Introducción al desarrollo web (iDESWEB)

Práctica 11: PHP 5 (sistema de ficheros)

1. Objetivos

- Aprender a acceder al sistema de ficheros desde PHP.
- Aprender a subir y almacenar un fichero con PHP.

2. Recursos

¿Cómo se accede al sistema de ficheros desde PHP? ¿Cómo se crea un fichero? ¿Cómo se lee un fichero? ¿Cómo se crea un directorio?

- **PHP Filesystem**[1]: documentación oficial de las funciones de manejo del sistema de ficheros en PHP.
- **PHP File Handling**[2]: explicación en W3Schools de las funciones de manejo de ficheros en PHP.
- **PHP Filesystem Functions**[3]: referencia en W3Schools de las funciones de manejo del sistema de ficheros en PHP.

¿Cómo se maneja el envío de ficheros con PHP?

- **Form-based File Upload in HTML**[4]: definición oficial del protocolo de envío de ficheros desde un navegador web al servidor.
- **Manejo de envío de archivos**[5]: documentación oficial de PHP que explica cómo manejar en el servidor web el envío de ficheros desde un navegador web.
- **PHP File Upload**[6]: explicación en W3Schools de la subida de ficheros en PHP.

3. ¿Qué tengo que hacer?

En esta práctica tienes que incluir en la página principal un apartado de "fotos seleccionadas" donde se muestran, de entre todas las fotografías existentes en los álbumes, algunas fotografías seleccionadas por críticos de fotografía. Las fotografías seleccionadas se seleccionan a mano y se gestionan mediante un fichero de texto en el que se almacena una referencia a las fotografías seleccionadas, junto con el nombre de la persona que ha seleccionado cada fotografía y un comentario que explica las razones de dicha selección. Cada vez que se cargue la página principal se tiene que elegir una fotografía distinta seleccionada de forma aleatoria: se tiene que mostrar la fotografía, la información disponible sobre ella, el nombre de la persona que la ha seleccionado y su comentario.

Además, también tienes que manejar la subida de una fotografía de un usuario cuando se registra en la "página con el formulario de registro como nuevo usuario". También tienes que permitir que la fotografía subida se pueda modificar o eliminar desde la opción "Modificar mis datos" del menú privado del usuario. **Importante:** todas las fotografías del perfil de los usuarios se tienen que almacenar en un mismo directorio, así que tienes que emplear un sistema de nombrado de ficheros que evite problemas cuando dos usuarios suban dos ficheros con el mismo nombre al servidor.

[1] http://es.php.net/manual/es/book.filesystem.php
[2] http://www.w3schools.com/php/php_fileasp
[3] http://www.w3schools.com/php/php_ref_filesystem.asp
[4] http://tools.ietf.org/html/rfc1867
[5] http://es.php.net/manual/es/features.file-upload.php
[6] http://www.w3schools.com/php/php_file_upload.asp

Por último, también tienes que manejar la subida de una fotografía desde la página "añadir foto a álbum" del menú privado del usuario. Se tiene que emplear alguna estrategia para evitar colisiones de nombres cuando dos usuarios suban dos fotografías (ficheros) con el mismo nombre al servidor o cuando un mismo usuario suba dos fotografías con el mismo nombre a diferentes álbumes.

4. ¿Cómo lo hago?

4.1. Sistema de ficheros

PHP proporciona un gran conjunto de funciones para acceder al sistema de ficheros y manejar los ficheros. La mayoría de las funciones permiten trabajar con ficheros que no están en local, sino que son accesibles a través de Internet mediante una URL.

Las principales funciones para trabajar con ficheros a nivel del sistema de ficheros son:

- `copy(origen, destino)`: realiza una copia del fichero `origen` a `destino`.
- `filemtime(fichero)`: devuelve la fecha y hora de la última modificación de un fichero.
- `filesize(fichero)`: devuelve el tamaño de un fichero.
- `is_file(fichero)`: devuelve `true` si es un fichero normal y `false` en caso contrario.
- `rename(nombreViejo, nombreNuevo)`: renombra o mueve de sitio un fichero.
- `unlink(fichero)`: elimina un fichero.

