2012

FICHEROS EN PHP

Lo distintos programas que utilizamos, necesitan guardar la información que manipulan en archivos. Estos archivos son guardados en un **formato** especial diseñado por la propia aplicación.

El concepto de **formato de archivo** es sencillo de comprender, puesto que en el ejemplo de un documento generado por una aplicación como el Word, éste debe guardar marcas que indiquen el fin de página, márgenes del documento, encabezados y pies, alineación de párrafos, saltos de línea, columnas, gráficos, texto como negritas, tipos de letra y tamaño de fuente, y un sin fin de particularidades que hacen en conjunto un documento de Word.

Todos los programas (en Windows) generan por lo tanto un archivo con un **formato**, que es especial y particular del programa, y a su vez, dicho formato suele ser comprimido de alguna forma, permitiendo que el tamaño del archivo sea reducido. En Windows este tipo de archivo se suele referenciar como archivos en **formato binario**.
PHP dispone de una serie de funciones que permiten manipular archivos de cualquier tipo. Esto significa que es posible leer un archivo generado por Word, un archivo .JPG o incluso un archivo ejecutable (.exe), pero por supuesto, lo que veremos será ilegible o incomprensible, puesto que no se conoce el formato (criterio) en que debe ser leído para interpretarlo.

Sin embargo, existe otro tipo de **formato de archivo** que es conocido como **formato de texto plano**, o simplemente **archivo de texto**. Este tipo de archivo es el que utilizamos al crear el código XHTML o PHP, y no contiene ningún tipo de marca especial, excepto, la marca de fin de línea y fin de archivo.

Estos archivos pueden ser generados y leídos por el Bloc de Notas de Windows o por cualquier *Editor de Texto*. Y lo diferente es que en este tipo de archivos, no existen conceptos como páginas, párrafos, alineación, tipo de letra, etc. Por eso los programas que trabajan con estos archivos en forma explícita, son conocidos o clasificados como *Editores de Texto*, y no *Procesadores de Textos* como lo es el Word.

Apertura de Ficheros

Antes de poder utilizar un archivo, es preciso abrirlo, incluso cuando éste aún no exista. Si bien esto puede resultar extraño, se debe a que en primera instancia es necesario asignar un **gestor del recurso o fichero**, es decir, un número que PHP asigna al fichero, para **identificarlo** y diferenciarlo de otros ficheros que podrían estar abiertos.
Este **gestor** o **identificador** es asignado en el momento de la apertura del archivo, y es importante puesto que cada vez que se desee realizar alguna tarea con el archivo (como escribir, leer, o añadir datos) deberá utilizarse dicho **identificador** o **gestor** del recurso o fichero.

La función que permite la apertura de un archivo o fichero es la siguiente:

```
$fp = fopen("NombreArchivo", "ModoApertura");
```

$fp es simplemente una variable que contendrá el valor devuelto por la función **fopen()**, es decir, el número de recurso asignado por PHP o *puntero del fichero (file pointer)*. Este valor será el utilizado en el resto de las funciones que permiten realizar operaciones con el archivo, como leer, escribir, etc.

NombreArchivo Es una cadena de texto o variable de tipo *string* que identifica el nombre del archivo.
El nombre del archivo puede contener el *path* (ruta de acceso) al mismo. Si se trata de un archivo que se encuentra dentro del ámbito de alcance por el servidor (es decir, dentro o por debajo de www), bajo plataforma Windows, es posible utilizar las barras **/** para separar el camino de acceso, al igual que se hace en Unix o Linux. Sin embargo, si el archivo se encuentra fuera del alcance del servidor y bajo plataforma Windows (por ejemplo, fuera y por encima de www), debe utilizarse las barras invertidas **** pero escapando las mismas para evitar interpretaciones de Windows. Esto se logra colocando **\\.**

```
$fp = fopen("c:\\datos\\info.txt", "r");
```

ModoApertura El modo de apertura es una cadena de texto o variable *string* que identifica el modo en que debe tratarse el archivo. En la siguiente tabla se puede apreciar los diferentes modos.

Modos de Aperturas para fopen()

modo	Descripción
r	Apertura para sólo lectura; ubica el puntero de archivo al comienzo del mismo. Este es el modo más usado para la lectura de un archivo.
r+	Apertura para lectura y escritura; ubica el puntero de archivo al comienzo del mismo.
w	Apertura para sólo escritura; ubica el puntero de archivo al comienzo de éste y lo trunca a una longitud de cero (elimina contenido). Si el archivo no existe, intenta crearlo.
w+	Apertura para lectura y escritura; ubica el puntero de archivo al comienzo de éste y lo trunca a una longitud cero (elimina contenido). Si el archivo no existe, intenta crearlo.
a	Apertura para sólo escritura; ubica el puntero de archivo al final del mismo. Si el archivo no existe, intenta crearlo. ***Permite agregar datos al final del archivo***.
a+	Apertura para lectura y escritura; ubica el puntero de archivo al final del mismo. Si el archivo no existe, intenta crearlo.

