

Tema 1

PHP: Funciones para tratamiento de ficheros y sistema de archivos

Apertura de ficheros: **fopen**

puntero **fopen**(ruta_fichero, modo_apertura)

Los archivos en PHP se abren con **fopen()**, que recibe 2 parámetros:

- Ruta del fichero a abrir, que debe tener los permisos adecuados.
 - Modo de apertura. Es el tipo de acceso que se tendrá al fichero, que puede ser:
- y devuelve un puntero que apunta al fichero, si se ha podido abrir o 0 en caso contrario.

Modo apertura	Descripción
'r'	Apertura para sólo lectura; coloca el puntero al principio del archivo.
'r+'	Apertura para lectura y escritura; coloca el puntero al principio del archivo. Debe tener permiso de escritura.
'w'	Apertura para sólo escritura; coloca el puntero al principio del archivo y trunca el archivo a longitud cero, por lo que borrará todo su contenido. Si el archivo no existe se intenta crear.
'w+'	Apertura para lectura y escritura; coloca el puntero al principio del archivo y trunca el archivo a longitud cero, por lo que borrará todo su contenido. Si el archivo no existe se intenta crear.
'a'	Apertura para sólo escritura; coloca el puntero al final del archivo. Si no existe se intenta crear.
'a+'	Apertura para lectura y escritura; coloca el puntero al final del archivo. Si no existe se intenta crear.

- El tercer parámetro es opcional y se trata de un booleano que indica si debe buscar el archivo en la directiva *include_path* en el archivo de configuración de PHP.

Cierre de ficheros: **fclose**

boolean **fclose**(puntero)

Cierra un fichero, tomando como parámetro el puntero al fichero a cerrar. Es importante acordarse de cerrar ficheros, si hemos ESCRITO en ellos.

Escritura de ficheros: **fwrite** y **fputs**

int **fwrite** (puntero , string_a_escribir , [longitud])

int **fputs** (puntero , string_a_escribir , [longitud]), que es un alias de fwrite

Escriben en un fichero abierto para escritura, en la posición actual del puntero. Reciben 2 parámetros: el puntero del fichero y la cadena a escribir. El tercer parámetro es opcional e indica la longitud en bytes que se va a escribir. Devuelve el número de bytes escritos o *FALSE*.

EJEMPLO

```
<?php
$f = fopen('archivo.txt', 'a');
if(!$f){
    echo 'No se puede abrir el fichero.'; exit();
}
$cadena="Hola, esto es un ejemplo de escritura en ficheros.";
fwrite($f, $cadena, strlen($cadena));
fclose($f);
?>
```

* *fwrite/fputs* NO agregan salto de línea. Si queremos escribir varias líneas en un fichero lo tendremos que especificar escribiendo `\n`

Comprobación de fin de fichero: **feof**

booleano **feof** (puntero)

Devuelve TRUE si el puntero del fichero se encuentra al final del mismo. Se utiliza cuando se recorre **un archivo línea por línea** o para la **lectura de grandes archivos**, mediante un condicional:

```
//Lectura de un archivo en bloques de 4092 bytes
$f = fopen("miarchivo.txt", "r");
while(!feof($f)){
    echo fread($f, 4092);
}
fclose($f);
```

Lectura de ficheros : **fgets, fgetc, fread, fscanf**

string **fread** (puntero, longitud)

Toma 2 parámetros, el puntero del fichero, y el número de bytes a leer, y los devuelve como cadena.

Ejemplo: Carga todo el contenido del fichero en una variable con la combinación de las ordenes fread y filesize. A continuación, mediante explode generamos un array con cada línea, el cual recorreremos y mostramos

```
<?php
    $f = 'fichero.txt';
    $fichero = fopen($nombre_fichero,'r');
    $contenido = fread($f, filesize($nombre_fichero));
    fclose($f);
    $lineas = explode("\n",$contenido);
    foreach ($lineas as $linea) {
        echo $linea."<br/>";
    }
?>
```

string **fgets** (puntero [, longitud])