Las principales funciones para trabajar con directorios son:

- `chdir(directorio)`: cambia de directorio.
- `closedir(id_directorio)`: cierra un directorio.
- `is_dir(directorio)`: devuelve `true` si es un directorio y `false` en caso contrario.
- `mkdir(directorio)`: crea un directorio nuevo.
- `opendir(directorio)`: abre un directorio para su posterior procesamiento con las funciones de manejo de directorios `closedir()`, `readdir()` y `rewinddir()`; devuelve un identificador de recurso.
- `rename(nombreViejo, nombreNuevo)`: renombra o mueve de sitio un directorio.
- `readdir(id_directorio)`: devuelve el nombre del siguiente fichero en un directorio.
- `rewinddir(id_directorio)`: sitúa el manejador de directorio al principio del directorio.
- `rmdir(directorio)`: elimina un directorio.

El siguiente código muestra en una página web el contenido de un directorio, tal como lo hacen muchos servidores web. Para cada fichero se muestra su nombre, la fecha de la última modificación y el tamaño que ocupa. En la Figura 1 se muestra un ejemplo de la página que genera.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Prueba de lectura de directorio</title>
</head>
<body>
<?php
  $nomdir = './';
  echo "<h1>Contenido de $nomdir</h1>\n";
  $dir = opendir($nomdir);
```

```
  echo "<pre>\n";
  echo "<b>";
  echo str_pad("Nombre", 30);
  echo str_pad("Fecha ult. mod.", 20);
  echo str_pad("Tamaño", 10);
  echo "</b></pre>\n";
  echo "<hr /><pre>\n";

  while(($fichero = readdir($dir)) != FALSE)
  {
    echo str_pad($fichero, 30);
    echo str_pad(date("d/m/Y H:i" , filemtime($nomdir . $fichero)), 20);
    if(is_dir($nomdir . $fichero))
    {
      echo "-";
    }
    else
    {
      echo str_pad(filesize($nomdir . $fichero), 10);
    }
    echo "<br />\n";
  }
  closedir($dir);

  echo "</pre><hr />\n";
?>
</body>
</html>
```

En el ejemplo anterior se emplea la función `str_pad(cadena, longitud)` para generar cadenas con una longitud definida que se rellenan con espacios en blanco. Además, la función `date(formato, tiempo)` se emplea para aplicar un formato al valor devuelto por la función `filemtime(fichero)`.

Las principales funciones para manejar los ficheros son:

- `fclose(id_fichero)`: cierra el fichero.
- `feof(id_fichero)`: devuelve `true` si se ha llegado al final del fichero y `false` en caso contrario.
- `fgetc(id_fichero)`: lee un carácter del fichero.
- `fgets(id_fichero)`: lee una línea del fichero.
- `file(nombre)`: lee todo el contenido de un fichero y lo almacena línea a línea en un array.
- `fopen(nombre, modo)`: abre un fichero según el modo indicado (lectura, escritura o lectura/escritura), devuelve un identificador de recurso que se emplea en otras funciones.
- `fscanf(id_fichero, formato, variable1, ...)`: lee datos de un fichero según el formato indicado.
- `fwrite(id_fichero, cadena)`: escribe la cadena en el fichero asociado al identificador de recurso.

El siguiente código muestra como leer todo el contenido de un fichero y visualizarlo línea a línea, indicando en cada línea el número de línea que es. En la Figura 2 se muestra como se visualiza el propio fichero.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
<head>
```