Puntero de Archivo
Cuando se abre un archivo, y a medida que se lee o graba sobre el mismo, un puntero interno apunta a la ubicación del registro en el que se encuentra el archivo. Es decir, imagine un archivo donde cada línea representa un registro de datos como el nombre, dirección, teléfono de una persona. Al leer una línea, el puntero interno se mueve una posición apuntando a la siguiente línea.

Fin de línea

Al igual que todos los archivos generados por diferentes aplicaciones utilizan marcas para manipular su información, los archivos de **formato texto** contienen una marca de **fin de línea**, esta marca difiere según la plataforma:

Unix **\n**
Win **\r\n**
Mac **\r**

Ejemplos de Apertura de Archivos

```php
<?php
   $fp = fopen("/home/rasmus/archivo.txt", "r");
   $fp = fopen("/home/rasmus/archivo.gif", "w");
   $fp = fopen("http://www.example.com/", "r");
   $fp = fopen("ftp://usuario:contrasenya@example.com/un_archivo.txt", "w");
   $fp = fopen("C:\\IncomingFiles\\un_archivo.txt", "w");
?>
```

Cierre de Ficheros

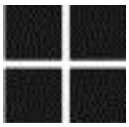
Todo fichero abierto debe ser cerrado luego de su uso, para permitir la liberación de los recursos utilizados. La forma de cerrar un archivo es muy sencilla, y siempre se hace referenciando el **identificador** o **gestor del fichero**.

```php
   fclose($fp);
```

Esta función devuelve **true** si el cierre pudo ser realizado, caso contrario devolverá **false**. En el ejemplo anterior **$fp** es una variable que contiene el **identificador** o **gestor del recurso o fichero**, que previamente fue asignado en forma automática por PHP al abrir el fichero con **fopen()**.

Ejemplos de Cierre de Archivos

```php
<?php
   $fp = fopen("/home/rasmus/archivo.txt", "r");
         ...código php...
   fclose($fp);
?>
```



2012

Lectura de Ficheros

En PHP existen varias funciones que permiten la lectura de un fichero, cada una tiene sus particularidades, sin embargo, veremos las más utilizadas para objetivos generales.

fgets() `fgets(id[,longitud]);`

Devuelve una línea del archivo o **FALSE** si ocurre un error.

id Es el puntero al archivo o identificador que fue asignado previamente por PHP al abrir el archivo con la función **fopen()**.

longitud La lectura finaliza cuando se han leído ***longitud* - 1** *bytes*, se alcanza un salto de línea (el cual se incluye en el valor devuelto), o en ***EOF*** (*end of file*), lo que ocurra primero. Si no se especifica una `longitud`, la función seguirá leyendo hasta que llegue al final de línea.

NOTA: Hasta PHP 4.3.0, al omitir este parámetro se asume 1024 como la `longitud` de línea. Si la mayoría de líneas en el archivo superan los 8KB (8192), es más eficiente en términos de recursos especificar la `longitud` máxima de línea en el *script*.
Se recomienda indicar siempre una longitud.

Ejemplo de Lectura de Archivos

```
<?php
   $fp = fopen("data/ficheros/frases.txt", "r");
   $fila = fgets($fp);
   echo "<span>$fila</span><br />\n";
   fclose($fp);
?>
```

Fin de Archivo (end of file – eof)

El final de un archivo esta determinado por una marca que puede ser obtenida con una función específica. Conocer esta marca permite leer todas las líneas de un archivo antes de cerrarlo.

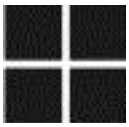
```
   feof(id);
```

Esta función devolverá **TRUE** si se llega al fin del archivo (*end of file* – ***eof***) apuntado por el identificador ***id***.

Ejemplo de Lectura de Archivos

```
<?php
   $fp = fopen("data/ficheros/frases.txt", "r");
   while (!feof($fp)) {
         $fila = fgets($fp,1024);
         echo "<span>$fila</span><br />\n";
   }
   fclose($fp);
?>
```

En el ejemplo anterior se puede apreciar la apertura de un fichero o archivo llamado *frases.txt*. Luego, mientras ***NO*** se llega al fin del archivo `(!feof($fp))`, se lee una línea y se imprime (observar que se especifica una longitud de 1024 caracteres, por lo que si la línea del archivo es mayor a esta cantidad, sólo se leerá 1023 caracteres. Por lo contrario, si la marca de fin de línea (\r\n) se encuentra antes, la línea es leída hasta la marca inclusive.



