U.T. 2: Características del lenguaje PHP. Parte 4: Cabeceras HTTP, ficheros y directorios

Contenidos:

- Cabeceras HTTP: Introducción
- Ficheros y directorios
  - Apertura y cierre
  - Procesamiento de ficheros
  - Manejo de directorios

1.- Cabeceras HTTP: Introducción

- Cabeceras HTTP
  - Aportan información en peticiones y respuestas a servidores
  - Función para enviar cabeceras HTTP:

    header("cabecera: valor");
  - Ejemplos:
    - header("location: http://www.upm.es");
    - header("HTTP/1.0 404 Not Found");
    - header("Pragma: no-cache");
    - Otras: Cache-Control, Expires, Last-Modified,etc.
- Es importante que esta orden se encuentre antes de la etiqueta <html> inicial o debe activar output_buffering en php.ini.
- Las cabeceras HTTP pueden también modificar el comportamiento del navegador que recibe la respuesta

1.- Cabeceras HTTP: Introducción

- Enviar cabeceras HTTP en PHP: header()
  - Debe aparecer antes de enviar cualquier otra cosa al cliente (antes de cualquier echo).
  - header ('Content-Type: text/html; charset=UTF-8');
- Extraer cabeceras HTTP del cliente en PHP:

  apache_request_headers()

```
<?php
    var_dump(apache_request_headers());
?>
```

- Redirigir al cliente a otra dirección.

```
header ('Location: acceso_no_autorizado.php');
```

- Mostrar un mensaje y redirigir al cliente a otra dirección.

```
header ('Refresh: 5; url=http://www.google.es');
echo 'Lo que busca no existe, le redirigiremos a Google en 5 segundos'
```

- Ocultar la versión de nuestro intérprete PHP.

```
header( 'X-Powered-By: adivina-adivinanza' );
```

- Alternativamente también podríamos ocultar nuestra versión de PHP asignando el valor 0 a la directiva *expose_PHP* del php.ini.

1.- Cabeceras HTTP: Introducción

- Ofrecer la descarga de un archivo desde PHP.

Vamos a mostrar un pdf

```
header('Content-type: application/pdf');
header('Content-Disposition: attachment;
filename="downloaded.pdf"');
readfile('original.pdf'));
```

Proporciona un nombre de fichero recomendado y fuerza al navegador a mostrar el diálogo para guardar el fichero.

La fuente del PDF se encuentra en original.pdf

1.- Cabeceras HTTP: Introducción

- Generar contenidos diferentes a páginas HTML con PHP.
  - Imágenes, documentos PDF, películas SWF de Flash en tiempo real (sin leerlos de un archivo)
  - Es posible gracias a librerías de PHP como GD, PDFlib, Ming, etc.

```
header("Content-type:image/jpeg");
header("Content-Disposition:inline ; filename=captcha.jpg");
```

El fichero se abre en el navegador en lugar de descargarse (attachment)

2.- Ficheros y Directorios

- Un fichero
  - Es un almacén de datos en dispositivos externos
  - Convierte los datos en persistentes al final de los programas
- Tipos de ficheros:
  - TEXTOS:
    - La información se almacena en ASCII, es legible mediante editores, los datos requieren separadores (‘ ’, ‘\n’, ‘\t’,’\r\n’)
  - BINARIOS:
    - La información es binaria, NO es legible mediante editores, los datos NO requieren separadores

2.- Ficheros y Directorios

- Las operaciones sobre ficheros suelen constar de tres fases:
  - Apertura del fichero
    - Se abre el fichero, indicando si se realizarán operaciones para leer, escribir o añadir al final del mismo.
    - La operación devuelve un descriptor de fichero que se usará en el resto de funciones.
  - Procesamiento del fichero
    - Lectura
    - Escritura.
  - Cierre del fichero.

2.1.- Apertura y Cierre

- Apertura fopen()

  fopen(nombre_fichero, modo, include_path)

  - nombre_fichero:
    - local o remoto (“http://” o “ftp://”)
  - modo:
    - ‘r’ Sólo lectura. Puntero al inicio.
    - ‘r+’ Lectura/escritura. Puntero al inicio.
    - ‘w’ Sólo escritura. Puntero al inicio. Si existe el fichero borra lo que había, sino lo intenta crear.
    - ‘w+’ Lectura/escritura. Puntero al principio. Si existe el fichero borra lo que había, sino lo intenta crear.
    - ‘a’ Sólo escritura. Puntero al final. Si no existe lo intenta crear.
    - ‘a+’ Lectura/escritura. Puntero al final.
    - ‘x’, ‘x+’ Igual que w, w+ pero si existe da un error.
    - ‘c’, ‘c+’ Igual que w , w+ pero sin truncar el fichero cuando ya existe.

