PHP File Handling

Prepared by:
Dr. Susheela

Introduction

- File handling is an important part of any web application.
- You often need to open and process a file for different tasks.

PHP Manipulating Files

- PHP has several functions for creating, reading, uploading, and editing files.
- *Common errors are: editing the wrong file, filling a hard-drive with garbage data, and deleting the content of a file by accident.*

PHP readfile() Function

- The readfile() **function reads a file and writes it to the output buffer.**
- The readfile() function is useful if all you want to do is **open up a file and read its contents.**
- Assume we have a text file called "webdictionary.txt", stored on the server, that looks like this:

AJAX = Asynchronous JavaScript and XML
CSS = Cascading Style Sheets
HTML = Hyper Text Markup Language
PHP = PHP Hypertext Preprocessor
SQL = Structured Query Language
SVG = Scalable Vector Graphics
XML = EXtensible Markup Language

The PHP code to read the file and write it to the output buffer is as follows (the readfile() function returns the number of bytes read on success):

```
<!DOCTYPE html>
<html>
<body>
<?php
$s = readfile("webdictionary.txt");
echo $s;
?>
</body>
</html>
```

PHP 5 File Open/Read/Close

- A better method to open files is with the **fopen() function**. This function gives you **more options than the readfile() function.**

- The first parameter of fopen() contains the **name of the file to be opened** and the second **parameter specifies in which mode the file should be opened.**

- The file may be opened in one of the following modes:

Modes	Description
r	**Open a file for read only**. File pointer starts at the beginning of the file
w	**Open a file for write only**. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file
a	**Open a file for write only**. The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist
x	**Creates a new file for write only**. Returns FALSE and an error if file already exists
r+	**Open a file for read/write**. File pointer starts at the beginning of the file
w+	**Open a file for read/write**. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file
a+	**Open a file for read/write**. The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist
x+	**Creates a new file for read/write**. Returns FALSE and an error if file already exists

Example

- ```
  <?php
  $myfile = fopen("webdictionary.txt", "r");
  // some code to be executed....
  fclose($myfile);
  ?>
  ```
  ```

PHP Read File - fread()

- The fread() function reads from an open file.
- The **first parameter** of fread() contains the **name of the file to read** from and the **second parameter specifies the maximum number of bytes to read.**
- The following PHP code opens and reads the "webdictionary.txt" file to the end:

PHP Close File - fclose()

- The **fclose() function is used to close an open file.**
- It's a good programming practice to close all files after you have finished with them.
- You don't want an open file running around on your server taking up resources.
- The **fclose() requires the name of the file (or a variable that holds the filename) we want to close**

Example

- <!DOCTYPE html>
  <html>
  <body>

  <?php
  $myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");
  echo fread($myfile,filesize("webdictionary.txt"));
  fclose($myfile);
  ?>

  </body>
  </html>

Note:

- The die() function prints a message and exits the current script.
- This function is an alias of the exit() function.

Syntax

message	Required. Specifies the message or status number to write before exiting the script. The status number will not be written to the output.

PHP Read Single Line - fgets()

- The **fgets() function is used to read a single line from a file.**
- The example below outputs the first line of the "webdictionary.txt" file:

Example

- <!DOCTYPE html>
  <html>
  <body>

  <?php
  $myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");
  echo fgets($myfile);
  fclose($myfile);
  ?>

  </body>
  </html>

- **Note:** After a call to the fgets() function, the file pointer has moved to the next line.

PHP Check End-Of-File - feof()

- The **feof() function checks if the "end-of-file" (EOF) has been reached.**
- The feof() function is useful for looping through data of unknown length.
- The example below reads the "webdictionary.txt" file line by line, until end-of-file is reached:

Example

```
<!DOCTYPE html>
<html>
<body>

<?php
$myfile = fopen("webdictionary.txt", "r") or die("Unable
to open file!");
// Output one line until end-of-file
while(!feof($myfile))
        {
                echo fgets($myfile) . "<br>";
        }
   fclose($myfile);
?>

