The following steps need to be followed to upload an image and display it on a website using PHP:

- Create a form using HTML for uploading the image files.
- Connect with the database to insert the image file.

Step 1: Create a Form Using HTML for Uploading the Image Files.

The following HTML code will create a simple form on your website, with a “choose file” option and a button to upload the chosen file.

```
<!DOCTYPE html>

<html>

<head>

   <title>Image Upload in PHP</title>

   <! link the css file to style the form >

   <link rel="stylesheet" type="text/css" href="style.css" />

</head>

<body>

   <div id="wrapper">

      <! specify the encoding type of the form using the

            enctype attribute >
```

```
      <form method="POST" action="" enctype="multipart/form-data">
         <input type="file" name="choosefile" value="" />
         <div>
            <button type="submit" name="uploadfile">
            UPLOAD
            </button>
         </div>
      </form>
   </div>
</body>
</html>
```

The following CSS code is for giving a basic styling to the HTML form.

```
#wrapper{
   width: 50%;
   margin: 20px auto;
   border: 2px solid #dad7d7;
}
```

```
form{
    width: 50%;
    margin: 20px auto;
}
form div{
    margin-top: 5px;
}
img{
    float: left;
    margin: 5px;
    width: 280px;
    height: 120px;
}
#img_div{
    width: 70%;
    padding: 5px;
    margin: 15px auto;
```

```
    border: 1px solid #dad7d7;
}
#img_div:after{
    content: "";
    display: block;
    clear: both;
}
```

Step 2: Connect with the Database to Insert the Image File.

The following PHP code will connect the database to insert the submitted data into the database.

```
<?php
error_reporting(0);
?>
<?php
$msg = "";
// check if the user has clicked the button "UPLOAD"
```

```
if (isset($_POST['uploadfile'])) {
    $filename = $_FILES["choosefile"]["name"];
    $tempname = $_FILES["choosefile"]["tmp_name"];
        $folder = "image/".$filename;
    // connect with the database
    $db = mysqli_connect("localhost", "root", "", "Image_Upload");
        // query to insert the submitted data
        $sql = "INSERT INTO image (filename) VALUES ('$filename')";
        // function to execute above query
        mysqli_query($db, $sql);
        // Add the image to the "image" folder"
        if (move_uploaded_file($tempname, $folder)) {
            $msg = "Image uploaded successfully";
        }else{
            $msg = "Failed to upload image";
    }
}
```

```
$result = mysqli_query($db, "SELECT * FROM image");

?>
```

Program Explanation

HTML Program

The above HTML program creates a simple HTML form having an option to choose a file from your system to upload, and an “UPLOAD” button. It will upload the image file using the POST method.

enctype attribute: This attribute is used to specify the encoding format, in which the data submitted in the form has to be encoded before sending it to the server. This attribute is very important and without specifying this, the image will not be uploaded to the server.

Syntax

enctype="multipart/form-data"

PHP Program

First, you need to create a database using the XAMPP/WAMP server. Here, you have to create a database and name it “Image_Upload”. Next, you need to create a new table in the database. You have created a new table named “Image”. Create two fields in the table:

Id – int(11)

Filename – varchar(100)

The PHP program discussed earlier will first connect to the database using the mysqli_connect() method. The data submitted using the POST method in the HTML form is stored in a variable $filename. To insert the data into the database, you have to apply an SQL query. The mysqli_query() method executes the SQL query and finally stores the submitted data into the database.

PHP Methods Used in the Program

isset

The isset method is an essential built-in PHP method. This method is used to find if it set a variable or not. A variable is said to be a set variable if it holds a value other than NULL. In other words, the isset method is used to determine if a variable has been declared and has a non-null value or not. It has a boolean return type. If it finds the variable to be set, this method will return true, otherwise, it will return false.

This method proves to be more useful in longer codes. In small pieces of code, you can easily track what and how many variables have been declared. But in the case of longer programs, you might face the problem of losing the track of variables. In such cases, the isset method is very helpful. You can pass the name of the variable that you want to check.

The syntax for checking the status of a variable using the isset method:

isset (mixed $var1 , mixed $var2, ...$vars)

This method accepts mixed parameters. Mixed variables mean that you can pass one or more variables of different data types to isset in PHP as parameters.

$_POST

$_POST is a global method in PHP that is used to send user data submitted into an HTML form from the browser to the webserver. The data is first encoded and then transferred

to the server via the QUERY_STRING HTTP header. Being a superglobal method, you can use anywhere it in the entire program. The isset method can also be used with the $_POST method to check whether the user has submitted a value or not.