2012

Filtrado de Etiquetas

Si el archivo contiene etiquetas HMTL o PHP, serán ejecutadas aquellas que correspondan a PHP e interpretadas por el navegador (*browser*) las que correspondan a HTML. Si se desea eliminar de la interpretación cualquier etiqueta HTML y PHP, deberá utilizarse ***fgetss()***.

fgetss() `fgetss(id,longitud,etiquetas);`

Esta función es idéntica a ***fgets()*** con la única diferencia que devuelve una línea del archivo sin las etiquetas de HTML y PHP que pudieran existir, o **FALSE** si ocurre un error.

id Es el puntero al archivo o identificador que fue asignado previamente por PHP al abrir el archivo con la función **fopen()**.

longitud La lectura finaliza cuando se han leído ***longitud*** **- 1** *bytes*, se alcanza un salto de línea (el cual se incluye en el valor devuelto), o en ***EOF*** (end of file), lo que ocurra primero.

etiquetas Se trata de un *string* que especifica qué etiquetas se permiten, es decir, que etiquetas serán interpretadas.

Ejemplo de Lectura de Archivos

```
<?php
   $fp = fopen("data/ficheros/frases.txt", "r");
   $fila = fgetss($fp,1024,"<b><u>");
   echo "<span>$fila</span><br />\n";
   fclose($fp);
?>
```

En el ejemplo anterior se indican como etiquetas permitidas **<b>** y **<u>** (negritas y subrayado respectivamente en HTML). Por tanto, si el archivo contiene estas etiquetas, o una combinación de ellas, no serán ignoradas y serán interpretadas. Obsérvese además que se utilizan únicamente las etiquetas de apertura y no las de cierre.

Lectura de Ficheros por caracteres

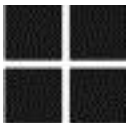
Existe una función que permite leer un archivo carácter a carácter, sin embargo el fin de archivo (**eof**) es leído también como un carácter y la función devuelve **FALSE** para dicha marca. A su vez, esta función además puede devolver un valor no-booleano que será evaluado como **FALSE**, como por ejemplo **0** (cero) o **“”** (cadena vacía). Por tal motivo se suele utilizar el *operador de comparación* ***idéntico*** **(===)** para comprobar el valor devuelto por esta función.

Ejemplo 1

```
<?php
    $fp = fopen("frases.txt", "r");
    do {
        $carac = fgetc($fp);
        echo $carac;
    } while (false !== $carac);
    fclose($fp);
?>
```

Ejemplo 2

```
<?php
  $fp = fopen("fichero.txt", "r");
  if (!$fp) {
     echo "o se pudo abrir el fichero.txt";
     exit;
  }
  while (false !== ($carac = fgetc($fp))) {
      echo $carac."\n";
  }
  fclose($fp);
?>
```



2012

Escritura de Ficheros

PHP permite además de leer archivos, la creación o escritura de ficheros. Resulta en ocasiones sumamente útil crear archivos con cierta información. Un ejemplo típico es crear archivos que mantienen información de configuración de un sistema, a modo de los archivos ***.ini*** de las que se valen varias aplicaciones de Windows.

La escritura de un fichero o archivo en PHP se realiza utilizando la función ***fwrite()***. Existe la función ***fputs()*** que es un *alias* de ***fwrite()***, es decir, es idéntica y equivalente.

```
fwrite(id, "cadena");
 fputs(id, "cadena");
```

id Es el puntero al archivo o identificador que fue asignado previamente por PHP al abrir el archivo con la función **fopen()**.

cadena Es un *string* con la cadena que se escribirá en el archivo, es decir, la línea de texto o información a escribir.

NOTA: el argumento o parámetro *cadena* debe incluir las marcas de fin de línea (\r\n). La marca de fin de archivo es creada en forma automática al cerrarlo.

Ejemplo de Escritura de Archivos

```
<?php
    $fp  = fopen("DataCli.txt","a");
    $reg = "Esta es una línea de texto a ser insertada en el archivo \r\n";
    fwrite($fp,$reg);
    fclose($fp);
    echo "<span>datos insertados</span>\n";
?>
```

En este ejemplo se graba una cadena de texto que representa simplemente una línea. Se puede observar como es incluida la marca de fin de línea (**\r\n**) al final de la misma. Esta marca en particular es la que utiliza *Windows*, recuerde que en *Unix* /*Linux* usa **\n** y Mac **\r**.

