PHP File Handling

- File handling is an important part of any web application.
- You often need to open and process a file for different tasks.
- PHP has several functions for creating, reading, uploading, and editing files.

PHP readfile() Function

- The readfile() function reads a file and writes it to the output buffer.
- Assume we have a text file called “data.txt", stored on the server, that looks like this:
- Content of data.txt

  Hello

  I am PHP

  I am Server side Scripting Language

  I am easy to Learn

Code of readfile() function

- The PHP code to read the file and write it to the output buffer is as follows
- readfile() function returns the number of bytes read on success.
- The readfile() function is useful if all you want to do is open up a file and read its contents.

```
<?php
        $filename=“data.txt";
       echo readfile($filename);
  ?>
```

Creating & Opening a File

We can use the fopen() function to open a file.

We can also use this function to create a file.

fopen() needs two parameters to work. First, it needs the name of the file that is being opened and the second parameter which specifies in which mode it is being opened.

Syntax

$var=fopen(filename,filemode);

e.g

$myfile=fopen(“data.txt” ,”r”);

The fopen() function has the following modes

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

Reading a File (fread())

- We can also use the **fread()** to read from an open file.
- This function also has two parameters, first one containing the name of the and the second one specifying the maximum number of bytes to read.
- Example:

```
fread($myfile,filesize("data.txt"));
```

Example Reading a File (fread())

```
<?php
        $myfile=fopen("myfile.txt", "r") or die("unable to open file");
      $data=fread($myfile, filesize("myfile.txt"));
      echo $data;
     fclose($myfile);



     ?>
```

Writing to a File with fwrite() function

- We can use the fwrite() function to write to a file.
- This function also takes 2 parameters. The first one is the filename and the second one specifies the string to be written.

```
<?php
   $myfile=fopen("newfile.txt", "w") or die("unable to create file");
   $data="Hello This is php";
   fwrite($myfile, $data);
   echo "data is wriiten in file";
   fclose($myfile);
  ?>
```

Note: We can also append data into the file by opening the file in append "a" mode.

Closing a File

- To close a file, use the **fclose()** function.

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

Reading a single line from file

We can use the **fgets()** to read a single line from a file.

```
<?php
    $myfile=fopen("myfile.txt", "r") or die("unable to open a file");
    $line= fgets($myfile);
    echo $line;
    fclose($myfile);

    ?>
```

Reading a a single character:

- We can use the **fgetc()** to read a single character from a file.

```
<?php
    $myfile=fopen("myfile.txt", "r") or die("unable to open a file");
    $char= fgetc($myfile);
    echo $char;
    fclose($myfile);

    ?>
```

PHP Check End-Of-File - feof()

- The feof() function checks if the "end-of-file" (EOF) has been reached.

```
<?php
    // put your code here
    $myfile=fopen("myfile.txt", "r") or die("unable to open a file");
    while(!feof($myfile))
    {
    $char= fgetc($myfile);
    echo $char;
    }
    fclose($myfile);
 ?>
```

Other File Operations

- Delete file
  - `unlink('filename');`
- Rename (file or directory)
  - `rename('old name', 'new name');`
- Copy file
  - `copy('source', 'destination');`
- And many, many more!
  - www.php.net/manual/en/ref.filesystem.php

Dealing With Directories

- Open a directory
  - `$handle = opendir('dirname');`
    - `$handle` 'points' to the directory
- Read contents of directory
  - `readdir($handle)`
    - Returns name of next file in directory
    - Files are sorted as on filesystem
- Close a directory
  - `closedir($handle)`
    - Closes directory 'stream'

Directory Example

```
$handle = opendir('./');

while(false !== ($file=readdir($handle)))
  {
  echo "$file<br />";
  }

closedir($handle);
```

Other Directory Operations

- Get current directory
  - `getcwd()`
- Change Directory
  - `chdir('dirname');`
- Create directory
  - `mkdir('dirname');`
- Delete directory (MUST be empty)
  - `rmdir('dirname');`
- And more!
  - www.php.net/manual/en/ref.dir.php

PHP File Upload

- Let’s understand PHP $_FILES First before understanding the file upload.
- The global predefined variable $_FILES is an associative array containing items uploaded via HTTP POST method.
- Uploading a file requires HTTP POST method form with enctype attribute set to multipart/form-data.