2.1.- Apertura y Cierre

- Apertura fopen()
  - Devuelve un **identificador** que se emplea en el resto de funciones (o FALSE en caso de error)
  - **Include_path** Especifica la lista de directorios donde las funciones require, include, fopen(), file(), readfile() y file_get_contents() buscarán ficheros.
  - **Include_path** = true -> el fichero debe buscarse en las rutas establecidoas en la directiva include_path de php.ini.
    - Ejemplo en Windows: include_path=".;c:\php\includes"
    - Ejemplo en Unix: include_path=".:/php/includes“
  - Por portabilidad, se recomienda encarecidamente que siempre use la bandera 'b' cuando se abran ficheros binarios con fopen().
- Cierre fclose()

fclose(identifcador)

2.1.- Apertura y Cierre

- ¿Para qué sirve el puntero que devuelve fopen()?
  - Define un canal a través del cual se accede al fichero.
  - Desde que el fichero está abierto se trabaja con el puntero.
  - Cuando se abre el fichero, el puntero se coloca al principio del fichero para esperar instrucciones.
- Ejemplos:
  - Apertura para lectura

```
$fichero = fopen("datos.txt", 'r');
```

  - Apertura para escritura silenciando errores con @

```
$fichero = @fopen("datos.txt", 'w');
```

  - Apertura para añadir, silenciando errores

```
$fichero = @fopen("datos.txt", 'a');
```

- Más ejemplos:
  - $gestor = fopen("/home/rasmus/fichero.txt", "r");
  - $gestor = fopen("/home/rasmus/fichero.gif", "wb");
  - $gestor = fopen("http://www.example.com/", "r");
  - $gestor = fopen("ftp://user:password@example.com/fichero.txt", "w");

2.1.- Apertura y Cierre

- Para verificar que la operación fopen() ha tenido éxito:
  - Apertura para lectura silenciando errores con el operador @

```
$fichero = @fopen("datos.txt", 'r');
```

  - Si no se ha podido abrir el fichero finaliza la ejecución del script devolviendo un mensaje de error:

```
if (!$fichero)
    die("ERROR: no se ha podido abrir el fichero de datos");
```

- Die() → Provoca la finalización de la ejecución del script mostrando el mensaje recibido como parámetro
- En caso contrario el script continúa ejecutándose.

2.2.- Procesamiento de ficheros

- Leer
  - string fgets ($puntero, [bytes])
    - Obtiene una línea desde el puntero del fichero
    - Se termina de leer cuando llega al final de línea, final del fichero o el último byte de datos
    - byte indica cuantos bytes (caracteres) queremos leer del fichero (opcional)
  - string fread ($puntero, longitud)
    - Es similar a fgets( ), pero se lee todo el fichero (o hasta el carácter longitud), no va línea por línea como fgets( ).
- Leer todo el contenido y almacenar cada línea en una posición del array que devuelve.
  - file (nombre_fichero)
- Leer todo el contenido y devolverlo en un string.
  - file_get_contents (nombre_fichero)

2.2.- Procesamiento de ficheros

- Ejemplo: leer el contenido de un fichero línea a línea:

```
$a = fopen('datos.txt', 'r');
while(!feof($a)){
    echo fgets($a) . '<br>';}
fclose($a);
```

- Ejemplo: leer el contenido de un fichero de una vez

```
$a = file('datos.txt');
foreach($a as $linea)
    echo $linea . '<br>';
```

- Ejemplo: mostrar una página web de otro sitio (las imágenes fallan si las URL son relativas)

```
$a =file('http://www.upm.es/index.html');
foreach($a as $linea)
    echo $linea;
```

- Ejercicio:
  - Realiza una página llamada lectura.php en la que lea el contenido de una de las páginas web hechas hasta ahora y lo muestre por pantalla.

2.2.- Procesamiento de ficheros

- Escribir (w ó a)
  - fwrite($fichero, cadena, longitud)
    - Escribe en *$fichero* los caracteres de cadena.
    - Si se añade el parámetro longitud, escribe hasta que termine la cadena o hasta que se alcancen los caracteres indicados en este parámetro, lo que antes ocurra.
    - Devuelve el número de caracteres escrito
  - fputs es una alias de fwrite.

```
$a = fopen('datos.txt', 'w+');
fwrite($a,"nueva linea\r\n");
fputs($a,"otra linea\r\n");
fclose($a);
```

- Verificar el final de fichero
  - feof(identificador) (TRUE -> fin, FALSE -> no fin)
- rewind(identificador)
  - permite colocar el puntero al principio del fichero.

