WEB PROGRAMMING AND APPLICATIONS (Tutorial 8)

Table of Contents

MySQL *3*

What's MySQL? 3

Should I Use MySQLi or PDO? 3

Open a Connection to MySQL 4

MySQLi Object-Oriented 4

MySQLi Procedural 5

Using PDO 5

Close the Connection 6

MySQLi Object-Oriented 6

MySQLi Procedural 6

Using PDO 6

Select Data With MySQLi 6

MySQLi Object-Oriented 6

MySQLi Procedural 6

Using PDO 7

Insert Data Into MySQL 7

MySQLi Object-Oriented 7

MySQLi Procedural 8

Using PDO 8

PHPMailer *9*

What is phpmailer? 9

Installing PHPMailer 9

via Composer 9

via PHPMailer library manually 9

Using PHPMailer in your PHP 10

Product management application *11*

db.php 11

utils.php 11

register.php 12

activate.php 13

Index.php 14

add_product.php **15**

References ***16***

MySQL

What's MySQL?

MySQL is a popular relational database management system widely used in web development, particularly in conjunction with PHP. Integrating MySQL with PHP allows developers to create dynamic and data-driven websites and applications.

PHP 5 and later can work with a MySQL database using:

- **MySQLi extension** (the "i" stands for improved)
- **PDO (PHP Data Objects)**

Should I Use MySQLi or PDO?

Choosing between MySQLi and PDO (PHP Data Objects) depends on various factors, including your specific project requirements, coding style, and personal preference. Here's a comparison to help you decide:

1. Ease of Use:
    - **MySQLi**: It closely resembles the traditional MySQL API and is relatively straightforward for developers familiar with MySQL. However, it requires more code for error handling and prepared statements.
    - **PDO**: It provides a more consistent and object-oriented interface for database access, making it easier to work with different database systems. PDO also supports features like named placeholders for prepared statements.
2. Database Support:
    - **MySQLi**: Specifically designed for MySQL databases. It offers support for features unique to MySQL, such as stored procedures and multiple statements in one query.
    - **PDO**: Supports multiple database systems, including MySQL, PostgreSQL, SQLite, and more. This makes it a better choice if you plan to switch databases in the future.
3. Prepared Statements:
    - **MySQLi**: Supports prepared statements with parameter binding, which helps prevent SQL injection attacks. However, it requires more verbose syntax for prepared statements compared to PDO.
    - **PDO**: Offers a simpler and more consistent way of working with prepared statements, including both positional and named parameters.
4. Error Handling:

- **MySQLi**: Requires explicit error checking after each database operation, which can lead to more verbose code.
- **PDO**: Offers more streamlined error handling through exceptions, making it easier to manage errors within a try-catch block.

5. Community and Support:
   - **MySQLi**: Being specific to MySQL, it benefits from a large community and extensive documentation tailored to MySQL.
   - **PDO**: Widely adopted due to its support for multiple database systems, resulting in a diverse community and comprehensive resources.
6. Performance:
   - **MySQLi**: Generally considered to be slightly faster than PDO due to its specialized nature and direct interaction with MySQL.
   - **PDO**: Provides good performance and can be optimized for speed, especially when used with prepared statements and parameter binding.

In conclusion, if you're working exclusively with MySQL and prefer a simpler, more traditional approach, MySQLi might be the better choice. However, if you value flexibility, portability, and a more consistent interface across different database systems, PDO is likely the way to go. Ultimately, both MySQLi and PDO are capable options, so the decision comes down to your specific project needs and personal preferences.

Open a Connection to MySQL

Opening a connection to a MySQL database typically involves using either the MySQLi extension or PDO (PHP Data Objects). Here's how you can open a connection using both methods:

MySQLi Object-Oriented

```php
// Database credentials
$servername = "localhost";
$username = "your_username";
$password = "your_password";
$database = "your_database";

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

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

MySQLi Procedural

```php
// Database credentials
$servername = "localhost";
$username = "your_username";
$password = "your_password";
$database = "your_database";

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

// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
} else {
    echo "Connected successfully";
}
```

Using PDO

```php
// Database credentials
$servername = "localhost";
$username = "your_username";
$password = "your_password";
$database = "your_database";

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

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

    echo "Connected successfully";
} catch(PDOException $e) {
    die("Connection failed: " . $e->getMessage());
}
```

Close the Connection

MySQLi Object-Oriented

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

MySQLi Procedural

```
mysqli_close($conn);
```

Using PDO

```
$conn = null;
```

Select Data With MySQLi

MySQLi Object-Oriented

```
$sql = "SELECT id, firstname, lastname FROM MyGuests";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
  // output data of each row
  while($row = $result->fetch_assoc()) {
    echo $row["firstname"]. " " . $row["lastname"]. "<br>";
  }
} else {
  echo "0 results";
}
```

MySQLi Procedural

