PHP

SUBIDA DE FICHEROS A TRAVES DE FROMULARIOS

Introducción

- Lo más laborioso a la hora de hacer un formulario que permita la subida de archivos es manejarlos y organizarlos una vez están en el servidor.
- Inicialmente los archivos se suben a un directorio temporal, y una vez se ha ejecutado script que recibe los datos, si no se relocaliza el fichero, este desaparece del directorio temporal, por lo que es necesario, una serie de operaciones para mantenerlo en un destino definitivo.

Subida de ficheros : cliente

- Las acciones a realizar al subir un fichero son:

En la parte de 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.

Subida de ficheros : cliente

- form method= "POST" enctype="multipart/form-data.
  - ✓El formulario donde esté el input ha de tener especificado el atributo enctype establecido con el valor mutipart/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 mutipart/form-data.
  - ✓ Este valor se emplea para transferir gran cantidad de texto u otros formatos de fichero entre cliente y servidor, mediante método POST.

Subida de ficheros : cliente

- El fichero debe tener un límite en cuanto a su tamaño. Este límite se fija de dos formas diferentes:

1. En el propio formulario(**Establecer tamaño en el cliente**).

   El tamaño de bytes que vamos a enviar se puede quedar establecido en el cliente, de modo que si el fichero tiene un tamaña mayor, no se envía al servidor.

   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.

Ejemplo parte cliente

```
<h2>Formulario subida de archivos</h2>
<html>
<body>
<form action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]); ?>" method="POST" enctype="multipart/form-data">
    <input type="hidden" name="MAX_FILE_SIZE" value="<?php echo $max_file_size; ?>" />
    <input type="file" name="imagen" />
    <input type="submit" name="submit" />
</form>
</body>
</html>
```

- Se incluye un input del tipo hidden que limita el tamaño máximo del archivo que se puede subir (se debe pasar en bytes).

Subida de ficheros : Servidor

2. El tamaño de fichero queda definido, en el servidor, por la directiva upload_max_filesize del fichero php.ini.

Existen Otras directivas relacionadas con la descargas de ficheros que están establecidas en php.ini, estas vienen con unos valores por defecto de configuración (para ver sus valores por defecto editar el fichero php.ini).

– file_uploads: (On / Off), pemite que haya o no cargas de archivos.

– uploads_max_filesize: tamaño máximo del archivo que se puede subir

– upload_tmp_dir: directorio temporal donde se guardan los archivos cargados

– post_max_size: tamaño máximo de todos los datos enviados por el método post

Subida de ficheros : Servidor

- Como ya se ha comentado la información sobre un archivo subido la proporciona el array multidiensional $_FILES.
- Este array se crea con el name que se indique en el input del formulario de tipo file.
- Cuando PHP recibe el archivo, lo almacena en el directorio upload_tmp_dir y rellena la matriz asociativa superglobal $_FILES["nombre_archivo_cliente"].
- $_FILES es un array asociativo con tantos elementos como input de tipo file vengan del formulario

Subida de ficheros : Servidor

- Cada posición a su vez contiene un array asociativo con información de ese fichero almacenada en 5 componentes:

  $_FILES['nombre_archivo_cliente']['name’]: Nombre que tenía el archivo cargado en el ordenador del cliente

  $_FILES['nombre_archivo_cliente']['type']: tipo MIME del archivo cargado

  $_FILES['nombre_archivo_cliente']['size']: tamaño del archivo cargado $_FILES['nombre_archivo_cliente']['tmp_name’]: Nombre temporal del archivo cargado en el directorio temporal del servidor $_FILES['nombre_archivo_cliente']['error']: código de error (en su caso)

El array $_FILES['nombre_archivo_cliente']['error'] especifica por qué no se ha podido subir el archivo, lo que permite especificar un mensaje de vuelta para cada tipo de error. Devuelve un integer con el número de error

Codigos de error de subida

Error	Valor	Significado
UPLOAD_ERR_OK	0	No hay errores
UPLOAD_ERR_INI_SIZE	1	Supera el tamaño máximo indicado en php.ini
UPLOAD_ERR_FORM_SIZE	2	Supera el tamaño máximo indicado en MAX_FILE_SIZE de html
UPLOAD_ERR_PARTIAL	3	Sólo se ha subido el archivo parcialmente
UPLOAD_ERR_NO_FILE	4	No se ha subido ningún archivo
UPLOAD_ERR_NO_TMP_DIR	6	Falta la carpeta temporal
UPLOAD_ERR_CANT_WRITE	7	No se puede escribir en el directorio especificado
UPLOAD_ERR_EXTENSION	8	Una extensión de PHP ha detenido la subida

