**Landing, Login, and Enrollment Pages Development**

Essence Jackson

The University of Arizona Global Campus

CST 499 Capstone for Computer Software Technology

Instructor: Joseph Rangitsch

April 29, 2024

**Running a PHP File in XAMPP**

Running PHP files in XAMPP involves configuring the XAMPP environment and accessing the files through a web browser. Here's a detailed guide:

1. **Install XAMPP**: Download the appropriate XAMPP installer from the Apache Friends website (<https://www.apachefriends.org/>) and follow the installation instructions for your operating system. For Windows, you typically run the installer and follow the setup wizard, while for macOS and Linux, you may need to extract the downloaded archive and run the installation script.
2. **Start XAMPP Control Panel**: After installation, launch the XAMPP Control Panel. On Windows, you can usually find it in the Start menu or by searching for "XAMPP Control Panel." On macOS and Linux, you can start XAMPP from the terminal by navigating to the XAMPP installation directory and running the command **sudo ./xampp start**. The Control Panel provides an interface to manage Apache, MySQL, PHP, and other services.
3. **Start Apache and MySQL Services**: In the XAMPP Control Panel, locate the Apache and MySQL modules, and click on the "Start" buttons next to each to initiate the web server and database server respectively. Wait until the status indicators turn green, indicating that the services are running. You may also need to configure these services to start automatically whenever you boot your system.
4. **Locate htdocs Directory**: Navigate to the htdocs directory within your XAMPP installation directory. This directory serves as the document root for your local web server. By default, it's located at **C:\xampp\htdocs** on Windows, **/Applications/XAMPP/htdocs** on macOS, and **/opt/lampp/htdocs** on Linux.
5. **Place PHP File**: Move your PHP file into the htdocs directory or create a subdirectory within it for organizing your files. Make sure that your PHP file has the **.php** extension and contains valid PHP code.
6. **Access PHP File via Browser**: Open your web browser and type **http://localhost/yourfile.php** in the address bar, where **yourfile.php** is the name of your PHP file. Press Enter to load the PHP file. XAMPP will process the PHP code and display the output in the browser window.

**MySQL Database Functions and Database Connection Custom Class**

In the provided code, MySQL functions are used to interact with the database, while a custom database connection class (DatabaseHandler) encapsulates database connectivity and query execution functionalities. Here's a breakdown:

1. **MySQL Functions Usage**:
   * The **mysqli** extension is utilized for database connectivity and query execution in PHP. It provides an object-oriented interface to interact with MySQL databases.
   * Functions like **mysqli::query()**, **mysqli::fetch\_assoc()**, and **mysqli::insert\_id()** are used to perform various database operations such as executing SELECT, INSERT, and other queries, fetching data from the result set, and retrieving the ID of the last inserted row.
2. **Database Connection Custom Class (DatabaseHandler)**:
   * The **DatabaseHandler** class abstracts database interactions and provides a cleaner interface for executing queries.
   * It includes methods such as **executeSelectQuery()**, **executeQuery()**, and **executeInsertQuery()** to execute different types of SQL queries and handle result sets.
   * The constructor initializes the database connection using the **mysqli** class, and it throws an exception if the connection fails.

**Steps to Create the Registration Page and Save User Information in the Database**

Creating the registration page and saving user information in the database involves several steps, including HTML form creation, form submission handling, and database interaction. Here's a more detailed breakdown:

1. **HTML Form Creation (registration.php)**:
   * Create an HTML form with input fields for user registration data, including email, password, confirm password, first name, last name, address, and phone number.
   * Ensure appropriate validation attributes and form structure to collect user input securely. Use client-side validation using JavaScript and server-side validation using PHP to ensure data integrity.
2. **Form Submission Handling (registration.php)**:
   * Handle form submission using PHP. Retrieve form data using **$\_POST** superglobal.
   * Validate form inputs, ensuring required fields are filled and passwords match. Handle any validation errors appropriately by displaying error messages to the user.
3. **Database Interaction (database.php)**:
   * Utilize the **registerUser()** function (not provided in the code snippet) to interact with the database.
   * This function executes an INSERT query to insert user data into the **users** table, using the **executeInsertQuery()** method of the **DatabaseHandler** class.
   * Sanitize user input to prevent SQL injection attacks by using prepared statements or parameterized queries.
4. **Redirecting and Error Handling**:
   * Upon successful registration, redirect the user to the login page (**login.php**). If registration fails, redirect back to the registration page with an appropriate error message appended to the URL.
   * Use **header()** function to perform redirection, and include error messages in the URL parameters for displaying error feedback.
5. **Database Class Usage (database.php)**:
   * Utilize the **DatabaseHandler** class to establish a database connection and execute SQL queries for user registration.
   * The class handles database connectivity and query execution, ensuring a secure and efficient interaction with the database. Encapsulating database operations within a class promotes code reusability and maintainability.

