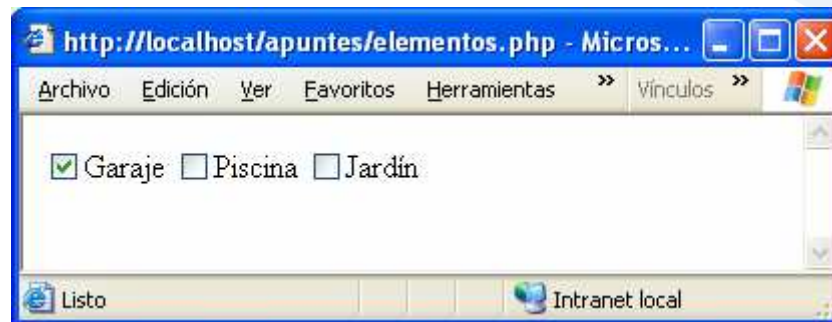
## CHECKBOX

```
<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 = $_POST['extras'];  
foreach($extras as $extra)  
    print("$extra<BR>\n");
```

```
?>
```

## EJERCICIO 3

### Calculadora2:

- Aprovecharemos que tenemos funciones definidas para hacer una calculadora que haga las diferentes operaciones de dos valores
- Formulario con dos campos de texto.
- Radio buttons con la operación a seleccionar
- 1 botón de envío de datos

# POST

## BUTTON

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



```
<?PHP
```

```
$actualizar= $_POST['actualizar'];  
if($actualizar)  
    print("Se han actualizado los datos");?
```

## EJERCICIO 4

### Ejercicio: Calculadora3:

- Aprovecharemos que tenemos funciones definidas para hacer una calculadora que haga las diferentes operaciones de dos valores
- Formulario con dos campos de texto.
- **4 botones de envío de datos**

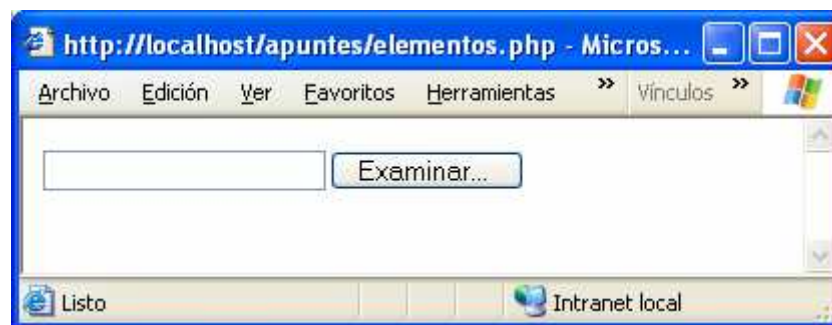
# POST

## FILE

```
<FORM ACTION="procesa.php" METHOD="post" ENCTYPE="multipart/form-data">
```

```
<INPUT TYPE="file" NAME="fichero">
```

```
</FORM>
```



# POST

## HIDDEN

INPUT TYPE='hidden' NAME='username' VALUE='\$usuario'



<?PHP

```
$username = $_POST['username'];  
print ($username);
```

?>

# POST

## PASSWORD

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



<?PHP

```
$clave = $_POST['clave'];  
print ($clave);
```

?>

## EJERCICIO 5

### LOGIN

- Mediante un campo de texto y uno de password, comprobaremos las credenciales de un usuario
- El usuario deberá ser “USER” y la password “PASSWORD”.
- Si se introducen correctamente devuelve un mensaje de **OK** en verde.
- Si se introducen incorrectamente devuelve un mensaje de **ERROR** en rojo.



