File Handling

What it is: File handling refers to a set of functions in PHP that allow you to interact with files on the server's disk. This includes operations like creating, reading, writing, appending, and deleting files.

Why it's important: File handling is essential for many web development tasks. examples:

- Uploading files (e.g., user avatars)
- Saving user data (e.g., login credentials)
- Reading configuration files
- Creating log files

Key Functions:

- `fopen()`: Opens a file for specific operations. Takes the filename and a mode as arguments (e.g., "r" for read, "w" for write, "a" for append).
- `fread()`: Reads a portion of a file into a variable. Takes the file pointer and number of bytes to read as arguments.
- `fwrite()`: Writes data to a file. Takes the file pointer and the data to write as arguments.
- `fclose()`: Closes a file after operations are complete. Always recommended to avoid resource leaks.
- Additional functions: `filesize()`, `fgets()`, `feof()`, `file_exists()`, `unlink()` (for deletion) - explore the PHP manual for details on their functionalities.

Example - Reading a File:

```php
<?php
$firstLine = fgets(fopen("myfile.txt", "r"));

if ($firstLine) {
  echo "First line: $firstLine";
} else {
  echo "Error: Could not open file or file is empty";
}
?>
```

Opening a File

- Use the `fopen()` function.
- Takes two arguments:
  - `$filename`: Path to the file (including filename).
  - `$mode`: Opening mode (read, write, append etc.).
- Returns a file handle on success or `FALSE` on error.
- Common modes:
  - `'r'`: Read only (throws error if file doesn't exist).
  - `'w'`: Write only (creates new file or overwrites existing).
  - `'a'`: Append only (creates new file or writes to the end).
  - `'r+'`: Read and write (existing file).
  - `'w+'`: Read and write (creates new file or overwrites).
- Example:

```php
$myfile = fopen("myfile.txt", "r") or die("Unable to open file!");
// Check if open succeeded before proceeding
```

Closing a File

- Use the `fclose()` function.
- Takes one argument: the file handle returned by `fopen()`.
- Releases resources associated with the file.
- Important to close files after use to avoid errors and improve performance.
- Example:

```php
// After working with the file...
fclose($myfile);
```

File Manipulation in PHP

Copying Files:

- Use the `copy()` function.
- Takes two arguments: source file path and destination file path.
- Returns `true` on success, `false` on failure.
- **Example:**

```php
$original_file = "image.jpg";
$copy_path = "uploads/image_copy.jpg";

if (copy($original_file, $copy_path)) {
  echo "File copied successfully!";
} else {
  echo "Failed to copy file.";
}
```

Renaming Files:

- Use the `rename()` function.
- Takes two arguments: old file path and new file path.
- Overwrites existing file with the same name if it exists.

Example:

```php
$old_name = "report.txt";
$new_name = "report_modified.txt";

if (rename($old_name, $new_name)) {
  echo "File renamed successfully!";
} else {
  echo "Failed to rename file.";
}
```

Deleting Files:

- Use the `unlink()` function.

- Takes one argument: the file path to delete.
- Returns `true` on success, `false` on failure.

Example:

```php
$file_to_delete = "temporary_file.dat";

if (unlink($file_to_delete)) {
  echo "File deleted successfully!";
} else {
  echo "Failed to delete file.";
}
```

Reading Files:

Several functions are available depending on your needs:

- `fread($filehandle, number_of_bytes)`: Reads a specific number of bytes from file.
- `fgets($filehandle)`: Reads a single line from the file.
- `file_get_contents($filename)`: Reads the entire file content into a string (better for smaller files).

Example (reading entire file):

```php
$content = file_get_contents("myfile.txt");
echo $content;
```

Writing Files:

- Use `fwrite($filehandle, data_to_write)` to write data to the file.

```php
$data = "This is some text to write to the file.";
fwrite($myfile, $data);
```

Database Handling

What it is:

Database handling in PHP refers to the capability of PHP scripts to interact with databases. This allows web applications to store, retrieve, update, and delete data dynamically.

Why it's important:

- Enables creation of dynamic and data-driven web applications.
- Stores user information, manages content, implements user authentication, and handles complex data relationships.

How it works:

- **Connection:** PHP establishes a connection to the database server using extensions like MySQLi or PDO. These extensions provide functions for connecting, executing queries, and handling results.
- **Interaction:** PHP scripts use SQL statements to interact with the database. SQL (Structured Query Language) is a standardized language for managing relational databases.
- **CRUD operations:** The core functionalities involve CRUD (Create, Read, Update, Delete) operations on the database tables.