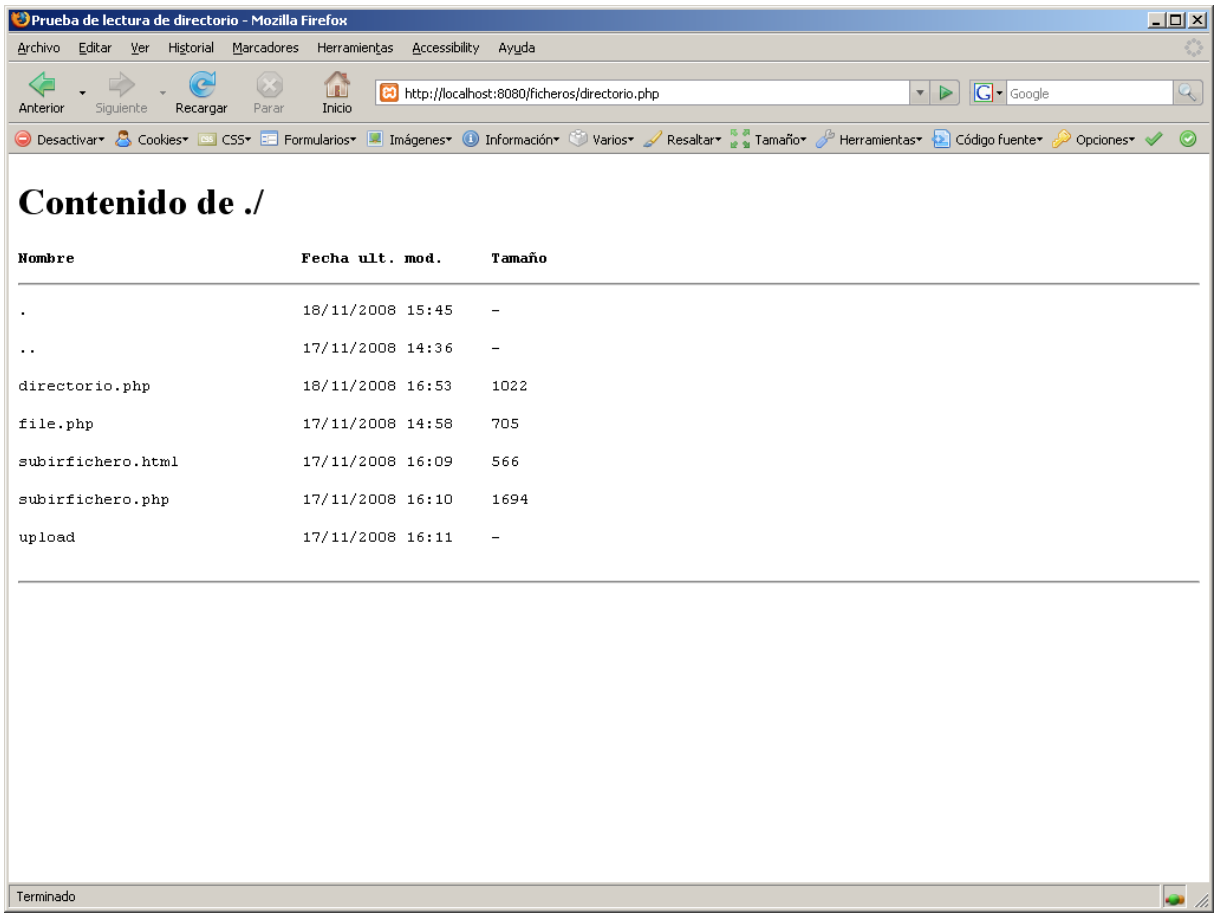

Figura 1: Visualización del contenido de un directorio

```
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Prueba de file</title>
</head>
<body>
<?php
  if(($fichero = @file("file.php")) == false)
  {
    echo "No se ha podido abrir el fichero";
  }
  else
  {
    echo "<pre>\n";
    foreach($fichero as $numLinea => $linea)
    {
      echo "Línea #<b>" . sprintf("%03d", $numLinea) . "</b> : ";
      echo htmlspecialchars($linea);
    }
    echo "</pre>\n";
  }
?>
</body>
</html>
```

En el ejemplo anterior, se emplea el operador `@` para evitar que se muestren en la página web los mensajes de error que puede producir la función `file(nombre)`, como por ejemplo al no existir el fichero indicado[7]. Además, se emplea la función `sprintf(formato, variable1, ...)` para generar una cadena con un formato establecido; en este ejemplo se emplea para generar una cadena de tres dígitos rellenada con ceros. Por último, se emplea la función `htmlspecialchars(cadena)` para convertir los caracteres especiales, como <, > y & a entidades de HTML: `<`, `>` y `&`.

4.2. Manejo de envío de ficheros

Para manejar el envío de ficheros a través de una página web desde el cliente al servidor web se emplea la variable predefinida superglobal `$_FILES`. Esta variable es un array multidimensional donde cada posición representa un fichero que se ha enviado y que contiene la siguiente información:

- `$_FILES['fichero']['name']`: nombre original del fichero en el cliente.
- `$_FILES['fichero']['tmp_name']`: nombre del fichero temporal que se genera en el servidor para guardar el fichero subido.
- `$_FILES['fichero']['size']`: tamaño en bytes del fichero subido.
- `$_FILES['fichero']['type']`: tipo MIME (`image/gif`, `image/jpg`, `application/pdf`, etc.), siempre que el navegador lo proporcione.
- `$_FILES['fichero']['error']`: código de error asociado con la subida del fichero.

El manejo del envío de ficheros se puede configurar en el fichero `php.ini` con las directivas `file_uploads`, `upload_max_filesize`, `upload_tmp_dir`, `post_max_size` y `max_input_time`.

El siguiente ejemplo muestra cómo manejar el envío de ficheros. El primer fragmento de código es la página web que muestra un formulario que permite al usuario enviar un fichero. Para que se pueda realizar el envío se tiene que cumplir que:

- El formulario se envíe mediante el método `post`.
- Se indique como codificación de los datos `enctype="multipart/form-data"`, necesario para poder enviar datos binarios como los contenidos en un fichero.

