

Tema 5. Manejo de ficheros externos

Utilización de ficheros externos

PHP dispone de funciones mediante las cuales se pueden crear, modificar, borrar y leer ficheros de cualquier tipo así como extraer información sobre ellos y sus contenidos.

1. Abrir o crear ficheros

Para crear o modificar ficheros se utiliza la instrucción:

`$f1=fopen(fichero,modo)`

\$f1 es una variable que recoge el identificador del recurso que será utilizado para referirnos a este fichero en instrucciones posteriores.

fichero es el nombre (con extensión) del fichero a abrir o crear y deberá escribirse entre comillas.

modo, que es una cadena que debemos poner entre comillas, el indicador del modo de apertura elegido .

Cerrar ficherosUna vez finalizado el uso de un fichero es necesario cerrarlo. Para ello PHP dispone de la siguiente intrucción:

`fclose($f1)`

Valores del parámetro <i>modo</i> de la función <i>fopen</i>	
Valor	Funcionalidad
r	Abre el fichero en modo lectura y coloca el puntero al comienzo del fichero
r+	Abre el fichero en modo lectura y escritura y coloca el puntero al comienzo del fichero
w	Abre el fichero en modo escritura y coloca el puntero al comienzo del fichero, reduce su tamaño a cero y si el fichero no existe intenta crearlo
w+	Abre el fichero en modo lectura y escritura y coloca el puntero al comienzo del fichero, reduce su tamaño a cero y si el fichero no existe intenta crearlo
a	Abre el fichero en modo escritura y coloca el puntero al final del fichero y si no existe intenta crearlo
a+	Abre el fichero en modo lectura y escritura y coloca el puntero al final del fichero y si no existe intenta crearlo

2. Escribir en un fichero

Una vez abierto un fichero -en modo que permita escritura- la función PHP que nos permite escribir en el es la siguiente:

`fwrite($f1,"texto",long);`

donde: \$f1 sigue siendo el identificador de recurso, texto la cadena de texto a insertar en el fichero y long el número máximo de caracteres que han de insertarse.

También puede utilizarse

`fputs($f1,"texto",long);`

que en realidad es un alias de la función anterior.

➤ [Ver ejemplo1_fwrite.php](#), [ejemplo2_fwrite.php](#)

3. Lectura de ficheros

Para la utilización de estas funciones los ficheros han de ser abiertos en un modo que permita la lectura.

➤ [Ver ejemplo_fgetcsv.php](#)

4. Punteros internos

PHP dispone de funciones para situar sus punteros internos y también para determinar la posición a la que apuntan en un momento determinado. Se trata de las siguientes:

feof(\$f1): Es un operador booleano que devuelve CIERTO (1) si el puntero señala el final del fichero y FALSO si no lo hace.

5. Borrado de ficheros

Para borrar ficheros se utiliza la siguiente instrucción:

unlink(fichero);

fichero ha de ser una cadena que contenga el nombre y la extensión del fichero y, en su caso, también el path.

6. Duplicado de ficheros

La función:

copy(fich1, fich2)

Copia el fichero fich1 (debe indicarse nombre y extensión) en otro fichero cuyo nombre y extensión se establecen en la cadena fich2. Esta función devuelve un valor booleano indicando si la copia se ha realizado con éxito TRUE (1) o FALSE (nul) si por alguna razón no ha podido realizarse la copia.

7. Renombrar ficheros

La función:

rename(fich1, fich2)

cambia el nombre del fichero fich1 (hay que poner nombre y extensión) por el indicado en la cadena fich2. También devuelve TRUE o FALSE.

8. Funciones informativas

PHP dispone de funciones que nos facilitan información sobre ficheros. Algunas de ellas son las siguientes:

- **file_exists(fichero)** Esta función devuelve TRUE si el fichero existe, en caso contrario devuelve FALSE.
- **filesize(fichero)** Devuelve el tamaño del fichero expresándolo en bytes. En caso de que el fichero no existiera nos dará un error.

- **filetype(fichero)** Devuelve una cadena en la que se indica el tipo del fichero. En caso de que el fichero no existiera nos dará un error.
- **filemtime(fichero)** Devuelve la fecha de la última modificación del fichero.

