RESUMEN DE PHP X – Subida y Descarga de ficheros

Subida de ficheros

Los formularios pueden contener un campo para subir ficheros al servidor. Antes de ver la lógica del sistema de subida de ficheros que vamos a implementar, vamos a explicar la variable super-global **$_FILES** de PHP y los atributos que hay que modificar en el formulario para permitir la subida de ficheros.

enctype es necesario y obligatorio para subir ficheros

```
<form method="post" action="subir.php" enctype="multipart/form-data">
   <input type="file" name="archivo" id="archivo" />
   <input type="submit" value="Subir fichero" />
</form>
```

Este formulario de ejemplo muy sencillo lo hemos diseñado con dos campos, un campo de tipo **file** para seleccionar el archivo que se subirá al servidor, y un botón de tipo **submit** que envía el formulario por el método **post** al script **subir.php**. Cabe destacar el uso del atributo **enctype="multipart/form-data"** que añadirá la cabecera en la petición HTTP para indicarle al servidor que el cuerpo de la petición incluye un archivo.

Cuando enviamos una petición HTTP con archivos al servidor, PHP detecta que estamos subiendo un archivo y lo almacena en un directorio temporal establecido en la directiva *upload_tmp_dir* del archivo de configuración de PHP. El tamaño del archivo no debe sobrepasar lo establecido en la directiva *upload_max_filesize*.

Podemos acceder al fichero subido a través del array super-global **$_FILES['name del campo file']** que nos ofrece un conjunto de propiedades a las que podemos acceder:

- **$_FILES['archivo']['tmp_name']**: El valor almacenado en esta clave devuelve el directorio temporal en el que se ha almacenado el archivo en el servidor.
- **$_FILES['archivo']['name']**: El nombre del archivo en el sistema del usuario.
- **$_FILES['archivo']['size']**: El tamaño en bytes del archivo subido.
- **$_FILES['archivo']['type']**: El tipo MIME del archivo, por ejemplo text/plain o image/gif.
- **$_FILES['archivo']['error']**: Código de error si se ha producido algún error.

Comprobando que se ha subido el fichero:

Podemos utilizar la función **is_uploaded_file()** que toma como parámetro el nombre temporal del fichero subido al servidor, no el nombre del fichero del usuario, para ver si se ha subido satisfactoriamente al servidor. Esta función devuelve TRUE si el archivo está en el directorio temporal, y FALSE en caso contrario.

```php
<?php
        if (is_uploaded_file($_FILES['archivo'][ 'tmp_name' ]))
        {
                echo 'El archivo se ha subido con éxito';
        }
?>
```

Comprobando que el fichero no sobrepasa el tamaño permitido:

Una simple comparación entre el tamaño que queramos permitir subir y **$_FILES['archivo']['size']** bastará para asegurarnos que no se sobrepasa el tamaño permitido.

```php
<?php
        if (is_uploaded_file($_FILES['archivo']['tmp_name']))
        {
                echo 'El archivo se ha subido con éxito';

                if($_FILES['archivo']['size']<(512*1024))
                {
                echo 'El archivo no sobrepasa el tamaño máximo: 512KB';
                }
        }
?>
```

Comprobando que el fichero tiene una extensión permitida:

Para comprobar si el fichero tiene una extensión permitida basta con comprobar el tipo MIME del fichero con las extensiones que permitimos.

```php
<?php
        if (is_uploaded_file($_FILES['archivo']['tmp_name']))
        {
                echo 'El archivo se ha subido con éxito';

                if($_FILES['archivo']['size']<(512*1024))
                {
                        echo 'El archivo no sobrepasa el tamaño máximo:
512KB';

                        if($_FILES['archivo']['type']=='image/jpeg')
                        {
                                echo 'La extensión JPEG está permitida';
                        }
                }
        }
?>
```

Mover el fichero del directorio temporal a nuestro directorio de subidas:

Si no movemos el archivo del directorio temporal a un directorio donde tengamos almacenados los ficheros subidos por los usuarios, el fichero se perderá en el limbo de PHP y no podremos recuperarlo. Para ello contamos con la función **move_uploaded_file()** que toma dos parámetros, la ruta del fichero en el directorio temporal y la nueva ruta donde queramos almacenarlo.

```php
<?php

if ( is_uploaded_file($_FILES['archivo']['tmp_name']) )
{
   echo 'El archivo se ha subido con éxito';

   if( $_FILES['archivo']['size']<(512*1024) )
   {
       echo 'El archivo no sobrepasa el tamaño máximo: 512KB';

        if( $_FILES['archivo']['type']=='image/jpeg')
        {
             echo 'La extensión JPEG está permitida';

             $rand = rand(1000,999999);
             $origen = $_FILES['archivo']['tmp_name'];
             $destino = 'uploads/'.$rand.$_FILES['archivo']['name'];
             move_uploaded_file($origen, $destino);

        }
   }
}
?>
```

El número aleatorio **rand** entre 1000 y 999999 lo utilizamos para prevenir la sobreescritura de ficheros con el mismo nombre. De esta forma ya hemos implementado un sistema de subida de archivos JPEG de tamaño menor a 512KB. Ahora bien, este sistema no es del todo seguro para evitar la subida de archivos maliciosos diseñados para atacar nuestra aplicación, como por ejemplo, una shell en php. Queda pendiente explicar como evitar esta vulnerabilidad de las aplicaciones web.

Descarga de ficheros

Para descargar ficheros normalmente ponemos la ruta al fichero en el enlace y el navegador ya se encarga de realizar la petición HTTP oportuna al servidor. Pero, ¿que ocurre cuando el archivo que se desea bajar se ha generado dinámicamente y aún no está guardando en ningún directorio dentro del servidor o solo queremos que se puedan descargar archivos a través de un script de PHP? Pues que tendremos que diseñar algún método que permita forzar la descarga de archivos.

La solución es enviar la petición HTTP de respuesta mediante código PHP enviando una serie de cabeceras HTTP mediante la función **header()** de PHP que solo funciona si no se ha enviado nada al flujo de salida.

Las cabeceras de la respuesta HTTP que vamos a enviar son:

- Content-Type: application/force-download
- Content-Disposition: attachment; filename=nombre_del_fichero
- Content-Transfer-Enconding: binary
- Content-Length: tamaño_del_fichero

Accederemos al script de descarga de la siguiente manera:

```
<a href="descarga.php?archivo=imagen5.jpeg" >Descargar imagen</a>
<?php
//Si la variable archivo que pasamos por URL no esta
//establecida acabamos la ejecucion del script.
if (!isset($_GET['archivo']) || empty($_GET['archvo'])) {
   exit();
}

//Utilizamos basename por seguridad, devuelve el
//nombre del archivo eliminando cualquier ruta.
$archivo = basename($_GET['archivo']);

$ruta = 'imagenes/'.$archivo;

if (is_file($ruta))
{
   header('Content-Type: application/force-download');
   header('Content-Disposition: attachment; filename='.$archivo);
   header('Content-Transfer-Encoding: binary');
   header('Content-Length: '.filesize($ruta));

   readfile($ruta);
}
else
   exit();
?>
```

Verificamos que el archivo existe, enviamos las cuatro cabeceras, y utilizamos la funcion readfile() que devuelve el contenido del archivo por el flujo de salida. Podemos mejorar mucho ese script añadiendo el content-type adecuado, y haciendo más comprobaciones de seguridad. Haremos una par de ejemplos más adelante para aclararlo.