Unit 5.2: File and directory access in PHP

1. Unit Structure

1. Learning Objectives
2. Introduction
3. File Handling Functions
4. Directory Handling Functions

2. Learning Objectives

After completing this unit, students will be able to

5. Read data from file.
6. Write data to file.
7. Read and Write data from random position.
8. Handle directories

3. Introduction

There are occasions where we need to interact with the file and directory in web applications. You might have some data available in text file or csv file and you need to read that data in your web application to have some analysis on it. These require you to open the file and perform some reading and writing operations on it. PHP provides different functions to interact with files and directories. PHP has several functions for creating, reading, writing, updating and uploading files. In this unit we will learn all these functions with examples.

4. File Handling

To perform any reading or writing operation on the file we need to open it first. PHP provides fopen() to open the file.

4.1. fopen(filename,mode)

This function is used to open the file for different operations. Filename is the name of the file that we want to open, and mode decides which operation we want to perform on the file.

The Different modes are:

9. r - open a file for reading only. File pointer starts from the beginning of the file.
10. w - open a file for write only. Deletes the contents of the file if already exists or creates a new file if it doesn't exist. File pointer starts at the beginning of the file.
11. a - Open a file for write only. The existing data in the file is not deleted if the file already exist. File pointer starts at the end of the file. Creates a new file if the file doesn't exist.
12. x - creates a new file for write only. Returns FALSE and an error if the file already exists.
13. r+ - open a file for read/write. File pointer starts at the beginning of the file.
14. 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.
15. a+ - open a file for read/write. The existing data in the file is not deleted. File pointer starts at the end of the file. Creates a new file if the file doesn't exist.
16. x+ - creates a new file for read/write. Returns FALSE and an error if file already exists.

4.2 fclose(filepointer)

This function is used to close the opened file.

Example:

```php
<?php
$fp=fopen("Myfile.txt","r");
fclose($fp)
?>
```

The above example opens the file Myfile.txt in read mode. If the file is available in the same directory, then there is no need to mention full path, else we need to write full path of the file. $fp is a file pointer which holds the file which is opened. The next statement closes the file which is pointed by $fp.

4.3 fread(filepointer, length)

This function reads the contents from the file and stops when it reaches the end of file or at specified length which comes first.

Example:

```
<?php
$fp=fopen("Myfile.txt","r");
echo fread($fp, filesize("Myfile.txt"));
?>
```

It will read all the contents from Myfile.txt. Instead of specifying length of bytes to be read, we have used filesize function which will return the size of Myfile.txt in bytes. If we write 5 as the first argument ($fp), then it will read the first 5 bytes of data from Myfile.txt file.

4.4 fgetc(filepointer)

This function is used to read a single character from the file.

4.5 fgets(filepointer)

This function is used to read a single line from the file.

4.6 fputs(filepointer,string)

This function is used to write single line into the file.

4.7 fwrite(filepointer,string,length)

The fwrite function is used to write string into file. Length is optional. If length is specified, then that number of bytes will be written into the file. If it reaches to the end of the file, then it stops writing. It returns the number of bytes written into the file.

Example:

```
<?php
$fp1=fopen("write.txt","w");
fwrite($fp1,"I am learning PHP");
?>
```

The above statements will write “I am learning PHP” text into write.txt file. If write.txt file has some contents already written into it, then they are erased because w mode starts writing from the beginning of the file. If you want to preserve the old contents and want to add the new contents at the end of the file, then instead of w mode use a (append) mode.

4.8 file_exist(file)

This function is used to check whether the specified file exists or not. Return true on success and false on failure.

Example:

```
<?php
if(file_exists("myfile.txt")) {
    $fp=fopen("myfile.txt","r");
    echo fread($fp,5);
    }
    else
    echo "<br> file does not exist";
?>
```

The above code first checks whether the file myfile.txt exists in the current directory or not? If it exists, then it will read the first 5 characters of that file and print it. If the file is not available in the current directory, then the code will print the message that "file does not exist".

Example of fgets and fputs

```
<?php
$fp=fopen("read.txt","r");
$fp1=fopen("write.txt","a");
while(!feof($fp)) {
//feof checks for end of file. Returns true if file pointer reaches to end of
file.
    $read=fgets($fp);
    fputs($fp1,$read);
}
fclose($fp);
?>
```

The Above code reads the contents from read.txt file and writes it into write.txt file.

5 Reading the data from Random position

In the above examples, we read or write data from beginning or end but sometimes we need to read or write data from random position of the file. To read or write data from a random position, we need to set the file pointer to the desired position. In PHP we have fseek() function to set file pointer at random position.

5.1 fseek(file , offset , whence)

The fseek function moves the file pointer from its current position to a new position, forward or backward, specified by the number of bytes. It is used for random location reading and writing in file. File position starts with 0.

- offset specifies the new position of the pointer. It is measured in bytes.
- whence can be SEEK_SET, SEEK_CUR, SEEK_END
  - SEEK_SET - It sets position equal to offset.
  - SEEK_CUR - It sets position to current location plus offset.
  - SEEK_END - It sets position to EOF plus offset. To move to a position before EOF, the offset must be a negative value.

Example 1:

```php
<?php
$fp=fopen("random.txt","r");
fseek($fp,0); // Sets the pointer to beginning of the file.
fseek($fp,7); // Sets the file pointer to 7th position.
$str=fread($fp,4); // read next 4 bytes from 7thposition and assign them to
$str variable.
echo "<br>".$str;
?>
```

Example 2:

```php
<?php
$fp=fopen("random.txt","r");
fseek($fp,7,SEEK_SET); // Sets the file pointer to 7th position and read from
there.
$str=fgets($fp);
echo "<br>Seek Set: ".$str; fseek($fp,7);
fseek($fp,8,SEEK_CUR); /*Sets the file pointer to 7th position and then move
8 position in forward direction and read from the new position which is 15.*/
$str=fgets($fp);
echo "<br>Seek Cur: ".$str;

fseek($fp,0);
fseek($fp,-10,SEEK_END);
$str=fgets($fp);
echo "<br>Seek end: ".$str;
?>
```

6. Directory Handling

The directory functions in PHP allow you to retrieve information about directories and to manipulate them. There are different functions available in PHP to work with directories. In this section we are going to discuss important directory handling functions of PHP.

6.1. getcwd()

This function is used to get the current working directory. It returns the current working directory and returns false in case of failure.

```php
<?php
echo getcwd();
?>
```

Output: C:\xampp\htdocs\test

6.2. chdir(directory)

The chdir() function is used to change the current directory. Directory is the name of directory that we need to change.

```php
<?php
echo getcwd();
chdir("Images");
echo "<br>";
echo getcwd();
?>
```

Output:

C:\xampp\htdocs\test

C:\xampp\htdocs\test\images

6.3. opendir()

This function opens the directory handle. You need to specify the path of the directory which you want to open.

6.4. closedir(dir)

This function is used to close the directory handle resource specified by dir.

6.5. readdir($dir)

This function returns the name of the next file from the opened directory. $dir is the directory handle resource opened with opendir(). It returns the name of the file or it returns FALSE on failure.

Example:

```php
<?php
$d = opendir("Images");
$i=1;
while($file = readdir($d)){
    echo "<br>"."file$i:".$file;
    $i++;
}
closedir($d);
?>
```

Output:

file1:. file2:..

file3:product1.jpg file4:product2.jpg file5:product3.jpg

The above code will open “Images” directory and then read files one by one from it.