9. Operaciones con ficheros.

- **Fusión de ficheros.**

Recorrer cada fichero introduciendo los datos en una tabla interna y después volcarlos en el fichero resultante de la fusión.

- **División de ficheros.**

Recorrer el fichero a dividir y según la condición de división guardar los datos en un fichero de salida (o tabla interna si se desea realizar más procesos con los datos) o en otro fichero de salida (u otra tabla).

- **Búsqueda en ficheros.**

Recorrer el fichero y procesar solo los elementos que cumplan la condición de búsqueda.

- **Ordenación de ficheros.**

Recorrer el fichero y volcarlo sobre un array (la ordenación se realizará sobre el array con la función `sort(array)` ó `rsort(array)`), a continuación se debe volcar el contenido del array ordenado al fichero origen u a otro (según se desee).

10. Transferencia de ficheros

Antes de empezar con este tema debemos comprobar cuál es la configuración de nuestro `php.ini`. Si por alguna circunstancia los valores no coincidieran con los que tenemos aquí a la derecha, tendríamos que abrir `php.ini` y modificar aquellas directivas.

file_uploads	On	On
upload_max_filesize	2M	2M
upload_tmp_dir	no value	no value


La transferencia de un fichero requiere dos documentos:

- Un *formulario* que la inicie.
- Un *script* que la recoja.

```

<HTML>
<BODY>
<FORM ENCTYPE="multipart/form-data" ACTION="ejemplo86.php" METHOD="post">
# con este input "oculto" establecemos el limite máximo
# del tamaño del fichero a transferir. En este ejemplo 1.000.000 bytes
<INPUT type="hidden" name="lim_tamano" value="1000000">
<p><b>Archivo a transferir<b><br>
<INPUT type="file" name="archivo"></p>
<p><INPUT type="submit" name="enviar" value="Aceptar"></p>
</FORM>
</BODY>
</HTML>

```



Se diferencia del que hemos visto en páginas anteriores en dos aspectos.

1. En la etiqueta **<form>** hemos de incluir —obligatoriamente— el **method="POST"** y **ENCTYPE = "multipart/form-data"** ya que no *soporta* ni otro método ni otra forma de encriptación.
2. El cuerpo del *formulario* ha de contener un nuevo tipo de input que utiliza la siguiente sintaxis:

<input type='file' name='nm'>

La transferencia.

Una vez enviado el formulario, *el fichero transferido* se guarda en un *directorio temporal* del servidor con un *nombre* que le es asignado de forma automática, y, además, en las *variables predefinidas* **\$_FILES** se recogerán datos relativos al contenido del fichero y a los resultados de la transferencia.

El formato de **\$_FILES** es el de *array bidimensional*.

- El *primero de sus índices* es el nombre de variable usado para la transferencia (el especificado como `name='nm'` en el `input type='file'`).
- Los segundos índices —se trata de un array asociativo— son estos: *name*, *type*, *tmp_name*, *error* y *size*.

\$_FILES['archivo']['name']

\$_FILES['archivo']['type']

\$_FILES['archivo']['tmp_name']

\$_FILES['archivo']['error']

\$_FILES['archivo']['size']

<?php

```

foreach ($_FILES['archivo'] as $indice=>$valor){
    print $indice."--->".$valor."<br>"; }

copy ($_FILES['archivo']['tmp_name'],'ficherosubido');

```

?>

Copia del fichero

Tal como hemos visto, el fichero transferido aún no está en el servidor. Por el momento se encuentra en un directorio temporal y será preciso hacer una copia en nuestro espacio de servidor. Para este proceso puede utilizarse una función que ya hemos anteriormente:

copy(fich1, fich2) donde

fich1 sería el fichero temporal. contenido en: `$_FILES['nm']['tmp_name']`

fich2 el del nuevo fichero.

➤ **Ver envio.php y recepción.php**

11. include

PHP dispone de funciones que permiten insertar en un documento una parte o la totalidad de los contenidos de otro. Esta opción resulta muy interesante, tanto desde el punto de vista operativo como en lo relativo a la seguridad. La sintaxis es la siguiente:

include("nom. del fichero");

➤ **Ver funciones.php y ejemplo_include.php**

Hay otras opciones como `require`, `require_once`, etc