Subida de ficheros : Servidor (Procedimiento general)

Si ha subido correctamente el fichero

    Asignar un nombre al fichero

    Mover el fichero a su ubicación definitiva

si no

    Mostrar un mensaje de error

fin-si

Subida de ficheros : Servidor (Procedimiento general)

- Para ver si se ha subido correctamente el fichero se procede:
- Esta acción la realiza la función **is_uploaded_file:** is_uploaded_file($_FILES[‘name-archivo']['tmp_name’])

  Esta función devuelve true si el archivo que se le pasa se ha subido por HTTP POST que es el método seguro(para evitar que un usuario intente manejar los ficheros no cargados por método POST) en este caso:

  - Se asigna un nombre único al fichero(si no queremos machacar el fichero en caso de que suba otro con el mismo nombre), para ello se suele añadir al nombre del fichero una marca de tiempo.
- Si se le pasa $_FILES['archivo_usuario']['name'] no funciona porque desde el directorio temporal solo es accesible el nombre-del-fichero-temporal.

Subida de ficheros : Servidor (Procedimiento general)

- Como el archivo del directorio temporal desaparecerá al acabar el script, es necesario mover el archivo a un directorio definitivo.
- Para realizar esta acción se utiliza la función **move_uploaded_file**

  move_uploaded_file($_FILE[‘name-archivo']['tmp_name'],$destino)

  Mueve el archivo temporal a la ubicación definitiva que esta en $destino, en caso de no poder moverlo da error.

Subida de ficheros : Servidor (Procedimiento general)

Esta función realiza un chequeo para asegurar que el archivo indicado por el primer parámetro sea un archivo cargado a través de HTTP POST.

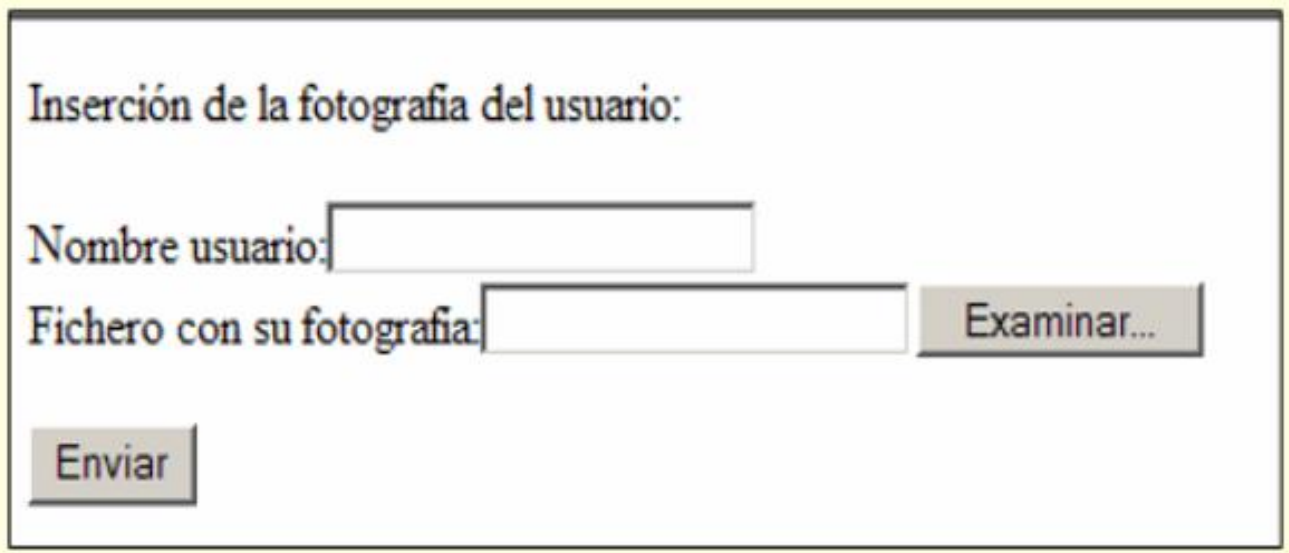
- SI el archivo es válido, será movido al nombre de archivo dado por destino.
- SI NO

  Si nombre_temporal_archivo no es un archivo cargado válido, no lo mueve , y devolverá FALSE.