[7] No es una buena práctica utilizar `@` para "esconder" los errores, lo correcto es utilizar un manejador de errores.

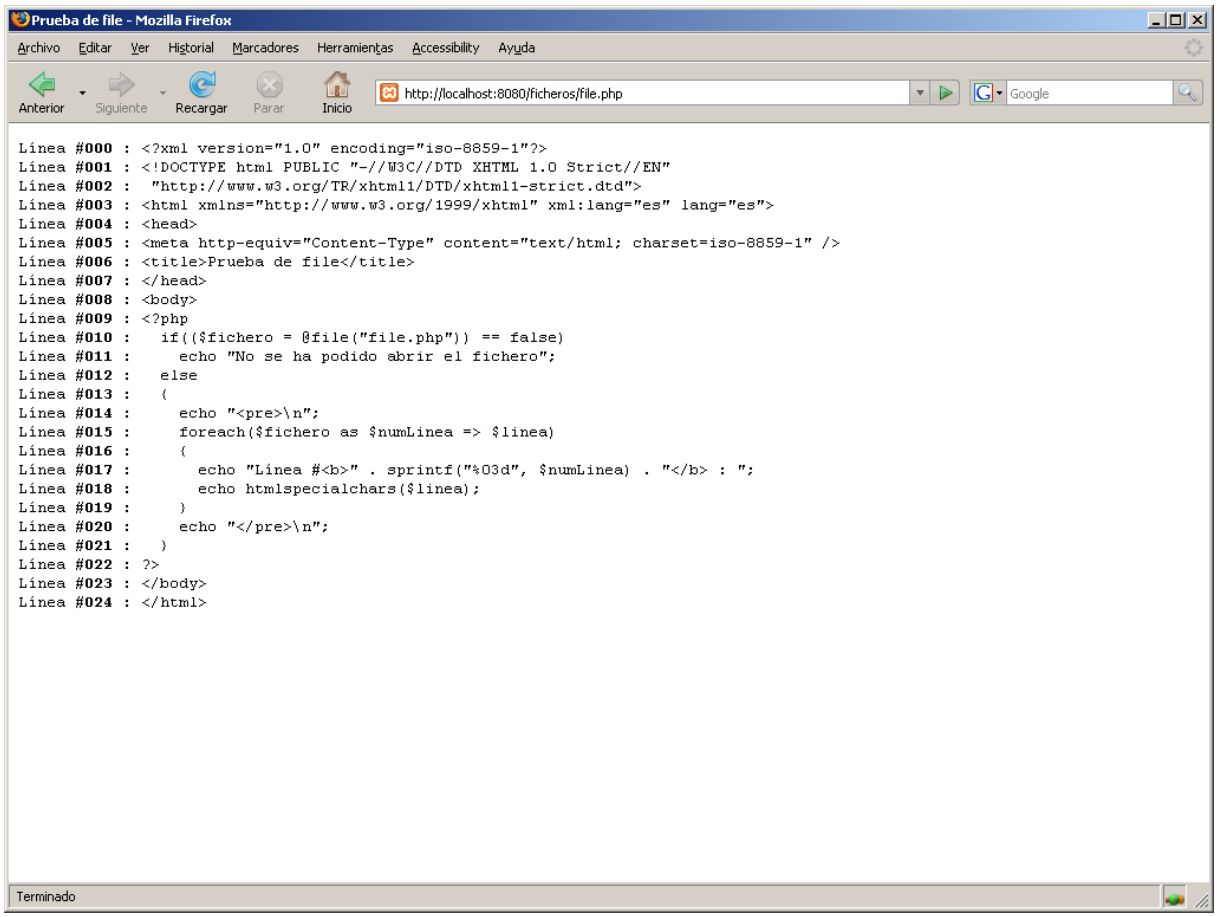

Figura 2: Lectura de un fichero con la función file

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Prueba de subir fichero</title>
</head>
<body>
<form action="subirfichero.php" method="post" enctype="multipart/form-data">
Fichero: <input type="file" name="fichero" />
<br />
<input type="submit" value="Enviar" />
</form>
</body>
</html>
```

El segundo fragmento de código es la página PHP que recibe el fichero enviado en el servidor. En este código se emplea la función `file_exists(nombre)` para comprobar si ya existe el fichero que se intenta subir y la función `move_uploaded_file(origen, destino)` para mover el fichero temporal de la subida a su destino definitivo.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Prueba de subir fichero</title>
</head>
<body>
<p>
<?php
  $msgError = array(0 => "No hay error, el fichero se subió con éxito",
                    1 => "El tamaño del fichero supera la directiva
                          upload_max_filesize el php.ini",
                    2 => "El tamaño del fichero supera la directiva
                          MAX_FILE_SIZE especificada en el formulario HTML",
                    3 => "El fichero fue parcialmente subido",
                    4 => "No se ha subido un fichero",
                    6 => "No existe un directorio temporal",
                    7 => "Fallo al escribir el fichero al disco",
                    8 => "La subida del fichero fue detenida por la extensión");

  if($_FILES["fichero"]["error"] > 0)
  {
    echo "Error: " . $msgError[$_FILES["fichero"]["error"]] . "<br />";
  }
  else
  {
    echo "Nombre original: " . $_FILES["fichero"]["name"] . "<br />";
    echo "Tipo: " . $_FILES["fichero"]["type"] . "<br />";
    echo "Tamaño: " . ceil($_FILES["fichero"]["size"] / 1024) . " Kb<br />";
    echo "Nombre temporal: " . $_FILES["fichero"]["tmp_name"] . "<br />";

    if(file_exists("upload/" . $_FILES["fichero"]["name"]))
    {
      echo $_FILES["fichero"]["name"] . " ya existe";
    }
```

```
    else
    {
      move_uploaded_file($_FILES["fichero"]["tmp_name"],
                         "upload/" . $_FILES["fichero"]["name"]);
      echo "Almacenado en: " . "upload/" . $_FILES["fichero"]["name"];
    }
  }
?>
</p>
</body>
</html>
```

4.3. Generación de números aleatorios

La mayoría de los mecanismos de generación de números aleatorios que se utilizan en los sistemas informáticos son en realidad procesos pseudo-aleatorios, ya que los números se generan a partir de una fórmula y, por tanto, una secuencia de números pseudo-aleatorios se puede reproducir si se conoce el primer número y la fórmula.

