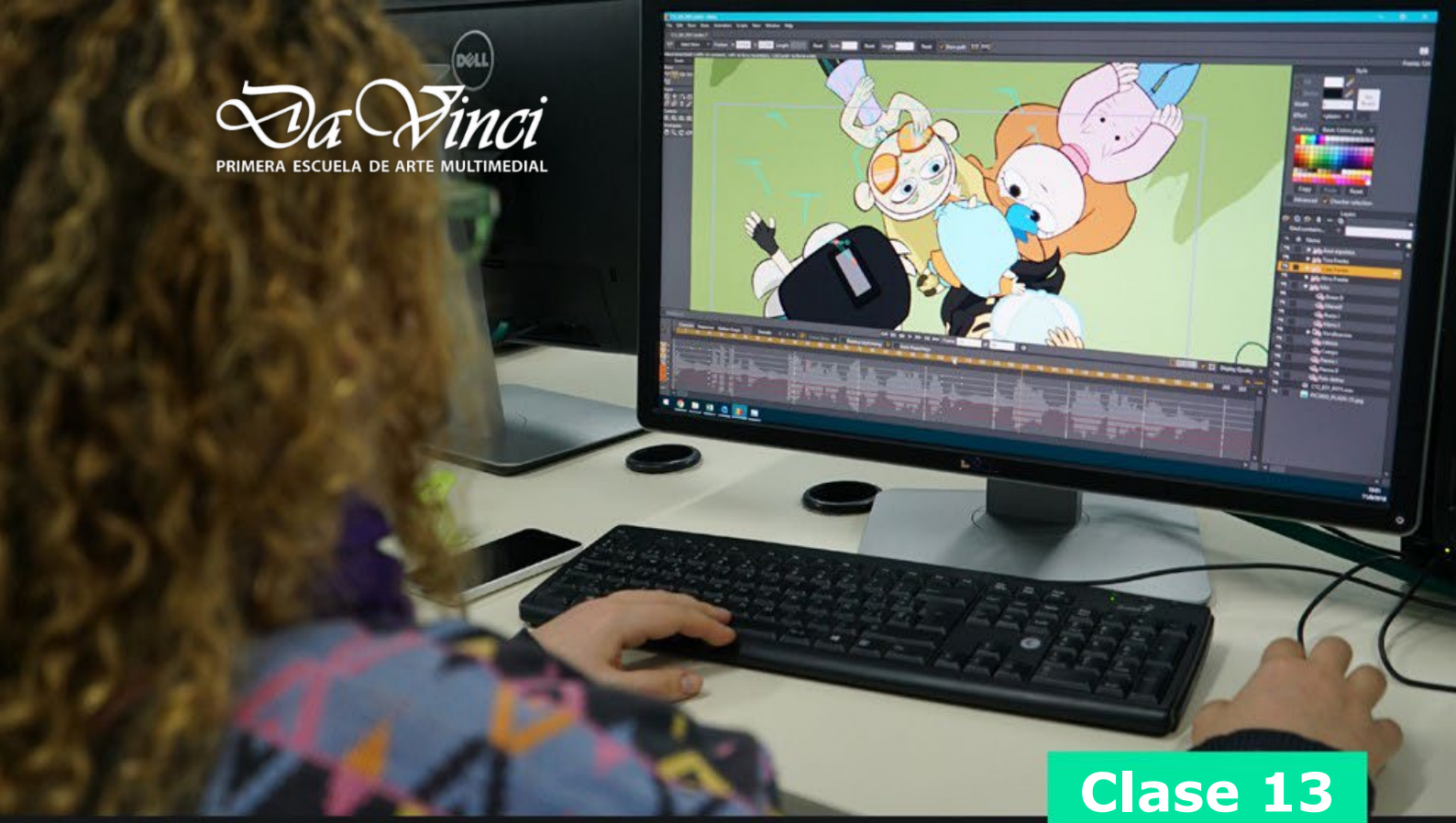

Clase 13

Análisis de Sistemas

Materia:
Programación Web II

Docente contenidista: ROLDÁN, Hernán

Revisión: Coordinación

Contenido

Clase 13

¡Te damos la bienvenida a la materia **Programación Web II**!

