File Uploads

File Uploads

Forms allow files to be uploaded by means of a “multi-part” HTTP POST transaction.
You can specify that you want to encode your POST using **multi-part** form data in your HTML by declaring a form something like this:

```
<form enctype="multipart/form-data" action="" method="post">
```

Note: that I've left the "action" attribute blank. By default, an HTML form will submit to the URI that it is served from.

File Handling

$_FILES is a superglobal variable in PHP that is used to handle file uploads from HTML forms.
When a user submits a form that includes a file input field, the data related to the uploaded file is made available in the $_FILES array.

The $_FILES array provides information about the uploaded file, including its name, type, size, and the temporary location on the server.

File Upload Form Example

File Uploads Example

```
<form action="upload.php" method="post" enctype="multipart/form-data">
   <label for="file">Select a file:</label>
   <input type="file" name="file" id="file" required>
   <br>
   <input type="submit" value="Upload File">
</form>
```

Maharah
Code Your Future

File Upload Handling Example

File Uploads Example (2)

```php
<?php

if (isset($_FILES['file'])) {
   $bytes = random_bytes(16);
   $uniqueFileName = bin2hex($bytes) . '.' . pathinfo($_FILES['file']['name'],
PATHINFO_EXTENSION);

   $targetDir = __DIR__ . '/uploads/';
   $targetFile = $targetDir . $uniqueFileName;

   if (move_uploaded_file($_FILES['file']['tmp_name'], $targetFile)) {
       echo 'File uploaded successfully!';
   } else {
       echo 'Error uploading file!';
   }
}
```

Exercise

04

1. Enhance the previous exercise (03) by allowing users to upload an image in addition to entering their name.

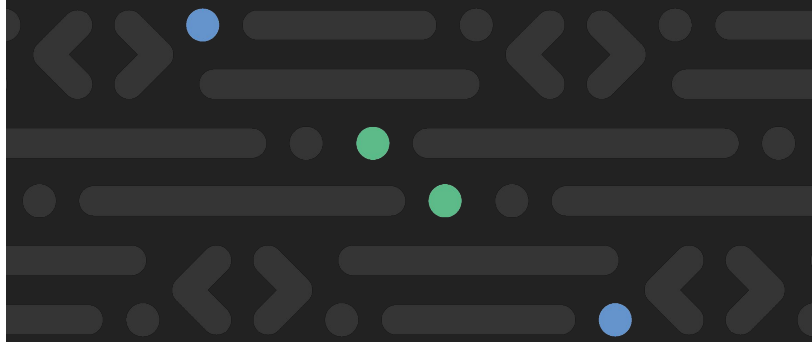
2. Modify the HTML form to include a file input field for uploading an image.

3. In the PHP script, extend the functionality to handle the uploaded image.

File Handling in PHP

Input-Output - I/O

In web development, the ability to work with files is fundamental. PHP equips developers with a robust set of file functions, allowing for seamless interaction with files and directories.



File Handling

PHP Files

PHP has several functions for creating, reading, uploading, and editing files.

There are two main groups of functions to deal with files: those that work with file resources, and those that work with a filename.

All the functions that deal with file resources begin with a single **f** letter and then have a verb describing their function. For example, fopen() opens a file resource.

Functions that work with the string name of a file all start with the word **file** and are followed by a verb descriptive of what they do. For example, file_get_contents() takes a string filename and returns the contents of that file.

Opening Files

open() function

The fopen() function in PHP is the gateway to file I/O operations. It allows us to open a file and returns a file pointer, which is essential for subsequent file operations.

You must pass two parameters to fopen():

- The name of the file in your file system
- The file **mode** that you want to open it with

```php
<?php

$file = fopen("example.txt", "r");
if ($file) {
   echo "File opened successfully!";
} else {
   echo "Unable to open the file.";
}
```

Opening Files

Modes for Opening Files

Modes determine the type of operations we can perform on the file. Common modes include:

- **r**: Open for reading only; place the file pointer at the beginning of the file.
- **r+**: Open for reading and writing; place the file pointer at the beginning of the file.
- **w**: Write-only. Truncates an existing file or creates a new file if it doesn't exist
- **a**: Append. Writes to the end of the file or creates a new file.

For more information about modes, visit:

https://www.php.net/manual/en/function.fopen.php

```php
<?php
$handle = fopen("example.txt", "r");
if ($handle) {
   echo "File opened successfully!";
} else {
   echo "Unable to open the file.";
}
```

Closing Files

fclose()

t is important to close files after performing operations to free up system resources.

fclose (resource $handle): bool

```php
<?php
$handle = fopen("example.txt", "r");
// Perform file operations

fclose($handle); // Close the file when done
```

Writing to Files

fwrite()

The **fwrite()** function in PHP is essential for writing data to files. It allows us to add or overwrite content in a file.

Syntax: **int fwrite (resource $handle , string $string [, int $length])**

```php
<?php
$handle = fopen("example.txt", "w");

// Writing data to the file
fwrite($handle, "Hello, PHP File Handling!");

fclose($handle);
```

Reading Files

fread()

`fread()` is a built-in PHP function used to read from a file or a URL.

Syntax: `fread($handle, $length);`

```php
<?php

$file = fopen("sample.txt", "r");
$content = fread($file, filesize("sample.txt"));
fclose($file);
```

Using file_* functions

file_get_contents():

`file_get_contents()` is a function that simplifies reading the entire content of a file into a string in a single function call.

Syntax: string file_get_contents (string $filename [, bool $use_include_path = FALSE [, resource $context [, int $offset = 0 [, int $maxlen]]]])

```php
<?php
// Reading entire content of a file into a string
$content = file_get_contents("example.txt");

// Displaying the content
echo "File Content: $content";
```

Using file_* functions

file_put_contents():

The **`file_put_contents()`** function is a convenient way to write data to a file. It appends content to the file without the need for explicit file opening and closing. This function is identical to calling fopen(), fwrite() and fclose() successively to write data to a file.

Syntax: int file_put_contents (string $filename , mixed $data [, int $flags = 0 [, resource $context]])

```php
<?php
file_put_contents("example.txt", "Appended Content");
```

Checking File Existence

file_exists():

The **file_exists()** function for validating whether a file exists before performing any file-related operations.

Syntax: bool file_exists (string $filename)

```php
<?php
$filename = "example.txt";

if (file_exists($filename)) {
   echo "The file $filename exists.";
} else {
   echo "The file $filename does not exist.";
}
```

Maharah
Code Your Future

Deleting Files

unlink():

`unlink()` is the function to remove or delete a file in PHP.

Syntax: bool unlink (string $filename [, resource $context])

```php
<?php
$filename = "example.txt";

if (unlink($filename)) {
   echo "File $filename has been deleted.";
} else {
   echo "Unable to delete $filename.";
}
```

Maharah
Code Your Future

Renaming Files

rename():

Use **rename()** to change the name or move a file to a different location.

Syntax: bool rename (string $oldname , string $newname [, resource $context])

```php
<?php
$oldname = "old_file.txt";
$newname = "new_file.txt";

if (rename($oldname, $newname)) {
   echo "File renamed successfully.";
} else {
   echo "Unable to rename the file.";
}
```

Exercise

10 Minutes!

05

- **Create an HTML page containing a comment input field.**
- **Create a PHP script to handle the submitted form and comment input and save it to a file, and then retrieve the comment from the file.**
- **If the file exists, it only displays the content and does not allow the user to add new or update the comment.**

Q&A

This project is funded by
the European Union

Maharah
Code Your Future

LibyanSpider