# POST

## 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= $_POST['color'];  
  print($color);  
?>
```

# POST

## SELECT múltiple

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

?>



# POST

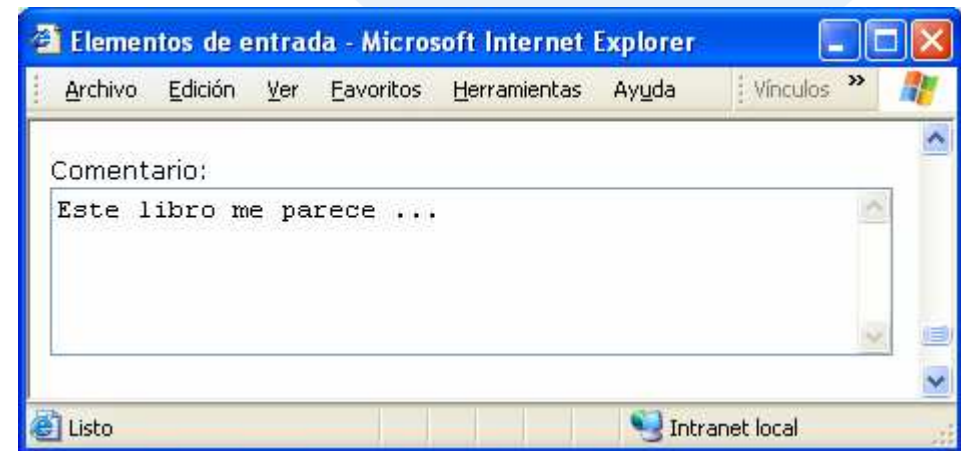
## TEXTAREA

Comentario:

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

Este libro me parece ...

```
</TEXTAREA>
```



```
<?PHP
```

```
$comentario= $_POST['comentario'];  
print($comentario);
```

```
?>
```

## EJERCICIO 6

### Sumatorio de un array

- Cargaremos 10 valores mediante un formulario
- Se guardaran en un array
- Tendremos que devolver el sumatorio Y el valor máximo y mínimo.
- Si teníamos una función previa que hacia esto **deberemos** utilizarla.

# COMENTARIOS PHP Y HTML

- Se debe ir con cuidado con los comentarios php y html a la vez ya que en el siguiente ejemplo el código php **se interpreta**:

```
<!-- comentario html  
con codigo php  
<?php mi codigo en php; ?>
```

de por medio

```
-->
```

- Si queremos mezclar comentarios la mejor manera de realizarlo es así:

```
<?php  
/*  
codigo html y codigo php, nada de aqui se interpreta  
*/  
?>
```

# MOSTRAR VARIABLES PHP EN HTML

Se pueden mezclar los códigos PHP con HTML o a la inversa para:

- Introducir un valor calculado con PHP en un campo HTML.
- Introducir cualquier texto que ha sido generado en PHP (funciones, cálculos,...)

DESDE **HTML** podemos:

```
<input type="number" id="res" name="res" value="<?php echo  
$resultat; ?>" />
```

O su alternativa más corta:

```
<input type="number" id="res" name="res" value="<?= $resultat; ?>"  
/>
```

# MOSTRAR VARIABLES PHP EN HTML

DESDE **PHP** podemos:

```
<?php
    echo "<input type=\"number\" id=\"res\" name=\"res\"
value=\"\${resultat}\" /> ";
?>
```

- Daremos por hecho que la variable `$resultat` contiene un valor calculado **anteriormente**.

# EJERCICIO 7

## Ejercicio: Calculadora4:

- Con cualquier versión de la calculadora anterior, introducir un campo de texto resultado que nos muestre el resultado calculado.



# Validación de formularios

- Toda la información que proviene de un formulario debe validarse antes de darla por buena y procesarla.
- Lo más eficiente es mostrar los errores sobre el propio formulario para facilitar su corrección:
  - 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

- Aún así, la manera más lógica de tratar con errores y validar formularios es con **Javascript**.
- De esta forma se evita enviar datos al servidor si no son correctos.
  - Incluir un button con un evento onClick en el formulario.
  - El código javascript hará sus comprobaciones. Si todo es correcto hará submit.

## EJERCICIO 8

**Dado un formulario con los campos:**

**Nombre \*: text**

**Apellidos : text**

**Edad: number**

**Email \*: text**

**Comentarios: textarea**

- Comprobar que los datos con asterisco son introducidos sino mostrar un error **junto al campo**.
- Si se ha introducido la edad (recordemos que es opcional ), debe ser mayor o igual de 18, sino mostrar un error **junto al campo**. Sino se ha introducido se debe saltar esta comprobación.
- Cuando se devuelve el formulario con o sin errores debe estar **rellenado para evitar que el usuario olvide que ha introducido**.

# GET

- Las variables también se pueden pasar mediante la url:

`<a href="destino.php?variable1=valor1&variable2=valor2&...">Mi enlace</a>`

`variable1=valor1`

- estas variables no tienen el símbolo \$ delante
- se usa el símbolo ? Para indicar que se empiezan a pasar variables
- se usa el símbolo & para separar más de una variable

PHP recoge estos valores mediante:

`$_GET['variable1'];`

# GET

## Fichero 1 origen:

