



DWES

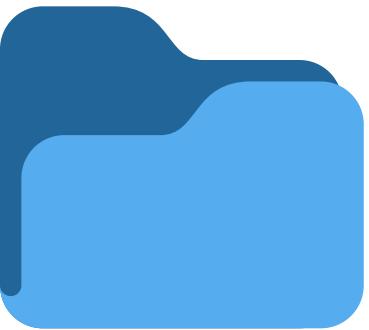
FICHEROS

Uso de Ficheros en PHP

ÍNDICE

Bloque I - Manejo de ficheros

Bloque II - Trabajo con ficheros en formularios



1. INTRODUCCIÓN

fichero

- Un fichero es una unidad de almacenamiento (texto, imagen, CSV, etc.) accesible desde PHP.
- PHP permite leer y escribir archivos del sistema usando funciones integradas.

ejemplos de uso

- Guardar logs o registros.
- Cargar productos desde un archivo CSV o TXT.
- Subir archivos desde formularios web.

2. FUNCIONES PRINCIPALES - I

Comprobación y apertura de archivos

Función	Descripción
file_exists(\$fichero)	Comprueba si el archivo existe.
readfile(\$fichero)	Muestra directamente el contenido del archivo.
fopen(\$fichero, \$modo)	Abre un archivo en el modo indicado.
fclose(\$manejador)	Cierra el archivo abierto.



2.1. MOSTRAR UN ARCHIVO SI EXISTE

- Primero comprobamos si el fichero existe con file_exists().
- Luego, readfile() lo muestra directamente en pantalla sin abrirlo manualmente.



```
$nombre = "datos.txt";

if (file_exists($nombre)) {
    echo "Contenido del archivo:<br>";
    readfile($nombre);
} else {
    echo "El archivo no existe.";
}
```

2.2. ABRIR Y CERRAR ARCHIVO

- @ evita el warning si el archivo no existe.
- fopen() abre el archivo en modo lectura ("r").
- Siempre se debe cerrar con fclose().



```
$archivo = "ejemplo.txt";
$manejador = @fopen($archivo, "r"); // r = solo lectura

if ($manejador) {
    echo "Archivo abierto correctamente.<br>";
    fclose($manejador);
} else {
    echo "Error al abrir el archivo.";
}
```

3. FUNCIONES PRINCIPALES - II

Lectura de archivos

Función	Descripción
fread(\$manejador, \$bytes)	Lee una cantidad de bytes.
fgets(\$manejador)	Lee una línea completa.
fgetc(\$manejador)	Lee un carácter.
feof(\$manejador)	Indica si se alcanzó el final del archivo.



3.1. LEER UN ARCHIVO COMPLETO

- fread() lee todo el contenido del archivo según su tamaño (filesize()).
- Ideal para archivos pequeños o medianos.



```
$archivo = "texto.txt";
$manejador = @fopen($archivo, "r");

if ($manejador) {
    $contenido = fread($manejador, filesize($archivo));
    echo $contenido;
    fclose($manejador);
} else {
    echo "No se pudo abrir el archivo.";
}
```

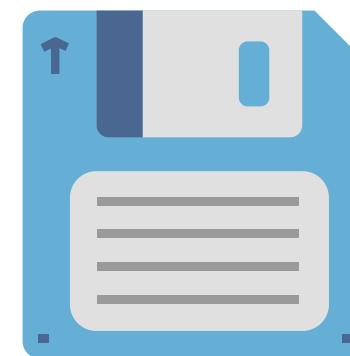
3.2. LEER LÍNEA A LÍNEA CON FGETS()

- feof() controla el fin del archivo.
- fgets() lee cada línea, ideal para procesar listas o CSV simples.



```
$archivo = "productos.txt";
$manejador = @fopen($archivo, "r");

if ($manejador) {
    while (!feof($manejador)) {
        $linea = fgets($manejador);
        echo $linea . "<br>";
    }
    fclose($manejador);
} else {
    echo "Error al abrir el archivo.";
}
```



3.3. LEER CARÁCTER A CARÁCTER

- fgetc() devuelve un carácter cada vez.
- Poco usado, pero útil para análisis detallado de texto.



```
$archivo = "mensaje.txt";
$manejador = @fopen($archivo, "r");

if ($manejador) {
    while (!feof($manejador)) {
        echo fgetc($manejador);
    }
    fclose($manejador);
}
```

4. FUNCIONES PRINCIPALES - III

Escritura en archivos

Función	Descripción
<code>fwrite(\$manejador, \$texto)</code>	Escribe texto en el archivo.
<code>fputs(\$manejador, \$texto)</code>	Igual que <code>fwrite()</code> .
<code>fputc(\$manejador, \$caracter)</code>	Escribe un carácter.
<code>feof(\$manejador)</code>	Útil también al escribir para verificar fin de archivo.



