Tema 08 – Gestión de Archivos y Directorios

2º DAW – Desarrollo Web Entorno Servidor

Profesor Juan Carlos Moreno

CURSO 2023/2024

Tabla de Contenido

8 Gestión de Archivos y Directorios en PHP 3
8.1 Introducción 3
8.2 Gestión de Archivos 3
8.2.1 Abrir Archivo 3
8.2.2 Cerrar Archivo 4
8.2.3 Escribir en un Archivo 5
8.2.4 Leer Archivo 6
8.2.5 Puntero de un archivo 7
8.2.6 Obtener información de un archivo 9
8.2.7 Copiar, renombrar, mover y eliminar archivos 10
8.3 Gestión de Directorios 11
8.3.1 Funciones de directorio 11
8.3.2 Función Glob() 13
8.3.3 Crear, renombrar, eliminar y cambiar de directorio 13
8.3.4 Otras funciones útiles sobre directorios 14
8.4 Utilidades 14
8.4.1 Subir archivos al servidor 14
8.4.2 Archivos CSV 16
8.4.3 Comprimir y Descomprimir Archivos 18

8 Gestión de Archivos y Directorios en PHP

8.1 Introducción

El lenguaje PHP dispone de una gran cantidad de funciones para realizar todo tipo de operaciones con archivos y carpetas, tanto básicas (como crear archivos y carpetas, modificar, eliminar...) como otras más avanzadas (para obtener y asignar permisos, crear enlaces simbólicos, etc.).

8.2 Gestión de Archivos

El tratamiento de archivos resulta ser una práctica muy común en cualquier sitio web. Muy a menudo nos vemos en la necesidad de procesar un texto para cambiarle el formato, buscar una cadena en su interior o cualquier otro tipo de operación.

PHP propone un sinfín de funciones para la gestión de archivos que van desde las más elementales de apertura, lectura y cierre a otras más rebuscadas como el cálculo de espacio en el disco duro, tamaño del archivo, gestión de derechos de acceso...

Las operaciones más elementales, copia, borrado y cambiar el nombre, requieren únicamente el nombre (y path) del archivo sobre el cual se ejerce la operación. Para operaciones más complejas, como la lectura de líneas o la escritura de texto dentro del archivo, se requiere de una previa apertura del archivo al cual le asignaremos un identificador *$id*.

Una vez abierto el archivo, podremos desplazarnos a lo largo de él por medio de un puntero imaginario que avanza o retrocede por las líneas de texto y mediante el cual nos situaremos en el lugar escogido para insertar, modificar o simplemente copiar una cadena.

8.2.1 Abrir Archivo

Los **archivos en PHP** se abren con la función ***fopen()***, que requiere dos parámetros: el **archivo que se quiere abrir** y el **modo en el que abrir el archivo**. La función devuelve un **puntero** en el archivo si es satisfactoria o **cero** si no lo es. Los archivos se abren para realizar operaciones de lectura o escritura.

```
$fp = fopen("miarchivo.txt", "r");
```

Si no es posible abrir el archivo, devuelve cero, por eso es frecuente utilizar esta función en una condición

```
if (!$fp = fopen("miarchivo.txt", "r")){
  echo "Error al abrir el archivo";
}
```

Los modos de acceso para ***fopen()***son los siguientes:

Modo	Descripción
r	Apertura para lectura. Puntero al principio del archivo

r+	Apertura para lectura y escritura. Puntero al principio del archivo
w	Apertura para escritura. Puntero al principio del archivo y lo sobrescribe. Si no existe se intenta crear.
w+	Apertura para lectura y escritura. Puntero al principio del archivo y lo sobrescribe. Si no existe se intenta crear.
a	Apertura para escritura. Puntero al final del archivo. Si no existe se intenta crear.
a+	Apertura para lectura y escritura. Puntero al final del archivo. Si no existe se intenta crear.
x	Creación y apertura para sólo escritura. Puntero al principio del archivo. Si el archivo ya existe dará error E_WARNING. Si no existe se intenta crear.
x+	Creación y apertura para lectura y escritura. Mismo comportamiento que x.
c	Apertura para escritura. Si no existe se crea. Si existe no se sobrescribe ni da ningún error. Puntero al principio del archivo.
c+	Apertura para lectura y escritura. Mismo comportamiento que C.