```
$sql = "SELECT id, firstname, lastname FROM MyGuests";
$result = mysqli_query($conn, $sql);

if (mysqli_num_rows($result) > 0) {
  // output data of each row
  while($row = mysqli_fetch_assoc($result)) {
    echo $row["firstname"]. " " . $row["lastname"]. "<br>";
  }
} else {
```

```
  echo "0 results";
}
```

Using PDO

```
try {
    // Create a PDO instance

    // SQL query
    $sql = "SELECT id, firstname, lastname FROM MyGuests";

    // Prepare statement
    $stmt = $conn->prepare($sql);

    // Execute statement
    $stmt->execute();

    // Check if any rows were returned
    if ($stmt->rowCount() > 0) {
        // Fetch data using fetch() or fetchAll()
        while ($row = $stmt->fetch(PDO::FETCH_ASSOC)) {
            // Output data of each row
            echo $row["firstname"] . " " . $row["lastname"] . "<br>";
        }
    } else {
        echo "0 results";
    }
} catch(PDOException $e) {
    echo "Error: " . $e->getMessage();
}
```

Insert Data Into MySQL

MySQLi Object-Oriented

```
$sql = "INSERT INTO MyGuests (firstname, lastname, email)
                        VALUES ('John', 'Doe', 'john@example.com')";

if ($conn->query($sql) === TRUE) {
  echo "New record created successfully";
} else {
  echo "Error: " . $sql . "<br>" . $conn->error;
}
```

MySQLi Procedural

```
$sql = "INSERT INTO MyGuests (firstname, lastname, email)
                          VALUES ('John', 'Doe', 'john@example.com')";

if (mysqli_query($conn, $sql)) {
  echo "New record created successfully";
} else {
  echo "Error: " . $sql . "<br>" . mysqli_error($conn);
}
```

Using PDO

```
try {
    // Create connection
     ...

    $sql = "INSERT INTO MyGuests (firstname, lastname, email)
                              VALUES ('John', 'Doe', 'john@example.com')";
    $conn->exec($sql);
    echo "New record created successfully";
} catch (PDOException $e) {
    echo $sql . "<br>" . $e->getMessage();
}
```

PHPMailer

What is phpmailer?

PHPMailer is a powerful and popular email-sending library for PHP. It provides a comprehensive set of features for creating and sending email messages from PHP applications. Here's an overview of PHPMailer:

1. **Sending Emails**: PHPMailer allows you to send emails directly from your PHP scripts without relying on the mail server configured on your server. This gives you more control over the email-sending process.
2. **SMTP Support**: It supports sending emails using SMTP (Simple Mail Transfer Protocol), which is a standard protocol for sending emails over the internet. This allows you to send emails through third-party email services like Gmail, Outlook, or your own SMTP server.
3. **Multiple Attachments**: PHPMailer supports attaching multiple files to your emails, making it easy to send documents, images, or any other type of file along with your email message.
4. **HTML and Plain Text Emails**: You can send both HTML-formatted and plain text emails using PHPMailer. This gives you the flexibility to send emails with rich formatting, including images, links, and styled text.
5. **Error Handling**: PHPMailer provides robust error handling mechanisms to help you diagnose and troubleshoot email sending issues.

Installing PHPMailer

via Composer

1. Open your terminal or command prompt.
2. Navigate to the root directory of your PHP project.
3. Run the following command to install PHPMailer:

```
composer require phpmailer/phpmailer
```

via PHPMailer library manually

We can still use PHPMailer without Composer by downloading the PHPMailer library manually and including it in your project. Here's how:

1. **Download PHPMailer**: Go to the PHPMailer GitHub releases page and download the latest version of PHPMailer as a ZIP file.
2. **Extract the ZIP file**: Extract the contents of the ZIP file you downloaded. You'll find a folder named something like **PHPMailer-x.y.z**, where **x.y.z** is the version number.

3. **Copy the PHPMailer files**: Copy the entire **PHPMailer-x.y.z** folder into your project directory.
4. **Include PHPMailer files in your PHP code**: In your PHP files where you want to use PHPMailer, include the necessary PHPMailer files using the **require** or **require_once** statement. You'll typically need to include the **PHPMailer.php** file.

   ```
   require 'path/to/PHPMailer-x.y.z/src/PHPMailer.php';
   ```

   Replace **'path/to/PHPMailer-x.y.z/src/PHPMailer.php'** with the actual path to the **PHPMailer.php** file in your project.

Using PHPMailer in your PHP

```
require 'PHPMailer-master/src/Exception.php';
require 'PHPMailer-master/src/PHPMailer.php';
require 'PHPMailer-master/src/SMTP.php'

try {
        $mail->isSMTP();
        $mail->Host = 'smtp.gmail.com'
        $mail->SMTPAuth = true;
        $mail->Username = $email;
        $mail->Password = $password;
        $mail->SMTPSecure = 'tls';
        $mail->Port = 587;