```
<HTML>
  <HEAD>
    <TITLE>origen.html</TITLE>
  </HEAD>
  <BODY>
    <a href="destino.php?saludo=hola&texto=Esto es una
variable texto">Paso variables saludo y texto a la página
destino.php</a>

  </BODY>

</HTML>
```

# GET

## Fichero 2 destino:

```
<HTML>
  <HEAD>
    <TITLE>destino.php</TITLE>
  </HEAD>
  <BODY>
    <?php
      echo "Variable saludo: $_GET['saludo'] <br>";
      echo "Variable texto: $_GET[' texto'] <br>"
    ?>
  </BODY>
</HTML>
```

# GET

- Más inseguro de usar que POST:
  - Cualquiera puede ver el valor de la variable!
- Se suele usar en:
  - Identificadores de productos.
  - Elementos públicos de ámbito general: nombres de secciones de la web...
- **Nunca** se debe usar en:
  - Controles de acceso: nombres de usuario y contraseñas.

## EJERCICIO 9

### PLANTILLA HTML y DISEÑO DE MENUS

El objetivo es realizar una página lo más dinámica posible:

- Tendremos una plantilla web estándar **única**: un `<html>...</html>` simple.
- Tendremos un menú web con 3 secciones y 3 links:  
-P.E: `<a href="index.php?pagina=1">Link1</a>`
- Mediante variables GET recuperaremos el valor pasado y cargaremos una información u otra. (puede ser texto simple): un switch o estructura `if...else` nos facilitará la decisión de que información cargaremos.



## EJERCICIO 9

### PLANTILLA HTML y DISEÑO DE MENUS

- El título de la página **deberá cambiar**.
- El texto de la página **deberá cambiar**.
- OPCIONAL 1: Añadir un estilo distinto en el menú a la sección que se está visitando.
- OPCIONAL 2: Añadir una barra de estado para indicar en que sección nos encontramos.

# SUBIR FICHEROS

- 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:
    - En el fichero de configuración php.ini
    - En el propio formulario

# SUBIR FI

Si tenemos wamp como hosting local:  
ir a C:\wamp64\bin\php\php7.1.9

abrir el php.ini. Es el fichero donde esta toda la configuracion de PHP.  
Si buscamos por las palabras nos llevara a la linea

## En el fichero de configuración 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
```

```
; Temporary directory for HTTP uploaded files (will use  
; specified).  
; http://php.net/upload-tmp-dir  
upload_tmp_dir = "c:/wamp64/tmp"  
; Maximum allowed size for uploaded files.  
; http://php.net/upload-max-filesize  
upload_max_filesize = 2M
```

## En el propio formulario

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

# SUBIR FICHEROS

- **Consideraciones:**

Debe darse al fichero un nombre que evite coincidencias con ficheros ya subidos. Se suele descartar el nombre original y se crea un nuevo nombre, por ejemplo:

- identificador único.
- fecha y hora actual.

El fichero subido se almacena en un directorio temporal y se debe mover al directorio de destino usando la función `move_upload_file()`;

Para controlar que se ha subido bien se usa la función `is_uploaded_file(...)` que nos devolverá un booleano.