Algunas **consideraciones** de la tabla:

- Si el archivo no es de escritura, abrirlo con r+ fallará, incluso cuando sólo se intenta leer.
- w y w+ eliminarán el contenido de cualquier archivo. Para sólo añadir y no borrar, se usa a y a+.
- Si quieres crear nuevos archivos y evitar sobrescribir sin querer un archivo existente, utiliza x o x+.
- Cuando se trabaja con archivos binarios, como imágenes, hay que añadir 'b' después del modo. Como rb o r+b

Veamos otros ejemplos

```
$fp = fopen("/apr2/fichero.txt", "r");
$fp = fopen("/apr2/fichero2.txt", "w");
$fp = fopen("http://www.aprenderaprogramar.com/texto.txt", "a+");
$fp = fopen("ftp://ftp.elmundo.es/fichero.txt", "w");
```

IMPORTANTE: los sistemas Windows usan **\r\n** como caracteres de final de línea, los basados en UNIX usan **\n** y los sistemas basados en sistemas MACintosh usan **\r**: para evitar incompatibilidades es recomendable usar el modificador **b**.

8.2.2 Cerrar Archivo

Una vez hayamos terminado de realizar las operaciones deseadas con el archivo es necesario cerrarlo usando la función ***fclose()***, que escribirá en él los cambios pendientes en el **buffer** (memoria en la que se guardan los datos antes de ir escribiéndolos).

```
$gestor = fopen('archivo.txt', 'r');
fclose($gestor);
```

8.2.3 Escribir en un Archivo

Para insertar texto en un archivo usaremos las funciones ***fwrite()*** o ***fputs()***, las cuales devolverán false en caso de error.

```
$file = "miarchivo.txt";
$texto = "Hola que tal";
$fp = fopen($file, "w");
fwrite($fp, $texto);
fclose($fp);
```

Podemos limitar la longitud de datos que queremos escribir (todos los datos que había en el archivo se borrarán por completo igualmente)

```
$file = "miarchivo.txt";
$texto = "Hola que tal";
$fp = fopen($file, "w");
fwrite($fp, $texto, 4); // Escribirá sólo: Hola
```

Cuando se escribe en un archivo los datos previamente son almacenados en un buffer de escritura (espacio en memoria RAM). El tamaño del buffer suele ser de 8K.

Si hay dos procesos esperando a escribir en un archivo, cada uno se pausará después de escribir 8K de información para permitir que el otro escriba.

Con la función de PHP ***set_file_buffer()*** es posible definir el tamaño del buffer.

Usando la función ***fflush()*** podemos forzar que se escriban en el archivo los cambios pendientes en el buffer de escritura. Tal y como se indica en el apartado anterior, con ***fclose()*** se escribirán también los cambios pendientes.

```
<?php
    // Abrir el archivo, creándolo si no existe:
    $archivo = fopen("datos.txt","w+b");
    if( $archivo == false ) {

            echo "Error al crear el archivo";
    }
      Else  {
           // Escribir en el archivo:
             fwrite($archivo, "Estamos probando\r\n");
             fwrite($archivo, "el uso de archivos ");
             fwrite($archivo, "en PHP");


            fflush($archivo);
    }
    // Cerrar el archivo:
    fclose($archivo);
?>
```

También podemos insertar texto en un archivo a partir de una cadena de texto usando la función de PHP ***file_put_contents()***:

```
<?php
    $cadena = file_get_contents("datos.txt");
```

```
    $cadena .= "\r\nMe encanta PHP!";
    file_put_contents("datos.txt", $cadena);
?>
```

8.2.4 Leer Archivo

La función de PHP ***fread()*** nos devolverá una cadena de texto con el contenido leído desde el archivo que le indiquemos como primer parámetro.

Como segundo parámetro opcional, podemos indicar el número de bytes que se deben leer, en caso de necesitar leer todo el archivo podemos usar la función ***filesize()***.

Tras usar ***fread()*** el puntero al archivo quedará en la última posición leída, en caso de ser necesario usaremos ***rewind()*** para volver a situarlo al principio del mismo.

Según la documentación oficial de PHP, en sistemas operativos que diferencian entre archivos de texto y archivos binarios (como Windows) se recomienda abrir el fichero usando el modificador b.

```
$file = "miarchivo.txt";
$fp = fopen($file, "r");
$contents = fread($fp, filesize($file));
```

La variable ***$contents*** guardará el contenido que obtengamos con la función ***fread()***. Esta función requiere dos parámetros, el archivo abierto y la longitud que queremos leer de dicho archivo (en bytes). En este caso hemos empleado la función ***filesize()*** para obtener el tamaño del archivo y así devolver todo su contenido.

```
<?php
    # Abrir el archivo en modo de sólo lectura:
    $archivo = fopen("datos.txt","rb");

    if( $archivo == false ) {
      echo "Error al abrir el archivo";
    }
    else
    {
     $cadena1 = fread($archivo, 18);
     rewind($archivo);
     $cadena2 = fread($archivo, filesize("datos.txt"));
     if( ($cadena1 == false) || ($cadena2 == false) )
        echo "Error al leer el archivo";
     else
        {
        echo "<p>\$contenido1 es: [".$cadena1."]</p>";
        echo "<p>\$contenido2 es: [".$cadena2."]</p>";
        }
    }

    # Cerrar el archivo:
    fclose($archivo);
?>
```