NOTA: La diferencia de las marcas de fin de línea para los diferentes sistemas, son importantes en el caso en que estos archivos sean leídos con *Editores de Textos* en cada sistema. Debido a que los archivos generados por nuestros programas, seguramente también sean leídos por programas que desarrollaremos, no es importante la marca que se utilice, incluso si desarrolla en plataforma Windows y luego es subido a un hosting con sistema Linux.

Ejemplo de Escritura de Archivos

```
<?php
    $Nums = array(10,40,100,3,50,300,31,5,4,9);
    $fp  = fopen("DataArray.txt","a");
    for ($i=0; $i<counts($Nums)) {
        $reg = $Nums[$i]."\r\n";
        fwrite($fp,$reg);
        echo "<span>Se insertó el valor: $Nums[$i] </span><br />";
    }
    fclose($fp);
?>
```

En el ejemplo anterior, se crea un archivo con cada valor del *array $Nums* al tiempo que se va imprimiendo el valor insertado. Se puede preciar como se agrega al final la marca de fin de línea **\r\n**.
Escriba y ejecute este script, luego abra el archivo .txt generado con el bloc de notas.

2012

Formato CSV - escritura

El formato CSV (del inglés *comma-separated values*) es un tipo de documento sencillo para representar datos en forma de tabla. Consiste en utilizar un separador entre los diferentes datos que conforman una línea, así, si por ejemplo se quiere almacenar información sobre productos, muy probablemente cada producto tendrá un número que lo identifique, una descripción y un precio. Estos tres datos conforman una línea en el archivo que representa un producto. Esto es conocido como *registro*, y cada dato que compone un *registro* como *campo*.

```
10,Televisor LCD 32i,1290
12,Televisor Plano 20i, 400
23, DVD Player MKi, 120
```

Aquí se puede apreciar un archivo con ***formato CSV***, el cual contiene cada valor separado por una coma (*Comma Separated Values*).

Esta forma de crear un archivo, es muy difundida para trasladar información entre diferentes aplicaciones. Por ejemplo, una aplicación de *Hojas de Cálculos* como *Microsoft Excel*, permite leer este tipo de archivos, y transformar cada línea (o registro) en una *fila* y cada valor separado por comas, en una *columna*.

Esto implica además, que se puede exportar una planilla de *Excel* como archivo en ***formato CSV*** y posteriormente ser procesada la información por un *script* PHP.

Las *celdas* en una *hoja de cálculo* pueden contener tanto valores fijos como *fórmulas*, y además la información en una celda puede ser de *tipo texto* o *tipo numérico*. Justamente que una celda contenga una fórmula, permite el cálculo del valor a presentar en la misma. Por tanto, al importar un archivo en ***formato CSV***, es posible que una celda calcule el valor recibido, en lugar de presentarlo.

Considere el siguiente ejemplo:

```
1020, Helen Cantero, 00598-2-480-1234
1021, Marcos Holst, 091-121-345
1024, Julian Mortensen, 090-333-458
```

En el caso anterior, al importar esta información, el último campo puede ser interpretado como una fórmula (la resta de los valores), en lugar de un número telefónico, como se pretende. Por este motivo, es muy común encontrar en un archivo de ***formato CSV*** valores entre comillas.

```
1020, “Helen Cantero”, “00598-2-480-1234”
1021, “Marcos Holst”, “091-121-345”
1024, “Julian Mortensen”, “090-333-458”
```

Cierto es también que es muy común utilizar como criterio general que todo dato que no represente explícitamente un valor numérico sea colocado entre comillas, como se puede apreciar en el ejemplo anterior.

Ejemplo de Escritura de Archivos CSV

```php
<?php
 // CREAR SEPARADORES
    $del = '"';     // delimitador de datos comillas dobles
    $sep = ",";     // separador de datos coma
    $eol = "\r\n";  // marca de fin de línea

 // ABRIR ARCHIVO
    $fp  = fopen("DataCli.csv","a");

 // CAPTURAR DATOS
    $nroCli  = $del.$_POST["nCli"].$del.$sep;
    $nombre  = $del.$_POST["NomCli"].$del.$sep;
    $Telefono= $del.$_POST["TelCli"].$del;

 // ARMAR REGISTRO
    $reg     = $nroCli.$nombre.$telefono.$eol;

 // ESCRIBIR REGISTRO
    fwrite($fp,$reg);

 // CERRAR REGISTRO

    fclose($fp);
?>
```

