Experiment No. : 12

Title: Write a program to develop file up-loader form to upload a file using PHP.

Objectives:

1. To learn to upload file to server.
2. To design file uploader form.
3. To learn saving uploaded file on server.
4. To learn putting restrictions on file to be uploaded.

Theory:

With PHP, it is possible to upload files to the server.

Create an Upload-File Form

Look at the following HTML form for uploading files with name fileloader.html:

```
<html>
<body>
<form action="upload_file.php" method="post" enctype="multipart/form-data">
<label for="file">Filename: </label>
<input type="file" name="file" id="file" />
<br />
<input type="submit" name="submit" value="Submit" />
</form>
</body>
</html>
```

The enctype attribute of the <form> tag specifies which content-type to use when submitting the form. "multipart/form-data" is used when a form requires binary data, like the contents of a file, to be uploaded.

The type="file" attribute of the <input> tag specifies that the input should be processed as a file. For example, when viewed in a browser, there will be a browse-button next to the input field.

Allowing users to upload files is a big security risk. Only permit trusted users to perform file uploads.

Saving the Uploaded File

The examples above create a temporary copy of the uploaded files in the PHP temp folder on the server.

The temporary copied files disappear when the script ends. To store the uploaded file we need to copy it to a different location:

Create upload_file.php and include following code in it.

```
<html>
<body>
<? php
```

DKTE CSE

```
if ((($_FILES["file"]["type"] == "image/gif") || ($_FILES ["file"] ["type"] == "image/jpeg")
|| ($_FILES ["file"] ["type"] == "image/pjpeg")) && ($_FILES ["file"] ["size"] < 20000))
 {
 if ($_FILES["file"]["error"] > 0)
  {
  echo "Return Code: " . $_FILES ["file"]["error"] . "<br />";
  }
 else
  {
  echo "Upload: " . $_FILES ["file"]["name"] . "<br />";
  echo "Type: " . $_FILES ["file"]["type"] . "<br />";
  echo "Size: " . ($_FILES ["file"]["size"] / 1024) . " Kb<br />";
  echo "Temp file: " . $_FILES ["file"]["tmp_name"] . "<br />";
  if (file_exists("upload/" . $_FILES ["file"]["name"]))
   {
   echo $_FILES ["file"]["name"] . " already exists. ";
   }
  else
   {
   move_uploaded_file ($_FILES["file"]["tmp_name"],"upload/".$_FILES ["file"]["name"]);
   echo "Stored in: " . "upload/" . $_FILES ["file"]["name"];
   }
  }
 }
else
 {
 echo "Invalid file";
 }
?>
</body>
</html>
```

By using the global PHP $_FILES array you can upload files from a client computer to the remote server.

The first parameter is the form's input name and the second index can be either "name", "type", "size", "tmp_name" or "error". Like this:

$_FILES ["file"] ["name"] - the name of the uploaded file.
$_FILES ["file"] ["type"] - the type of the uploaded file.
$_FILES ["file"] ["size"] - the size in bytes of the uploaded file.
$_FILES ["file"] ["tmp_name"] - the name of the temporary copy of the file stored on the server.
$_FILES ["file"] ["error"] - the error code resulting from the file upload.

DKTE CSE

In this script we add some restrictions to the file upload. The user may only upload .gif or .jpeg files and the file size must be under 20 kb:

This is a very simple way of uploading files. For security reasons, you should add restrictions on what the user is allowed to upload.

The script above checks if the file already exists, if it does not, it copies the file to the specified folder.

Note: This example saves the file to a new folder called "upload"

Key concepts: $_FILES, file_exists, move_uploaded_file, file_exists

Algorithm:

- Create fileloader.html and put above shown html code in it and save it in /xampp/htdocs directory
- Create upload_file.php and write above code in it and save it in /xampp/htdocs directory.
- Open browser and type localhost/fileloader.html
- Browse the file and upload any *.gif,*.jpeg or *.pjpeg image with size less than 20000 bytes

DKTE CSE