En esta clase vamos a ver los siguientes temas:

- Introducción al manejo de archivos.
- Apertura de archivos.
- Lectura de archivos.
- Cierre de archivos.
- Leer todo el contenido de un archivo.
- Lectura línea por línea.
- Verificar fin de archivo.
- Escritura de archivos.

PHP File Open/Read/Close

Introducción al manejo de archivos

El manejo de archivos en PHP es una parte esencial del desarrollo web, ya que nos permite interactuar con el sistema de archivos del servidor y trabajar con información de manera persistente. PHP proporciona un conjunto de funciones específicas para abrir, leer, escribir y cerrar archivos, lo que facilita la manipulación de datos almacenados en archivos de texto, CSV, XML, entre otros formatos.

Apertura de archivos

La función **fopen()** (file open) se utiliza para abrir un archivo en PHP antes de realizar operaciones de lectura o escritura en él. Su sintaxis es la siguiente:

```
resource fopen ( string $filename , string $mode [, bool
$use_include_path = FALSE [, resource $context ]] )
```

Parámetros:

- **$filename:** El nombre del archivo que se va a abrir, especificando la ruta completa o relativa al archivo.

- **$mode:** El modo de acceso para abrir el archivo. Puede ser uno de los siguientes valores:
  - "**r**" (predeterminado): Apertura en modo lectura. El puntero del archivo se coloca al inicio del archivo.
  - "**r+**": Apertura en modo lectura y escritura. El puntero del archivo se coloca al inicio del archivo.
  - "**w**": Apertura en modo escritura. Si el archivo no existe, se crea uno nuevo. Si ya existe, se trunca (vacía) el contenido existente.
  - "**w+**": Apertura en modo lectura y escritura. Si el archivo no existe, se crea uno nuevo. Si ya existe, se trunca el contenido existente.
  - "**a**": Apertura en modo escritura al final del archivo (append). Si el archivo no existe, se crea uno nuevo.

- "**a+**": Apertura en modo lectura y escritura al final del archivo (append). Si el archivo no existe, se crea uno nuevo.
- "**x**": Apertura en modo escritura exclusiva. Se crea un nuevo archivo y falla si ya existe.
- "**x+**": Apertura en modo lectura y escritura exclusiva. Se crea un nuevo archivo y falla si ya existe.

- **$use_include_path** (opcional): Si es TRUE, busca el archivo en el directorio especificado por include_path además del directorio actual.

- **$context** (opcional): Un contexto de flujo creado con stream_context_create() que contiene opciones de configuración adicionales.

Ejemplo de uso de la función fopen():

```php
<?php

  $archivo = "C:\wamp64\www\archivos\datos.txt";
  $modo = "r"; // Apertura en modo lectura

  $manejador = fopen($archivo, $modo);

  if($manejador){
    echo "El archivo se abrió correctamente.";
    // Realizar operaciones de lectura o escritura
    fclose($manejador); // Cerrar el archivo al finalizar
  }
  else{
    echo "No se pudo abrir el archivo.";
  }

?>
```

En este ejemplo, la función **fopen()** abre el archivo "datos.txt" en modo lectura ("r"). Luego, se verifica si el archivo se abrió correctamente, y se procede a realizar las operaciones de lectura o escritura según sea necesario. Finalmente, se cierra el archivo con **fclose()** para liberar recursos del sistema y evitar problemas de acceso concurrente.

Es importante resaltar que al utilizar la función **fopen()** en modo lectura ("r"), el archivo que se intenta abrir debe existir previamente en el sistema de archivos. Si el archivo no existe, la función **fopen()** devolverá false, y no se podrá realizar ninguna operación de lectura sobre él.