$_FILES

- The $_FILES array contains following properties –
- **$_FILES['file']['name']** - The original name of the file to be uploaded.
- **$_FILES['file']['type']** - The mime type of the file.
- **$_FILES['file']['size']** - The size, in bytes, of the uploaded file.
- **$_FILES['file']['tmp_name']** - The temporary filename of the file in which the uploaded file was stored on the server.
- **$_FILES['file']['error']** - The error code associated with this file upload.

Example of $_FILES
coding of index.php

```
<html>
   <head>
      <meta charset="UTF-8">
      <title></title>
   </head>
   <body>
      <form method="post" action="newfile.php" enctype="multipart/form-data">
         <input type="file" name="myfile">
         <input type="submit" name="submit">
      </form>

   </body>
</html>
```

Coding of myfile.php

```
<?php
    // put your code here
    $filename=$_FILES['myfile']['name'];
    $filetype=$_FILES['myfile']['type'];
    $filesize=$_FILES['myfile']['size'];
    $tempname=$_FILES['myfile']['tmp_name'];
    $error=$_FILES['myfile']['error'];
    echo "File name=".$filename."<br>";
    echo "File type=".$filetype."<br>";
    echo "File size=".$filesize."<br>";
    echo "Temp Name=".$tempname."<br>";
    echo "error=".$error."<br>";
 ?>
```

File Upload Example
coding of index.php

```
<html>
   <head>
      <meta charset="UTF-8">
      <title></title>
   </head>
   <body>
      <form method="post" action="next.php" enctype = "multipart/form-data">
         <label> Name</label> <input type='text' name='txtname'>
         <label> Per</label> <input type='text' name='txtper'>
          <input type='file' name='image'>
         <input type="submit" value="upload">
      </form>

   </body>
</html>
```

Coding of next.php

```php
<?php
    if (isset($_FILES['image'])) {
        $errors = array();
        $file_name = $_FILES['image']['name'];
        $file_size = $_FILES['image']['size'];
        $file_tmp = $_FILES['image']['tmp_name'];
        $file_type = $_FILES['image']['type'];
        $file_ext = strtolower(end(explode('.', $_FILES['image']['name'])));
        $extensions = array("jpeg", "jpg", "png");
if (in_array($file_ext, $extensions) === false) {
            $errors[] = "extension not allowed, please choose a JPEG or PNG file.";
        }
        if ($file_size > 2097152) {
            $errors[] = 'File size must be excately 2 MB';
        }
        if (empty($errors) == true) {
            move_uploaded_file($file_tmp, "images/" . $file_name);
            echo "Success";
        } else {
            print_r($errors);
        }
    }
```

Cont….

```
try {
        $name = $_POST['txtname'];
        $per = $_POST['txtper'];
        $path = "images/$file_name";
        $hostname = "localhost";
        $username = "root";
        $password = "";
        $database = "mydb";
        $conn = new mysqli($hostname, $username, $password, $database);
        if ($conn->connect_error) {
            die("Connection Faield" . $conn->connect_error);
        }
         $sql = "INSERT INTO `student` ( `name`, `per`, `photo`) VALUES ('$name',
'$per', '$path')";
                if ($conn->multi_query($sql) === TRUE) {
                    echo 'Data Is Inserted';
                } else {
                    echo 'Error' . $conn->error;
                }
                $conn->close();
            } catch (Exception $e) {
                echo 'Data Is Not Inserted';
            }
            ?>
            <a href="/fileupload/display.php">Click to display all data </a>
        </body>
</html>
```

Coding of display.php

```
<?php
    // put your code here
    try {
        $hostname = "localhost";
        $username = "root";
        $password = "";
        $database = "mydb";
        $conn = new mysqli($hostname, $username, $password, $database);
        if ($conn->connect_error) {
            die("Connection Faield" . $conn->connect_error);
        }
$sql = "Select * From student ";
        $result = $conn->query($sql);
        if ($result->num_rows > 0) {
            while ($row = $result->fetch_assoc()) {

                echo "<table> <tr> <td> $row[rno] </td> <td>$row[name] </td> <td> $row[per]</td> <td><img src=$row[photo]> </td> </tr> </table>";
}
        }
    } catch (Exception $e) {
        echo "error";
    }
    ?>
```

Multiple file upload example code of index.php

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="UTF-8">
      <title></title>
   </head>
   <body>

<form method="post" action="upload.php" enctype="multipart/form-data">
   <input type="file" name="files[]" multiple>
   <input type="submit" value="Upload">
</form>

