-

TEMA 03. Formularios y tratamiento de ficheros. Parte 2

Desarrollo web entorno servidor
CFGS DAW

Francisco Aldarias
Raya
paco.aldarias@ceedcv.es

2019/2020

18/09/19

Versión:191025.0918

Licencia

Nomenclatura

A lo largo de este tema se utilizarán distintos símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

Importante

Atención

Interesante

Reconocimiento.

Reconocimiento a mi compañera del CEED Cristina Alvarez por sus apuntes del curso 2016-17.

INDICE

UD03. Ficheros. Parte 2

1. MANEJO DE FICHEROS

PHP permite guardar y recuperar información a través de los ficheros, para lo cual dispone de funciones y procedimientos que crean, destruyen o modifican su contenido.

Vamos a ver cómo podemos usar PHP para acceder al sistema de archivos del servidor con el objetivo de guardar o leer datos.

1.1 ABRIR UN FICHERO

Para acceder a un archivo primero es necesario abrirlo. Realizaremos las operaciones que sean necesarias y lo cerraremos.

Para abrir un archivo usaremos la función **fopen()** *que* tiene dos argumentos, el nombre del archivo a acceder y el modo de acceder a este. La sintaxis es:

fopen(ruta_al_archivo, modo_de_acceso)

La ruta del archivo es el nombre exacto del archivo que se desea abrir. Podemos usar en nuestros scripts previamente la función **chdir()** para conocer exactamente sus rutas.

El segundo parámetro es el método de apertura, que debe ponerse entre comillas. PHP ofrece los siguientes modos de acceso:

Modo de acceso	Descripción
r	Apertura de sólo lectura. Ubica el apuntador del archivo al comienzo del mismo.
r+	Apertura de lectura y escritura. Ubica el apuntador del archivo al comienzo del mismo.
a	Apertura de sólo escritura. Ubica el apuntador del archivo al final del mismo. Si no existe el archivo, intenta crearlo.
a+	Apertura de lectura y escritura. Ubica el apuntador del archivo al final del mismo. Si no existe el archivo, intenta crearlo.
w	Apertura de sólo escritura. Cualquier contenido del archivo será borado. Si el archivo no existe, intenta crearlo.

w+	Apertura de lectura y escritura. Cualquier contenido del archivo será borado. Si el archivo no exite intenta crearlo.

Tabla 1: Modos de acceso a ficheros en PHP

En la tabla vemos que aparece la palabra "**apuntador**". Esto es la posición del archivo en la que leemos o escribimos. Observamos que lo más habitual es lo situemos al principio para leer todo su contenido, o al final para ir añadiendo datos.

La función **fopen** devuelve un manejador o identificador que guardaremos en una variable porque la usaremos posteriormente para manipular el archivo. Si no ha podido acceder por alguna causa (permisos, ruta, memoria…) devuelve FALSE.

Veamos un ejemplo:

```
if (fopen("Programa.php", "r")
echo "<b> El archivo \"Programa.php\" existe y ha quedado abierto. </b>";
else
echo "<b> El archivo \"Programa.php\" no existe. </b>";
```

Otra forma de hacerlo es:

```
@fopen("Programa.php", "r") or die ("<b> El archivo no se pudo abrir.
</b><p>");
```

De esta manera más elegante, se evita con @ el error que se genera y se dejan de ejecutar las siguientes líneas del código.

Aprovechando este estilo, podemos obtener el nombre y el identificador del archivo en dos variables distintas:

```
<?php
$archivo1 = "Programa.php"; //se guarda el nombre del archivo
$id_fichero1 = @fopen("Programa.php", "r") or die("<b> El archivo no se pudo
abrir.</b><p>"); //se guarda el identificador del archivo
echo "<b> El archivo $archivo1 existe y ha quedado abierto en modo lectura.
</b><p>";
```

Aún así, nosotros no usaremos este modo ya que es de más difícil lectura y requiere mayor experiencia.

