ICT726 Web Development

Tutorial 8
Database and State Management

Overview

- Use PHP database API to connect with the database
- Writing codes to write perform insert, update and select statements
- Using POST and GET to process FORM data
- Create a Rest API
- Understand uses of cookies and sessions

Guidelines

Start by putting everything into a folder named "ict726_tutorial8".

Note: Since absolute links will only work on your computer, we recommend checking your website to make sure all the links work and use the same structure on your home computer.

Recommended Reading:

https://www.php.net/manual/en/book.mysql.php
https://www.tutorialrepublic.com/php-tutorial/php-mysql-introduction.php
https://www.w3schools.com/php/php_mysql_intro.asp

Introduction

In this lab you will need to connect with the MySQL database and run the basic data manipulation queries. Before writing the code please make sure that you have XAMPP is installed on your machines. Start the Apache and MySQL instance in the XAMPP control panel. Make sure to put all your script **.php** files inside C:\xampp\htdocs\Lab8.

Now please find the following parameters from your installation of MySQL

```
$servername = "localhost";
$username = "root";
$password = "";
```

Your parameter values will be different. Once you know your parameter values then you can start with the exercise.

Exercise 1:

Connect to the database using my MySQLi and PDO. You can use the following code to connect to the database

```
<?php
 $servername = "localhost";
 $username = “root";
 $password = "";
// Create connection

$conn = new mysqli($servername, $username, $password);

// Check connection

if ($conn->connect_error) {

  die("Connection failed: " . $conn->connect_error);

}
echo "Connected successfully";

?>
```

Exercise 2:

Create a new database called **“myDb”** using PHP code and create a table named "MyGuests", with five columns: "id", "firstname", "lastname", "email" and "reg_date": Your table must have the following schema

```
CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP)
```

Exercise 3:

Write a PHP code to insert the values into the table created in "*Exercise 2*". Also write the code to select the store values from the table and display. Write the code using both MySQLi and PDO.

Exercise 4:

Create a database "**Office**" using phpMyAdmin. Add a table called employee in the database with five fields (id, firstname, lastname, address and position). The id filed is a primary key and auto increment. Use the following code and explain what it is doing and then use it to demonstrate that it is working

```
<html>
<body>
<?php
if ($submit) {
//Open MYSQL server connection
    $db = mysql_connect("localhost", "root","");

//Select the database using MYSQL server connection
    mysqlI_select_db("mydb",$db);

/*Write insert query and assign the query in $sql Variable*/
      $sql="INSERT INTO employees (first,last,address,position) VALUES ('$first', '$last',
'$address', '$position')";

//Execute the query
        $result = mysql_query($sql);
       echo "Thank you! Information entered.";}
      else{
    //display form?>

<form method="post" action="<?php echo $PHP_SELF?>">
First name:<input type="Text" name="first"><br>
Last name:<input type="Text" name="last"><br>
Address:<input type="Text" name="address"><br>
Position:<input type="Text" name="position"><br>
<input type="Submit" name="submit" value="Enterinformation"></form>
<?php}// end if?>
</body></html>
```

Exercise 5:

Build a simple PHP application to implement all CURD (Create, Update, Read and Delete) operations for the following employee table.

```
Employees ( EmpID, EmpName, EmpAddress, Salary, Hiredate)
```

You need to implement your code using PDO API. For reference, you can use a similar application on the following link
https://www.tutorialrepublic.com/php-tutorial/php-mysql-crud-application.php

Exercise 6:

Read the articles related to cookies and sessions on the following links.

1. https://www.tutorialrepublic.com/php-tutorial/php-cookies.php
2. https://www.tutorialrepublic.com/php-tutorial/php-sessions.php

Exercise 7:

Create a simple Login System using MySQL and manage cookies/sessions.
Step1: Create a new directory on your server or local development environment. Inside this directory, create the following files:

- index.php: The main page where users will log in.
- welcome.php: The page users will be redirected to after successful login.
- logout.php: The page to log out and end the session.
- config.php: A configuration file for database connection and other settings.