The data sent using the $_POST method can be of any size. Also, this method is secure since, unlike the $_GET method, the data is invisible and cannot be accessed by anyone from the URL. This method accepts an array as its parameter and supports multiple data types like string, integer, and binary.

mysqli_connect

This method is used to connect a PHP application to the web server by establishing a secure connection between them. Another method called mysql_connect also does the same task, but mysqli_connect is more secure and the “i” here represents an improved version.

The syntax to establish a new connection using the mysqli_connect method:

mysqli_connect ("host","username","password","database_name")

This method accepts 4 parameters:

host: This is an optional parameter. If you are on a local server, then pass NULL or the keyword “localhost” to specify that the connection has to be made with the local webserver.

username: This parameter is used to specify the username of MySQL. This is also an optional parameter. If you are on a local server, then the username will be “root”.

password: This parameter is used to specify the password of the user in MySQL. You can skip this parameter as it is an optional parameter.

name_of_database: This parameter is the name of the database you want to establish the connection with. It will perform all the queries on this specified database. This parameter is also optional.

mysqli_query

This method is used to execute the query that is passed as a parameter to it. The queries insert, select, update, and delete can be executed using the mysqli_query method. If the queries are successfully executed, then this method will return an object or a boolean value TRUE, depending upon the type of query executed.

The syntax to execute a MySQL query using the mysqli_query method in PHP:

mysqli_query($connection, query, mode)

This method accepts 3 parameters:

connection: This is a mandatory parameter, and it represents the object of the server with which the connection is established.

query: This parameter is also mandatory, and it represents the SQL query string that needs to be executed.

mode: This is an optional parameter. It represents the mode of the result. It is a constant and is used to store the return value of the method.

move_uploaded_file

This method is used to move a specified file (that has already been uploaded) to a new location. This method first validates the specified file, and if it is valid, then moves it to

the destination location. If there is already a file in the destination location, then it is overwritten by the new file.

The syntax to move a file to a specific location using the move_uploaded_file method:

move_uploaded_file(file_name, destination_path)

This method accepts 2 parameters:

file_name: This parameter is a string that represents the name of the file to be moved to a new location.

destination_path: This is a string specifying the destination location where the file needs to be moved.

Steps to Exceed the Size of Image Upload

The program depicted above can upload a file of up to 2MB in size. This is the default file size in PHP. This size limit can be updated and exceeded according to your choice. To increase the size limit for file upload, follow the steps discussed below:

Go to the C drive and open the folder named WAMP or XAMPP server.

Click on “bin” to open this folder.

Open the folder named as the PHP version (the version which you are using).

In this folder, search and go to “php.ini”.

Now search for the variables:

upload_max_size = 100M

post_max_filesize = 100M

Update the new values of these variables and save them.

Now go to this path: “C:\wamp64\bin\apache\apache2.4.27\bin”.

Search and go to “php.ini” and make the same changes.

Save the changes.

Finally, restart your WAMP or XAMPP server.

Run your code on the server.

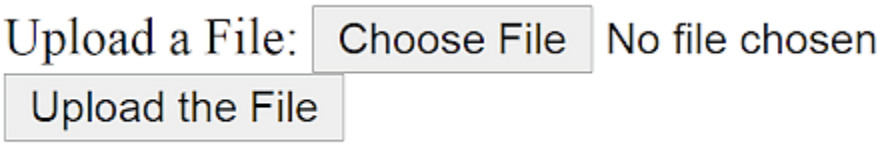
Creating the HTML Form for File Upload in PHP

Now that we are done with the configuration settings let's move ahead with creating an HTML form for uploading the file. For this, we will be creating the index.php file and save it inside a folder. Below is the code for the index.php file.

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<head>
 <title>PHP File Upload</title>
</head>
<body>
 <?php
  if (isset($_SESSION['message']) && $_SESSION['message'])
  {
   printf('<b>%s</b>', $_SESSION['message']);
   unset($_SESSION['message']);
  }
 ?>
 <form method="POST" action="fileUpload.php" enctype="multipart/form-data">
  <div>
   <span>Upload a File:</span>
   <input type="file" name="uploadedFile" />
  </div>
  <input type="submit" name="uploadBtn" value="Upload the File" />
 </form>
</body>
</html>
```

When you run the above code through your local server, it will give the following output.

Output:



Some Important Things to Note From the Form

There are some things to note in the above HTML form.

- action="fileUpload.php": The value in the action field refers to the file that will handle the file upload in PHP. We will create the file in a moment.
- method="POST": This value indicates the browser about the script's action to upload the selected file.
- enctype="multipart/form-data": This value refers to the content type of the files that will be accepted for uploading. It also indicates the type of encoding that the PHP script will use for uploading. The multipart/form-data value enables us to upload files using the POST method. It also ensures the characters of the files are not encoded while submitting the form. Besides multipart/form-data, enctype also accepts application/x-www-form-urlencoded and text/plain values.
- The message variables used at the start of the form will display the status of the upload. They will also show successful or error messages depending on the status of the upload.

Running the file through the server will allow you to browse and choose any file from your computer.

Creating the Upload Logic for File Upload in PHP

The HTML form represents the client-side code. Now that our form is ready, let's move towards the server-side scripting to handle the file upload in PHP. Below is the code that you need to copy in the fileUpload.php file.