Es responsabilidad del programador asegurarse de que el archivo a abrir en modo lectura ya exista. En caso contrario, es necesario realizar las verificaciones adecuadas para evitar errores o excepciones. Para abrir un archivo en modo escritura ("w" o "a"), se creará un nuevo archivo si no existe, pero en modo lectura, la existencia del archivo es requerida para poder acceder a su contenido.

Ejemplo de uso de la función fopen() con verificación de la existencia del archivo que se desea abrir:

```php
<?php

  $archivo = "C:\wamp64\www\archivos\datos.txt";
  $modo = "r"; // Apertura en modo lectura

  if(file_exists($archivo)){
    $manejador = fopen($archivo, $modo);

    if($manejador){
      echo "El archivo se abrió correctamente.";
      // Realizar operaciones de lectura o escritura
      fclose($manejador); // Cerrar el archivo al finalizar
    }
    else{
      echo "No se pudo abrir el archivo.";
    }
  }
  else{
    echo "El archivo no existe.";
  }

?>
```

En este ejemplo, antes de intentar abrir el archivo en modo lectura, se utiliza la función **file_exists()** para verificar si el archivo "datos.txt" existe en el sistema de archivos. Solo si el archivo existe, se procede a abrirlo con **fopen()** y realizar las operaciones correspondientes. En caso contrario, se muestra un mensaje indicando que el archivo no existe. Esta verificación previa evita que se intente acceder a un archivo inexistente, evitando posibles errores en la ejecución del programa.

Lectura de archivos

La función fread() en PHP se utiliza para leer contenido desde un archivo abierto con la función fopen(). Nos permite especificar la cantidad de bytes que queremos leer desde el puntero actual del archivo. Su sintaxis es la siguiente:

```
string fread ( resource $handle , int $length )
```

Parámetros:

- **$handle:** El identificador del archivo devuelto por la función fopen().
- **$length:** El número de bytes que deseamos leer desde el archivo.

Funcionamiento de fread():

- Cuando llamamos a **fread()**, se lee la cantidad de bytes especificada por $length desde el puntero actual del archivo representado por el $handle.
- El puntero del archivo avanza automáticamente después de la lectura, por lo que en cada llamada a **fread()**, el contenido se leerá desde el lugar donde se detuvo la lectura anterior.
- Si llegamos al final del archivo o si no se pueden leer los bytes especificados, **fread()** devuelve una cadena vacía (“”).

Supongamos que tenemos un archivo "datos.txt" con el siguiente contenido:

Hola, este es un archivo de ejemplo.

Estamos aprendiendo sobre fread() y fclose().

El código sería el siguiente:

```php
<?php

  $archivo = "C:\wamp64\www\archivos\datos.txt";
  $modo = "r"; // Apertura en modo lectura

  $manejador = fopen($archivo, $modo);

  if($manejador){
    // Obtener el tamaño total del archivo
    $tamanioArchivo = filesize($archivo);

    // Leer todo el contenido del archivo
    $contenido = fread($manejador, $tamanioArchivo);

    echo $contenido; // Salida: contenido completo del
archivo

    fclose($manejador); // Cerrar el archivo al finalizar
  }
  else{
    echo "No se pudo abrir el archivo.";
  }

?>
```

Vamos a explicar paso a paso el código del ejemplo:

1. Primero, se establece el modo de apertura del archivo como lectura **("r")**. Esto significa que el archivo se abrirá para ser leído, y el puntero del archivo se posicionará al inicio.

2. Se utiliza la función **fopen()** para abrir el archivo **"datos.txt"** en el modo de apertura especificado por la variable **$modo**. El resultado de esta operación se almacena en la variable $manejador. La función **fopen()** devuelve un identificador de archivo (handle) que se utiliza para referenciar el archivo en las operaciones posteriores.

3. Se verifica si el archivo se ha abierto correctamente. Si **$manejador** es válido (diferente de false), significa que el archivo fue abierto con éxito y podemos proceder con las operaciones de lectura. Si **$manejador** es false, se muestra el mensaje "No se pudo abrir el archivo." indicando que ha ocurrido un problema en la apertura del archivo.