   </body>
</html>
```

Coding of upload.php

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
    </head>
    <body>
        <?php

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    // Loop through each uploaded file
    foreach ($_FILES['files']['name'] as $index => $filename) {
        $file_tmp = $_FILES['files']['tmp_name'][$index];
        // echo "$index=>$filename";
        //$file_name = basename($filename);
 $file_type = $_FILES['files']['type'][$index];
        $file_size = $_FILES['files']['size'][$index];


        // Handle the uploaded file (e.g. move it to a specific directory)
        move_uploaded_file($file_tmp, 'images/' . $filename);
    }
}
?>

    </body>
</html>
```

Namespace

- In PHP, a namespace is a mechanism that allows developers to group related classes, functions, and constants together under a specific name.
- This helps to avoid naming conflicts and makes it easier to organize and maintain code.

Creating a Namespace

- To create a namespace in PHP, you use the namespace keyword followed by the name of the namespace.
- Here is an example:

Syntax:

namespace MyNamespace;

Define classes , function and constant in namespace

This declares a new namespace called MyNamespace. You can then define your classes, functions, and constants within this namespace.

```
namespace MyNamespace;

class MyClass {
  // class definition
}

function myFunction() {
  // function definition
}

const MY_CONSTANT = 123;
```

Importing a Namespace

- To use code from a namespace in your PHP script, you need to import it using the use keyword. Here is an example:

Syntax:

```
use MyNamespace\MyClass;
$obj = new MyClass();
```

- This imports the MyClass class from the MyNamespace namespace and allows you to create an instance of it.

Import multiple classes from a namespace

- You can also import multiple classes from a namespace using the curly braces syntax:

Syntax:

```
use MyNamespace\{MyClass, MyOtherClass};


$obj1 = new MyClass();
$obj2 = new MyOtherClass();
```

Aliasing a Namespace

- If you have multiple namespaces with the same class names, you can use aliases to differentiate between them. Here is an example:

Syntax:

```
use MyNamespace\MyClass as MyClass1;
use AnotherNamespace\MyClass as MyClass2;

$obj1 = new MyClass1();
$obj2 = new MyClass2();
```

- This imports the MyClass class from two different namespaces and creates aliases for them (MyClass1 and MyClass2). You can then use these aliases to differentiate between the two classes.

Coding of A.php

```
<?php
namespace MyNamespace;
class MyClass {
  public function sayHello() {
    echo "Hello from MyNamespace\MyClass!"."<br>";
  }
}

function myFunction() {
  echo "Hello from MyNamespace\myFunction!"."<br>";
}

const MY_CONST = "Hello from MyNamespace\MY_CONST!"."<br>";

?>
```

Coding of B.php

```
<?php
namespace YourNamespace {

    class MyClass {

        public function sayHello() {
            echo "Hello from YourNamespace\YourClass!" . "<br>";
        }

    }

}
?>
```

Coding of index.php

```
<?php
    require_once('A.php');
    include 'B.php';
    use MyNamespace\MyClass as Class1;
    use YourNamespace\MyClass as Class2;
    use function MyNamespace\myFunction;
    use const MyNamespace\MY_CONST;
    $obj1 = new Class1();
    $obj1->sayHello();
    myFunction();
    echo MY_CONST;
    $obj2 = new Class2();
    $obj2->sayHello();
    ?>
```

CSV (Comma Separated Values)

- CSV (Comma Separated Values) is a file format used to store data in a structured way.
- PHP provides built-in functions to read, write and manipulate CSV files.

Reading a CSV file

```
<?php
    // Open the file for reading
    $file = fopen('student.csv', 'r');
    $data = array();
    while (($row = fgetcsv($file)) !== false) {
       $data[] = $row; // Add each row to an array
    }
    foreach ($data as $value) {
            echo "$value[0]"."$value[1]"."$value[2]"."</br>";

     //echo implode(', ', $value) . '<br>';
    }
        fclose($file); // Close the file
    ?>
```

Writing to CSV file

```php
<?php
    $file = fopen('data.csv', 'w'); // Open the file for writing
    $data = array(
        array('1', 'Ram Patel', '70'),
        array('2', 'Laxman Prajapati', '80'),
        array('3', 'Bharat', '90')
    );

    foreach ($data as $row) {
        fputcsv($file, $row); // Write each row to the file
    }
    fclose($file);
    ?>
```