RESUMEN DE PHP IX – Lectura y Escritura de ficheros

Hasta ahora casi todo lo que hemos visto son algunos aspectos de PHP en lo referente al lenguaje, como:

- Manipular datos en variables, ya sean cadenas, enteros, flotantes, booleanos o arrays.
- Utilizar funciones definidas en la API de PHP y crear nuestras propias funciones.
- Utilizar operadores para realizar cálculos y las estructuras de control para cambiar el flujo de nuestros scripts.
- Hemos accedido a las datos que los usuarios envían a través de formularios con el método POST y GET.

Pero aún estamos muy lejos del camino de poder juntar todo lo aprendido para realizar aplicaciones webs dinámicas. En este capítulo vamos a ver como podemos manejar ficheros en el servidor con PHP.

En PHP5 existen un conjunto de funciones para manipular ficheros, no vamos a ver todas, solo las más utilizadas. Si queréis ver más ejemplos o una documentación más detallada como siempre podéis acudir a la documentación oficial en línea de PHP.

Apertura de ficheros

fopen()

Para abrir un fichero en PHP se utiliza la función **fopen()** que devuelve un recurso que apunta al fichero abierto. Los 2 parámetros que se pasan a esta función son los siguientes:

- *Nombre del fichero:* PHP5 puede acceder a ficheros locales o ficheros remotos mediante HTTP y FTP. El fichero que se quiere abrir debe tener los permisos adecuados.
- *Modo de apertura:* Especifica el tipo de acceso que se tendrá al fichero. Os incluyo la tabla de la especificación oficial para que veáis los diferentes modos de apertura:

Modo	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.
'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 el archivo no existe se intenta crear.
'a+'	Apertura para lectura y escritura; coloca el puntero al final del archivo. Si el archivo no existe se intenta crear.
'x'	Creación y apertura para sólo escritura; coloca el puntero al principio del archivo. Si el archivo ya existe, la llamada a **fopen()** fallará devolviendo **FALSE** y generando un error de nivel

Modo	Descripción
	E_WARNING. Si el archivo no exite se intenta crear.
'x+'	Creación y apertura para lectura y escritura; coloca el puntero al principio del archivo. Si el archivo ya existe, la llamada a **fopen()** fallará devolviendo **FALSE** y generando un error de nivel **E_WARNING**. Si el archivo no exite se intenta crear.

También se puede especificar **b** de binario o **c** de texto en combinación con los diferentes modos de apertura. Solo en sistemas Windows se diferencia entre archivos binarios y de texto, así que si se quiere lograr la mayor portabilidad del código sería más eficiente abrir los archivos en modo binario.

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

En este ejemplo abrimos un fichero en modo binario para leer y escribir al final del fichero, sin borrar el contenido. El tercer parámetro lo establecemos a *true* para que busque el fichero en el *include_path*. Utilizamos el operador @ para suprimir los posibles errores que muestre la función **fopen()** si algo va mal. Comprobamos si el puntero es *FALSE*, y en ese caso mostramos un mensaje de error.

```
<?php
  $fichero = @fopen('archivo.txt', 'a+b', true );
  if(!$fichero)
  {
          echo 'No se puede abrir el fichero.';
  }
?>
```

Escritura de ficheros

💻 fwrite() y fputs():

PHP cuenta con una serie de métodos para escribir en un fichero abierto para escritura. Podemos utilizar la función **fwrite()** o **fputs()**. A estas dos funciones que funcionan de la misma manera, ya que **fputs()** es un alias de **fwrite()**, se les tiene que pasar dos parámetros, el recurso apuntador del fichero y la cadena que quiere escribirse. El tercer parámetro es opcional e indica la longitud en bytes que se va a escribir. La función devuelve el número de bytes escritos o *FALSE* si hubo algún error.

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

Lectura de ficheros

PHP también cuenta con una serie de funciones para leer ficheros. Podemos utilizar por ejemplo la función **fread()**. Esta función toma dos parámetros, el recurso que apunta al fichero, y el número de bytes que queremos leer. Esta función es poco manejable cuando queremos buscar en un fichero, así que PHP cuenta con funciones más precisas para leer líneas del fichero apoyándose en otras funciones relacionadas.

💻 feof():

Esta función toma como parámetro el recurso que apunta al fichero y devuelve TRUE si el puntero se encuentra al final del archivo.

💻 fgets():

Esta función 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 la longitud establecida en el segundo parámetro opcional. Utilizar esta función es la forma ideal de ir analizando el fichero línea a línea para encontrar la información que estamos buscando.

Vamos a ver un ejemplo de lectura utilizando la función **fgets()** y **feof()** para leer un fichero línea a línea y mostrarlo en pantalla:

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

💻 fgetc():

Esta función es muy parecida a la anterior pero devuelve un solo carácter.

💻 readfile() y file_get_contents():

Estas dos funciones sirven para leer todo el fichero de una vez. La función **readfile()** toma como parámetro la ruta absoluta del fichero y muestra todo su contenido en pantalla sin la necesidad de haber abierto el fichero para lectura. La función **file_get_contents()** funciona de manera similar pero no imprime el contenido en la salida estándar sino que lo guarda en una variable de tipo cadena.

```php
<?php
  $archivo = file_get_contents('archivo.txt', true);
  echo $archivo;
?>
```

Cierre de ficheros

💻 fclose():

Cerrar un fichero es tan fácil como abrirlo. Para ello utilizamos la función **fclose()** que toma como parámetro el puntero al fichero que queremos cerrar.

```php
<?php
  $archivo = file_get_contents('archivo.txt', true);
  echo $archivo;
  fclose($archivo);
?>
```

Sistema de ficheros

PHP cuenta con una serie de funciones para manipular el sistema de ficheros, con ellas podemos copiar, renombrar y mover ficheros.

🡆 copy():

Esta función copia un archivo a un destino. Toma dos parámetros, la ruta al archivo que se quiere copiar, y una ruta donde se quiere copiar el fichero. Devuelve TRUE en caso de éxito o FALSE en caso contrario.

```php
<?php
	$archivo = 'archivo.txt';
	$nuevo_archivo = 'copia.txt';

	if (!copy($archivo, $nuevo_archivo)) {
		echo 'Error al copiar.';
	}
?>
```

🡆 rename():

Esta función renombra un archivo. Toma dos parámetros la ruta al archivo que se quiere renombrar, y el nuevo nombre del archivo.

```php
<?php
	$archivo = 'archivo.txt';
	$nuevo_archivo = 'nuevo.txt';

	if (!rename($archivo, $nuevo_archivo)) {
		echo 'Error al renombrar.';
	}
?>
```

↳ unlink():

Esta función borra un archivo. Toma como parámetro la ruta al archivo.

```php
<?php
	$archivo = 'archivo.txt';
	if (!unlink($archivo)) {
		echo 'Error al borrar.';
	}
?>
```

↳ file_exists():

Esta función comprueba si existe un fichero que se le pase como parámetro. Devuelve TRUE si existe, FALSE en caso contrario. Hay que tener en cuenta que el parámetro no es un puntero al fichero, sino el nombre del fichero.

```php
<?php
	$archivo = 'archivo.txt';

	if (file_exists($nombre_archivo)) {
		echo "El archivo $archivo existe";
	} else {
		echo "El archivo $archivo no existe";
	}
?>
```