1.2 CERRAR UN FICHERO

Mientras hacemos operaciones con el archivo lo debemos mantener abierto. El sistema

operativo lo mantendrá bloqueado para que otros programas no puedan escribir ni destruir mutuamente lo que escriben.

Pero al terminar de trabajar con él, hay que cerrarlo para que el sistema operativo puede disponer de él de nuevo.

Para cerrar un archivo abierto se usa la función **fclose()** pasándole como parámetro la variable que contiene el manejador del archivo:

```
fclose(id_archivo)
```

Veámoslo con un ejemplo. Vamos a escribir un código que permita abrir un fichero en modo lectura y devuelva un mensaje indicando el éxito o fracaso de la misión. Cuando termine, debe cerrar el archivo:

```
<?php
$ruta = "utils.php";
$id_archivo = fopen($ruta, "r"); if ($archivo) {
      print "Archivo $ruta abierto para lectura.";
} else {
      print "No se pudo abrir el archivo: $ruta.";
}
fclose($id_archivo);
```

La ruta del fichero que abrimos puede ser absoluta o relativa. Vamos a usar la barra normal "/" para separar los directorios, ya que es la empleada en Linux y en Windows. PHP permite la barra y la contrabarra.

```
<?php
$ruta_absoluta = "c:/CursoPHP/htdocs/index.php";
$ruta_relativa = "../practicasPHP/config.php";
$id_archivo1 = fopen($ruta_absoluta, "r");
$id_archivo2 = fopen($ruta_relativa, "r"); fclose($id_archivo1);
fclose($id_archivo2);
?>
Incluso podemos abrir archivos que estén alojados en otros servidores, aunque lo
normal es que sólo tengamos permisos de lectura:
<?php
$url = "http://www.google.es/index.html";
$archivo = fopen($url, "r"); fclose($id_archivo);
?>
```

1.3 LEER UN FICHERO

Lo más habitual es que queramos leer un archivo. PHP ofrece muchas formas de hacerlo. Una de las más sencillas es mediante la función **fread()** que lee un número de caracteres de un archivo. En

conjunción con la función **filesize()** que nos devuelve el tamaño del archivo en bytes se puede usar para leer todo el archivo. La sintaxis es:

fread(archivo, tamaño)

Donde archivo es el nombre completo (ej. "programas.txt") o el identificador del archivo (el que devuelve *fopen()*).

Veamos un ejemplo. Vamos a leer el archivo "prueba.txt". Para que funcione debes crearlo antes en la misma carpeta que este script, con el contenido que quieras. Ahora vamos a leerlo:

```
<pre>
<?php
$archivo = fopen("prueba.txt", "r");  //abrimos el archivo
$tamano = filesize("prueba.txt");     //miramos cuánto ocupa
$texto = fread($archivo, $tamano);    //guardamos el contenido del archivo en $texto
echo $texto;              //mostramos por pantalla el contenido
fclose($archivo);  //como hemos acabado, cerramos el archivo
?>
</pre>
```

Otra función similar es **fgets()**, quien se diferencia sólo de *fread()* en que sólo lee una cadena hasta el final de línea (retorno de carro), aunque el número de caracteres indicado sea mayor. La función **feof()** indica la marca del final de archivo o retorno de carro.

1.4 ESCRIBIR EN UN FICHERO

Otra acción que querremos hacer habitualmente es añadir datos a un archivo. Para ello se usa una función muy sencilla fwrite(), que escribe en la posición en la que está el apuntador. Por lo general, si hemos abierto el archivo en modo "a" escribiremos al final del archivo.

La función fwrite() usa dos parámetros. El primero el manejador del archivo y el segundo la cadena que queremos escribir. Su sintaxis es:

```
fwrite(archivo, cadena_de_texto)
```

Vamos a hacer un ejemplo: escribiremos dos frases en un archivo:

```
<?php
$archivo = fopen("refranes.txt", "a");
fwrite($archivo, "Si las barbas de tu vecino ves cortar ...\r\n");
fwrite($archivo, "...pon las tuyas a remojar \r\n"); fclose($archivo);
?>
```

Fíjate que el fichero (refranes.txt) se crea al acceder a la página del script con el navegador.