4. A continuación, se utiliza la función **filesize($archivo)** para obtener el tamaño total del archivo **"datos.txt"** y se almacena en la variable **$tamanioArchivo**. Esto nos proporciona la cantidad total de bytes que contiene el archivo.

5. Luego, se utiliza la función **fread($manejador, $tamanioArchivo)** para leer todo el contenido del archivo desde la posición actual del puntero hasta el final del archivo. Al pasar **$tamanioArchivo** como segundo argumento a **fread()**, le indicamos que queremos leer la cantidad total de bytes del archivo.

6. El contenido leído se almacena en la variable **$contenido**.

7. Finalmente, se imprime el contenido del archivo completo utilizando **echo $contenido**;. El resultado será todo el contenido del archivo **"datos.txt"**, que en este caso sería:

```
 Hola, este es un archivo de ejemplo.
Estamos aprendiendo sobre fread() y fclose().
```

8. Finalmente, el archivo se cierra usando **fclose($manejador);**. Esto libera los recursos asociados al archivo y asegura que no haya problemas de acceso concurrente.

En resumen, este ejemplo muestra cómo abrir y leer un archivo utilizando las funciones **fopen()** y **fread()**. Al utilizar **filesize()** para obtener el tamaño del archivo, podemos leer todo el contenido del archivo sin tener que especificar la cantidad exacta de bytes a leer, lo que facilita la lectura de archivos de diferentes tamaños.

Cierre de archivos

La función **fclose()** en PHP se utiliza para cerrar un archivo que ha sido previamente abierto con la función **fopen()**. Su sintaxis es sencilla:

```
bool fclose ( resource $handle )
```

Parámetros:

- **$handle:** El identificador del archivo devuelto por la función fopen().

Funcionamiento de fclose():

- Al cerrar el archivo con **fclose()**, se liberan los recursos asociados a ese archivo y se cierra la conexión con el mismo.
- Es importante cerrar los archivos correctamente después de finalizar las operaciones de lectura o escritura para liberar los recursos del servidor y evitar problemas de acceso concurrente a los archivos.
- Si no se cierran los archivos explícitamente, es posible que queden bloqueados o que no se liberen los recursos adecuadamente, lo que podría causar problemas en la ejecución de la aplicación.

Para analizar su funcionamiento, en el ejemplo anterior cuando vimos cómo usar la función fopen() también usamos la función fclose() para cerrar el archivo.

El uso adecuado de la función **fclose()** es una buena práctica en el manejo de archivos en PHP, ya que garantiza que no haya pérdida de datos y evita problemas de acceso concurrente cuando varios procesos intentan acceder al mismo archivo. Cerrar los archivos adecuadamente es especialmente importante en aplicaciones web donde se manejan múltiples solicitudes al mismo tiempo.

Leer todo el contenido de un archivo

La función **filesize()** devuelve el tamaño total del archivo especificado en bytes. Si el archivo no existe o no se puede acceder a él, la función retornará false.

Una vez que tenemos el tamaño del archivo con **filesize()**, podemos utilizar la función **fread()** para leer todo el contenido del archivo en una sola operación. Si no especificamos la cantidad de bytes a leer en **fread()**, esta función leerá desde la posición actual del puntero hasta el final del archivo.

Para ver su funcionamiento, en el ejemplo anterior cuando vimos cómo usar la función fopen() también usamos la función fsize() para conocer el tamaño total del archivo.

Es importante destacar que este método es útil cuando se necesita leer el archivo completo en una sola lectura. Sin embargo, para archivos de gran tamaño, es posible que leer todo el contenido de una vez ocupe una cantidad significativa de memoria. En tales casos, puede ser más conveniente leer el archivo en bloques más pequeños para evitar problemas de rendimiento y uso excesivo de memoria.