</body>
</html>
```

PHP Read Single Character - fgetc()

- The **fgetc() function is used to read a single character from a file.**
- The example below reads the "webdictionary.txt" file character by character, until end-of-file is reached:

Example

- <!DOCTYPE html>
  <html>
  <body>

  <?php
  $myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");
  // Output one character until end-of-file
  while(!feof($myfile))
  {
  echo fgetc($myfile);
  }
  fclose($myfile);
?>
</body>
</html>

Note: After a call to the fgetc() function, the file pointer moves to the next character.

PHP 5 File Create/Write

PHP Create File - fopen()

- The **fopen() function is also used to create a file**. Maybe a little confusing, but in PHP, a file is created using the same function used to open files.

- If you use fopen() on a file that does not exist, it will create it, given that the file is opened for writing (w) or appending (a).

- The example below creates a new file called "testfile.txt". **The file will be created in the same directory where the PHP code resides.**
    - **$myfile = fopen("testfile.txt", "w")**

fwrite() Function

- The **fwrite() function is used to write to a file.**
- The first parameter of fwrite() contains the **name of the file to write to and the second parameter is the string to be written**.
- The example below writes a couple of names into a new file called "newfile.txt”.

Example

- <?php
  $myfile=fopen("newfile.txt", "w") or die("Unable to open file!");
  $txt = "John Doe”;
  fwrite($myfile, $txt);
  $txt = "Jane Doe”;
  fwrite($myfile, $txt);
  fclose($myfile);
  ?>

- Notice that we wrote to the file "newfile.txt" twice. Each time we wrote to the file we sent the string $txt that first contained "John Doe" and second contained "Jane Doe". After we finished writing, we closed the file using the fclose() function.
- If we open the "newfile.txt" file it would look like this:

John DoeJane Doe

PHP Overwriting

- Now that "newfile.txt" contains some data we can show what happens when we open an existing file for writing. **All the existing data will be ERASED and we start with an empty file.**
- In the example below we open our existing file "newfile.txt", and write some new data into it:

Example

- <?php
  $myfile= fopen("newfile.txt", "w") or die("Unable to open file!");
  $txt = “Avita Katal";
  fwrite($myfile, $txt);
  $txt = “Jammu and Kashmir";
  fwrite($myfile, $txt);
  fclose($myfile);
  ?>

- If we now open the "newfile.txt" file, both John and Jane have vanished, and only the data we just wrote is present:

Avita KatalJammu and Kashmir

copy() Function

- The copy() function copies a file.
- This function returns TRUE on success and FALSE on failure.
- **Syntax**

copy(file,to_file)

Parameter	Description
file	Required. Specifies the file to copy
to_file	Required. Specifies the file to copy to

- **Note:** If the destination file already exists, it will be overwritten.

Example

```
<?php
echo copy("webdictionary.txt","webdictionary1.txt");
?>
```

- The output of the code above will be:

1

unlink() Function

- The unlink() function deletes a file.
- This function returns TRUE on success, or FALSE on failure.
- **Syntax**

unlink(filename)

Parameter	Description
filename	Required. Specifies the file to delete

Example

- <?php
  $file = “Random.php";
  if (!unlink($file))
  {
  echo ("Error deleting $file");
  }
  else
  {
  echo ("Deleted $file");
  }
  ?>

file_exists() Function

- Definition and Usage
- The file_exists() function checks whether or not a file or directory exists.
- This function returns TRUE if the file or directory exists, otherwise it returns FALSE.
- **Syntax**

file_exists(path)

-

Parameter	Description
path	Required. Specifies the path to check

Example

```
<?php
echo file_exists("webdictionary.txt");
?>
```

The output of the code above will be:

- 1

file() Function

- The file() reads a file into an array.
- Each array element contains a line from the file, with newline still attached.
- **Syntax**

file(path)

Parameter	Description
path	Required. Specifies the file to read

Example