Se utiliza para leer línea a línea un fichero. Toma como parámetro el puntero al fichero y opcionalmente una longitud y leerá una nueva línea o hasta que alcance dicha longitud

```
<?php
$f= fopen('archivo.txt', 'r' );
if(!$f)
    echo 'No se puede abrir el fichero.';
else{
    $num=1;
    while (!feof($f)) {
        $linea = fgets ($f) ;
        echo 'Linea '.$num.': '.$linea."<br />";
        $num++;
    }
    fclose ($f) ;
}
?>
```

string **fgetc** (puntero)

Para leer un carácter del fichero

EJEMPLO: Leer fichero carácter a carácter, mostrando un carácter por línea

```
<?php
$f = fopen('archivo.txt', 'r');
if (!$f) {
    echo 'No se pudo abrir archivo.txt';
}
while (false !== ($carácter = fgetc($f))) {
    echo "$carácter<br/>";
}
fclose($f);
?>
```

array **fscanf** (puntero, formato)

Lee del fichero que se pasa como primer parámetro, ajustándose al formato indicado en el segundo parámetro. Devuelve un array con los datos leídos

EJEMPLO: Leer un fichero por líneas, (2 string y 1 int por línea, separados por tabuladores)

```
<?php
$f = fopen("usuarios.txt", "r");
while ($userinfo = fscanf($f, "%s\t%s\t%d\n")) {
    list ($nombre, $profesión, $edad) = $userinfo;
    //... hacer algo con los valores
}
fclose($f);
?>
```

SISTEMA DE ARCHIVOS

PHP cuenta con una serie de funciones para manipular el sistema de ficheros. Con ellas podemos copiar, renombrar, mover ficheros, etc, como si lo hiciéramos con comandos del sistema operativo, sin necesidad de abrirlos (crear puntero...). He aquí algunas:

Estas funciones NO operan sobre archivos abiertos con fopen, sino que usan **los nombres** de los mismos

file_get_contents (ruta)	Recibe la ruta de un fichero y devuelve su contenido en forma de cadena , sin necesidad de abrirlo para lectura.
copy (origen, destino)	Crea una copia de un archivo. Devuelve TRUE en caso de éxito
rename (antiguo, nuevo)	Renombra un archivo.
unlink (ruta)	Borra un archivo.
boolean file_exists (ruta)	Devuelve si un archivo existe
string basename (ruta [, sufijo])	Devuelve la parte del path correspondiente al nombre del archivo. Si el fichero acaba en el sufijo que se le pase como 2º parámetro, éste se corta. <?php \$path = "/home/httpd/html/index.php"; \$file = basename(\$path); // \$file toma el valor "index.php" \$file = basename(\$path, ".php"); // \$file toma el valor "index" ?>
int filesize (ruta)	Devuelve tamaño del fichero en bytes
int filemtime (ruta)	Obtiene el momento (timestamp) de última modificación del fichero
file_type (ruta)	Devuelve el tipo del fichero. Los valores posibles son fifo, char, dir, block, link, file, socket y unknown.
bool is_dir (ruta) bool is_file (ruta) bool is_link (ruta)	Devuelve si el fichero es un directorio, un fichero regular, un enlace, ...
bool is_readable (ruta)	Devuelve si el fichero existe y es legible
bool is_uploaded_file (ruta)	Indica si el archivo fue subido mediante HTTP POST. Recibe un argumento como \$_FILES['archivo_usuario']['tmp_name']
move_uploaded_file (origen, destino)	Mueve un archivo subido a una nueva ubicación
rmdir (ruta)	Intenta eliminar el directorio indicado
mkdir (ruta)	Intenta crear el directorio indicado en la ruta
directorio opendir (ruta)	Abre el directorio indicado y le asigna un identificador, que devuelve.
entrada_directorio readdir (directorio)	Lee la siguiente entrada de un directorio abierto con opendir y desplaza el puntero al elemento siguiente
closedir (directorio)	Cierra el directorio abierto previamente con opendir
scandir (directorio)	Array con el listado de archivos de un directorio

PHP: Subida de ficheros

En un formulario HTML los usuarios pueden subir archivos del navegador a un servidor Web. Los archivos pueden ser de texto o archivos binarios, hojas de cálculo o cualquier otro tipo de datos. Para subir archivos necesitaremos crear un formulario con un input de tipo file.