Lectura línea por línea

La función **fgets()** en PHP se utiliza para leer un archivo línea por línea. Esto es especialmente útil cuando se trabaja con archivos de texto que contienen registros o líneas de información separados por saltos de línea. La función **fgets()** nos permite procesar archivos línea por línea sin agotar la memoria, lo que es especialmente beneficioso para archivos grandes. Su sintaxis es la siguiente:

```
string|false fgets ( resource $handle [, int $length ] )
```

Parámetros:

- **$handle:** El identificador del archivo devuelto por la función fopen().
- **$length (opcional):** La longitud máxima de caracteres que se leerán de cada línea. Si no se especifica, fgets() lee una línea completa hasta el salto de línea o hasta el final del archivo.

Funcionamiento de fgets():

- **fgets()** lee una línea completa del archivo apuntado por el identificador de archivo ($handle) y mueve el puntero al inicio de la siguiente línea.
- Cada línea se define por el delimitador de fin de línea, que puede ser un salto de línea ("\n"), un retorno de carro ("\r"), o una combinación de ambos ("\r\n").
- La función devuelve la línea leída como una cadena de caracteres, incluyendo el delimitador de fin de línea, o false si se ha alcanzado el final del archivo o si ha ocurrido un error.

Ejemplo de uso de la función fgets():

```php
<?php

  $archivo = "C:\wamp64\www\archivos\datos.txt";
  $modo = "r"; // Apertura en modo lectura

  $manejador = fopen($archivo, $modo);

  if($manejador){
      // Leer el archivo línea por línea hasta el final
    while(($linea = fgets($manejador)) !== false) {
      // Procesar la línea leída
      echo $linea;
    }

    fclose($manejador); // Cerrar el archivo al finalizar
  }
  else{
    echo "No se pudo abrir el archivo.";
  }

?>
```

En este ejemplo, abrimos el archivo **"datos.txt"** en modo lectura con **fopen()**. Luego, utilizamos un bucle while para leer el archivo línea por línea hasta que se alcance el final. En cada iteración del bucle, la función **fgets()** lee una línea del archivo y la almacena en la variable $linea.

Dentro del bucle, puedes realizar cualquier tipo de procesamiento que desees con la línea leída, como imprimirla en pantalla, procesar datos, o almacenarla en una base de datos, entre otras acciones.

El uso de **fgets()** es muy eficiente cuando se trabaja con archivos grandes, ya que solo se lee una línea a la vez y el contenido no se carga por completo en memoria. Esto evita el agotamiento de la memoria RAM y permite procesar archivos de gran tamaño de manera eficiente.

Recordá que es importante cerrar el archivo con **fclose($manejador)** una vez que hayas terminado de leerlo para liberar los recursos del servidor.

Verificar fin de archivo

La función **feof()** en PHP se utiliza para comprobar si se ha alcanzado el final de un archivo después de realizar operaciones de lectura con la función fread(). Su sintaxis es la siguiente:

```
bool feof ( resource $handle )
```

Parámetros:

- **$handle:** El identificador del archivo devuelto por la función fopen().

Funcionamiento de feof():

- La función **feof()** toma como argumento el identificador de archivo (handle) devuelto por fopen().
- Retorna true si se ha alcanzado el final del archivo, es decir, si la lectura ha llegado al final del archivo.
- Retorna false si el puntero de lectura aún no ha alcanzado el final del archivo.

Uso de feof() con fread():

La función **feof()** es comúnmente utilizada en combinación con fread() para determinar cuándo ha terminado la lectura del archivo. Esto se realiza en un bucle mientras no se haya llegado al final del archivo.

Ejemplo de uso de la función feof():

```php
<?php

  $archivo = fopen("C:\wamp64\www\archivos\datos.txt", "r")
or die("No se pudo abrir el archivo.");

  // Leer el archivo hasta que se alcance el final
  while(!feof($archivo)){

    // Mostrar el contenido leído
    echo fgets($archivo);

  }

  // Cerrar el archivo al finalizar
  fclose($archivo);

?>
```

Escritura de archivos

La función **fwrite()** en PHP se utiliza para escribir datos en un archivo. Esta función es parte del manejo de archivos y nos permite agregar o sobrescribir contenido en un archivo ya existente o crear un nuevo archivo si este no existe. Su sintaxis es la siguiente:

```
int|false fwrite ( resource $handle , string $string [, int $length ] )
```

Parámetros:

- **$handle:** El identificador del archivo devuelto por la función fopen().
- **$string:** El contenido que se va a escribir en el archivo.
- **$length (opcional):** La cantidad de bytes a escribir. Si no se especifica, se escribirá todo el contenido de $string.

