File Handling in PHP

1. File Read, Write, and Close

Overview

PHP provides functions to read from and write to files. Files must be opened before performing operations and closed afterward to free up resources.

Key Functions

- `fopen()` : Opens a file.
- `fread()` : Reads from a file.
- `fwrite()` : Writes to a file.
- `fclose()` : Closes a file.
- `file_get_contents()` : Reads the entire file into a string.
- `file_put_contents()` : Writes data to a file.

Practical Examples

Example 1: Reading a File

```
<?php
$file = fopen("example.txt", "r"); // Open file in read mode
if ($file) {
    echo fread($file, filesize("example.txt")); // Read the entire file
    fclose($file); // Close the file
} else {
    echo "Unable to open file.";
}
?>
```

Example 2: Writing to a File

```
<?php
$file = fopen("example.txt", "w"); // Open file in write mode
if ($file) {
    fwrite($file, "Hello, World!"); // Write data to the file
    fclose($file); // Close the file
    echo "Data written to file.";
} else {
    echo "Unable to open file.";
```

```
}
?>
```

Example 3: Using `file_get_contents()` and `file_put_contents()`

```
<?php
// Read the entire file
$content = file_get_contents("example.txt");
echo $content;

// Write data to a file
file_put_contents("example.txt", "New content added.");
?>
```

2. File Upload

Overview

PHP allows users to upload files to the server. The uploaded file is stored in a temporary directory and can be moved to a permanent location.

Key Concepts

- `$_FILES` **Superglobal**: Contains information about the uploaded file.
- `move_uploaded_file()` : Moves the uploaded file to a new location.

Practical Example

Example: File Upload

```
<?php
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    if (isset($_FILES['file']) && $_FILES['file']['error'] == UPLOAD_ERR_OK) {
        $tmp_name = $_FILES['file']['tmp_name'];
        $name = basename($_FILES['file']['name']);
        move_uploaded_file($tmp_name, "uploads/$name");
        echo "File uploaded successfully!";
    } else {
        echo "Error uploading file.";
    }
}
?>
<form method="POST" enctype="multipart/form-data">
    <input type="file" name="file">
```

```
    <input type="submit" value="Upload">
</form>
```

3. Parsing CSV Files

Overview

CSV (Comma-Separated Values) files are commonly used to store tabular data. PHP provides functions to read and parse CSV files.

Key Functions

- `fgetcsv()` : Reads a line from a CSV file and parses it into an array.

Practical Example

Example: Parsing a CSV File

```
<?php
$file = fopen("data.csv", "r");
if ($file) {
    while (($data = fgetcsv($file)) !== FALSE) {
        print_r($data); // Display each row as an array
    }
    fclose($file);
} else {
    echo "Unable to open file.";
}
?>
```

4. Parsing JSON Files

Overview

JSON (JavaScript Object Notation) is a lightweight data interchange format. PHP provides functions to encode and decode JSON data.

Key Functions

- `json_decode()` : Decodes a JSON string into a PHP variable.
- `json_encode()` : Encodes a PHP variable into a JSON string.

Practical Examples

Example 1: Decoding JSON

```
<?php
$json = '{"name": "John", "age": 30, "city": "New York"}';
$data = json_decode($json, true); // Decode JSON into an associative array
print_r($data);
?>
```

Example 2: Encoding JSON

```
<?php
$data = array("name" => "John", "age" => 30, "city" => "New York");
$json = json_encode($data); // Encode array into JSON
echo $json;
?>
```

Summary of Key Points

FeatureDescriptionFile Read/Write Use `fopen()`, `fread()`, `fwrite()`, and `fclose()` for file operations.**File Upload**Use `$_FILES` and `move_uploaded_file()` to handle file uploads.**Parsing CSV Files**Use `fgetcsv()` to read and parse CSV files.**Parsing JSON Files**Use `json_decode()` and `json_encode()` to work with JSON data.

Practical Questions

1. Write a PHP script to read the contents of a text file and display them.
2. Create a form to upload a file and save it to the server.
3. Parse a CSV file and display its contents in an HTML table.
4. Decode a JSON string and display its data in a readable format.
5. Encode a PHP array into a JSON string and save it to a file.