```
<?php
session_start();
$message = '';
if (isset($_POST['uploadBtn']) && $_POST['uploadBtn'] == 'Upload the File')
{
  if (isset($_FILES['uploadedFile']) && $_FILES['uploadedFile']['error'] ===
UPLOAD_ERR_OK)
  {
    // uploaded file details
    $fileTmpPath = $_FILES['uploadedFile']['tmp_name'];
    $fileName = $_FILES['uploadedFile']['name'];
    $fileSize = $_FILES['uploadedFile']['size'];
    $fileType = $_FILES['uploadedFile']['type'];
    $fileNameCmps = explode(".", $fileName);
    $fileExtension = strtolower(end($fileNameCmps));
    // removing extra spaces
    $newFileName = md5(time() . $fileName) . '.' . $fileExtension;
    // file extensions allowed
    $allowedfileExtensions = array('jpg', 'gif', 'png', 'zip', 'txt', 'xls', 'doc');
    if (in_array($fileExtension, $allowedfileExtensions))
    {
      // directory where file will be moved
      $uploadFileDir = 'C:\xampp\htdocs\test';
      $dest_path = $uploadFileDir . $newFileName;
      if(move_uploaded_file($fileTmpPath, $dest_path))
      {
```

```
      $message = 'File uploaded successfully.';
    }
    else
    {
      $message = 'An error occurred while uploading the file to the destination directory. Ensure that the web server has access to write in the path directory.';
    }
  }
  else
  {
    $message = 'Upload failed as the file type is not acceptable. The allowed file types are:' . implode(',', $allowedfileExtensions);
  }
 }
 else
 {
  $message = 'Error occurred while uploading the file.<br>';
  $message .= 'Error:' . $_FILES['uploadedFile']['error'];
 }
}
$_SESSION['message'] = $message;
header("Location: index.php");
```

Some Important Things to Note From the Upload Code

There are some things to note in the above upload code.

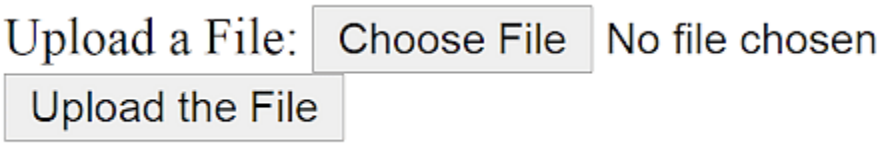
- The first thing we did was to check if the file had come from a valid source or not. We used the if statement and cross-checked them with the upload button's variables' values.
- When the file is uploaded, the $_Files superglobal variable is populated with the following information:

- tmp_name: Temporary path that holds the upload file
- name: Name of the file
- size: The size of the uploaded file in bytes
- type: Information about the mime type of the uploaded file
- error: In case of any error while uploading, this variable gets the appropriate error message

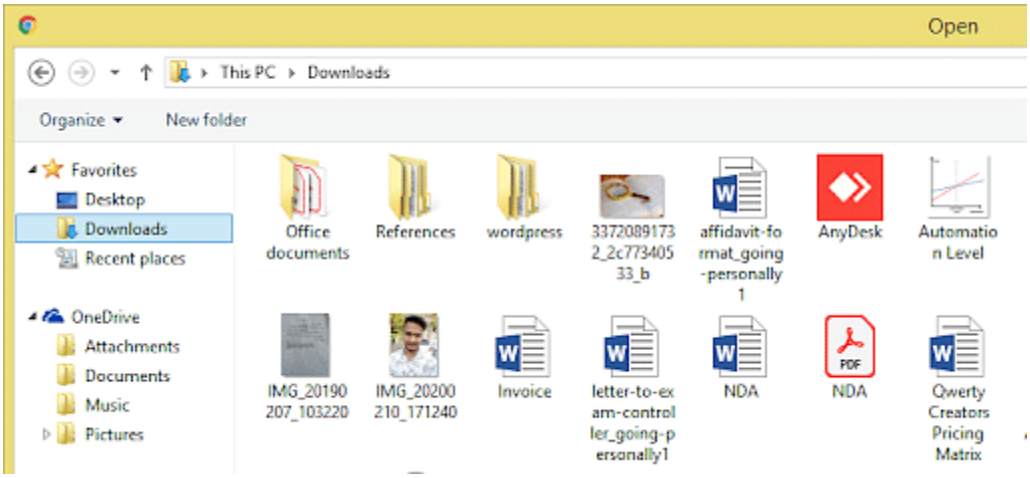
- Through the inner if statement, we checked whether the file upload in PHP was successful or not.
- We then use the multi-dimensional $_Files array that holds the information as discussed above.
- We then figured out the file's extension and matched it with the allowed types.
- Next, we removed all the additional spaces from the file.