File_get_contents()

Aparte de ***fread()*** nos puede resultar muy útil ***file_get_contents()***, puesto que con ella podemos obtener todo el contenido del archivo en una cadena de texto, pudiendo también

indicarle la posición inicial a partir de la que deseamos extraer el texto (comenzando desde cero), y el número de caracteres a obtener. Devuelve false en caso de error.

```
<?php
    $cadena = file_get_contents("datos.txt");
    echo "<p>\$cadena es [".$cadena."]</p>";

    # Leer 14 caracteres comenzando desde el carácter número 21
    $cadena = file_get_contents("datos.txt", null, null, 20, 14);
    echo "<p>Ahora \$cadena es [".$cadena."]</p>";
?>
```

File()

Otra función que puede resultar interesante es ***file()***, con la que obtendremos un array que contendrá cada línea de texto del archivo en una posición del mismo. Devuelve false si se ha producido un error.

```
<?php
    $aCadena = file("datos.txt");
    print_r($aCadena);
?>
```

fgets() y fgetss()

Aparte de las ya comentadas, también podemos utilizar ***fgets()*** y ***fgetss()*** que nos devolverá una cadena de texto con el contenido de la línea en la que se encuentra el puntero al archivo. La diferencia entre ambas es que ***fgetss()*** limpará los caracteres HTML encontrados (si los hay). Ambas devuelven false en caso de error.

```
<?php
    # Abrir el archivo en modo de sólo lectura:
    $archivo = fopen("datos.txt","rb");

    # Recorremos el archivo mostando el contenido de cada línea:
    while( !feof($archivo)
    {
        Echo fgets($archivo). "<br />";
    }
    fclose($archivo);
?>
```

8.2.5 Puntero de un archivo

Funciones relacionadas

- rewind(): situa el puntero del archivo al principio del mismo
- ftell(): nos devolverá la posición actual del puntero al archivo dentro del mismo.
- fseek(): nos permite desplazarlo hacia una posición exacta.
- feof(): informa de si el puntero al archivo se encuentra al final del mismo.

Un puntero de archivo (file pointer o handle) es una variable que hace referencia a un archivo. Es una variable que apunta a un archivo en concreto, y normalmente se obtiene cuando se abre con ***fopen()***.

PHP y su recolección de basuras cierra todos los punteros de archivos al final de la ejecución del script, aunque se considera una buena práctica cerrar los archivos manualmente con ***fclose()***.

Además de apuntar a un archivo, apunta a una posición concreta en ese archivo. En la mayoría de los casos cuando se abre un archivo el puntero apunta al principio (posición 0) o al final del archivo.

Feof()

La función ***feof()*** es utilizada con frecuencia en el manejo de archivos en PHP. Esta función comprueba si el puntero se encuentra al final del archivo. Se utiliza cuando se recorre un archivo línea por línea o para la lectura de grandes archivos, mediante un condicional:

```
$archivo = "miarchivo.txt";
# Abrimos el archivo
$fp = fopen($archivo, "r");
# Loop que parará al final del archivo, cuando feof sea true:
while(!feof($fp)){
   echo fread($fp, 4092);
}
```

El código anterior sólo cargará 4kb de datos de vez, lo que reduce el uso de memoria para grandes archivos.

Seeking es mover el puntero de un archivo. Para mover el puntero se puede usar la función ***fseek()***, y para mostrar la posición de un puntero dado ***ftell()***.

Vamos a ver algunos ejemplos:

```
$file = "miarchivo.txt";
$texto = "Hola que tal 12345";
$fp = fopen($file, "w");
fwrite($fp, $texto);
fclose($fp);
```

Ahora abriremos el archivo para ir viendo ***fseek*** y ***ftell***:

```
$fp = fopen($file, "r");

# Si lo hemos abierto con r, siempre empieza desde el principio:
Echo ftell($fp) . "<br>"; // Devuelve 0

# Colocamos el apuntador en la posición 10:
fseek($fp, 10);

# Mostramos la posición actual:
echoftell($fp) . "<br>"; // Devuelve 10

# Se puede indicar una posición sin contenido:
fseek($fp, 1000);
echo ftell($fp) . "<br>"; // Devuelve 1000

# Para ir al final del archivo se emplea un tercer argumento en fseek:
fseek($fp, 0, SEEK_END);
echo ftell($fp) . "<br>"; // Devuelve 18

# Para mover el apuntador a una posición relativa se emplea SEEK_CUR:
```

```
fseek($fp, -5, SEEK_CUR);
echo ftell($fp) . "<br>"; // Devuelve 13
```

