UPLOAD DE ARCHIVOS con PHP

¿Qué es lo más importante cuando hacemos un UPLOAD con PHP?

SEGURIDAD

¿Por qué?

Estamos permitiendo a cualquier usuario de nuestra página web que nos suba un archivo al servidor.

Debemos controlar en todo momento qué es lo que hace y cómo.

¿Cómo configurar el Form?

```
<form
 action="subearchivo.php"
 method="post"
 enctype="multipart/form-data"
>
```

Subir Archivos

Al subir archivos hemos de tener en cuenta ciertas configuraciones en el archivo php.ini:

- **file_upload** debe estar activado.
- **upload_tmp_dir** debe estar establecido.
- **upload_max_filesize** y **post_max_size** indican el tamaño máximo a subir al servidor.
- Si el archivo es grande, deberemos tocar el **memory_limit** y **max_execution_time** para que le dé tiempo a subirse bien.

Configurar el Formulario

```
<form action="." method="post" enctype="multipart/form-data>
<p>
 <label for="untextoplano">Nombre</label>
 <input type="text" name="untextoplano">
</p>
<p>
 <label for="archivo">Archivo a subir</label>
 <input type="hidden" name="MAX_FILE_SIZE" value="30000" />
 <input type="file" name"archivo">
</p>
</form>
```

A través de **$_POST** recibiremos lo que se escriba en la caja de texto pero no el archivo. Éste se recibe a través de **$_FILES**.

El campo oculto nos permite definir el tamaño máximo del archivo a subir, y se mide en bytes.

$_FILES

Es un array de dos dimensiones. En la primera tenemos los archivos y en la segunda las propiedades de los archivos.

- **$_FILES['archivo']['name']** El nombre original del fichero en la máquina del cliente.
- **$_FILES['archivo']['type']** El tipo mime del fichero.
- **$_FILES['archivo']['size']** El tamaño en bytes del fichero recibido.
- **$_FILES['archivo']['tmp_name']** El nombre del fichero para almacenar el archivo recibido en el servidor.
- **$_FILES['archivo']['error']** Un código de error si procede. http://www.php.net/manual/es/features.file-upload.errors.php

move_uploaded_file()

Una vez subido el archivo a nuestro servidor, se almacena en la carpeta temporal que tenemos asignada.

Tendremos que validar las características del archivo y si son correctas, moverlo a su ubicación definitiva.

Para moverlo usamos:

```
move_uploaded_file($archivo_origen, $ruta_destino);
```

http://www.php.net/manual/es/function.move-uploaded-file.php

Vamos a desarrollar un ejemplo, el cual guardaremos como index.php:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Upload de archivos</title>
</head>
<body>
  <?php
    if(!$_POST){
  ?>
  <form action="." method="post" enctype="multipart/form-data>
    <p>
      <label for="untextoplano">Algo de Texto</label>
      <input type="text" name="untextoplano">
    </p>
    <p>
      <label for="archivo">Archivo a subir</label>
      <input type="hidden" name="MAX_FILE_SIZE" value="5000000" />
      <input type="file" name"archivo">
    </p>
    <p>
      <input type="submit" value="Enviar">
    </p>
  </form>
  <?php
    }else{
    //Estoy recibiendo datos por $_POST
    }
  ?>
</body>
</html>
```

```
//Estoy recibiendo datos por $_POST
$tam_max = 2 * 1024 * 1024 //declaramos como tam maximo 2 MB
$carpeta = "./uploads/";
echo "<p>El dato de texto plano: " . $_POST["untextoplano"] . "</p>";
//Lo primero es comprobar si recibimos algún fichero
if(!isset($_FILES["archivo"])) {
  echo "No estoy recibiendo el archivo";
}elseif($_FILES["archivo"]["size"] == 0) {
//Si el tamaño es 0, es porque el archivo no se envía al servidor
//y puede ser porque supera MAX_FILE_SIZE del formulario o de php.ini
  echo "El archivo no ha llegado correctamente";
}elseif($_FILES["archivo"]["size"] > $tam_max){
  echo "El archivo no puede superar $tam_max bytes";
}elseif($_FILES["archivo"]["type"] != 'image/jpeg') {
  echo "No se permiten archivos diferentes de jpg";
  //Esto no es seguro porque sólo comprueba la extensión del fichero.
} else {
  //Nos podemos fiar sólo en parte
  $destino = $carpeta . $_FILES["archivo"]["name"];
  if(move_uploaded_file($_FILES["archivo"]["tmp_name"], $destino)) {
    echo "Tu archivo ha sido cargado correctamente";
  }else{
    echo "Fallo al cargar el archivo";
  }
}
```

Problemas que pueden surgir

A veces, cuando el archivo ya está subido, nos encontramos que no podemos acceder a él. Eso suele ser problema de permisos, ya que el usuario propietario es Apache. Lo conveniente es cambiarle dichos permisos. Regresamos a nuestro ejemplo:

```
if(move_uploaded_file($_FILES["archivo"]["tmp_name"], $destino)) {
  echo "Tu archivo ha sido cargado correctamente";
  chmod($destino, "0755");
  //Se pone un 0 delante porque es un número octal
}else{
  echo "Fallo al cargar el archivo";
}
```

Más problemas

El usuario, desde su sistema de archivos, podría cambiar la extensión del archivo a subir. Así, p.e., puede cambiar un .exe por un .jpg y subirnos una "foto". Necesitamos hacer más comprobaciones a las realizadas en el código anterior.

Para obtener información sobre el fichero tenemos:

http://www.php.net/manual/es/ref.fileinfo.php

Vamos a ver cual es el nuevo código que debemos poner en el else.

Solución

```
} else {
  $mimeinfo = finfo_open(FILEINFO_MIME);
  if(!$mimeinfo){
    echo "Por motivos de seguridad no puedo analizar el archivo";
  }else{
    $mimereal = finfo_file($mimeinfo, $_FILES["archivo"]["tmp_name"]);
    if(strpos($mimereal, 'image/jpeg') !== 0) {
      echo "El mime real no corresponde. $mimereal";
    }else{
      //Nos podemos fiar completamente
      $destino = $carpeta . $_FILES["archivo"]["name"];
      if(move_uploaded_file($_FILES["archivo"]["tmp_name"], $destino)) {
        echo "Tu archivo ha sido cargado correctamente";
      }else{
        echo "Fallo al cargar el archivo";
      }
    }
  }
}
```

Comprobaciones de seguridad

- Nunca hemos de fiarnos de los datos que el navegador ofrece. Tenemos que comprobarlos sobre el propio archivo a través de PHP.
- Es muy recomendable subir los archivos a una carpeta externa a HTDOCS si se puede, ya que así no tendrán acceso directo los usuarios.
- Tenemos que gestionar los permisos adecuadamente. Los permisos "ideales" son
  - Para archivos: 644
  - Para carpetas: 755

Nunca usar 777, ya que daría permisos a todo el mundo. En todo caso, y sólo si no hay más remedio, el valor más permisivo que deberíamos colocar es 775

Carpeta fuera de htdocs

Si creamos una carpeta fuera de htdocs para guardar nuestros archivos subidos por los clientes, éstos no podrán acceder directamente desde la barra de su navegador. Recordar que si las imágenes están dentro de htdocs, ya sea agrupadas en una carpeta o no, sí pueden acceder.

Pero entonces, ¿Cómo podemos permitir a un usuario que vea las imágenes subidas?

Vamos a crear un script “imagen.php” al mismo nivel que “index.php” que recibirá por GET el nombre de la imagen, y será el propio script el que acceda y muestre la imagen.

Para ello tendremos una carpeta “uploads” al mismo nivel que “htdocs”.

Código de imagen.php

```
<?php
if(!isset($_GET["imagen"])){
 echo "No sé que imagen quieres ver";
}else{
 $ruta = "../uploads/" . $_GET["imagen"] . ".jpg";
 if(!file_exists($ruta)){
  echo "Esa imagen no existe";
 }else{
  header("content-type: imagen/jpeg");
  echo file_get_contents($ruta);
 }
}
```

Para acceder a una imagen escribiremos:

imagen.php?imagen=nombreimagen

De esta forma, en este script podemos añadir comprobaciones extra a las ya colocadas en el script de subida.

Mejoras para imagen.php

¿Qué es el SEO?

El posicionamiento en buscadores u Optimización de motores de búsqueda es el proceso de mejorar la visibilidad de un sitio web en los resultados orgánicos de los diferentes buscadores.En ingles, SEO (Search Engine Optimization).

Mejoras para imagen.php

¿Qué es el SEO?

El posicionamiento en buscadores u Optimización de motores de búsqueda es el proceso de mejorar la visibilidad de un sitio web en los resultados orgánicos de los diferentes buscadores.En ingles, SEO (Search Engine Optimization).

¿Y para qué?

Como programadores en PHP debemos conseguir que nuestra aplicación esté lo más arriba en los buscadores. No vamos a dar ahora SEO, pero debemos saber que si tenemos en nuestra aplicación dos páginas iguales, bajamos en el buscador.

Y ahora, si buscan imágenes que no están en la carpeta uploads, dan el mismo resultado, aunque la URL es distinta (páginas distintas, mismo resultado)

Vamos a tratar de mejorar ese detalle.

Para ello nos vamos a crear en “imagen.php” la siguiente función:

```
function genera_error_404($error = “No se encuentra la imagen”){
 header(“HTTP/1.0 404 Not Found”);
 header(“Status: 404 Not Found”);
 echo $error;
}
```

Y entonces, imagen.php queda como:

Código de imagen.php

```
<?php
if(!isset($_GET["imagen"])){
 genera_error_404();
}else{
 $ruta = "../uploads/" . $_GET["imagen"] . ".jpg";
 if(!file_exists($ruta)){
  genera_error_404();
 }else{
  header("content-type: imagen/jpeg");
  echo file_get_contents($ruta);
 }
}
```

Otra mejora para imagen.php

Aunque hemos comprobado durante la subida (index.php) el tipo MIME del archivo, también sería conveniente comprobarlo aquí. Así pues, nos definimos la siguiente función:

```
function es_jpg($archivo){
  $mimeinfo = finfo_open(FILEINFO_MIME);
  if(!$mimeinfo){
    return false;
  }else{
    $mimereal = finfo_file($mimeinfo, $archivo);
    if(strpos($mimereal, 'image/jpeg') !== 0) {
      return false;
    }else{
      return true;
    }
  }
}
```

Código de imagen.php

```
<?php
if(!isset($_GET["imagen"])){
 genera_error_404();
}else{
 $ruta = "../uploads/" . $_GET["imagen"] . ".jpg";
 if(!file_exists($ruta)){
  genera_error_404();
 }elseif(es_jpg($ruta)){
  header("content-type: imagen/jpeg");
  echo file_get_contents($ruta);
 }else{
  genera_error_404("No es el MIME correcto");
 }
}
```

Última mejora para imagen.php

Si queremos acceder a una imagen en el servidor, hemos visto que la URL a usar sería:

www.midominio.algo/imagen.php?imagen=nombreimagen

Lo cual, según los buscadores, no es una URL amigable. Eso nos haría bajar posiciones.

Para solucionarlo, debemos recordar que los archivos de configuración de Apache son .htaccess, y sólo configuran el directorio en el que se encuentran.

Así pues, en nuestro directorio creamos uno cuyo contenido:

```
RewriteEngine on
RewriteRule îmg/upload/([a-zA-Z0-9_-]+).php$ imagen.php?
imagen=$1
```

Y ¿qué conseguimos con eso?

Pues ahora, para acceder a nuestras imágenes nos basta con escribir:

www.midominio.algo/img/upload/nombreimagen.php

Y eso, la regla que acabamos de escribir en el archivo de configuración lo traduce por:

www.midominio.algo/imagen.php?imagen=nombreimagen

lo cual vemos que es una URL más amigable.

Pero todavía nos queda anular la posibilidad de que puedan acceder por la forma antigua.

Para hacer dicha anulación, hemos de añadir un nuevo condicional al principio de imagen.php, quedando de la siguiente manera:

Código de imagen.php

```
<?php
if(strpos($_SERVER[“REQUEST_URI”], “/img/upload”) !== 0){
 genera_error_404(“No se puede entrar por ahí”);
}elseif(!isset($_GET[“imagen”])){
 genera_error_404();
}else{
 $ruta = “../uploads/“ . $_GET[“imagen”] . “.jpg”;
 if(!file_exists($ruta)){
  genera_error_404();
 }elseif(es_jpg($ruta)){
  header(“content-type: imagen/jpeg”);
  echo file_get_contents($ruta);
 }else{
  genera_error_404(“No es el MIME correcto”);
 }
}
```