Web Framework and Services Class Assignment Q&A

1. What is PHP? Explain Features of PHP.

PHP (Hypertext Preprocessor) is a widely-used, open-source server-side scripting language especially suited for web development. It can be embedded directly into HTML, which makes it easy to create dynamic web pages. When a user requests a PHP page, the server executes the PHP code and sends the resulting HTML to the user's browser.

Key Features of PHP:

- **Simplicity and Ease of Use**: PHP has a straightforward syntax that is similar to C, making it easy for new developers to learn and use.
- **Cross-Platform Compatibility**: It is platform-independent, meaning it can run on various operating systems like Windows, Linux, and macOS.
- **Database Support**: PHP has excellent built-in support for a wide range of databases, including MySQL, PostgreSQL, Oracle, and others, making it a great choice for database-driven websites.
- **Open Source**: PHP is free to download and use, which has contributed to its large and active community that provides extensive support and documentation.
- **Performance**: PHP scripts generally execute faster than scripts written in other languages like ASP and JSP. With modern versions (PHP 8+), performance has been significantly improved with features like the JIT (Just-In-Time) compiler.

---

2. Explain Directory Handling functions in PHP

Directory handling functions in PHP are used to interact with the file system on the server. They allow you to perform operations such as reading the contents of a directory, creating new directories, and deleting them.

Common Directory Handling Functions:

- **opendir($path)**: Opens a directory handle to be used in other directory functions.
- **readdir($dir_handle)**: Reads an entry from the directory handle. It returns the name of the next file or directory inside the directory.
- **closedir($dir_handle)**: Closes the specified directory handle.
- **mkdir($pathname)**: Creates a new directory at the specified path.
- **rmdir($dirname)**: Removes an empty directory.
- **scandir($directory)**: Returns an array of files and directories from the specified path.

---

3. Explain different types of operators in PHP.

PHP supports several types of operators to perform various operations:

- **Arithmetic Operators**: Used to perform common mathematical operations. Examples: + (addition), - (subtraction), * (multiplication), / (division), % (modulus).

- **Assignment Operators**: Used to assign values to variables. The basic assignment operator is =. Combined operators like +=, -=, *= perform an operation and then assign the value.
- **Comparison Operators**: Used to compare two values. Examples: == (equal), != (not equal), > (greater than), < (less than), >= (greater than or equal to).
- **Logical Operators**: Used to combine conditional statements. Examples: && (and), || (or), ! (not).
- **Increment/Decrement Operators**: Used to increment or decrement a variable's value. Examples: ++$x (pre-increment), $x++ (post-increment), --$x (pre-decrement).
- **String Operators**: There are two operators for string manipulation: . (concatenation) and .= (concatenating assignment).

---

4. Discuss different condition statements available in PHP.

Conditional statements are used to perform different actions based on different conditions.

- **if Statement**: Executes a block of code if a specified condition is true.

```
if ($condition) {
   // code to execute if condition is true
}
```

- **if-else Statement**: Executes one block of code if a condition is true and another block if it is false.

```
if ($condition) {
   // code if true
} else {
   // code if false
}
```

- **if-elseif-else Statement**: Used to check multiple conditions. It executes the block of code corresponding to the first true condition it finds.

```
if ($condition1) {
   // code for condition1
} elseif ($condition2) {
   // code for condition2
} else {
   // code if all conditions are false
}
```

- **switch Statement**: Selects one of many blocks of code to be executed. It is often used as an alternative to a long if-elseif-else statement.

```
switch ($variable) {
   case value1:
      // code to execute
      break;
   case value2:
      // code to execute
      break;
```

```
    default:
        // default code
}
```

---

5. Explain steps of Database connectivity of PHP and MySql

Connecting PHP to a MySQL database typically involves the following steps using the MySQLi or PDO extension:

1. **Establish a Connection**: Use the mysqli_connect() function or create a new PDO object. You need to provide the database server name (hostname), username, password, and database name.
2. **Check the Connection**: After attempting to connect, it's crucial to check if the connection was successful. If it fails, the script should stop and display an error message.
3. **Execute SQL Queries**: Once connected, you can execute SQL queries using functions like mysqli_query() or PDO's prepare() and execute() methods. This is used for operations like SELECT, INSERT, UPDATE, and DELETE.
4. **Fetch Results (for SELECT queries)**: If you run a SELECT query, you'll need to fetch the results from the database. Functions like mysqli_fetch_assoc() or PDO's fetch() are used to retrieve one row at a time.
5. **Close the Connection**: After you have finished all database operations, it is a good practice to close the connection using mysqli_close() or by setting the PDO object to null. This frees up server resources.