No es necesario emplear ***fseek()*** para mover el puntero, también se puede hacer cuando se lee o se escribe un archivo:

```
# Cambiar el puntero leyendo un archivo:
$file = "miarchivo.txt";
$fp = fopen($file, "r");

# Leemos 10 bytes
$datos = fread($fp, 10);
Echo ftell($fp); // Devuelve 10

# Cambiar el puntero escribiendo un archivo:
$file = "miarchivo.txt";
$texto = "12345";
$fp = fopen($file, "w");

# Escribimos los 5 bytes del texto:
fwrite($fp, $texto);
echo ftell($fp); // Devuelve 5
```

8.2.6 Obtener información de un archivo

Algunas funciones que nos devolverán información sobre un archivo son:

- filesize(): devuelve el tamaño de un archivo en bytes.
- filetype(): devuelve el tipo de fichero de que se trata: file, dir, block, char, fifo, link o unknown.
- filemtime(): devuelve la fecha y hora de la última modificación del archivo, en formato UNIX Timestamp.
- fileperms(): devuelve la mascara de permisos (en sistemas UNIX).
- stat() y fstat(): devuelven información sobre un archivo abierto con fopen(). La diferencia entre ellas es que a stat() se le pasa como parámetro el nombre del archivo, y a fstat() el recurso obtenido con fopen().

Otras funciones con las que podemos obtener información sobre los archivos son:

- file_exists(): comprueba la existencia de un archivo
- is_executable(): devuelve true si el archivo es ejecutable.
- is_readable(): comprueba si existe un archivo se puede leer.
- is_writable(): comprueba si existe un archivo y se puede escribir en él.
- is_file(): indica si el nombre pasado como parámetro es un archivo.
- is_link(): indica si el nombre pasado como parámetro es un enlace simbólico.
- is_dir(): indica si el nombre pasado como parámetro es un directorio.

Stat()

Se puede obtener información de un archivo además de su contenido: tamaño, última vez que se ha accedido o modificado, número de links, etc. La función principal para obtener esta

información es con la función ***stat()***, en esta tabla se pueden ver los 12 elementos que devuelve el array.

```
$file = "miarchivo.txt";
$texto = "Todos somos muy ignorantes, lo que ocurre es que no todos ignoramos
las mismas cosas.";

$fp = fopen($file, "w");
fwrite($fp, $texto);

$datos = stat($file);

echo $datos[3] . "<br>"; // Número de enlaces, 1
echo $datos[7] . "<br>"; // Tamaño en bytes, 85
echo $datos[8] . "<br>"; // Momento de último acceso, 1444138104
echo $datos[9] . "<br>"; // Momento de última modificación, 1444138251
```

8.2.7 Copiar, renombrar, mover y eliminar archivos

Para copiar y eliminar archivos disponemos de las funciones ***copy()*** y ***unlink()***, respectivamente.

Hay que tener en cuenta que al copiar un archivo a otro directorio, si existe uno con el mismo nombre será sobrescrito.

Si lo que necesitamos es renombrar o mover archivos, en ambos casos utilizaremos la función ***rename()***.

Todas estas funciones devuelven true o false si ha ocurrido algún error.

```
<?php
    # Copiar el archivo a otra ruta;
    copy("datos.txt", "c:\\datos.txt");

    # Copiar el archivo con otro nombre:
    copy("datos.txt", "datos-2.txt");
    copy("datos.txt", "datos-3.txt");
    copy("datos.txt", "datos-4.txt");

    # Renombrar el archivo:
    rename("datos-2.txt", "datos---2.txt");

    # Renombrar carpetas:
    rename("miCarpeta1", "miCarpeta1-1");
    rename("./miCarpeta2", "./miCarpeta2-2");

    # Mover el archivo:
    rename("datos-3.txt", "c:\\datos-3-3.txt");

    # Eliminar el archivo:
    unlink("datos-4.txt");

    echo "Proceso finalizado";
?>
```

8.3 Gestión de Directorios

8.3.1 Funciones de directorio

Las funciones de directorios vienen de la extensión directories de PHP. Hay un total de 9 funciones disponibles:

- Chdir() — Cambia de directorio
- chroot() — Cambia el directorio raíz
- closedir() — Cierra un gestor de directorio
- dir() — Devuelve una instancia de la clase Directory
- getcwd() — Obtiene el directorio actual
- opendir() — Abre un gestor de directorio
- readdir() — Lee una entrada desde un gestor de directorio
- rewinddir() — Regresar el gestor de directorio
- scandir() — Enumera los ficheros y directorios ubicados en la ruta especificada

8.3.1.1 chdir()

Cambia el directorio actual

```
Bool chdir (string $directory)
```

8.3.1.2 getcwd()

Obtiene el directorio actual

```
String getcwd ( void )
```

```
echo getcwd() . "\n"; // Directorio: /directorio/actual
chdir('nuevo/directorio');
echo getcwd() . "\n"; // Directorio: /directorio/actual/nuevo/directorio
```

