

DESARROLLO WEB EN ENTORNO SERVIDOR

CAPÍTULO 7: Programación entorno servidor con PHP Archivos y Directorios

Trabajando con Archivos y Directorios en PHP

- Leer archivos

El entorno de PHP nos permite leer archivos en los siguientes casos:

- Una cadena de texto o en una matriz
- Desde un sistema de archivos local o un URL remoto, por líneas, bytes o caracteres

Trabajando con Archivos y Directorios en PHP

- Leer archivos locales

- La manera más sencilla de leer el contenido de un archivo de disco en PHP es con la función [file_get_contents\(\)](#). Acepta el nombre y la ruta de acceso de un archivo de disco, y lee el archivo completo de un solo golpe en una variable de cadena.

```
<?php
// lee el archivo en una cadena
$cadena = file_get_contents('ejemplo.txt') or die ('ERROR: No se
encuentra el archivo');
echo $cadena;
?>
```

(Copia éste código en un fichero nuevo y guárdalo como **prueba lectura.php**)

- Esta función se limita a leer el fichero no codifica caracteres.

Trabajando con Archivos y Directorios en PHP

- Leer archivos locales
- Un método opcional para leer datos de un archivo es la función **file()** de PHP, que acepta el nombre y la ruta de acceso de un archivo y lo lee en una matriz, donde cada elemento de ésta corresponde a una línea del texto en el archivo.

```
<?php  
// lee archivo en una matriz  
$matriz = file('ejemplo.txt') or die('ERROR: No se encuentra el  
archivo');  
foreach ($matriz as $lineaa){  
    echo $lineaa;  
}  
?>
```
- Esta función se limita a leer el fichero no codifica caracteres.

Trabajando con Archivos y Directorios en PHP

- Leer archivos Remotos

- Tanto `file_get_contents()` como `file()` también pueden leer datos de una URL, utilizando el protocolo HTTP o FTP.

```
<?php
$cadena = file_get_contents('http://www.marca.com/') or die ('ERROR:
No se encuentra el archivo');
echo $cadena;
?>
```

Varia tu ejercicio de prueba para leer el texto de ésta dirección

- En el caso de vínculos en redes lentas, resulta más eficiente leer un archivo remoto en “fragmentos”, para maximizar la eficiencia del ancho de banda disponible en la red. Para ello, utiliza la función `fgets()` con el fin de leer una cantidad específica de bytes del archivo.

Trabajando con Archivos y Directorios en PHP

- Leer segmentos específicos de un archivo

También es posible la lectura sólo de un bloque específico de líneas a partir de cierto punto, lo que puede realizarse con una combinación de las funciones **fseek()** y **fgets()**

Para que lo comprendas mejor crea el ejercicio 51 de tu juego de clase

- Observa las funciones empleadas:

- **fopen()**: acepta el nombre de la fuente del archivo y un argumento que indica si será abierto en modo solo lectura ('r'), solo escritura ('w') u otras opciones: consultar en php.net : y crea un puntero al archivo (enlace)
- **fgets()**, que lee una cantidad específica de bytes del archivo y los añade a una variable de cadena, es posible especificar el número de bytes a leer.
- **feof()** adquiere un valor *true* (verdadero), indicando que se ha alcanzado el final del archivo.
- **fclose()** destruye el puntero al archivo creado con fopen();

Trabajando con Archivos y Directorios en PHP

■ Escribir archivos

PHP cuenta con un par de maneras para crear y escribir ficheros.

- La primera es la función **file_put_contents()**, esta función acepta el nombre del archivo y su ruta de acceso, junto con los datos que se escribirán en el archivo y luego escribe estos últimos en el lugar indicado.

Para que lo comprendas mejor crea el ejercicio 52 de tu juego de clase

- Si el archivo especificado en la invocación de `file_put_contents()` ya existe en el disco, la función lo sobrescribirá, como opción por defecto.
- **NOTA:** El directorio, de ponerlo en la ruta de creación del fichero, debe existir previamente o te saldrá un error, podemos comprobar si existe dicho directorio con la función **file_exists()**

Trabajando con Archivos y Directorios en PHP

- Leer y evaluar archivos externos
 - Para leer y evaluar archivos externos desde tu script PHP, utiliza las funciones **include()** o **require()**.
 - Una aplicación muy común de estas funciones es incluir un encabezado estándar (*header*), *un pie de página (footer)* o *una nota de derechos de autor (copyright)* en todas las páginas Web de un sitio.

Trabajando con Archivos y Directorios en PHP

- Observa el siguiente código

```
<body>  
  <?php require('encabezado.php'); ?>  
  <p/>  
  Éste es el cuerpo de la página.  
  <p/>  
  <?php include('pie.php'); ?>  
</body>
```

- Ambas funciones admiten cualquier tipo de archivo.

Trabajando con Archivos y Directorios en PHP

- Diferencias entre `include()` y `require()`
 - Una función **`include()`** perdida generará una alerta, pero la ejecución del script continuará.
 - Una función **`require()`** perdida generará un error fatal que detendrá la ejecución del script.
 - La ruta del fichero especificada puede ser una URL siempre y cuando en la configuración de **PHP** se permite la inclusión de ficheros externos, esto se hace activando el la directiva [allow_url_fopen](#)

Trabajando con Archivos y Directorios en PHP

- Contenidos en ficheros externos.
 - Los archivos "a incluir" suelen tener la nomenclatura **nombre.inc.php** por mera organización de archivos.
 - Los scripts almacenados en los ficheros a incluir deben contener las etiquetas de entrada y salida. Si almacenamos código HTML generalmente no nos interesará añadir etiquetas de encabezado o estructurales ya incluidas en el documento llamante.
 - Los ficheros con contenido que no sean funciones pueden llamarse recursivamente desde diferentes partes del documento, los que contienen funciones NO. Se generaría un error por recurrencia.

Trabajando con Archivos y Directorios en PHP

- Sobre `include()` y `require()`
 - **`include_once("fichero")` y `require_once("fichero")`** van a **impedir** que un mismo fichero pueda incluirse dos veces, si tienes clara el número de llamadas al mismo fichero y si es única, ésta sería la forma más correcta de llamar a la fichero externo.
 - Los fichero a "incluir" conviene organizarlos por tipos de contenidos, es decir, funciones php por un lado y contenidos por otro.

Gracias por tu interés

Fin de la presentación