# SUBIR FICHEROS

- **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

fsi

# SUBIR FICHEROS

```
<INPUT TYPE="HIDDEN" NAME="MAX_FILE_SIZE" VALUE="102400">  
<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 la máquina cliente

**–\$\_FILES['imagen']['type'] //NO ES LA EXTENSIÓN**

Tipo mime del fichero. Por ejemplo, "image/gif"

**–\$\_FILES['imagen']['size']**

Tamaño en bytes del fichero subido

**–\$\_FILES['imagen']['tmp\_name']**

Nombre del fichero temporal en el que se almacena el  
fichero subido en el servidor

**–\$\_FILES['imagen']['error']**

Código de error asociado al fichero subido

# SUBIR FICHEROS

## Ejemplo 1:

```
If (is_uploaded_file($_FILES['imagen']['tmp_name']))  
{//si se ha subido el fichero....  
    $nombreDirectorio= "img/";  
    $idUnico= time();  
    $nombreFichero= $idUnico. "-" . $_FILES['imagen']['name'];  
  
    move_uploaded_file(  
        $_FILES['imagen']['tmp_name'],  
        $nombreDirectorio. $nombreFichero);  
  
}else  
    print("No se ha podido subir el fichero\n");
```

# SUBIR FICHEROS

## Ejemplo 2:

```
If (is_uploaded_file($_FILES['imagen']['tmp_name']))  
{ //si se ha subido el fichero....  
    $nombreDirectorio= "img/";  
    $nombreFichero= $_FILES['imagen']['name'];  
    $nombreCompleto= $nombreDirectorio. $nombreFichero;  
  
    if(is_file($nombreCompleto))  
    {  
        $idUnico= time();  
        $nombreFichero= $idUnico. "-" . $nombreFichero;  
    }  
    move_uploaded_file($_FILES['imagen']['tmp_name'],$nombreDirectorio.  
    $nombreFichero);  
}else  
    print("No se ha podido subir el fichero\n");
```

Se puede poner la fecha/hora como parte del fichero



# SUBIR FICHEROS

- **Ejercicio 10:** Descripción de un producto
  - Formulario que permita:
    - Nombre del producto
    - Descripción
    - Precio
    - Imagen del producto
    - Fecha actual (campo oculto: lo podemos añadir como hidden o no añadir en el form pero si debe aparecer en el resumen de la información)
  - Si se sube correctamente la información, se muestra por pantalla

# FUNCIONES DE FICHEROS

Una vez se han subido ficheros, deberíamos saber como acceder a ellos para futuras referencias y consultas

**opendir( path)** Abre la carpeta para trabajar en ella

**path** es la ruta de la carpeta que queremos abrir

Esta función nos abre la carpeta en cuestión para poder trabajar con ella y usar las siguientes funciones

Nos devuelve un elemento RESOURCE, no visto hasta ahora: es un **recurso externo** de php, independiente del lenguaje, en este caso representa una **CARPETA**

Permite leer la carpeta

readdir (opendir (.....))

se ha de correr dentro de un bloque por que lee uno a uno todos los elementos

# FUNCIONES DE FICHEROS

**readdir(recurso\_a\_carpeta)**

**recurso\_a\_carpeta** es una variable de tipo recurso que solemos recoger usando **opendir**.

Esta función devuelve el **nombre** de la siguiente entrada de un directorio, que puede ser una carpeta o un fichero, por orden en que fueron almacenadas. Esta función suele usarse en un **while**.

Dice si es un directorio o un fichero

**is\_dir( path/elemento)**

**path/elemento** es una ruta a un fichero en cuestión

Esta función nos dice si un elemento con la ruta indicada es una carpeta o no.

Por lo general intentaremos evitar trabajar con subcarpetas, que nos pueden dar mayor complicación a la hora de recorrer una carpeta general.

# FUNCIONES DE FICHEROS

**unlink( path/elemento)**

**path/elemento** es una ruta a un fichero en cuestión

Esta función **elimina un elemento permanentemente del sistema**

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

# FUNCIONES DE FICHEROS

```
function listarArchivos( $path ){  
    // Abrimos la carpeta que nos pasan como parámetro  
    $dir = opendir($path);  
    while ( $elemento = readdir($dir)){    // Leo todos los ficheros de la carpeta  
        // Tratamos los elementos . y .. que tienen todas las carpetas  
        if( $elemento != "." && $elemento != ".."){  
            if( is_dir($path.$elemento) ){    // Si es una carpeta  
                // Muestro la carpeta  
                echo "<p><strong>CARPETA: ". $elemento."</strong></p>";  
            } else {    // Si es un fichero  
                echo "<br />". $elemento;    // Muestro el fichero  
            }  
        }  
    }  
}  
  