2.2.- Procesamiento de ficheros

- Otras operaciones:
  - file_exists()
    - Determina si existe un archivo o directorio
  - fgetss()
    - Idéntica a *fgets* con la diferencia de que los tags html son eliminados del archivo a medida que se lee el mismo.
    - Opcionalmente puede pasarse una lista de tags que no deben ser eliminados.
    - Ejemplo:

      $string=fgetss($des,999999,"<b> <i> <table> <tr> <td>");

    - Lee una línea (de cualquier longitud) eliminando los tags html excepto los indicados como segundo parámetro. Los tags que cierran éstos tampoco son eliminados. (</td>,..)
  - readfile(path)
    - Lee e imprime un archivo

2.2.- Procesamiento de ficheros

- Otras operaciones:
  - copy (origen, destino)
    - Copiar un fichero
  - rename (nombre_original, nombre_final)
    - Renombrar (o mover) un fichero
  - unlink(fichero)
    - Borrar un fichero
  - ftruncate (descriptor, longitud)
    - Trunca el archivo a la longitud en bytes dada.
  - filesize(path)
    - Tamaño de un fichero en bytes
  - filemtime(path)
    - Devuelve marca de tiempo en que se modificó por última vez el fichero.
    - Es necesario formatearla con date

2.2.- Procesamiento de ficheros

- Otras operaciones .
  - chgrp : Cambia el grupo de un archivo.
  - chmod : Cambia permisos de un archivo
  - chown : Cambia el propietario de un archivo
  - is_executable : Indica si el archivo es ejecutable
  - is _file :Indica si el archivo es un archivo regular
  - is_link : Indica si el archivo es un enlace simbólico
  - is_readable : Indica si es posible leer el archivo
  - is_uploaded_file Indica si un archivo fue cargado a través de HTTP POST
  - is_writable : Indica si el nombre de archivo es escribible
  - is_writeable : Alias de is_writable
  - filetype(path) : Devuelve el tipo de un archivo.

2.2.- Procesamiento de ficheros

- Otras operaciones .
  - int exif_imagetype (string $filename): Lee los primeros bytes de una imagen y comprueba su firma.
  - devuelve el valor de la constante apropiada al tipo o FALSE si no se reconoce su tipo
  - Algunas constantes utilizadas por la función:

1	IMAGETYPE_GIF
2	IMAGETYPE_JPEG
3	IMAGETYPE_PNG
4	IMAGETYPE_SWF
5	IMAGETYPE_PSD
6	IMAGETYPE_BMP

2.2.- Procesamiento de ficheros

- int exif_imagetype (string $filename)

```
<?php
    if (exif_imagetype('imagen.gif') != IMAGETYPE_GIF) {
        echo 'La imagen no es gif';
    }
?>
```

2.3.- Manejo de directorios

- Similar al procesamiento de ficheros secuenciales:
  - Se abre directorio
  - Después se procesa cada entrada del mismo
  - Se cierra el directorio

2.3.- Manejo de directorios

- Funciones:
  - Determinar la existencia
    - is_dir(directorio): Determina si existe y es un directorio
  - Apertura
    - opendir(directorio): Abre directorio y devuelve un descriptor
  - Lectura
    - readdir($descriptor):
      - Devuelve el nombre del siguiente fichero en el directorio.
      - Los nombres de archivo son devueltos en el orden en que están en el sistema de archivos. ¡Ojo! los primeros “.” y “..”
  - Cierre
    - closedir($descriptor): Cierra el recurso $descriptor

2.3.- Manejo de directorios

- Ejemplo: Mostrar los ficheros de un directorio con su tamaño:

```
$dir = opendir("."); //abre el directorio actual
while(false !== ($fichero = readdir($dir))){
    if (is_dir($fichero))
        echo "Directorio: $fichero<br>";
    else
        echo "$fichero: " .filesize($fichero).'bytes <br>';}
closedir($dir);
```

- chdir(directorio) : Cambia de directorio
- getcwd(directorio) :Devuelve el directorio actual
- mkdir(directorio) : Crea un directorio
- rmdir(directorio) : Elimina un directorio
- basename(path) : Devuelve la parte del path correspondiente al nombre del archivo
- rewinddir(directorio): Rebobina el puntero interno de un directorio para que apunte nuevamente al comienzo del mismo.