---

6. Explain Multidimensional Array in details.

A **multidimensional array** is an array that contains one or more other arrays as its elements. It's like a table or a matrix, where each element can be accessed using multiple indices. In PHP, you can create arrays of arrays to represent data in a two-dimensional (or higher) structure.

This is useful for storing grouped data, such as a list of students where each student has multiple properties like name, age, and grade.

Example:

Here is a two-dimensional array that stores information about students:

```
$students = array(
    array("name" => "Alice", "age" => 20, "grade" => "A"),
    array("name" => "Bob", "age" => 22, "grade" => "B"),
    array("name" => "Charlie", "age" => 21, "grade" => "A")
);

// To access Bob's age:
echo $students[1]["age"]; // Outputs: 22
```

Web Framework and Services Class Assignment Q&A

7. What is meaning of LAMP and WAMP.

LAMP and **WAMP** are acronyms for software stacks used for web development. They bundle all the necessary components to run a dynamic web server.

- **LAMP**: Stands for **L**inux, **A**pache, **M**ySQL, and **P**HP. This is the standard software stack for building and deploying web applications on a Linux operating system.
- **WAMP**: Stands for **W**indows, **A**pache, **M**ySQL, and **P**HP. This is the equivalent stack for developers working on a Windows operating system.

---

8. What is Session? Exaplain in details with example.

A **session** in PHP is a way to store information about a user across multiple page requests. Unlike cookies, session data is stored on the server, and only a unique session ID is stored on the client's computer in a cookie. This makes sessions a more secure way to store sensitive information.

Sessions are commonly used for user authentication, shopping carts, and preserving user-specific settings.

Example:

To use a session, you must first start it using session_start().

page1.php

```
<?php
// Start the session
session_start();

// Store session data
$_SESSION["username"] = "JohnDoe";
$_SESSION["favColor"] = "Green";

echo "Session variables are set.";
?>
```

page2.php

```
PHP
<?php
// Start the session
session_start();

// Access session data
echo "Welcome, " . $_SESSION["username"] . "<br>";
echo "Your favorite color is " . $_SESSION["favColor"] . ".";
?>
```

When you navigate from page1.php to page2.php, the user's information is preserved and can be accessed.

Web Framework and Services Class Assignment Q&A

9. Explain move_uploaded_file() function with example.

The move_uploaded_file() function in PHP is a built-in function used to move an uploaded file from its temporary server directory to a new, permanent location. This function is the recommended way to handle file uploads because it performs a crucial security check: it verifies that the file specified was actually uploaded via a POST request, which helps prevent potential attacks.

Syntax:

```
bool move_uploaded_file(string $from, string $to)
```

- $from: The temporary filename of the uploaded file (accessible via $_FILES['input_name']['tmp_name']).
- $to: The new destination path for the file.

Example:

This code processes a file upload from an HTML form.

```
<?php
if (isset($_FILES['myFile'])) {
    $target_dir = "uploads/";
    $target_file = $target_dir . basename($_FILES["myFile"]["name"]);
    $tmp_file = $_FILES["myFile"]["tmp_name"];

    // Move the file from the temporary directory to the "uploads" folder
    if (move_uploaded_file($tmp_file, $target_file)) {
        echo "The file " . htmlspecialchars(basename($_FILES["myFile"]["name"])) . " has been uploaded.";
    } else {
        echo "Sorry, there was an error uploading your file.";
    }
}
?>
```

---

10. Write a code for the following

A. Insert a record in table of your choice

Assuming a table named users with columns name and email:

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
```

```
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

// SQL to insert a record
$sql = "INSERT INTO users (name, email) VALUES ('John Doe',
'john.doe@example.com')";

if ($conn->query($sql) === TRUE) {
  echo "New record created successfully";
} else {
  echo "Error: " . $sql . "<br>" . $conn->error;
}

$conn->close();
?>
```

B. Delete record according ID passed in textbox

Assuming an HTML form with a textbox named user_id and a users table with a primary key column id:

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "myDB";

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

// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

// Check if the form was submitted and ID is set
if (isset($_POST['user_id']) && !empty($_POST['user_id'])) {
    // Sanitize the input
    $id_to_delete = intval($_POST['user_id']);

    // Prepare and bind
    $stmt = $conn->prepare("DELETE FROM users WHERE id = ?");
    $stmt->bind_param("i", $id_to_delete);

    // Execute the statement
    if ($stmt->execute()) {
      echo "Record with ID " . $id_to_delete . " deleted successfully.";
    } else {
      echo "Error deleting record: " . $conn->error;
    }

    $stmt->close();
}

$conn->close();
?>

<form action="" method="post">
  Enter ID to Delete: <input type="text" name="user_id">
  <input type="submit" value="Delete Record">
</form>
```