/* Llamamos a la función para que nos muestre el contenido de la carpeta  
galería que se encuentra en la misma carpeta */  
listarArchivos("galeria/");
```

# SUBIR FICHEROS

- **Ejercicio 11:** Galería de imágenes
  - Formulario que permita subir ficheros.
  - Asegurarse que el fichero subido sea una imagen.
  - Modificar el nombre de la imagen a:  
img\_1.extensión, img\_2.extensión, img\_3.extensión,...
  - También podéis usar el día y hora actual para renombrar las imágenes de manera única.
  - Mostrar una lista de imágenes inferior.

# SESIONES

- A veces es necesario mantener el estado de una conexión entre distintas páginas o entre distintas visitas a un mismo sitio:
  - Configuraciones personales
  - Carrito de la compra
  - Control de usuarios
- HTTP es un protocolo sin estado: cada conexión entre el cliente y el servidor es independiente de las demás.
- Para mantener el estado entre diferentes conexiones hay que establecer lo que se conoce como una **sesión**.

# SESIONES

- Las sesiones permiten disponer de unas variables con valores persistentes durante toda la conexión del usuario.
- Estas variables pueden almacenarse en el cliente mediante cookies o en el servidor.
- PHP dispone de una biblioteca de funciones para la gestión de sesiones.



# SESIONES

- Funciones de PHP para el manejo de sesiones:
  - Requisito indispensable: register\_globals ON en php.in
  - Advertencia Esta característica ha sido declarada OBSOLETA desde PHP 5.3.0 y ELIMINADA a partir de PHP 5.4.0.

Genera una array y puede almacenar varias sesiones

- **session\_start();** ←

Inicializa una sesión y le asigna un identificador de sesión único.  
Si la sesión ya esta iniciada, **carga todas las variables de sesión.**

- **session\_register(variable);**  
Registra una variable de sesión.
- **session\_unregister(variable);**  
Elimina una variable de sesión.

**Session\_start()**

ha de estar en todas las paginas que necesitamos la sesion

**Session\_register()**

solo de define una vez al principio


**\$\_session[]=valor**


Damos valor a la variable de sesion

# SESIONES

- Funciones de PHP para el manejo de sesiones:
  - **session\_is\_registered(variable);**  
Comprueba si una variable está registrada.  
Devuelve true en caso afirmativo y false en caso contrario.
  - **session\_destroy();**  
Cierra una sesión

# SESIONES

- Funciones de PHP para el manejo de sesiones:
  - Requisito indispensable: register\_globals OFF en php.ini
  - **session\_start();**  
Inicializa una sesión y le asigna un identificador de sesión único.  
Si la sesión ya esta iniciada, carga todas las variables de sesión.
  - **\$\_SESSION['nombre'] = valor;**  
Registra una variable de sesion 

Se define una sola vez el valor
  - **unset( \$\_SESSION['nombre'] );**  
Elimina una variable de sesión 

Elimina sin destruir la sesion

# SESIONES

- Funciones de PHP para el manejo de sesiones:
  - **If(isset(\$\_SESSION['nombre']))**  
Comprueba si una variable está registrada.  
Devuelve true en caso afirmativo y false en caso contrario.
  - **session\_destroy();**  
Cierra una sesión

# SESIONES

- El manejo de las sesiones se realiza de la siguiente forma:
  - Todas las páginas deben realizar una llamada a `session_start()`; como su primera sentencia;
  - Esta llamada debe estar colocada antes de cualquier código HTML o PHP.
  - Conviene llamar a `session_destroy()` para cerrar la sesión al finalizar el documento

# AUTENTICACIÓN DE USUARIOS

- Una cuestión frecuente en un sitio web es controlar el acceso de los usuarios a una zona determinada del mismo.
- La autenticación de usuarios debe realizarse en el propio servidor web.
- Para ello se usa PHP + bases de datos para controlar el acceso de los usuarios. Se deben utilizar sesiones para ello.

# AUTENTICACIÓN DE USUARIOS

- Ejemplo