Funcionamiento de fwrite():

- La función **fwrite()** toma el identificador del archivo ($handle) y una cadena de caracteres ($string) como argumentos. Escribe los datos de $string en el archivo asociado con el handle $handle.
- Si el archivo no ha sido abierto en modo escritura ('w', 'a', 'x', 'c', etc.), la función **fwrite()** no funcionará correctamente y retornará false.
- La función devuelve la cantidad de bytes escritos en el archivo, o false si ocurre un error durante la escritura.

Ejemplo de uso de la función fwrite():

```php
<?php

  $archivo =
"C:\wamp64\www\desafios_semanales\clase_13\datos.txt";
  $modo = "w"; // Apertura en modo escritura

  $manejador = fopen($archivo, $modo);

  if($manejador){
    $contenido = "Hola, este es un archivo de ejemplo.\n";
    $bytesEscritos = fwrite($manejador, $contenido);

    if($bytesEscritos !== false){
      echo "Se han escrito $bytesEscritos bytes en el
archivo.";
    }
    else{
      echo "Error al escribir en el archivo.";
    }

    fclose($manejador); // Cerrar el archivo al finalizar

  }
  else{
    echo "No se pudo abrir el archivo.";
  }

?>
```

Como cierre del tema de manejo de archivos con PHP, aquí dejamos algunas recomendaciones importantes a tener en cuenta al utilizar las funciones vistas:

Verificar la existencia del archivo:
Antes de abrir un archivo, siempre es recomendable verificar que el archivo existe en la ubicación especificada. Podés utilizar la función file exists() para realizar esta comprobación.

***Manejo de errores**:*
Siempre es importante manejar adecuadamente los errores al trabajar con archivos.
Asegurate verificar el resultado de las funciones que abren archivos, como fopen(), fread(), fwrite(), etc., y manejar cualquier error que pueda ocurrir.

***Cerrar archivos correctamente**:*
Una vez que hayas terminado de trabajar con un archivo, asegurate cerrarlo utilizando la función felose(). Esto libera los recursos asociados al archivo y evita posibles bloqueos o problemas de acceso concurrente.

Hemos llegado así al final de esta clase en la que vimos:

- Introducción al manejo de archivos.
- Apertura de archivos.
- Lectura de archivos.
- Cierre de archivos.
- Leer todo el contenido de un archivo.
- Lectura línea por línea.
- Verificar fin de archivo.
- Escritura de archivos.

Te esperamos en la **clase en vivo** de esta semana.
No olvides realizar el **desafío semanal**.

¡Hasta la próxima clase!

Bibliografía

Documentación Oficial de PHP

W3Schools: PHP Tutorial

Cabezas Granado, Luis. González Lozano, F. (2018) Desarrollo web con PHP y MySQL. Anaya Multimedia.

Heurtel, O. (2016) Desarrolle un sitio web dinámico e interactivo. Eni Ediciones.

Welling,Luke Thompson, L. (2017) Desarrollo Web con PHP y MySQL. Quinta Edición (2017, Editorial Anaya)

Para ampliar la información

PHP: fopen()

PHP: fclose

PHP: fread

PHP: file_exists

PHP: filesize

PHP: fgets

PHP: feof

PHP: fwrite

PHP File Open/Read/Close

PHP: The Right Way

Todo sobre PHP

PHP Programming Language Tutorial - Full Course

PHP Full Course for non-haters

CURSO de php desde cero

MDN: Introducción al lado Servidor

Guía de HTML