Web Development with PHP
Practical - 10
PHP-Mysql Database

Understanding database connectivity in PHP is crucial for building dynamic web applications. PHP provides various extensions and libraries, such as MySQLi and PDO, to connect to databases like MySQL, PostgreSQL, and SQLite. These extensions offer functions and methods to establish connections, execute SQL queries, and retrieve results. The process involves providing database credentials, including hostname, username, password, and database name. PHP's database connectivity allows developers to interact with databases seamlessly, perform CRUD (Create, Read, Update, Delete) operations, and handle data efficiently.

Database connectivity in PHP is a fundamental aspect of web development that allows PHP applications to interact with relational databases. By establishing a connection between PHP and a database management system (DBMS), developers can perform various database operations, such as storing, retrieving, updating, and deleting data.

PHP provides robust database connectivity features, supporting a wide range of DBMSs, including MySQL, PostgreSQL, SQLite, Oracle, and more. This flexibility allows developers to work with their preferred database system seamlessly. To establish a database connection in PHP, developers typically provide connection details, such as the server address, username, password, and database name. Once the connection is established, PHP applications can execute SQL queries and retrieve the results, enabling efficient data manipulation and retrieval.

Database connectivity in PHP facilitates the creation of dynamic and data-driven web applications. It allows developers to store user information, manage content, implement user authentication, and handle complex data relationships. With PHP's database connectivity, applications can seamlessly integrate and leverage the power of relational databases.

It is important to note that database connectivity in PHP also involves considerations for security, such as protecting against SQL injection attacks by properly sanitizing and validating user input, implementing prepared statements or parameterized queries, and applying appropriate access controls to safeguard sensitive data.

PHP's database connectivity capabilities provide developers with the necessary tools to create robust and interactive web applications that effectively store, retrieve, and manipulate data from various relational databases.

MySQL

MySQL is an older PHP extension that allows connectivity to MySQL databases. However, it is no longer recommended for new projects, as it has been deprecated since PHP version 5.5 and removed in PHP 7.0 and later versions. It is advised to use either MySQLi or PDO for new projects.

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

     // Create a connection
     $conn = mysql_connect($servername, $username, $password);

     // Check the connection
     if (!$conn) {
         die("Connection failed: " . mysql_error());
     }

     // Select Database
     $db_selected = mysql_select_db($dbname, $conn);
     if (!$db_selected) {
         die ("Database selection failed: " . mysql_error());
     }

     // Perform queries and other operations

     // Close connection
     mysql_close($conn);
?>
```

MySQLi

MySQLi (MySQL improved) is an improved version of the MySQL extension and is recommended for new PHP projects. It provides both a procedural and object-oriented interface for connecting to and interacting with MySQL databases.

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

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

     // Check the connection
     if (!$conn) {
         die("Connection failed: " . mysqli_connect_error());
     }

     // Perform queries and other operations

     // Close connection
     mysqli_close($conn);
?>
//-----------------------------------------------------------------
<?php
     $servername = "localhost";//Replace with your database server
     name
     $username = "username";// Replace with your database username
     $password = "password";// Replace with your database password
     $dbname = "database";// Replace with your database name

     $conn = new mysqli($servername, $username, $password,
     $dbname);
     if ($conn->connect_error) {
         die("Connection failed: " . $conn->connect_error);
     }


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

PDO (PHP Data Objects)

PDO is a database abstraction layer in PHP that supports multiple databases, including MySQL. It provides a consistent API for connecting to and working with different databases, making it easier to switch between database systems.

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

     try {
         // Create a connection
          $conn = new PDO("mysql:host=$servername;dbname=$dbname",
     $username, $password);

         // Set the PDO error mode to exception
                              $conn->setAttribute(PDO::ATTR_ERRMODE,
     PDO::ERRMODE_EXCEPTION);

         // Perform queries and other operations

         // Close connection
         $conn = null;
     } catch(PDOException $e) {
         echo "Connection failed: ". $e->getMessage();
     }
?>
```

Function	Description
affected_rows()	Returns the number of affected rows in the previous MySQL operation
connect()	Opens a new connection to the MySQL server
connect_error()	Returns the error description from the last connection error
fetch_all()	Fetches all result rows as an associative array, a numeric array, or both
fetch_array()	Fetches a result row as an associative, a numeric array, or both
fetch_row()	Fetches one row from a result-set and returns it as an enumerated array
field_count ()	Returns the number of columns for the most recent query
fetch_fields()	Returns an array of objects that represent the fields in a result-set
close()	Closes a previously opened database connection
multi_query()	Performs one or more queries on the database
query()	Performs a query against a database

Exercise

1. Do the following example.
    a. From the phpMyAdmin dashboard create database Person and create a table personinfo having field *name, age* and *city*. Create a PHP application for user registration by making a HTML registration page which takes the person name,age,city. When the user clicks on the insert button it inserts the record into the table person. When the user clicks on the display button it will display all records in appropriate html format. Use MySQLi Procedural methods for the same.
    b. Consider the above example for Person, extend it by adding the functionality of update. Create a page for the user where all records are displayed in the table along with the Update button as the last column. When the user presses the update button the respective record is opened in a new page for updating where the user can update the value of city and age. When the user clicks on save. The records should get updated and the updated list or records should be displayed back along with the update button. Use MySQLi Class for the implementation.
    c. Consider the above example for Person, add the functionality of delete. Create a page for the user where all records are displayed in the table along with the Delete button as the last column. When the user presses the delete button the respective record gets deleted from the table and displays an appropriate message to the user along with the updated list or records should be displayed back along with the delete button. Use PDO for the implementation.

2. Create an Employee records maintaining application in PHP. Application will maintain personal and salary data of the employee. The application will have the following functionalities.
    a. Add an employee
    b. Edit salary details of the employee
    c. Delete an employee.
    d. Display single employee details in a decorated HTML page.
    e. Display all employees in a decorated HTML page.

    Make use of prepared statements for this example. You can either select MySQLi Class or PDO for the same