HYPERTEXT PREPROCESSOR(PHP)

UNIT-4

What is File Handling in PHP?

PHP allows you to work with files stored on the server. File handling involves tasks like reading data from files, writing data to files, and managing file resources.

Opening a File:

When you want to work with a file in PHP, you need to open it first. You do this using the fopen() function.

Think of fopen() as the key to unlock the door to your file.

Imagine you have a file named "data.txt" that you want to read. You would use fopen() like this:

file = fopen("data.txt", "r");

Here, "data.txt" is the name of the file, and "r" tells PHP that you want to open the file for reading (r stands for "read").

Understanding Modes for Opening Files:

- The second parameter in fopen() is the mode, which tells PHP how you want to interact with the file.
- Think of modes as different ways to use your key to unlock the door.

Here are some common modes:

a. r: Opens the file for reading only. It starts at the beginning of the file.

b. w: Opens the file for writing only. If the file doesn't exist, it creates it. If it does, it truncates it to zero length.

c. a: Opens the file for writing only. It starts at the end of the file if the file exists. If the file doesn't exist, it creates it.

d. For example, if you want to write to a file named "log.txt", you'd use:

$file = fopen("log.txt", "w");

Closing a File:

a. Once you're done with a file, it's important to close it properly using the fclose() function.

b. Think of fclose() as locking the door again after you've finished with the room.

c. If you forget to close a file, it's like leaving a door unlocked, which can cause problems.

To close a file, you simply use fclose() with the file pointer you got from fopen():

fclose($file);

Why Closing a File Matters:

a. Closing a file releases the resources associated with it, like memory, making your program more efficient.

b. It also ensures that any data you've written to the file is saved properly.

c. Forgetting to close a file can lead to issues like memory leaks or problems accessing the file later.

Copying a File:

a. When you need to make a copy of a file in PHP, you use the copy() function.

b. It's like making a photocopy of a document.

Example:

$sourceFile = "original.txt";

$destinationFile = "copy_of_original.txt";

copy($sourceFile, $destinationFile);

In this example, "original.txt" is the file you want to copy, and "copy_of_original.txt" is the new file created as a copy.

Renaming a File:

a. To rename a file, PHP offers the rename() function.

b. Think of it as giving a new name to a file.

Example:

$oldName = "old_name.txt";

$newName = "new_name.txt";

rename($oldName, $newName);

Here, "old_name.txt" is the current name of the file, and "new_name.txt" is the desired new name.

Deleting a File:

a. When a file is no longer needed, you can delete it using unlink().

b. It's like throwing away a piece of paper you no longer require.

Example:

$fileToDelete = "file_to_delete.txt";

unlink($fileToDelete);

Here, "file_to_delete.txt" is the file you want to remove from your system.

Error Handling:

a. It's important to anticipate and handle potential errors.

b. For instance, if a file doesn't exist when you try to copy or rename it, PHP will throw an error.

c. You can use conditional statements or try-catch blocks to manage these errors gracefully.

Precautions:

a. Always ensure you have appropriate permissions to perform file operations.

b. Double-check file paths and names to avoid unintended actions.

c. Deleting files is irreversible, so be certain before executing the delete operation.

Reading from a File:

When you want to retrieve data from a file, you use functions like fread() or fgets().

It's like opening a book and reading its contents.

Example using fread():

$file = fopen("data.txt", "r");

$content = fread($file, filesize("data.txt"));

fclose($file);

echo $content;

Here, "data.txt" is opened in read mode ("r"), and fread() reads the entire content of the file into a variable.

Writing to a File:

If you want to add or overwrite content in a file, you use functions like fwrite() or file_put_contents().

It's like writing notes in a notebook.

Example using fwrite():

$file = fopen("data.txt", "w");

fwrite($file, "Hello, World!");

fclose($file);

Here, "data.txt" is opened in write mode ("w"), and fwrite() writes the string "Hello, World!" to the file.

Appending to a File:

If you want to add content to the end of an existing file without overwriting its contents, you use append mode ("a").

Example:

$file = fopen("data.txt", "a");

fwrite($file, "Appending more content!");

fclose($file);

Here, "Appending more content!" is added to the end of "data.txt".

Using file_put_contents():

This function simplifies writing data to a file by handling the file opening, writing, and closing automatically.

Example:

$content = "Content to write to the file.";

file_put_contents("data.txt", $content);

This writes the content directly to "data.txt" without needing to explicitly open or close the file.

Error Handling and Checking:

a. Always check if file operations are successful and handle errors gracefully.

b. Use functions like file_exists() to check if a file exists before reading from or writing to it.

Connecting to a MySQL Database:

To connect to a MySQL database, you use the mysqli_connect() function.

You need to provide the hostname, username, password, and database name.

Example:

```
$hostname = "localhost";

$username = "root";

$password = "your_password";

$database = "your_database";

$conn = mysqli_connect($hostname, $username, $password, $database);

if (!$conn) {

    die("Connection failed: " . mysqli_connect_error());

}
```

Performing Basic Database Operations:

Once connected, you can perform basic operations like insert, delete, update, and select.

Insert Data:

```
$sql = "INSERT INTO table_name (column1, column2, ...) VALUES ('value1', 'value2', ...)";

if (mysqli_query($conn, $sql)) {

    echo "Record inserted successfully";

} else {

    echo "Error: " . $sql . "<br>" . mysqli_error($conn);

}
```

Delete Data:

```
$sql = "DELETE FROM table_name WHERE condition";

if (mysqli_query($conn, $sql)) {

    echo "Record deleted successfully";

} else {

    echo "Error: " . $sql . "<br>" . mysqli_error($conn);

}
```

Update Data:

```
$sql = "UPDATE table_name SET column1='value1', column2='value2' WHERE condition";

if (mysqli_query($conn, $sql)) {

   echo "Record updated successfully";

} else {

   echo "Error: " . $sql . "<br>" . mysqli_error($conn);

}
```

Select Data:

```
$sql = "SELECT * FROM table_name";

$result = mysqli_query($conn, $sql);

if (mysqli_num_rows($result) > 0) {

   while ($row = mysqli_fetch_assoc($result)) {

      echo "Column1: " . $row["column1"]. " - Column2: " . $row["column2"]. "<br>";

   }

} else {

   echo "0 results";

}
```

Query Handling:

a. Construct SQL queries as per your requirements, using proper syntax.

b. Execute queries using mysqli_query() function.

c. Handle query results using mysqli_fetch_assoc() or other appropriate functions.

d. Always sanitize user input to prevent SQL injection attacks.

MOCK QUESTIONS

Explore the fundamentals of file handling in PHP, examining the process of interacting with files on the server filesystem. Discuss the various file operations supported by PHP, including opening, closing, reading, writing, copying, renaming, and deleting files. Analyze different file handling modes and their implications for file access and manipulation. Provide detailed examples demonstrating the use of file handling functions in practical PHP applications.

Investigate the intricacies of database handling in PHP, focusing on the integration of PHP with MySQL databases for data storage and retrieval. Discuss the process of establishing a connection to a MySQL database using PHP, and explore the mechanisms for performing basic database operations such as insert, delete, update, and select queries. Analyze best practices for database security, transaction management, and error handling in PHP applications.