Programación III
PHP

Clase 3

Maximiliano Neiner

Temas a Tratar

- Envío de datos
- Manipulación de archivos
- Subir archivos al servidor

Temas a Tratar

- Envío de datos
  - HTTP
  - GET
  - POST
- Manipulación de archivos
- Subir archivos al servidor

HTTP (HTTP: Hypertext Transfer Protocol)

- HTTP está diseñado para permitir comunicaciones entre clientes y servidores.
- HTTP funciona como un protocolo de pedido-respuesta entre cliente y servidor.
- Un navegador web puede ser el cliente y una aplicación sobre un computador que aloja un sitio web puede ser el servidor.

Temas a Tratar

- Envío de datos
  - HTTP
  - GET
  - POST
- Manipulación de archivos
- Subir archivos al servidor

Método GET

- El par de nombres/valores es enviado en la dirección URL (texto claro).
- Las peticiones GET se pueden almacenar en caché.
- Permanecen en el historial del navegador.
- Pueden ser marcadas (book marked).
- Nunca debe ser utilizado cuando se trata de datos confidenciales.
- Tiene limitaciones de longitud de datos (longitud máxima de 2048 caracteres en la URL).

Temas a Tratar

- Envío de datos
  - HTTP
  - GET
  - POST
- Manipulación de archivos
- Subir archivos al servidor

Método POST

- El par de nombres/valores es enviado en el cuerpo del mensaje HTTP.
- Las peticiones POST no se almacenan en caché.
- No permanecen en el historial del navegador.
- No pueden ser marcadas.
- No poseen restricciones de longitud de datos.

Manejo de Formularios

- Tanto GET como POST crean un *array asociativo.*
- Dicho array contiene pares de clave-valor, dónde las claves son los nombres (atributo ***name***) de los controles del formulario y los valores son la entrada de datos del usuario.
- PHP utiliza las ***super globales*** $_GET, $_POST y $_REQUEST para recolectar datos provenientes de un Form.
- $_GET es un array pasado por GET.
- $_POST es un array pasado por POST.
- $_REQUEST es un array asociativo que contiene $_GET, $_POST y $_COOKIE.

Temas a Tratar

- Envío de datos
- Manipulación de archivos
- Subir archivos al servidor

Temas a Tratar

- Envío de datos
- Manipulación de archivos
  - Generalidades
  - Abrir archivos
  - Cerrar archivos
  - Leer archivos
  - Escribir archivos
  - Copiar archivos
  - Borrar archivos
- Subir archivos al servidor

E/S con Archivos

- El manejo de archivos (de texto o binarios) es una parte importante de una aplicación web.
- PHP nos provee de una extensa gama de funciones de acceso a archivos.
- En esta clase veremos las funciones básicas:
  - fopen (arbrir)
  - fclose (cerrar)
  - fread/fgets (leer)
  - fwrite/fputs (escribir)
  - copy (copia)
  - unlink (elimina)

fopen() (1/2)

- Nos permite abrir un archivo ya sea de manera local o externa (http:// o ftp://).
- El primer parámetro de ***fopen*** contiene el nombre del archivo a ser abierto, y el segundo especifica el modo en que el archivo será abierto.
- El valor de retorno de ***fopen*** es un entero. Nos servirá para referenciar al archivo abierto.

```
<?php
        int fopen(archivo, modo);
?>
```

fopen() (2/2)

Modo	Descripción
r	Abre un archivo para sólo lectura. El cursor comienza al principio del archivo.
w	Abre un archivo para sólo escritura. Si no existe, crea uno nuevo. Si existe, borra el contenido.
a	Abre un archivo para sólo escritura. Si no existe, crea uno nuevo. Si existe, mantiene el contenido. El cursor comienza en el final del archivo.
x	Crea un nuevo archivo para sólo lectura. Retorna FALSE y un error si el archivo existe.
r+	Abre un archivo para lectura/escritura. Ídem r.
w+	Abre un archivo para lectura/escritura. Ídem w.
a+	Abre un archivo para lectura/escritura. Ídem a.
x+	Crea un archivo para lectura/escritura. Ídem x.

fclose()

- Nos permite cerrar un archivo abierto.
- ***fclose*** requiere el indicador del archivo a ser cerrado (la variable que referencia al archivo).
- Retorna TRUE si tuvo éxito, FALSE caso contrario.

```
<?php
        $ar = fopen(archivo, modo);
        //MAS CODIGO
        fclose($ar);
?>
```

fread()

- Nos permite leer de un archivo abierto.
- El primer parámetro de ***fread*** contiene el indicador del archivo a ser leído, y el segundo especifica el número máximo de bytes que serán leídos.
- Retorna un string que representa al contenido del archivo leído.

```
<?php
    echo fread(indicador_archivo, filesize(archivo));
    //Lee el archivo completo
?>
```

fgets()