4.1. GUARDAR TEXTO EN UN ARCHIVO

a (append)

- Modo "a" agrega al final sin borrar el contenido anterior.
- Usamos @ para evitar warnings si el archivo no existe.



```
function guardarArchivo($texto)
{
    $archivo = "cesar.txt";
    $manejador = fopen($archivo, "a"); // modo append
    if ($manejador) {
        fwrite($manejador, $texto . "\n");
        fclose($manejador);
        echo "Texto guardado correctamente.";
    } else {
        echo "No se pudo abrir el archivo.";
    }
}
```



4.2. CREAR O SOBREESCRIBIR UN ARCHIVO `w` (write)

- Modo "w" crea el archivo si no existe, o borra su contenido si ya existía.



A hand holding a blue pen is shown writing in a white spiral-bound notebook with a yellow cover. The hand is positioned as if in the middle of writing a line of text.

```
$archivo = "nuevo.txt";
$manejador = fopen($archivo, "w"); // sobrescribe si existe
fwrite($manejador, "Primera línea del archivo.\n");
fclose($manejador);
```

5. FUNCIONES PRINCIPALES - IV

Funciones de mantenimiento de archivos

Función	Descripción	Ejemplo
unlink(\$archivo)	Elimina un archivo.	unlink("borrar.txt");
rename(\$viejo, \$nuevo)	Cambia el nombre o mueve un archivo.	rename("a.txt", "b.txt");
copy(\$origen, \$destino)	Copia un archivo a otro lugar.	copy("datos.txt", "backup.txt");
filesize(\$fichero)	Devuelve el tamaño en bytes.	echo filesize("datos.txt");



Bloque II – Trabajo con ficheros en formularios



1. INTRODUCCIÓN

ficheros en formularios

- Para poder trabajar con ficheros enviados desde formularios, hay que tener en cuenta varios aspectos de configuración y funciones.
- PHP permite que el usuario suba archivos (imágenes, textos, etc.) desde un formulario HTML, pero para ello el servidor debe estar correctamente configurado y el código PHP debe validar y procesar los datos de forma segura.

2. CONFIGURACIÓN PREVIA EN PHP.INI

Antes de subir archivos, debemos asegurarnos de que el servidor lo permita.

php.ini

- Activar la subida de archivos:



```
file_uploads = On
```

- Ajustar el tamaño máximo permitido (según necesidades):



```
upload_max_filesize = 2M  
post_max_size = 8M
```

Importante: *post_max_size siempre debe ser mayor o igual que upload_max_filesize*



3. FUNCIONES PRINCIPALES

Funciones de mantenimiento de archivos en formularios

Además de las vistas anteriormente: fopen, fclose, feof, fread, fgets, ...

Función	Descripción
basename(\$ruta)	Devuelve solo el nombre del archivo (sin ruta).
move_uploaded_file(\$tmp, \$destino)	Mueve el archivo subido desde la carpeta temporal a su ubicación final.
pathinfo(\$ruta)	Devuelve información (nombre, extensión, etc.) de un archivo.
file_get_contents(\$archivo)	Lee todo el contenido de un archivo en una sola llamada.
file_put_contents(\$archivo, \$contenido)	Escribe directamente texto o datos en un archivo (crea o sobrescribe).
uniqid()	Genera un nombre único (útil para evitar sobrescribir archivos).
unlink(\$archivo)	Elimina un archivo del servidor.



3.1. EL ARRAY `$_FILES`

`enctype="multipart/form-data"`

- Cuando un formulario HTML usa `enctype="multipart/form-data"`, PHP crea el array global `$_FILES` para almacenar la información de los archivos subidos.

Clave	Descripción
<code>\$_FILES["fichero"]["name"]</code>	Nombre original del archivo.
<code>\$_FILES["fichero"]["type"]</code>	Tipo MIME (por ejemplo, <code>image/jpeg</code>).
<code>\$_FILES["fichero"]["tmp_name"]</code>	Ruta temporal donde PHP guarda el archivo.
<code>\$_FILES["fichero"]["size"]</code>	Tamaño en bytes.
<code>\$_FILES["fichero"]["error"]</code>	Código de error (0 si fue correcto).