En PHP, las principales funciones para generar números pseudo-aleatorios son:

- `rand(min, max)`: devuelve un número pseudo-aleatorio entre `min` y `max`; si no se indican `min` y `max`, se genera un número entre 0 y `getrandmax()`.
- `mt_rand(min, max)`: similar a `rand()`, pero es un generador mejorado y más rápido.
- `getrandmax()`: devuelve el máximo número pseudo-aleatorio que se puede generar con `rand()`.
- `mt_getrandmax()`: similar a `getrandmax()`.
- `srand()`: fija la semilla de inicio para `rand()`; desde PHP 4.2 no es necesario su uso ya que se hace de forma automática.
- `mt_srand()`: similar a `srand()`.

Por ejemplo, el siguiente código genera 10 números pseudo-aleatorios:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Prueba de números aleatorios</title>
</head>
<body>
<p>
<?php
  for($i = 1; $i <= 10; $i++)
  {
    echo rand() . "<br />";
  }
?>
</p>
</body>
</html>
```

5. Recomendaciones

El manual de PHP te lo puedes descargar en diferentes formatos de su sitio web[8] para tenerlo siempre a mano y poder hacer las búsquedas de información rápidamente.

[8] http://es.php.net/download-docs.php

Cuando trabajes con ficheros debes realizar todas las comprobaciones que puedas, ya que los ficheros suelen ser origen de muchos problemas durante el tiempo de ejecución de un programa: un fichero puede ser borrado, puede cambiar su nombre o puede ser movido de sitio. Además, otro problema son los permisos del sistema de archivos: el fichero puede existir, pero puede ser que no tengas permisos para acceder a él.

Permitir que los usuarios puedan subir ficheros al servidor web puede suponer un agujero de seguridad del sitio web, por lo que hay que tomar todas las medidas de protección que se pueda, como por ejemplo limitar el tamaño máximo de fichero que se puede subir o comprobar las extensiones de los ficheros que se suben.

Las funciones de números aleatorios no suelen controlar que no se repita el mismo número en peticiones sucesivas de números aleatorios. Este comportamiento es el esperado en un suceso con probabilidad no condicionada, como puede ser el lanzamiento de una moneda para elegir cara o cruz. Sin embargo, a veces interesa obtener una secuencia de números aleatorios, pero sin que se repita un número en toda la secuencia o que no se repita al menos en dos posiciones consecutivas. En estos casos, es el programador el que debe controlar la generación de los números aleatorios para lograr el resultado esperado.