La cadena **\r\n** sirve para insertar un salto de línea en el archivo en el sistema operativo *Windows*. En *Linux* se usa solo la cadena " **\n** ". Si lo que queremos es sobrescribirlo abriremos el archivo en modo "w", de tal forma que se borre el contenido al abrirlo y dispondremos de un fichero vacío .

1.5 CREAR UN FICHERO

Como se deduce del ejemplo del apartado anterior, para crear un archivo basta con abrirlo en modo “a”o “w”. Al abrirlo, si el archivo no existe se creará.

2. FICHEROS SEPARADOS POR COMAS

De igual manera que podemos leer o escribir en un fichero de texto, también podemos leer o escribir línea a línea en ficheros estándar CSV.

Los ficheros CSV (del inglés comma-separated values) son los llamados ficheros de valores separados por comas. Dentro se ordenan fila a fila listas de elementos separados por comas, como su nombre indica.

Todo ésto lo podemos manejar con el uso general de un fichero abriendo y cerrando ficheros como veniamos haciendo con **fopen** y **fclose**. Y ahora usaremos **fgetcsv** o **fputcsv** para leer línea a línea del fichero. Dejemos de tanto hablar de lo mismo y vamos a ver un ejemplo..

Supongamos que tenemos el siguiente fichero:

```
21,89,7,16,76,18,52,51,58,52
80,3,20,6,83,64,21,73,14,80
7,30,23,75,71,9,96,56,47,25
```

Tenemos 3 filas de 10 elementos separados por comas.

Supongamos ahora que tenemos el siguiente código PHP:

```php
<?php
$fichero = 'nombreFichero.csv';

// si el fichero lo abrimos para leer correctamente
if (($manejador = fopen($fichero, "r")) !== false) {
	echo "Leyendo el fichero: <br>n";
	// leemos hasta que se acaba fila a fila
	while (($arrayFila = fgetcsv($manejador, 1000, ",")) !== false) {
	// escribimos cada fila por pantalla
	for ($i = 0; $i < count($arrayFila); $i ++) {
		echo $arrayFila [$i] . " - ";
```

```
        }
        echo "<br>n";
        }
        fclose($manejador);
}

if (($manejador = fopen($fichero, "w")) !== false) {
        echo "Escribiendo en el fichero. <br>n";
        // escribimos 3 filas
        for ($i = 0; $i < 3; $i ++) {
        $arrayFila = null;
        // de 10 elementos cada fila
        for ($j = 0; $j < 10; $j ++) {
                // llenamos con números aleatorios
                $arrayFila [$j] = rand(1, 100);
        }
        //echo var_dump($arrayFila) . "<br>n";
        fputcsv($manejador, $arrayFila, ",");
        }

        fclose($manejador);
}
```

Si lo ejecutamos veremos por pantalla lo siguiente en un navegador:

Leyendo el fichero:

21 – 89 – 7 – 16 – 76 – 18 – 52 – 51 – 58 – 52 –

80 – 3 – 20 – 6 – 83 – 64 – 21 – 73 – 14 – 80 –

7 – 30 – 23 – 75 – 71 – 9 – 96 – 56 – 47 – 25 –

Escribiendo en el fichero.

También podemos ejecutarlo desde línea de comandos con 'php nombrefichero.php', para los que no tengan instalado un servidor web.

La función var_dump($parametro) nos muestra el contenido de la variable pasada como parámetro.

Documentación oficial: http://php.net/manual/es/function.fgetcsv.php

3. MANEJO DE DIRECTORIOS

PHP ofrece muchas funciones para manejar directorios, pero lo más probable es que en la

mayoría de los casos lo único que nos interese de un directorio es conocer los archivos que tiene. Una vez que conozcamos este dato podremos construir rutas relativas a sus subdirectorios y a su vez listarlos y así de forma sucesiva.