  Si nombre_temporal_archivo es un archivo cargado válido, pero no puede ser movido por alguna razón, devolverá FALSE, dará una advertencia y generara un error.

Ejemplo(I)

```
<html><body>
Inserción de la fotografía del usuario:
<form action="inserta.php" method="post" enctype="multipart/form-data">
<?
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>
```



- En la parte servidora vamos a centrarnos solo en el código de subida de ficheros y obviaremos la parte de saneado y validación de datos, en este caso el código correspondiente podría ser el siguiente:

Ejemplo(II) inserta.php

```
<html><body>
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_file($nombreCompleto))
   {
      $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
   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
```

Subida múltiple de archivos

- En la parte cliente, el formulario HTML, lo único que cambia es que se añade un atributo **multiple** y que en el atributo name del input file se añade corchetes para indicar que es un array:
- Varios ficheros

```
<input type="file" name="imagenes[]" multiple="multiple" />
```

- Solo 1 fichero

```
    <input type="hidden" name="MAX_FILE_SIZE" value="<?php echo $max_file_size; ?>" />
    <input type="file" name="imagen" />
    <input type="submit" name="submit" />
</form>
</body>
</html>
```

Subida múltiple de archivos

- En cuanto al backend, omitiendo validaciones/saneamiento de datos, esta puede ser una forma de hacerlo:

```php
$directorioSubida = "uploads/";
$max_file_size = "51200";
if(isset($_POST["submit"]) && isset($_FILES['imagenes'])){
    $nombres = $_FILES['imagenes']['name'];
    $temporales = $_FILES['imagenes']['tmp_name'];
    $tipos = $_FILES['imagenes']['type'];
    $errores = $_FILES['imagenes']['error'];
    // Iteramos sobre los arrays creados
    for ($i = 0; $i < count($temporales); $i++){
        if(move_uploaded_file($temporales[$i], $directorioSubida.$nombres[$i])){
            echo "Se ha subido {$nombres[$i]} correctamente <br>";
        } else {
            echo "Ha habido algún error al subir algún archivo";
        }
    }
}
```

```php
<?php
$directorioSubida = "definitivo/";
$max_file_size = "51200";
if(isset($_POST["enviar"]) && isset($_FILES['imagenes'])){
    //print"<br>".print_r($_FILES['imagenes']).print "</pre>\n";
    $nombres = $_FILES['imagenes']['name'];
    $temporales = $_FILES['imagenes']['tmp_name'];
    $tipos = $_FILES['imagenes']['type'];
    $errores = $_FILES['imagenes']['error'];
    // Iteramos sobre los arrays creados
    for ($i = 0; $i < count($temporales); $i++){
        if (is_uploaded_file($temporales[$i]))
        {
            if(move_uploaded_file($temporales[$i], $directorioSubida.$nombres[$i])){
                echo "Se ha subido {$nombres[$i]} correctamente <br>";}
        else {
            echo "El error reportado en la subida es";
            }
        }
    }
} else {
?>
<h2>Formulario subida de archivos</h2>
<html>
<body>
<form action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]); ?>" method="POST" enctype="multipart/form-data">
    <input type="hidden" name="MAX_FILE_SIZE" value="<?php echo $max_file_size; ?>" />
    <input type="file" name="imagen" />
    <input type="file" name="imagenes[]" multiple="multiple" />
    <input type="submit" name="enviar" />
</form>
</body>
<?php
}
?>
```

Funciones de comprobación de archivos y directorios: is_file, is_dir, y file_exists

- is_file devuelve true sólo si la ruta pasada a la función es efectivamente un archivo existente.
- Is_dir devuelve true sólo si la ruta pasada a la función es efectivamente un directorio existente.
- file_exists devuelve true tanto si la ruta pasada es un archivo como un directorio válido (utiliza is_dir si quieres comprobar específicamente si una ruta es un directorio pero no un archivo).
- Esta diferencia es muy importante. Si tu objetivo son únicamente archivos y no directorios is_file es tu función. Si quieres comprobar un directorio o un archivo indiferentemente elige file_exists.

Funciones de comprobación de archivos y directorios: is_file, is_dir, y file_exists

```
$file ='/home/misitio/public_html/directorio/archivo.php’;
$directory ='/home/misitio/public_html/directorio/’;
    $vble = is_file( $file ); // Devuelve true
    $vble= is_file( $directory ); // Devuelve false
    $vble= file_exists( $file ); // Devuelve true
    $vble = file_exists( $directory ); // Devuelve TRUE
```