```
<?PHP
session_start();
?>
<HTML >
  <HEAD> ...</HEAD>
  <BODY>
    <?PHP
      if(isset($_SESSION["usuario_valido"]))
        // Código para usuarios autorizados
      else
        // Mensaje de acceso no autorizado
    ?>
  </BODY>
</HTML>
```

# SESIONES

- **Mejorar el login que ya teníamos:**
  - Añadir un botón para salir de la sesión.
  - Si el usuario se loguea correctamente iniciar sesión y mostrar su nombre.
  - Sino mostrar el <form> del login.

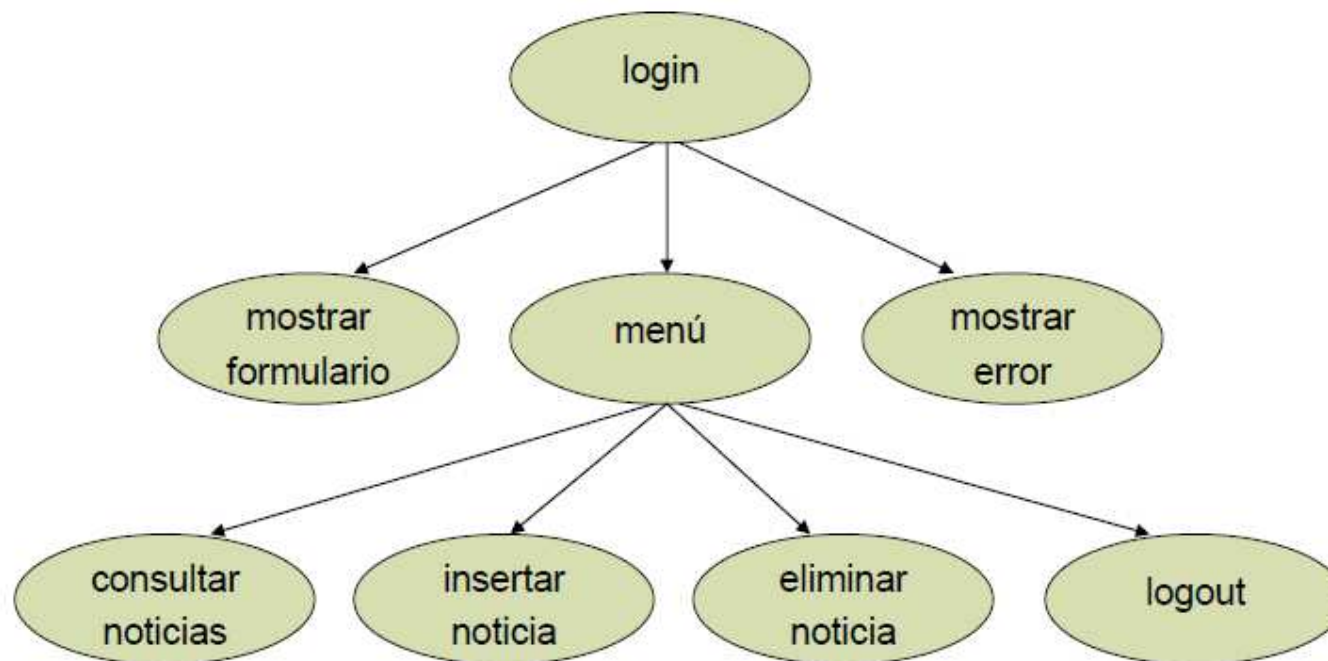


# AUTENTICACIÓN DE USUARIOS

- **Ejercicio 12:** Gestión de noticias parte 1
  - Para nuestro sistema de gestión de noticias se va a restringir el acceso a las operaciones a unos usuarios identificados por un nombre y contraseña.
  - La información de los usuarios por ahora será estática.
  - Las contraseñas estarán encriptadas.
  - Detalles del ejercicio en los siguientes esquemas:

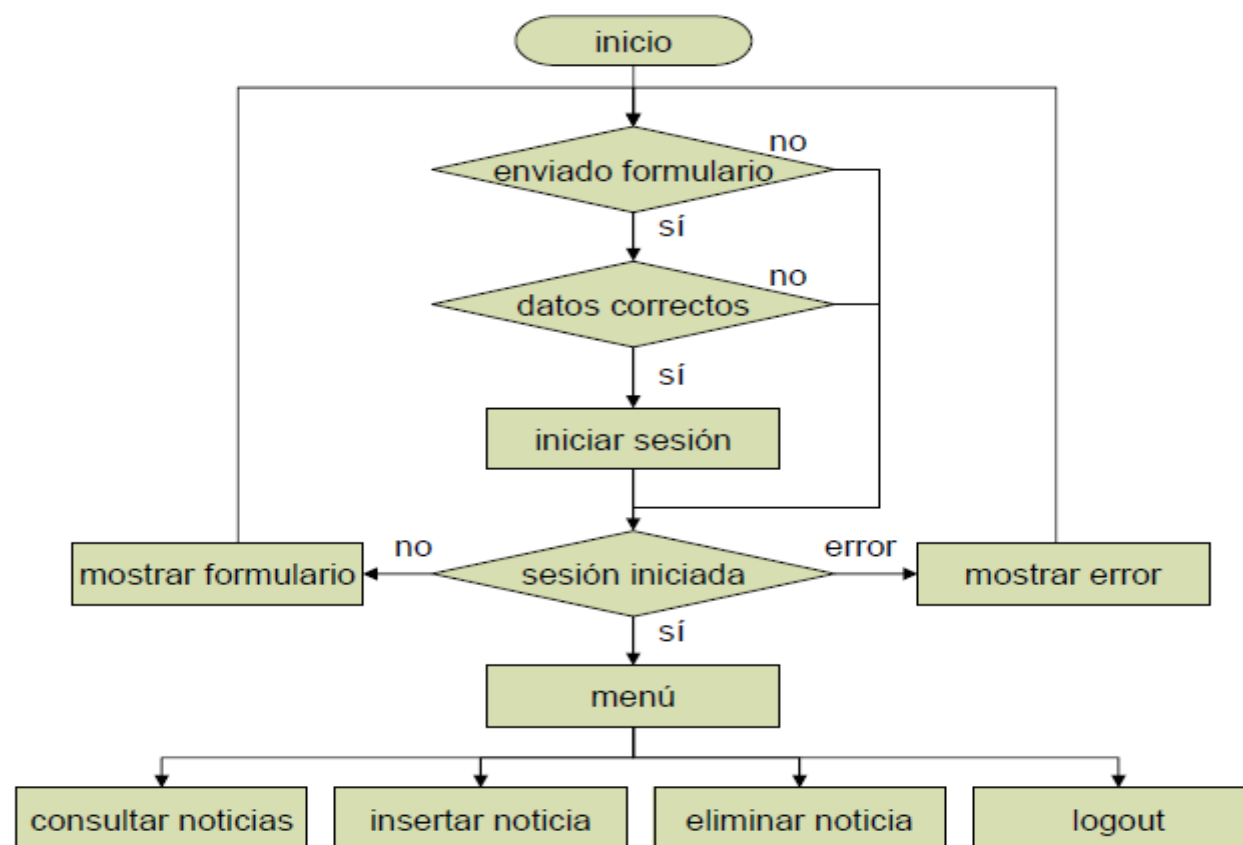
# AUTENTICACIÓN DE USUARIOS

- **Ejercicio 12:** Gestión de noticias parte 1



# AUTENTICACIÓN DE USUARIOS

- **Ejercicio 12:** Gestión de noticias parte 1



# AUTENTICACIÓN DE USUARIOS

- **Ejercicio 12: Gestión de noticias parte 1**
  - Insertar noticia será un formulario **solo válido para usuarios con login**.
  - Eliminar noticia será un formulario **solo valido para usuarios con login (apartado vacio por ahora)**.
  - Consultar noticia será una **sección pública (apartado vacio por ahora)**.

# AUTENTICACIÓN DE USUARIOS

- **Ejercicio 12:** Gestión de noticias parte 1
  - Insertar noticia será un formulario con:
    - Titulo de noticia
    - Texto de noticia
    - Imagen asociada
    - Nombre del autor
    - Fecha
    - Botón de submit
  - Cuando se introduce la noticia se vuelve a mostrar la información por pantalla.