3.1 ELIMINAR UN FICHERO

Para borrar un fichero usamos la función **unlink()**. Ésta recibe como parámetro una ruta al fichero a borrar. En el caso de que no lo haya conseguido, por no tener permisos o sencillamente porque no existe el archivo, devuelve FALSE. La sintaxis es:

```
unlink(archivo)
```

Vamos a probarlo. El siguiente programa intenta borrar el archivo refranes.txt que habíamos creado antes y en caso de no conseguirlo muestra mensaje de error:

```
<?php
if (!unlink("refranes.txt")) {
echo "No se ha podido borrar el archivo.";
}
?>
```

3.2 ATRIBUTOS DE UN FICHERO

Para manejar de forma segura es conveniente que conozcamos sus atributos. Por ejemplo, antes de abrir un archivo para escritura convendría comprobar si tenemos permisos de lectura. Las siguientes funciones nos ayudarán en esta tarea.

Funciones	Descripción
file_exists()	Devuelve TRUE si existe el archivo por el que preguntamos
file_size()	Devuelve el tamñao en bytes del archivo
is_file()	Devuelve TRUE si el archivo es un fichero
is_dir()	Devuelve TRUE si el archivo es un directorio
is_readable()	Devuelve TRUE si el archivo se puede abrir para lectura
is_writeable()	Devuelve TRUE si el archivo se puede abrir para escritura

Tabla 2: Funciones que devuelven atributos de un fichero

Veamos un ejemplo de uso. Vamos a obtener las propiedades del archivo “prueba.txt” que habíamos creado en los apartados anteriores:

```
<?php
$ruta = "prueba.txt"; if (file_exists($ruta)) {
```

```
      echo "$ruta tiene un tamaño de ";
      echo filesize($ruta) . " bytes.<br>";
      if (is_file($ruta)) {
      echo "$ruta es un fichero.<br>";
      }
      if (is_dir($ruta)) {
      echo "$ruta es un directorio.<br>";
      }
      if (is_readable($ruta)) {
      echo "$ruta se puede abrir para lectura.<br>";
      }
      if (is_writable($ruta)) {
      echo "$ruta se puede abrir para escritura.<br>";
      }
} else {
      echo "$ruta no existe.";
}
```

3.3 ABRIR UN DIRECTORIO

De forma similar a como sucedía con los archivos, antes de trabajar con un directorio lo tendremos que abrir. Nos serviremos de la función **opendir()** que devuelve un manejador del directorio.

opendir(ruta_al_directorio)

3.4 LISTAR LOS ARCHIVOS DE UN DIRECTORIO

Para conocer los ficheros y subdirectorios que contiene un directorio se usa la función **readdir()**. Por cada vez que llamemos a esta función nos devolverá el nombre de cada archivo encontrado o FALSE, si no quedan más archivos en el directorio.

readdir(directorio)

Veámoslo en un ejemplo. Vamos a escribir un programa que lista todos los contenidos del directorio actual (referenciado mediante una ruta relativa ".") y que indica a su vez si son ficheros o directorios:

```
<?php
$directorio = opendir(".");
while ($archivo = readdir($directorio)) {
      if (is_file($archivo)) {
      echo "$archivo es un fichero.<br>";
      }
      if (is_dir($archivo)) {
      echo "$archivo es un directorio.<br>";
```

```
        }
}
```

4. BIBLIOGRAFÍA

Libros

- [CIB12] Cibelli, C. (2012): *PHP. Programación web avanzada para profesionales*, Marcombo - Alfaomega, Barcelona
- [DUR13] Duran, C. (2013): *Recuperación y utilización de la información proveniente del cliente web*
- [LOP12] López, M.; Vara, JM; Verde, J.; Sánchez, D.M.; Jiménez, J.J.; Castro, V. (2012): *Desarrollo web en entorno servidor*, RA-MA, Madrid

Web:

- Trabajando con ficheros CSV desde PHP https://jnjsite.com/trabajando-con-ficheros-csv-desde-php/