8.3.1.3 scandir()

```
array_scandir (string $directory [, int $sorting_order =
SCANDIR_SORT_ASCENDING [, resource $context ]] )
```

Devuelve un array con los archivos y directorios que se encuentran en $directory. El _$sortingorder indica el orden en que devolverá el listado: SCANDIR_SORT_ASCENDING (0), SCANDIR_SORT_DESCENDING (1) o SCANDIR_SORT_NONE (sin orden):

```
$directorio = "Slim";
$archivos = scandir($directorio, 1);
print_r($archivos);
/*
Array
(
    [0] =>View.php
    [1] =>Slim.php
    [2] =>Router.php
```

```
    [3] => Middleware
    [4] =>LogWriter.php
    [5] =>Log.php
    [6] => Http
    [7] =>Helper
    [8] =>Environment.php
    [9] =>..
    [10] => .
)
 */
```

8.3.1.4 Chroot()

```
Bool chroot (string $directory)
```

Cambia el directorio raíz al directorio ***$directory***. Requiere privilegios de administrador. Es necesario tener en cuenta que el directorio que se indique ha de estar preparado incluyendo los archivos necesarios para ser un directorio root.

8.3.1.5 Opendir()

```
Resource opendir (string $path [, resource $context ] )
```

Abre un gestor de directorio para ser usado en llamadas posteriores como ***closedir()***, ***readdir()*** y ***rewinddir()***.

8.3.1.6 Readdir()

```
String readdir ([ resource $dir_handle])
```

Lee una entrada desde un gestor de directorio. ***$dirhandle*** es el gestor de directorio previamente abierto por ***opendir()***. Si no se especifica se usa la última conexión abierta por ***opendir()***. Devuelve el nombre de la siguiente entrada del directorio.

8.3.1.7 Closedir()

```
Void closedir ([resource $dir_handle ])
```

Cierra un gestor de directorio abierto $dirhandle. Si no se especifica se asumirá la última conexión abierta por opendir().

Ejemplo de las funciones ***opendir***, ***readdir*** y ***closedir***:

```
if ($gestor = opendir('Slim')) {

    echo "Gestor de directorio: $gestor\n";
    echo "Entradas:\n";

    # Iteramos sobre el directorio:
    while ( $entrada = readdir($gestor)) {
          echo "$entrada\n";
    }
    closedir($gestor);
}
# Devuelve todos los archivos del directorio especificado
```

8.3.1.8 Rewinddir()

```
Void rewinddir ([resource $dir_handle])
```

Restablece la secuencia de directorio indicada por $dirhandle al comienzo del directorio. De nuevo, si no se especifica el gestor, se asumirá la última conexión abierta por opendir().

8.3.2 Función Glob()

La función glob () devuelve un array de nombres de archivos o directorios que coinciden con un patrón especificado.

```
Array glob ( string $pattern [, int $flags = 0 ] )
```

La gran ventaja de esta función con respecto a ***scandir()*** es que permite especificar un patrón para los elementos a listar.

En el siguiente ejemplo veremos cómo mostrar sólo los archivos con extensión ".txt" del directorio upload usando ***scandir()***.

```
if ($gestor = opendir('uploads/')) {
while ($archivo = readdir($gestor)) {
    if (strpos($archivo, ".txt") !== false) {
        echo "$archivo\n";
    }
}
closedir($gestor);
}
```

En el siguiente ejemplo queremos listar los archivos con extensión ".png" que se encuentran en la carpeta /home/DaPa/images/ usando la función ***glob()***

```
$directorio = "/home/DaPa/images/";
$archivos = glob("" . $directorio . "*.png");
foreach($archivos as $archivo) {
   echo "$archivo\n";
}
```

Veamos otro ejemplo básico del uso de ***glob()***

```
<?php
    # Cambiar el directorio actual:
    chdir("c:\\");

    # Mostrar el contenido del directorio actual:
    foreach( glob("*.*") as $archivo )
      echo $archivo."<br />";

?>
```

8.3.3 Crear, renombrar, eliminar y cambiar de directorio

Ya sabemos que la función de PHP ***getcwd()*** nos devuelve el directorio actual, y con ***chdir()*** podremos cambiar de directorio.

Para crear un directorio usaremos la función ***mkdir()***, indicando como primer parámetro su nombre y como segundo parámetro (opcional) los permisos que deberá tener (sólo en sistemas UNIX, no es aplicable en Windows).

Asimismo, podremos también crear directorios de forma recursiva asignando el valor true como tercer parámetro.