```
<?php
print_r(file("webdictionary.txt"));
?>
```

Output:

- Array ([0] => synchronous JavaScript and XML [1] => CSS = Cascading Style Sheets [2] => HTML = Hyper Text Markup Language [3] => PHP = PHP Hypertext Preprocessor [4] => SQL = Structured Query Language [5] => SVG = Scalable Vector Graphics [6] => XML = EXtensible Markup Language)

file_get_contents() Function

- The file_get_contents() reads a file into a string.
- This function is the preferred way to read the contents of a file into a string. Because it will use memory mapping techniques, if this is supported by the server, to enhance performance.

- **Syntax**

file_get_contents(path,**include_path,context,start, max_length**)

Parameter	Description
path	Required. Specifies the file to read
include_path	Optional. Set this parameter to '1' if you want to search for the file in the include_path (in php.ini) as well
context	Optional. Specifies the context of the file handle. Context is a set of options that can modify the behavior of a stream. Can be skipped by using NULL.
start	Optional. Specifies where in the file to start reading. This parameter was added in PHP 5.1
max_length	Optional. Specifies how many bytes to read. This parameter was added in PHP 5.1

Example

```
<?php
// Read 14 characters starting from the 21st character
$homepage=file_get_contents('webdictionary.txt',NULL, NULL,20,14);
echo $homepage;
?>
```

file_put_contents() Function

- Write a string to a file.
- This function follows these rules when accessing a file:
  - If FILE_USE_INCLUDE_PATH is set, check the include path for a copy of *filename*
  - Create the file if it does not exist
  - Open the file
  - Lock the file if LOCK_EX is set
  - If FILE_APPEND is set, move to the end of the file. Otherwise, clear the file content
  - Write the data into the file
  - Close the file and release any locks
  - **This function returns the number of character written into the file on success, or FALSE on failure.**

Syntax:

- file_put_contents(file,data,**mode,context**)

Parameter	Description
file	Required. Specifies the file to write to. If the file does not exist, this function will create one
data	Required. The data to write to the file. Can be a string, an array or a data stream
mode	•Optional. Specifies how to open/write to the file. Possible values: •FILE_USE_INCLUDE_PATH •FILE_APPEND •LOCK_EX
context	Optional. Specifies the context of the file handle. Context is a set of options that can modify the behavior of a stream.

Example 1

```
<?php
$file = "webdictionary.txt";
// Open the file to get existing content
$current = file_get_contents($file);
echo $current;
// Append a new person to the file
$current .= "John Smith\n";
// Write the contents back to the file
file_put_contents($file, $current);
echo file_get_contents($file);
?>
```

rename() Function

- The rename() function renames a file or directory.
- This function returns **TRUE on success, or FALSE on failure.**

Syntax

- rename(oldname,newname)

Parameter	Description
oldname	Required. Specifies the file or directory to be renamed
newname	Required. Specifies the new name of the file or directory

Example

```
<?php
rename("images","pictures");
?>
```

fileperms() Function

- The fileperms() function returns the permissions for a file or directory.
- This function **returns the permission as a number on success or FALSE on failure.**

Syntax

- fileperms(filename)

Parameter	Description
filename	Required. Specifies the file to check

Example 1

- <?php
  echo fileperms("test.txt");
  ?>

The output of the code above could be:

- 33206

Example 2

- Display permissions as an octal value:
- <?php
  echo substr(sprintf("%o",fileperms("test.txt")), -4);
  ?>
- Output:0666

Note:

1. The sprintf() function writes a formatted string to a variable. The arg1 %o, parameters will be inserted at percent (%) signs in the main string.
2. The substr() function returns a part of a string.

fileowner() Function

- The fileowner() function returns the user ID (owner) of the specified file.
- This function **returns the user ID on success or FALSE on failure.**
- **Syntax**

fileowner(filename)

Parameter	Description
filename	Required. Specifies the file to check

Note: This function doesn't produce meaningful results on Windows systems.

Example

- <?php
  echo fileowner("test.txt");
  ?>

- Output:

0

chmod() Function

- The chmod() function changes permissions of the specified file.
- Returns **TRUE on success and FALSE on failure.**
- **Syntax**

chmod(file,mode)

Parameter	Description
file	Required. Specifies the file to check
mode	Required. Specifies the new permissions. The mode parameter consists of four numbers: •The first number is always zero •The second number specifies permissions for the owner •The third number specifies permissions for the owner's user group •The fourth number specifies permissions for everybody else Possible values (to set multiple permissions, add up the following numbers): •**1 = execute permissions** •**2 = write permissions** •**4 = read permissions**

Example

- <?php
  // Read and write for owner, nothing for everybody else
  chmod("test.txt",0600);

  // Read and write for owner, read for everybody else
  chmod("test.txt",0644);

  // Everything for owner, read and execute for everybody else
  chmod("test.txt",0755);

  // Everything for owner, read for owner's group
  chmod("test.txt",0740);
  ?>

chown() Function

- The chown() function changes the owner of the specified file.
- Returns **TRUE on success and FALSE on failure.**
- **Syntax**

chown(file,owner)

Parameter	Description
file	Required. Specifies the file to check
owner	Required. Specifies the new owner. Can be a **user name or a user ID.**

Example

- <?php
  chown("test.txt",“avita")
  ?>

dirname() Function

- The dirname() function returns **the directory name from a path.**
- **Syntax**

dirname(path)

Parameter	Description
path	Required. Specifies the path to check

Example