Atributos de <form> y el elemento archivo

Para subir archivos, la etiqueta <form> tiene 3 atributos:

- El atributo action que especifica el script PHP que procesará el formulario.
- El atributo enctype, que determina cómo codifica el navegador los datos del formulario. El valor predeterminado es application/x-www-form-urlencoded. Sin embargo, si vamos a enviar archivos, debemos especificar que los datos son de tipo multipart/form-data.
El navegador codifica los datos de tipo multipart/form-data de modo diferente
- El atributo method tiene que ser POST

Además de añadir estos 3 atributos al <form>, debemos agregar un input de tipo "file"

```
<input type="file" name="upload" />
```

El navegador muestra los un botón de navegación para entrar en modo de selección de archivos. También se puede especificar el máximo tamaño de fichero a subir, en un campo oculto denominado "MAX_FILE_SIZE" aunque este dato es burlable y es el archivo php.ini quien determina finalmente dicho tamaño de carga máxima, mediante **upload_max_filesize** y **post_max_size**. Si incluimos el campo oculto, éste deberá preceder al campo de entrada del archivo. Los archivos se almacenan en el directorio temporal predeterminado del servidor, determinado por la directiva **upload_tmp_dir** del archivo php.ini.

La matriz \$_FILES

Cuando el archivo se envía al servidor, PHP almacena toda la información del mismo en la matriz superglobal \$_FILES. El primer índice de la matriz es el nombre del input y el segundo índice es uno de los atributos del archivo.

Matriz	Descripción
\$_FILES['xxxxxx']['name']	Nombre del archivo en la máquina cliente
\$_FILES['xxxxxx']['type']	Tipo MIME del archivo, si el navegador lo reconoce. Ejemplo: "image/gif"
\$_FILES['xxxxxx']['size']	Tamaño del archivo en bytes
\$_FILES['xxxxxx']['tmp_name']	Nombre temporal del archivo que se carga en el servidor
\$_FILES['xxxxxx']['error']	Código de error asociado al envío del archivo

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

  <input type="hidden" name="MAX_FILE_SIZE" value="30000"/>
  Escoge archivo a subir: <input type="file" name="uploadfile" />
  <input name="submit" type="submit" value="Enviar archivo" />

</form>
```

```
<?php

$f=fopen($_FILES['uploadfile']['tmp_name'], "r");
while (!feof($f)){
    $texto = fgets ($f);
    echo $texto . "<br/>";
}

?>
```

Después de subir al servidor un fichero desde un campo “input file”, hay 2 funciones que nos pueden interesar:

is_uploaded_file (\$_FILES['xxxxxx']['tmp_name'])

→ para comprobar si realmente ha podido subir el fichero

move_uploaded_file (\$_FILES['xxxxxx']['tmp_name'] , rutadestino);

→ para mover el fichero (desde la carpeta temporal con su nombre temporal) a una carpeta del servidor dándole si se desea otro nombre

```
if (isset($_POST['submit'])){  
    if (!is_uploaded_file($_FILES['fich']['tmp_name'])){  
        echo "Archivo no subido";  
    }  
    else{  
        move_uploaded_file($_FILES['fich']['tmp_name'], UPLOAD_DIR . "/" . basename($_FILES['fich']['name']));  
        echo "Archivo subido";  
    }  
}
```

NOTAS:

- Si creamos un fichero de texto a mano TABULADO (tipo agenda con líneas) y posteriormente lo queremos leer línea a línea con `fgets`, dentro de la línea no va a reconocer los tabuladores como `\t`
Soluciones:
 - No crearlo a mano, sino programáticamente:
`fwrite` o `fputs ($f, "cadena\tcadena\t");`
 - Utilizar como separador de campos un espacio o coma o en vez de un tabulador
- Si creamos un fichero de texto a mano insertando saltos de línea con la tecla "Return", cada salto de línea se traduce a 2 caracteres: `\r\n`