FSD Laboratory 04

Aim: Write server-side script in PHP to perform form validation and create database application using PHP and MySQL to perform insert, update, delete and search operations.

Objectives:

1. To understand Server-side Scripting.
2. To learn database connectivity using PHP-MySQL.
3. To perform insert, update, delete and search operations on database.

Theory:

1. PHP Architecture.
   PHP (Hypertext Preprocessor) is a server-side scripting language widely used for web development. Its architecture follows a typical request-response cycle, where PHP scripts are executed on the server to generate dynamic web pages. Here's an overview of PHP's architecture:

   1. **Client-Server Model:** PHP operates within the client-server model of web development. The client (typically a web browser) sends a request to the web server for a specific web page. The web server, which has PHP installed, processes the request and sends back an HTML response to the client.

   2. **HTTP Request:** When a user interacts with a web application, they send an HTTP request to the web server. The request may include various components, such as the URL, HTTP method (e.g., GET or POST), headers, and possibly data (e.g., form inputs).

   3. **Web Server:** The web server, such as Apache, Nginx, or Microsoft IIS, receives the HTTP request and forwards it to the PHP interpreter for processing. The server is responsible for managing incoming requests, handling security, and routing requests to the appropriate PHP scripts.

   4. **PHP Interpreter:** The PHP interpreter is a server-side component responsible for executing PHP scripts. It interprets the PHP code contained in the requested script file and generates dynamic content (usually HTML) based on the script's logic and data processing.

   5. **PHP Scripts**: PHP scripts are files containing PHP code mixed with HTML or other content. These scripts can include logic to interact with databases, process form submissions, perform calculations, and more. PHP scripts are often embedded within HTML markup and are saved with a ".php" file extension.

   6. **Data Sources**: PHP can interact with various data sources, including databases (e.g., MySQL, PostgreSQL, SQLite), external APIs, and files. It retrieves, manipulates, and stores data from these sources as needed to generate dynamic content.

   7. **HTML Response**:After processing the PHP script, the PHP interpreter generates an HTML response, which is sent back to the web server.

   8. **HTTP Response:** The web server receives the HTML response from PHP and sends it back to the client as an HTTP response. The response may include additional HTTP headers, cookies, and other metadata.

9. **Client Presentation**: The client's web browser receives the HTML response and renders it to display the web page to the user. JavaScript, CSS, and other client-side technologies may be included in the HTML to enhance the user interface and interactivity.

10. **User Interaction**: The user interacts with the web page in their browser, initiating new requests to the server by clicking links, submitting forms, or performing other actions.

11. **Repeat Cycle**: The request-response cycle continues as the user interacts with the web application, and PHP scripts process each request, generating dynamic content as needed.

Overall, PHP's architecture is designed for server-side scripting, allowing developers to create dynamic and data-driven web applications by embedding PHP code within HTML documents and executing it on the server in response to client requests. This architecture provides the foundation for building a wide range of web applications, from simple websites to complex web applications and APIs.

2. Steps for Database connectivity in PHP.
Certainly, here are the simplified steps for establishing database connectivity in PHP:

1. **Choose a Database**: Select a database system (e.g., MySQL, PostgreSQL, SQLite).

2. **Install Necessary Extensions**: Ensure the PHP extension for your chosen database is enabled.

3. **Create Database Credentials:** Gather hostname, username, password, and database name.

4. **Establish Connection:**
   - For MySQL using `mysqli`:
     ```php
     $conn = new mysqli($hostname, $username, $password, $dbname);
     ```
   - For MySQL using `PDO`:
     ```php
     $conn = new PDO("mysql:host=$hostname;dbname=$dbname", $username, $password);
     ```

5. **Perform Database Operations:** Execute SQL queries and operations.

6. **Close Connection (Optional):**
   - For `mysqli`, use `$conn->close()`.
   - For `PDO`, the connection closes automatically when out of scope.

That's a concise summary of the steps involved in connecting to a database in PHP. Remember to adapt the code to your specific database system and needs.

FAQ:

1. What are the advantages of Server-side Scripting?
Server-side scripting offers several advantages:

1. **Security**: Server-side scripts can securely handle sensitive operations and data access, reducing the risk of exposing critical information to clients.

2. **Data Handling**: They enable efficient data processing, manipulation, and validation, ensuring data integrity and reliability.

3. **Scalability:** Server-side code can handle multiple requests concurrently, making it suitable for scaling web applications.

4. **Access Control:** Server-side scripting allows enforcing access control and authentication mechanisms to protect resources.

5. **Integration**: It enables integration with databases, APIs, and external services, enhancing functionality.

6. **SEO:**Server-side rendering improves search engine optimization, as search engines can crawl content.

7. **Code Privacy:** Sensitive code and business logic are hidden from clients, reducing the risk of intellectual property theft.

8. **Consistency**: Server-side code ensures consistent behavior across various client devices and browsers.

9. **Dynamic Content**: It enables dynamic content generation and personalization based on user input or other factors.

10. **Error Handling:** Server-side scripts provide detailed error handling, improving application robustness and debugging capabilities.

2. What is XAMPP and phpMyAdmin?

XAMPP (short for Cross-Platform, Apache, MySQL, PHP, and Perl): XAMPP is a free and open-source web server solution that simplifies the setup and management of a local web development environment. It includes Apache (web server), MySQL (database server), PHP (scripting language), and Perl. XAMPP allows developers to create and test web applications on their local machines before deploying them to live servers.

phpMyAdmin:phpMyAdmin is a web-based graphical user interface (GUI) tool for managing MySQL databases. It provides a user-friendly way to create, modify, delete, and query databases, tables, and records. phpMyAdmin simplifies database administration tasks and is commonly used in conjunction with web development and database management on local and remote servers.