- Nos permite leer una línea de una archivo abierto.
- Requiere como parámetro el indicador del archivo a ser leído, y retorna un string que representa la línea que fue leída.
- Después de cada llamada a ***fgets***, el cursor se mueve a la siguiente línea.

```
<?php
      string fgets(indicador_archivo);
?>
```

feof() (End Of File)

- Retorna un booleano que indica si se ha llegado al fin del archivo.
- Requiere cómo parámetro el indicador.

```
<?php
        $ar = fopen(archivo, modo);
        //Lee línea a línea hasta EOF
        while(!feof($ar))
        {
            echo fgets($ar), "<br/>";
        }
        fclose($ar);
?>
```

fwrite() - fputs()

- Nos permite escribir en un archivo abierto.
- La función parará cuando llegue al fin del archivo o cuando alcance la longitud especificada.
- El primer parámetro contiene el archivo a ser leído, y el segundo es el string a ser escrito. El tercer parámetro es opcional e indica la cantidad de bytes a ser escritos.
- Retorna la cantidad de bytes que se escribieron o FALSE si hubo error.

```
<?php
    fwrite(indicador_archivo, texto [,longitud]);
    fputs(indicador_archivo, texto [,longitud]);
?>
```

copy()

- Permite copiar un archivo.
- Los parámetros que recibe son los nombres de los archivos. El primero es el archivo origen, luego el destino de la copia.
- Retorna TRUE en caso de éxito o FALSE si hubo algún error.

```
<?php
	copy(archivo_origen, archivo_destino);
?>
```

unlink()

- Permite eliminar un archivo.
- Recibe el nombre del archivo a ser borrado como primer parámetro.
- Retorna TRUE en caso de éxito o FALSE si hubo algún error.

```
<?php
	unlink(archivo);
?>
```

file_get_contents()

- Lee un archivo completo.
- Es la forma preferida para leer el contenido de un archivo en una cadena. Utilizará técnicas de mapeo de memoria (si el SO lo permite) para mejorar el rendimiento.
- Retorna la cadena del archivo leído o FALSE si hubo algún error.

```
<?php
      $contenido = file_get_contents(archivo);
?>
```

file_put_contents()

- Escribe datos en un archivo.
- Es equivalente al llamado de fopen(), fwrite() y fclose() sucesivamente.
- Si el archivo no existe se creará. Si existe se sobreescribirá a menos que se use FILE_APPEND.
- Retorna la cantidad de bytes escritos o FALSE si hubo algún error.

```
<?php
      file_put_contents(archivo, data);
?>
```

Temas a Tratar

- Envío de datos
- Manipulación de archivos
- Subir archivos al servidor

Subir archivos en PHP(1/3)

- Para poder subir archivos al servidor, es necesario crear un formulario en HTML que le permita a los usuarios seleccionar un archivo.

```
<!doctype html>
<html>
<head>
   <title>Subir Archivos</title>
</head>
<body>
   <form action="upload.php" method="post"
         enctype="multipart/form-data" >

         <input type="file" name="archivo" />
         <input type="submit" value="Subir" />
   </form>
</body>
</html>
```

Subir archivos en PHP(2/3)

- Algunas reglas a seguir para el formulario HTML.
  - El método del form debe ser POST.
  - El form necesita del atributo ***enctype***. Dicho atributo especifica el contenido/tipo a usarse cuando se 'submitea' el form.
  - El input de tipo **FILE** permite mostrar una ventana modal para navegar en busca del archivo a ser subido.

- Sin los requerimientos indicados la subida de archivos fallará.

Subir archivos en PHP(3/3)

- Del lado del servidor, tenemos que manipular el archivo recibido en $_FILES.
- Utilizando funciones de PHP deberemos mover el archivo subido desde su ubicación temporal a la ubicación definitiva dentro del servidor.

```
<?php

$destino = "uploads/".$_FILES["archivo"]["name"];

move_uploaded_file($_FILES["archivo"]["tmp_name"], $destino);

?>
```

$_FILES

- Es una **super global** que existe a partir de PHP 4.1.0.
- Es un array asociativo de elementos cargados al script actual a través del método POST.
  - **[name]** => nombre del archivo (con su extensión).
  - **[type]** => tipo del archivo (dado por el navegador).
  - **[tmp_name]** => carpeta temporal dónde se guardará el archivo subido.
  - **[error]** => código de error (si es 0, no hubo errores).
  - **[size]** => tamaño del archivo, medido en bytes.

Demo

Atributos

- El nuevo atributo booleano ***multiple*** (de HTML5) permite que los usuarios seleccionen varios archivos para ser subidos al servidor.
- El atributo ***accept***, permite filtrar (en el cliente) los tipos de archivos que se permitirán subir al servidor.

```
<form action="upload.php" method="post"
      enctype="multipart/form-data" >

      <input type="file" name="archivos[]" multiple
      accept="image/png,.jpg,image/gif" />
      <input type="submit" value="Subir" />
</form>
```

Demo

Ejercitación