By following these steps, you can create a registration page that securely collects user information and stores it in a MySQL database using PHP and XAMPP. Ensure to follow best practices for security, such as hashing passwords before storing them in the database and validating user input to prevent common vulnerabilities like SQL injection and cross-site scripting (XSS).

**Screenshots:**

**A white background with black text

Description automatically generatedLanding Page**

**A screenshot of a login screen

Description automatically generatedLogin Page:**

A login form with text

Description automatically generated**Registration Page:**

A screenshot of a computer

Description automatically generated**Database User Table:**

**Registration Source Code:**

<?php

// Include the database handler

require\_once 'database.php';

// Check if the registration form is submitted

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

    // Retrieve form data

    $email = $\_POST["email"];

    $password = $\_POST["password"];

    $confirmPassword = $\_POST["confirm\_password"]; // Corrected name for confirm password field

    $firstName = $\_POST["first\_name"];

    $lastName = $\_POST["last\_name"];

    $address = $\_POST["address"];

    $phone = $\_POST["phone"];

    // Check if passwords match

    if ($password !== $confirmPassword) {

        header("Location: registration.php?error=password\_mismatch");

        exit();

    }

    // Insert user data into the database using the $databaseHandler object

    $success = registerUser($databaseHandler, $email, $password, $firstName, $lastName, $address, $phone);

    if ($success) {

        // Registration successful, redirect to login page

        header("Location: login.php");

        exit();

    } else {

        // Registration failed, redirect back to registration page with an error message

        header("Location: registration.php?error=registration\_failed");

        exit();

    }

}

?>

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Registration</title>

</head>

<body>

    <header>

        <!-- Header content here -->

    </header>

    <main>

        <h1>Registration</h1>

        <!-- Registration form here -->

        <form action="<?php echo htmlspecialchars($\_SERVER["PHP\_SELF"]); ?>" method="post">

            <label for="email">Email:</label>

            <input type="email" id="email" name="email" required><br>

            <label for="password">Password:</label>

            <input type="password" id="password" name="password" required><br>

            <label for="confirm\_password">Confirm Password:</label>

            <input type="password" id="confirm\_password" name="confirm\_password" required><br>

            <label for="first\_name">First Name:</label>

            <input type="text" id="first\_name" name="first\_name"><br>

            <label for="last\_name">Last Name:</label>

            <input type="text" id="last\_name" name="last\_name"><br>

            <label for="address">Address:</label>

            <input type="text" id="address" name="address"><br>

            <label for="phone">Phone:</label>

            <input type="text" id="phone" name="phone"><br>

            <button type="submit">Register</button>

        </form>

    </main>

    <?php include 'footer.php'; ?>

</body>

</html>

**References:**

Tsui, F., Karam, O., & Bernal, B. (2018). [*Essentials of software engineering*](https://uagc.instructure.com/courses/130505/modules/items/6659009)(4th ed.). Jones & Bartlett Learning.

Mikoluk, K. (2013, September 18). [XAMPP tutorial: How to use XAMPP to run your own web serverLinks to an external site.](https://blog.udemy.com/xampp-tutorial). *Udemy.*<https://blog.udemy.com/xampp-tutorial>

SiteGround. (n.d.). [*phpMyAdmin create and populate tables tutorial.*Links to an external site.](https://www.siteground.com/tutorials/phpmyadmin/create-populate-tables/) https://www.siteground.com/tutorials/phpmyadmin/create-populate-tables/