3. What are the two ways to connect to a database in PHP?

The two common ways to connect to a database in PHP are:

1. Using MySQLi Extension (MySQL Improved): MySQLi is a built-in PHP extension that provides a powerful and secure way to interact with MySQL databases. It offers both a procedural and an object-oriented API for connecting to and querying databases.

2. Using PDO (PHP Data Objects): PDO is a database abstraction layer in PHP that provides a consistent interface for connecting to various database systems, including MySQL, PostgreSQL, SQLite, and more. It offers a flexible and object-oriented approach to database connectivity.

Both MySQLi and PDO are widely used and have their advantages, depending on your specific project requirements and coding preferences.

Output: Screenshots of the output to be attached.

Sample Problem Statements:

PHP CRUD Operations

1.Student can create a PHP form or use existing/ implemented HTML form for Student's Registration System with the fields mentioned: First name,Last name, Roll No/ID, Password, Confirm Password,Contact number and perform following operations

1.Insert student details -First name,Last name, Roll No/ID, Password, Confirm Password,Contact number
2.Delete the Student records based on Roll no/ID
3.Update the Student details based on Roll no/ID- Example students can update their contact details based on searching the record with Roll no.
4.Display the Updated student details or View the Students record in tabular format.

Apply Form Validation on the necessary fields using PHP/Javascript

2. Student can create a PHP form or use existing/ implemented HTML form for Library Management System with the fields mentioned: Book name, ISBN No, Book title, Author name , Publisher name and perform following operations

1.Insert Book details -Book name, ISBN No, Book title, Author name, Publisher name
2.Delete the Book records based on ISBN No
3.Update the Book details based on ISBN No- Example students can update wrong entered book details based on searching the record with ISBN No.
4.Display the Updated Book details or View the Book Details records in tabular format.

Apply Form Validation on the necessary fields using PHP/Javascript

3. Student can create a PHP form or use existing/ implemented HTML form for Employee Management System with the fields mentioned: Employee name, Employee ID, Department_name, Phone number , Joining Date and perform following operations

1.Insert Employee details -Employee name, Employee ID, Department_name, Phone number , Joining Date
2.Delete the Employee records based on Employee ID
3.Update the Employee details based on Employee ID- Example students can update Employee details based on searching the record with Employee ID.
4.Display the Updated Employee details or View the Employee Details records in tabular format.

Apply Form Validation on the necessary fields using PHP/Javascript

4. Student can create a PHP form or use existing/ implemented HTML form for Flight Booking Management System with the fields mentioned: Passenger name, From, to, date,Departure date,Arrival date, Phone number , Email ID and perform following operations

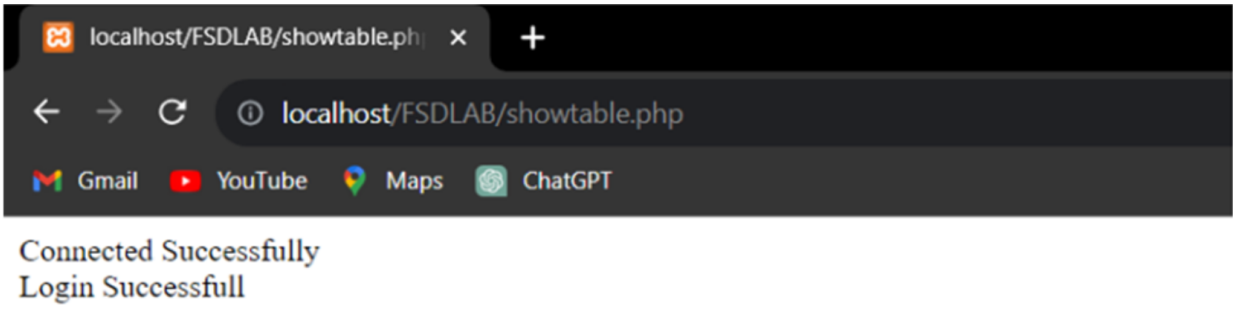
1.Insert Passenger details -Passenger name, From, to, date,Departure date,Arrival date, Phone number , Email ID

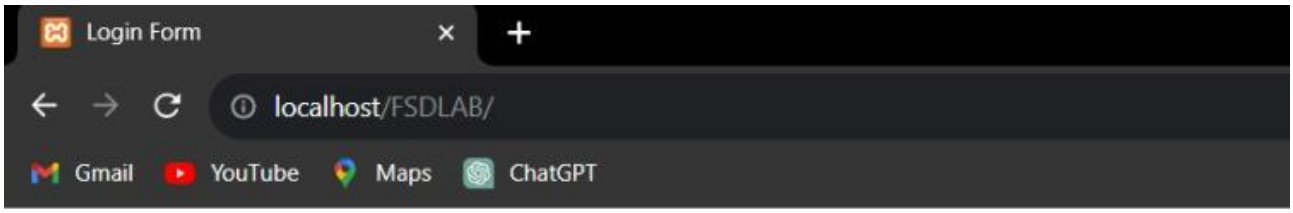
2.Delete the Passenger records based on Phone Number
3.Update the Passenger details based on Phone Number - Example students can update Flight Booking details based on searching the record with Phone Number.
4.Display the Updated Flight Booking details or View the Flight Booking Details records in tabular format.

Apply Form Validation on the necessary fields using PHP/Javascript.

Technologies Student Should Use:
XAMPP
PHP for Server-side Scripting
MySQL as a backend Database

Output:





Hosted on Local Server

Username

Devanshu

Password

••••••••••

Submit