```
<?php
echo dirname("C:\xampp\htdocs\Class\strcmp1.php");
?>
```

Output:

C: mpp\htdocs\Class

disk_free_space() Function

- The disk_free_space() function returns the **free space, in bytes**, of the specified directory.

Syntax

- disk_free_space(directory)

Parameter	Description
directory	Required. Specifies the directory to check

Example

- <?php
  echo disk_free_space("C:");
  ?>
- Output:

46595321856

disk_total_space() Function

- The disk_total_space() function returns the **total space, in bytes,** of the specified directory.

Syntax

- disk_total_space(directory);

Parameter	Description
directory	Required. Specifies the directory to check

Example

- <?php
  echo disk_total_space("C:");
  ?>

- Output:

104333307904

fileatime() Function

- The fileatime() function **returns the last access time of the specified file.**
- This function returns the last access time as a **Unix timestamp on success**, **FALSE on failure.**

Syntax

- fileatime(filename)

Parameter	Description
filename	Required. Specifies the file to check

Example

```
<?php
echo fileatime("file1i.php");
echo "<br />";
echo "Last access: ".date("F d Y H:i:s.",fileatime("file1i.php"));
?>
```

1488329154
Last access: March 01 2017 01:45:54.

Note:

- F - A full textual representation **of a month** (January through December)
- d - **The day of the month** (from 01 to 31)
- H - 24-hour format of an hour (00 to 23)
- Y - A four digit
- i - Minutes with leading zeros (00 to 59)
- s - Seconds, with leading zeros (00 to 59)
- date(*format,timestamp);*

filectime() Function

- The filectime() function returns the **last time the specified file was changed.**
- This function checks for the inode changes as well as regular changes. Inode changes is **when permissions, owner, group or other metadata is changed.**
- This function returns the last change time as a Unix timestamp on success, FALSE on failure. **Syntax**
- filectime(filename)

Parameter	Description
filename	Required. Specifies the file to check

Example

```
<?php
echo filectime("file1i.php");
echo "<br />";
echo "Last change: ".date("F d Y H:i:s.",filectime("file1i.php"));
?>
```

Output: 1488329154
Last change: March 01 2017 01:45:54.

Exercise

Ques1: Write a script to store the roll no, name, age, address, city into the file called studentdetails.txt and the record should be stored in the format given below:

- 101:priya:19:bangalore

Ques 2: Write a function to retrieve the student information from the studentdetails.txt file.

Ques3: Write a script to display the records containing XML from webdictionary.txt.

Ques 4: Write a script to copy content of one file into another.(without using copy())

isset() function

The isset () function is used to check whether a variable is set or not. If a variable is already unset with unset() function, it will no longer be set. The isset() function return false if testing variable contains a NULL value.

Example

```
<?php
$var1 = 'test';
var_dump(isset($var1));
```

Output:

```
bool(true)
```