Si deseamos eliminar un directorio vacío cuyos permisos lo permitan, usaremos la función ***rmdir()***

```
?php
    // Obtener el directorio actual:
    echo "El directorio actual es: [".getcwd()."]<br />";

    // Cambiar el directorio actual:
    chdir("c:\\");

    // Obtener el directorio actual:
    echo "Ahora el directorio actual es: [".getcwd()."]<br />";

    // Crear directorios:
    mkdir("miCarpeta58975-01-1");
    mkdir("./miCarpeta58975-02-1");
    mkdir("./miCarpeta58975-03-1/miCarpeta58975-03-2/miCarpeta589752-03-3/",
null, true);

    // Renombrar directorio
    rename("miCarpeta58975-01-1", "miCarpeta58975--01--1");

    // Borrar un directorio (no borra los subdirectorios):
    rmdir("./miCarpeta58975-02-1");
?>
```

Otra función útil es ***is_dir()***, que nos informará de si el nombre pasado como parámetro corresponde a un directorio.

8.3.4 Otras funciones útiles sobre directorios

Otras funciones útiles para el manejo de directorios son:

- dirname(): devuelve el directorio padre de la ruta pasada como parámetro.
- is_dir(): indica si el nombre pasado como parámetro es un directorio.
- pathinfo(): devuelve información sobre una ruta de archivo.

8.4 Utilidades

8.4.1 Subir archivos al servidor

Para hacer ***upload***, es decir, subir archivos al servidor web, en primer lugar debemos crear un formulario HTML con los atributos ***method="post"*** y ***enctype="multipart/form-data"***, y que contenga un componente de tipo ***<input type="file" ... />***.

Tras seleccionar un archivo y enviarse el formulario, la información del archivo subido quedará guardada en el array asociativo ***$_FILES***, que tiene las siguientes claves:

- ***name***: nombre original del archivo.
- ***tmp_name***: nombre temporal del archivo subido junto con la ruta temporal (los archivos subidos se guardan en un directorio temporal).

- ***type***: tipo de archivo.
- **error**: código de error, si sucedió alguno.
- ***size***: tamaño del archivo en bytes.

Para acceder al archivo subido utilizaremos las funciones de PHP ***is_uploaded_file()*** (para comprobar si el archivo fue subido al directorio temporal usando el método POST) y ***move_uploaded_file()*** (mueve el archivo temporal hacia la ruta actual).

Upload.html

```
<html>

<head>
<title>Enviar E-Mail desde PHP | informaticapc.com</title>
</head>

<body>
<form name="frmUpload" action="upload.php" method="post" enctype="multipart/form-data">

   Archivo: <input type="file" name="txtFile" id="txtFile" />

   <input type="submit" name="btnSubmit" value="Enviar" />

</form>
</body>

</html>
```

Upload.php

```
<?php
if( empty($_FILES['txtFile']['name']) == false )
    {
    if (is_uploaded_file($_FILES['txtFile']['tmp_name']))
        {
        if( move_uploaded_file($_FILES['txtFile']['tmp_name'],
$_FILES['txtFile']['name']) == false )
            echo "No se ha podido el mover el archivo.";
        else
            echo "Archivo [".$_FILES['txtFile']['name']."] subido y movido al
directorio actual.";
        }
    else
        {
         echo "Posible ataque al subir el archivo
[".$_FILES['txtFile']['nombre_tmp']."]";
        }
    }
else
    {
     echo "No se seleccionó ningún archivo.";
    }
?>
```

Hay que tener en cuenta que si no se selecciona ningún archivo, el array **$_FILE** contendrá un elemento con un valor en la clave **error** de '4': dicho código informa de que no se subió ningún archivo.

8.4.2 Archivos CSV

Los archivos de texto delimitado o csv son archivos de texto normales en los que se utiliza un determinado carácter para separar cada campo.

En el siguiente ejemplo de archivo CSV los datos de cada línea se encuentran separados mediante un punto y coma:

```
Jose;Mena;Pérez;http://www.josemenap33982.com/
Fernando;Ruiz;Morote;http://www.fernandorm38347.com/
Marta;Rodríguez;Perdomo;http://www.martarp37294.com/
María;Perdomo;Mena;http://www.mariapmd82744.com/
Roberto;Valido;Morote;http://www.robertovm3873624.com/
```

Para el manejo de archivos CSV disponemos de las funciones de PHP ***fgetcsv()*** y ***fputcsv()***, con las que podremos leer y escribir en ellos respectivamente, si bien podemos usar también las explicadas anteriormente como fwrite(), fread(), etc.

La función ***fgetcsv*** obtiene una línea del puntero a un archivo y la examina para tratar campos CSV. La función ***fputcsv*** da formato a una línea como CSV y la escribe en un puntero a un archivo.

A continuación te mostramos un ejemplo de cómo escribir en un archivo pasando como parámetro a **fputcsv()** tanto una cadena de texto como un array, que contienen los datos que se desean escribir:

```
<?php
    $linea  = "Antonia,Martel,Calvo,http://www.antoniamc74924.com/";
    $aDatos = array("Rosa", "Castellano", "Herrera",
"http://www.rosach3729023.com/");

    // Abrimos el archivo situando el puntero al final del archivo:
    $archivo = fopen( "datos.csv", "ab" );

fputcsv( $archivo, explode(",", $linea), ";" );
fputcsv( $archivo, $aDatos, ";" );

fclose( $archivo );
?>
```

El archivo CSV quedaría como el que se muestra al principio de esta página.