Connecting to MySQL Database

Connection Details: You'll need:

- **Hostname:** Server where MySQL is running (often "localhost").
- **Username:** Authorized user for database access.
- **Password:** Username's corresponding password.
- **Database Name:** Specific database you want to connect to.

Methods: PHP offers two main methods for connecting to MySQL databases:

1. MySQLi (Improved MySQL): Recommended approach, offers object-oriented and procedural styles.

- Define connection details (hostname, username, password, database name).
- Use `mysqli_connect()` to establish connection.
- Check for connection errors with `mysqli_connect_error()`.
- Execute queries and interact with the database.

- Close the connection using `mysqli_close()`.

2. PDO (PHP Data Objects): General-purpose interface, works with multiple database types besides MySQL. (Not Important)

MySQLi Example (Procedural):

```php
<?php
$servername = "localhost";
$username = "your_username";
$password = "your_password";
$dbname = "my_database";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);

// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}

echo "Connected successfully";

mysqli_close($conn);
?>
```

Performing Basic Database Operations (CRUD)

CRUD stands for **Create**, **Read**, **Update**, and **Delete**, which are the fundamental operations for interacting with data in a database. Here's a breakdown for performing these in PHP:

- Establish a connection to the MySQL database using libraries like `mysqli` or `PDO`.

1. Insert (Create)

- Use the `INSERT` statement with the table name and column names.
- Bind values using prepared statements for security against SQL injection.
- **Example:**

```php
$sql = "INSERT INTO users (name, email) VALUES (?, ?)";
$stmt = mysqli_prepare($conn, $sql);

mysqli_stmt_bind_param($stmt, "ss", $name, $email);

$name = "John Doe";
$email = "john.doe@example.com";

mysqli_stmt_execute($stmt);

mysqli_stmt_close($stmt);
```

2. Select (Read):

- Use the `SELECT` statement to retrieve data from the table.
- Specify columns and filter data with `WHERE` clause.
- **Example:**

```php
$sql = "SELECT * FROM users WHERE id = ?";
$stmt = mysqli_prepare($conn, $sql);
mysqli_stmt_bind_param($stmt, "i", $id);
$id = 1; // Replace with desired ID
mysqli_stmt_execute($stmt);
$result = mysqli_stmt_get_result($stmt);
// Process results (check for rows and loop)
mysqli_stmt_close($stmt);
```

3. Update (Modify):

- Use the `UPDATE` statement to modify existing data in the table.
- Specify columns to update and filter with `WHERE` clause.
- **Example:**

```php
$sql = "UPDATE users SET email = ? WHERE id = ?";
$stmt = mysqli_prepare($conn, $sql);
mysqli_stmt_bind_param($stmt, "si", $email, $id);
$email = "updated@example.com";
$id = 1;
mysqli_stmt_execute($stmt);
mysqli_stmt_close($stmt);
```

4. Delete (Remove):

- Use the `DELETE` statement to remove data from the table.
- Filter data with `WHERE` clause for specific deletion.
- **Example:**

```php
$sql = "DELETE FROM users WHERE id = ?";
$stmt = mysqli_prepare($conn, $sql);
mysqli_stmt_bind_param($stmt, "i", $id);
$id = 1;
mysqli_stmt_execute($stmt);
mysqli_stmt_close($stmt);
```

Query Handling in PHP

Connecting to Database:

- Use `mysqli` extension for secure connections.

Executing Queries:

- Use `mysqli_query(connection, query_string)` to execute queries.
- Function returns `mysqli_result` object for SELECT queries, `TRUE` for others.
- Example (SELECT):

```php
$mysqli = mysqli_connect("localhost", "user", "password",
"mydatabase");
$result = mysqli_query($mysqli, "SELECT * FROM users");

if ($result) {
    // Process results here
} else {
    echo "Error: " . mysqli_error($mysqli);
}
```

Handling Results:

- Use functions like `mysqli_num_rows($result)` to get the number of rows.
- Fetch results row by row using:
    - `mysqli_fetch_assoc($result)` (associative array)
    - `mysqli_fetch_array($result)` (numeric array or both)
    - `mysqli_fetch_all($result)` (all rows into an array)
- **Example (Fetching data):**

```php
while ($row = mysqli_fetch_assoc($result)) {
    echo "ID: " . $row["id"] . ", Name: " . $row["name"] . "<br>";
}
```

Important Functions:

- `mysqli_free_result($result)`: Frees memory associated with the result set.
- `mysqli_close($connection)`: Closes the connection to the database.