- After cleaning the file data, we used the move_uploaded_file function to transfer the file to the desired location.
- In the end, we wrote the success and error messages that would be fetched and shown by the error variable discussed earlier.

How Both Files Work Together to Enable File Upload in PHP

When both the files are in place, run the webserver to see the file upload in action. In the beginning, you will see the HTML form that we have created in the 'index.php' file. This will allow you to choose a file.

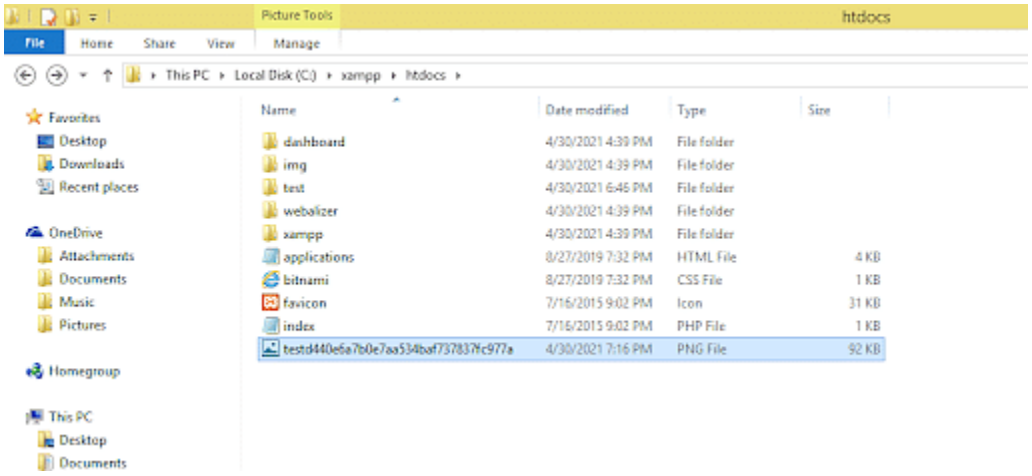


Ensure selecting a file with an acceptable extension. For this example, let's choose a 'png' file.



Once you select the file, you can click on the "Upload the File" button. This will initiate the file upload in PHP. When the form is submitted, the file transfer will take place. If everything goes well, you will see the success message. But if an error occurs, you will see a relevant error message.

Once the upload is successful, you will be able to see the uploaded file in the target directory, set to 'C:\xampp\htdocs\test' in the above code. You can set it to anything according to your preference.



Note: The name of the file will be changed.

Resolving the Common Errors That May Be Encountered While a File Upload in PHP

Although file upload in PHP is easy, you may encounter some common errors. The error messages will give you a hint about what error has occurred during the upload. However, you can use the below code to get the exact and detailed reason.

$_Files[‘uploadedFile’][‘error’]

Here are some standard errors that you may face.

The File Is Too Large

You may face UPLOAD_ERR_INI_SIZE or UPLOAD_ERR_FROM_SIZE errors if the size of the selected file is larger than the specified limits. You can easily conquer the error by changing the configuration of the maximum file size key.

Temporary Folder is Missing

Sometimes you can encounter two types of folder errors. The first one is the UPLOAD_ERR_NO_TMP_DIR error thrown if the temporary folder to hold the uploaded files is missing. You will see the UPLOAD_ERR__NO_FILE error if no file is selected to be uploaded.

Partial Upload

The UPLOAD_ERR_PARTIAL error is thrown if the server is not able to upload the file completely.

Can’t Write to Disk

You will get the UPLOAD_ERR_CANT_WRITE error when the server cannot write the file to the disk for whatever reason.

A PHP Extension Stopped the File Upload

You will get the UPLOAD_ERR_EXTENSION if the upload is halted due to any extension error. Finding the actual extension that caused the error might be tricky here, especially if you have uploaded multiple files with different extensions.