Observa que como tercer parámetro podemos indicar el **carácter delimitador** que debe escribirse en el archivo CSV.

En caso de que hubiese **espacios en un campo** éste será escrito entre comillas dobles en el archivo de texto, como puedes ver en el siguiente ejemplo:

```
<?php
    $linea  = "Antonia;Martel H;C/Cuadrada,8;http://www.antoniamh74924.com/";
    $aDatos = array("Pedro", "Ramírez Quevedo", "C/Redonda,7",
"http://www.pedrorq89876.com/");

    // Abrimos el archivo (si existe será sobreescrito):
    $archivo = fopen( "datos.csv", "w+b" );
```

```
fputcsv( $archivo, explode(";", $linea), ";" );
fputcsv( $archivo, $aDatos, ";" );

fclose( $archivo );
?>
```

Si queremos usar otro carácter podemos indicarlo en un cuarto parámetro (opcional):

```
<?php
    $aDatos = array("Rosa", "Castellano Herrera", "",
"http://www.rosach3729023.com/");

    // Abrimos el archivo situando el puntero al final del archivo:
    $archivo = fopen( "datos.csv", "ab" );
fputcsv( $archivo, $aDatos, ";", "-" );
fclose( $archivo );
?>
```

Si lo que deseamos es que no se use ningún carácter podemos utilizar la función **fwrite()**.

Para obtener los datos de un archivo csv o delimitado usaremos **fgetcsv()** pasándole como parámetro un puntero al archivo abierto con **fopen()** y como segundo parámetro (opcional) el número de caracteres a leer, debiendo ser un número mayor que la longitud de la línea más larga (no se leerán caracteres de otra línea). En PHP 5 este parámetro ya no es necesario (aunque será ligeramente más lento).

Como tercer y cuarto parámetros (opcionales también) podemos indicar el carácter delimitador y el carácter de cierre, respectivamente.

Si **fgetcsv()** encuentra una línea en blanco se devolverá un array conteniendo un sólo campo con valor **null**, no siendo considerado como un error.

Veamos a continuación un ejemplo:

```
<?php
    $archivo = fopen( "datos.csv", "rb" );

    // Leer la primera línea:
    $aDatos = fgetcsv( $archivo, 100, ";");
    print_r( $aDatos );

    echo "<br />";

    // Tras la lectura anterior, el puntero ha quedado en la segunda línea:
    $aDatos = fgetcsv( $archivo, 100, ";" );
    print_r( $aDatos );
    echo "<br />-------------------------------------<p />";

    // Volvemos a situar el puntero al principio del archivo:
    fseek($archivo, 0);

    // Recorremos el archivo completo:
    while( feof($archivo) == false )
    {
        $aDatos = fgetcsv( $archivo, 100, ";");

    echo "Nombre: ".$aDatos[0]."<br />";
    echo "Apellido 1: ".$aDatos[1]."<br />";
```

```
    echo "Apellido 2: ".$aDatos[2]."<br />";
    echo "WEB: ".$aDatos[3]."<br />";

    echo "-------------------------<br />";
    }

fclose( $archivo );
?>
```

Finalmente veamos otro ejemplo de uso de la función ***fgetcsv()***

```
<?php

$path = "/home/usr/data3.csv";
if (!file_exists($path))
   exit("File notfound");
$file = fopen($path, "r");
echo "<html><body><tableborder=1>";
echo
"<tr><th>Country</th><th>Area</th><th>Population</th><th>Density</th></tr>";

while ($fields = fgetcsv($file,",")) {
     echo "<tr><td>".$fields[0]."</td><td>".$fields[1]."</td><td>".
         $fields[2]."</td><td>".$fields[3]."</td></tr>";
}
echo "</table></body></html>";
fclose($file);

?>
```

8.4.3 Comprimir y Descomprimir Archivos

Desde PHP podremos comprimir y descomprimir archivos y carpetas de forma muy sencilla, pudiendo hacerlo en los principales formatos de compresión de archivos: gz, rar, zip, etc.

Para la creación de archivos ZIP se usa la clase **ZipArchive** que está integrada en PHP.