En este ejemplo se puede observar como el script captura datos de un formulario enviado por método ***POST***, concatenando con el delimitador definido en la variable ***$del*** (comillas), delante y después del valor ingresado en el formulario. También se concatena el separador (coma) definido en la varible ***$sep***.
Finalmente se arma el registro concatenando los valores que lo conforman, ya entrecomillados y separados con coma, finalizando el registro con la marca de fin de línea, definida en la variable ***$eol***.
Una vez armado el registro, se procede a grabarlo en el archivo por medio de la función ***fwrite()***, la cual recibe como primer parámetro el identificador devuelto por la función ***fopen()*** y el string (registro) a guardar. Finalmente se cierra el archivo.

Formato CSV – lectura

PHP provee una función especial para leer archivos en ***formato CSV***, se trata de la función ***fgetcsv()*** y es similar a la función ***fgets()***, con la diferencia que ésta devuelve el resultado en un *array* donde cada elemento del mismo representa un *campo* del registro del archivo ***CSV***.

```php
fgetcsv(id,longitud,separador,delimitador);
```

id Es el *identificador* de archivo devuelto por la función ***fopen()*** al momento de abrirlo.

longitud Debe ser mayor que la línea más grande (en caracteres) a ser encontrada en el archivo **CSV** (permitiendo los caracteres de fin de línea). Se volvió opcional en PHP 5. Omitiendo este parámetro (o poniéndolo en 0 en PHP 5.0.4 y superior) no hay tamaño máximo para una línea, lo cuál lo hace un poco lento. Por lo que se recomiendo ***siempre*** utilizar una longitud.

separador Es un valor opcional y fija el *delimitador* o *separador* del campo (sólo un caracter). El valor por defecto es la coma. Se recomienda ***siempre*** especificar este dato.

delimitador Es un valor opcional y fija el caracter de delimitación (sólo un caracter). El caracter por defecto son las dobles comillas. Se recomienda ***siempre*** especificar este valor.

NOTA: Una línea en blanco en un archivo ***CSV*** será regresada como un *matriz* que contiene un campo ***null*** y no será tratada como error.

Ejemplo de Lectura de Archivos CSV

```
<html>

 <head>
  <title>Ejemplo de lectura de archivo CSV</title>
 </head>

 <body>
  <h3>Ejemplo de lectura de archivo CSV</h3>

  <table border="1">
   <tr>
    <th>ID</th>
    <th>CLIENTE</th>
    <th>TELEFONO</th>
   <tr>

<?php
// CREAR SEPARADORES
   $del = '"';     // delimitador de datos comillas dobles
   $sep = ",";     // separador de datos coma
   $eol = "\r\n";  // marca de fin de línea

// ABRIR ARCHIVO
   $fp  = fopen("DataCli.csv","r");

// LEER ARCHIVO
   while (($reg = fgetcsv($fp, 1024, $sep,$del)) {

     echo "<tr>\n";
     echo " <td>$reg[0]</td>\n";
     echo " <td>$reg[1]</td>\n";
     echo " <td>$reg[2]</td>\n";
     echo "</tr>\n";
   }
   fclose($handle);
?>

  </table>
 <body>
<html>
```

En el ejemplo anterior, se utilizan variables para guardar los símbolos que representan el separador de datos, el delimitador y el fin de línea. Esto permite evitar usar estos símbolos al momento de especificarlos en la lectura del registro. Por otro lado, aportan un poco de claridad al código y contribuyen a un rápido mantenimiento del programa, puesto que en caso de decidir utilizar otros símbolos, bastaría con asignar los nuevos valores en estas variables, sin necesidad de cambiarlos en todos los programas.

El ***while*** contiene una línea especial que utiliza ***$reg*** como nombre del ***array*** que recibirá los datos del registro (línea) del archivo. Recuerde que la función ***fgetcsv()*** devuelve un ***array*** con tantas posiciones como datos separados por comas existan en el archivo. Por este motivo, para mostrar cada valor correspondiente a un dato (campo) del registro, es necesario acceder a cada posición del ***array $reg***. Así, ***$reg[0]*** corresponde al dato antes de la primer coma, ***$reg[1]*** al siguiente y así sucesivamente.
Observe en este ejemplo como se utiliza una tabla de XHTML para crear una presentación en pantalla acorde a los datos. Al llegar al fin del archivo (marca ***eof***), esta función devuelve falso, por lo que el ***while*** dejará de cumplir la condición y terminará el bucle.