3.2 ERROR - CONSTANTES \$_FILES["fichero"]["error"]

- Constantes que podemos usar para los distintos errores

Constante	Valor	Significado
UPLOAD_ERR_OK	0	✓ No hubo error, el archivo se subió correctamente.
UPLOAD_ERR_INI_SIZE	1	⚠ El archivo excede el tamaño máximo permitido por upload_max_filesize en php.ini.
UPLOAD_ERR_FORM_SIZE	2	⚠ El archivo excede el tamaño máximo indicado en el formulario HTML (atributo MAX_FILE_SIZE).
UPLOAD_ERR_PARTIAL	3	⚠ El archivo se subió solo parcialmente (por error en la conexión o interrupción).
UPLOAD_ERR_NO_FILE	4	✗ No se subió ningún archivo (el usuario no seleccionó ninguno).
UPLOAD_ERR_NO_TMP_DIR	6	✗ Falta el directorio temporal donde PHP guarda los archivos antes de moverlos.
UPLOAD_ERR_CANT_WRITE	7	✗ Error al escribir el archivo en el disco (por permisos o espacio insuficiente).
UPLOAD_ERR_EXTENSION	8	⚠ Una extensión de PHP detuvo la subida (por ejemplo, un filtro de seguridad).

3.3. EJEMPLO BÁSICO DE SUBIDA DE ARCHIVOS

Formulario HTML



```
<form action="subir.php" method="post" enctype="multipart/form-data">
  <label>Selecciona un archivo:</label>
  <input type="file" name="fichero"><br><br>
  <input type="submit" value="Subir archivo">
</form>
```

3.4. EJEMPLO BÁSICO DE SUBIDA DE ARCHIVOS

Código PHP (subir.php)

- move_uploaded_file() mueve el archivo desde la carpeta temporal.
- basename() se usa para evitar rutas no deseadas.
- Siempre es recomendable validar tipo, tamaño y nombre antes de guardar.



```
if (isset($_FILES["fichero"])) {  
    $archivo = $_FILES["fichero"];  
    $nombre = basename($archivo["name"]);  
    $destino = "uploads/" . $nombre;  
  
    if ($archivo["error"] == 0) {  
        if (move_uploaded_file($archivo["tmp_name"], $destino)) {  
            echo "Archivo subido correctamente: $nombre";  
        } else {  
            echo "Error al mover el archivo.";  
        }  
    } else {  
        echo "Error en la subida: " . $archivo["error"];  
    }  
}
```

3.5. VALIDACIONES RECOMENDADAS

Antes de guardar un archivo subido:



Comprobar tamaño:

```
if ($_FILES["fichero"]["size"] > 2000000) {  
    echo "El archivo supera el tamaño máximo permitido (2MB);  
}
```



Comprobar extensión y tipo MIME:

```
$ext = strtolower(pathinfo($_FILES["fichero"]["name"], PATHINFO_EXTENSION));  
if ($ext != "jpg" && $ext != "jpeg" && $ext != "png") {  
    echo "Tipo de archivo no permitido. Solo imágenes JPG o PNG.";  
}
```



Evitar sobrescribir archivos existentes:

```
$nombreUnico = uniqid() . "_" . basename($_FILES["fichero"]["name"]);
```

3.6. EJEMPLOS

Leer con file_get_contents()

- file_get_contents() lee el contenido completo del archivo y lo devuelve como una cadena.
- Es más sencillo que usar fopen() y fread() cuando solo queremos leer todo el archivo de una vez.



```
$contenido = file_get_contents("texto.txt");
echo $contenido;
```

Escribir con file_put_contents()

- file_put_contents() escribe texto en un archivo.
- Si el archivo no existe, lo crea automáticamente.
- Si ya existe, lo sobrescribe, salvo que usemos la bandera FILE_APPEND, que añade el texto al final.



```
$texto = "Nueva línea añadida al archivo.\n";
file_put_contents("log.txt", $texto, FILE_APPEND);
```

3.7. EJEMPLOS

Eliminar un archivo con unlink()

- unlink() elimina un archivo del servidor.
- Antes de usarlo, conviene comprobar con file_exists() si el archivo realmente está disponible.



```
$archivo = "uploads/imagen_temp.jpg";
if (file_exists($archivo)) {
    unlink($archivo);
    echo "El archivo ha sido eliminado correctamente.";
} else {
    echo "El archivo no existe o ya fue eliminado.";
}
```

FICHEROS EN PHP

FIN

Realizado por: @pepelluyot