Los métodos más interesantes de esta clase son los siguientes:

- ***Open(string $filename [, int $flags])***. Abre un archivo en formato ZIP
- ***Close()***: Cierra fichero abierto o creado y guarda los cambios. Este método es automáticamente llamado al finalizar el script.
- ***addFile(string $filename [, string $localname = NULL [, int$start = 0 [, int $length = 0]]])*** Añade un fichero al archivo ZIP para la ruta dada. De sus parámetros los más útiles son:
  - ***Filename:*** La ruta del fichero a añadir
  - ***Localname:*** Si corresponde, este es el nombre local dentro del archivo ZIP que reemplazará el Filename
- ***extractTo(string $destination [,mixed $entries]).*** Extrae el archivo completo o los ficheros dados en la ruta que se especifique.
  - ***destination***:Destino en donde extraer los ficheros.

- ***entries***: Las entradas a extraer. Acepta tanto un solo nombre o un array de nombres.

8.4.3.1 Crear archivo comprimido

En el siguiente ejemplo puedes ver cómo crear un archivo comprimido en formato ZIP utilizando la Clase ***ZipArchive***:

```
<?php
    $archivo_origen1 = "logo.jpg";
    $archivo_origen2 = "archivo.html";
    $archivo_zip = "comprimido.zip";

    // Creamos una instancia de la clase ZipArchive:
    $zip = new ZipArchive();

    // Creamos el archivo zip:
if ($zip->open($archivo_zip, ZipArchive::CREATE) === true )
    {
        // Añadimos archivos:
        $zip->addFile( $archivo_origen1 );
        $zip->addFile( $archivo_origen2 );

        // Cerramos el archivo zip:
        $zip->close();

echo "Proceso finalizado";
    }
else
    {
echo "Ha ocurrido un error";
    }
?>
```

En caso de que el archivo "comprimido.zip" exista en el directorio se machacaría. Podríamos usar entonces

```
if (  is_file($nombreZip)  ) {
      exit("El archivo ya existe");
}
```

Podemos cambiar el nombre del archivo que estamos comprimiendo indicándolo en el segundo parámetro.

```
$miInstanciaZip->addFile("miarchivo.jpg", "nuevoNombre.jpg");
```

8.4.3.2 Descomprimir Archivo

Para descomprimir todos los archivos usaremos el método ***extractTo()*** (si la carpeta en la que se desea extraer los archivos no existe será creada):

```
<?php
    $archivo_zip = "comprimido.zip";

    // Creamos una instancia de la clase ZipArchive:
    $zip = new ZipArchive();

    // Abrimos el archivo zip:
if( $zip->open($archivo_zip) === true )
    {
```

```
        $zip->extractTo('./temp/');
        $zip->close();

echo 'Archivo descomprimido';
    }
else
    {
echo 'Error al descomprimir';
    }
?>
```

Si quisiéramos extraer sólo determinados archivos, como segundo parámetro pasaríamos un array con sus respectivos nombres y extensiones.

```
<?php
$zip = new ZipArchive();
if ($zip->open('test_im.zip')) {
    $zip->extractTo('/my/destination/dir/', array('pear_item.gif',
'testfromfile.php'));
    $zip->close();
echo 'ok';
} else {
echo 'error';
}
?>
```

8.4.3.3 Crear Carpeta en Archivo Comprimido

Podemos crear una carpeta dentro de un archivo comprimido para ello se usa la función ***addEmptyDir()***

```
<?php
// Creamos un instancia de la clase ZipArchive
 $zip = new ZipArchive();
// Creamos y abrimos un archivo zip temporal
 $zip->open("miarchivo.zip",ZipArchive::CREATE);
 // Añadimos un directorio
 $dir = 'miDirectorio';
 $zip->addEmptyDir($dir);
 // Añadimos un archivo en la raiz del zip.
 $zip->addFile("imagen1.jpg","mi_imagen1.jpg");
 //Añadimos un archivo dentro del directorio que hemos creado
 $zip->addFile("imagen2.jpg",$dir."/mi_imagen2.jpg");
 // Una vez añadido los archivos deseados cerramos el zip.
 $zip->close();
?>
```

8.4.3.4 Descarga Archivo ZIP

Para forzar su descarga creamos las cabeceras necesarias indicando que es un archivo de tipo zip y especificando su nombre, en este caso le hemos llamado miarchivo.zip.En la última línea eliminamos el archivo zip temporal que hemos creado.

```
header("Content-type: application/octet-stream");
header("Content-disposition: attachment; filename=miarchivo.zip");
// leemos el archivo creado
readfile('comprimido.zip');
unlink('miarchivo.zip');//Destruye el archivo temporal
```

Veamos otro ejemplo con enlace de descarga

Fichero descarga.php

```
<?php
header("Content-disposition: attachment; filename=Hipercubo.pdf");
header("Content-type: application/pdf");
readfile("Hipercubo.pdf");
?>
```

Mediante el siguiente enlace en otro documento

```
<a href="descarga.php">Descargar PDF</a>
```

Enlaces de interés

http://informaticapc.com/tutorial-php/crear-archivos.php

http://programacion-php.readthedocs.io/es/latest/Tutorial4_Archivos.md.html

https://diego.com.es/certificacion-php