`

AMRITA VISHWA VIDYAPEETHAM

AHEAD Online

Web Application Development

Guruprakash J
Department of Computer Science
Amrita Vishwa Vidyapeetham

Objective

Database Programming: Inputting and Outputting Data from MySQL using PHP

- Introduction to MySQL database and its integration with PHP
- Establishing a database connection in PHP
- Executing SQL queries using PHP and MySQL
- Inserting, updating, retrieving, and deleting data from MySQL database using PHP

Introduction

- Welcome students to today's session on integrating MySQL databases with PHP.
- In this session, we'll explore how PHP can be used to interact with MySQL databases.
- We'll learn how to establish a connection, execute SQL queries, and manage data using PHP.

MySQL Integration

- MySQL is a popular open-source relational database management system.
- PHP provides functions to interact with MySQL databases, making it an excellent choice for web applications.
- Integration involves establishing a connection between PHP and MySQL.

Establishing Connection

- mysqli and PDO are two common PHP extensions for database connection.
- Establishing a connection requires the database server name, username, password, and database name.
- Successful connection provides a link to execute queries.

Executing Queries

- SQL queries are executed using mysqli_query() or PDO::query() functions.
- SELECT retrieves data, while INSERT, UPDATE, and DELETE modify data.
- These functions return results or error messages.

Display.php

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

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

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

//Retrieve the comments from the database
$sql = "SELECT * FROM comments";
$result = $conn->query($sql);
if($result->num_rows > 0)
{
    // Output data of each row
    while($row = $result->fetch_assoc())
    {
        echo "Name: " . $row["name"]. " - Comment: " .
$row["comments"]. "<br>";
    }
}
else
{
    echo "No commnets yet ";
}

// Close connection
$conn->close();

?>
```

Index.php

```
<body>
    <form action="process.php" method="post">
        <lable for="name">Name:</lable>
        <input type="text" name="name" id="name" placeholder="Enter your name" required><br>
        <label for="comment">Comment:</label><br>
        <textarea name="comment" id="comment" cols="30" rows="10" placeholder="Enter your
comment" required></textarea><br>
        <input type="submit" value="Submit">
    </form>

    <?php include 'display.php'; ?>

</body>
```

Process.php

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

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

// Check connection
if ($conn->connect_error)
{
    die("Connection failed: " . $conn->connect_error);
}
//Retrieve the data
$name = $_POST['name'];
$comment = $_POST['comment'];
```

```
// Insert data into database
$sql = "INSERT INTO comments (name, comments) VALUES
('".$_POST["name"]."','".$_POST["comment"]."')";
if ($conn->query($sql) === TRUE)
{
    echo "New record created successfully";
}
else
{
    echo "Error: " . $sql . "<br>" . $conn->error;
}

// Close connection
$conn->close();
?>
```

Introduction

- Welcome to today's session on state management in web applications.
- In this session, we'll delve into the concept of state management and focus specifically on cookies as a means to preserve state.

Introduction to Cookies

- Cookies are small pieces of data stored on a user's browser.
- They are used to store user-related information for a specific website.
- Cookies play a vital role in maintaining state between different requests and sessions.

Setting and Retrieving Cookies

- PHP provides functions like setcookie() to create and send cookies to the user's browser.
- The $_COOKIE superglobal array is used to retrieve cookies.
- Cookies can store various types of data, such as strings, numbers, and timestamps.

Managing Expiration and Security

- Cookies can have an expiration time, after which they are automatically deleted.
- Secure and HttpOnly flags can enhance cookie security.
- It's important to be mindful of the information stored in cookies due to potential security risks.

Cookies

- A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.

- Create Cookie with PHP
  - A cookoe is created with the setcookie() function.
  - Synatx:

    setcookie(name, value, expire, path, domain, secure, httponly);

Introduction to Session Management

- Sessions are a way to store and manage user data across multiple pages during a single visit.
- In PHP, sessions allow the preservation of data between different requests.
- This is particularly useful for maintaining user authentication and personalization.

Working with Session Variables

- Session variables are used to store and retrieve data throughout a user's session.
- These variables are stored on the server but linked to a user via a unique session ID.
- PHP provides a superglobal array named $_SESSION for working with session variables.

Storing and Retrieving Session Data

- To store data, assign values to $_SESSION keys: $_SESSION['key'] = 'value';.
- To retrieve data, access the stored value using the same key: $value = $_SESSION['key'];.

Session Security and Best Practices

- Session security is crucial for preventing unauthorized access to user data.
- Always start sessions with session_start() at the beginning of your script.
- Avoid storing sensitive data in session variables.
- Implement session timeout to invalidate sessions after a period of inactivity.