Step2: In config.php, set up your database connection and create a users table. Here's a basic example:

```
<?php
// Database configuration
$servername = "your_database_server";
```

```
$username = "your_database_username";
$password = "your_database_password";
$dbname = "your_database_name";

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

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

// Create users table if not exists
$create_users_table = "CREATE TABLE IF NOT EXISTS users (
   id INT AUTO_INCREMENT PRIMARY KEY,
   username VARCHAR(50) NOT NULL,
   password VARCHAR(255) NOT NULL
)";

$conn->query($create_users_table);
?>
```

Step3: Create a simple login form in index.php.

In the login logic:

- After verifying the username, we use password_verify to check the password against the hashed password stored in the database.
- If the login is successful, we set the $_SESSION['user_id'] variable to store the user's session.
- Additionally, we set a cookie (user_id) to remember the user's ID for 30 days.

```
<?php
require 'config.php';
```

```
session_start();

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
   $username = $_POST['username'];
   $password = $_POST['password'];

   $login_query = $conn->query("SELECT * FROM users WHERE username =
'$username'");

   if ($login_query->num_rows == 1) {
      $user_data = $login_query->fetch_assoc();

      // Verify the password using password_verify
      if (password_verify($password, $user_data['password'])) {
         $_SESSION['user_id'] = $user_data['id'];

         // Set a cookie to remember the user
         setcookie('user_id', $user_data['id'], time() + (86400 * 30), "/"); // 86400
seconds = 1 day

         header('Location: welcome.php');
      } else {
         $login_error = "Invalid password.";
      }
   } else {
      $login_error = "Invalid username.";
   }
}
$conn->close();
?>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-
fit=no">
  <title>Login</title>
</head>
<body>

<div style="margin: 100px; text-align: center;">
  <h2>Login</h2>
  <form action="index.php" method="post">
    <label for="username">Username:</label>
    <input type="text" id="username" name="username" required><br>

    <label for="password">Password:</label>
    <input type="password" id="password" name="password" required><br>

    <button type="submit">Login</button>
  </form>

  <?php
  if (isset($login_error)) {
    echo "<p style='color: red;'>$login_error</p>";
  }
  ?>
</div>

</body>
</html>
```

Step4: Create a simple welcome page in welcome.php. In the welcome page:

- We prioritize the session over the cookie. If the session is not set, we check for the user ID in the cookie.

```
<?php
require 'config.php';

session_start();

if (!isset($_SESSION['user_id']) && !isset($_COOKIE['user_id'])) {
   header('Location: index.php');
   exit();
}

// Prioritize session over cookie
$user_id = isset($_SESSION['user_id']) ? $_SESSION['user_id'] : $_COOKIE['user_id'];
$user_query = $conn->query("SELECT * FROM users WHERE id = $user_id");
$user_data = $user_query->fetch_assoc();

$conn->close();
?>

<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta  name="viewport"  content="width=device-width,  initial-scale=1,  shrink-to-
fit=no">
   <title>Welcome</title>
</head>
<body>

<div style="margin: 100px; text-align: center;">
```

```
    <h2>Welcome, <?php echo $user_data['username']; ?>!</h2>
    <a href="logout.php">Logout</a>
</div>

</body>
</html>
```

Step5: Create a simple logout page in logout.php. In the logout logic:

- We unset all session variables and destroy the session.
- We also delete the session cookie and the user ID cookie.

```
<?php
session_start();

// Unset all session variables
$_SESSION = array();

// Delete the session cookie
if (isset($_COOKIE[session_name()])) {
    setcookie(session_name(), '', time() - 3600, '/');
}
// Destroy the session
session_destroy();

// Delete the user ID cookie
if (isset($_COOKIE['user_id'])) {
    setcookie('user_id', '', time() - 3600, '/');
}
header('Location: index.php');
?>
```

Step 6: Practice

1. Testing Login:

- Test the login system with both correct and incorrect credentials.
- Verify that users are redirected to the welcome page upon successful login.

2. Cookie Inspection:
   - Use browser developer tools to inspect cookies. Verify that a cookie named user_id is set after successful login.
   - Test the login page by disabling cookies in your browser and check if the login still works.
3. Logout Functionality:
   - Test the logout functionality by clicking the "Logout" link on the welcome page.
   - Verify that both the session and cookie are destroyed, and the user is redirected to the login page.
4. Session Timeout:
   - Experiment with session timeout settings. Change the session timeout duration in the PHP configuration or use session_set_cookie_params to set a custom timeout.

~~~~~The END~~~~~~~~~~~
~~~~~