        $mail->setFrom($email, $full_name);
        $mail->addAddress($to);

        $mail->isHTML(true);
        $mail->Subject = $subject;
        $mail->Body = $body;

        $mail->send();
        return true;
    } catch (Exception $e) {
        return $mail->ErrorInfo;
    }
```

Product management application

db.php

```php
$servername = "localhost";
$db_username = "";
$db_password = "";
$dbname = "";

// Create connection
$conn = new mysqli($servername, $db_username, $db_password, $dbname);
$conn->set_charset("utf8");

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

utils.php

```php
use PHPMailer\PHPMailer\PHPMailer;
use PHPMailer\PHPMailer\Exception;

require 'PHPMailer-master/src/Exception.php';
require 'PHPMailer-master/src/PHPMailer.php';
require 'PHPMailer-master/src/SMTP.php';

function send_mail($to, $subject, $body)
{
    $email = "";
    $password = "";
    $first_name = "";
    $last_name = "";
    $mail = new PHPMailer(true);

    try {
        $mail->isSMTP();
        $mail->Host = 'smtp.gmail.com';
        $mail->SMTPAuth = true;
        $mail->Username = $email;
        $mail->Password = $password;
        $mail->SMTPSecure = 'tls';
        $mail->Port = 587;
```

```
        $mail->setFrom($email, $first_name . ' ' . $last_name);
        $mail->addAddress($to);

        $mail->isHTML(true);
        $mail->Subject = $subject;
        $mail->Body = $body;

        $mail->send();
        return true;
    } catch (Exception $e) {
        return $mail->ErrorInfo;
    }
}
function generateToken($length = 20) {
    return bin2hex(random_bytes($length));
}

if( !function_exists('random_bytes') )
{
    function random_bytes($length = 6)
    {
        $characters = '0123456789';
        $characters_length = strlen($characters);
        $output = '';
        for ($i = 0; $i < $length; $i++)
            $output .= $characters[rand(0, $characters_length - 1)];

        return $output;
    }
}
```

register.php

```
require "utils.php";
require "db.php";

$error = '';
$first_name = '';
$last_name = '';
$email = '';
$user = '';
$pass = '';
$pass_confirm = '';

if (isset($_POST['first']) && isset($_POST['pass-confirm']) && ...) {
    $first_name = $_POST['first'];
```

```
    $last_name = $_POST['last'];
    $email = $_POST['email'];
    $user = $_POST['user'];
    $pass = $_POST['pass'];
    $pass_confirm = $_POST['pass-confirm'];

    $activate_token = generateToken(5);

    // Hash the password
    $hashed_password = password_hash($pass, PASSWORD_DEFAULT);

    // Prepare and bind statement
    $stmt = $conn->prepare("INSERT INTO account (username, firstname,
lastname, email, password, activate_token) VALUES (?, ?, ?, ?, ?, ?)");

    $stmt->bind_param("ssssss", $user, $first_name, $last_name, $email,
                                    $hashed_password, $activate_token);

    // Execute the statement
    if ($stmt->execute()) {
        $activation_link = '.../activate.php?email='
                                    . urlencode($email)
                                    . '&activate_token='
                                    . urlencode($activate_token);
        $body = "click to activate your account: <a href="'
                        . $activation_link . '">Activate</a>';

        send_mail($email, "Account Activation", $body);
    } else {
        echo "Error: " . $stmt->error;
    }

    $stmt->close();
    $conn->close();
}
```

activate.php

```
require "db.php";

$error = '';
if (isset($_GET['email']) && isset($_GET['activate_token'])) {
  $email = $_GET['email'];
  $token = $_GET['activate_token'];
```

```
  $sql = "SELECT * FROM account WHERE email = '$email' AND
                                        activate_token = '$token'";

  $result = mysqli_query($conn, $sql);
  if (mysqli_num_rows($result) > 0) {
    $update_sql = "UPDATE account SET activated = 1 WHERE
                    email = '$email' AND activate_token = '$token'";
    if (mysqli_query($conn, $update_sql)) {
      // Account activated successfully
    } else {
      $error = 'Error updating record: ' . mysqli_error($conn);
    }
  } else {
    $error = 'Invalid activation link';
  }
} else {
  $error = 'Invalid activation link ';
}
```

Index.php

```
require "db.php";

$sql = "SELECT * FROM product";
$result = mysqli_query($conn, $sql);


while ($row = mysqli_fetch_assoc($result)) {
    echo "<tr>";
    echo "<td class='align-middle'><img
                                src='images/{$row['image']}'></td>";
    echo "<td class='align-middle'>{$row['name']}</td>";
    echo "<td class='align-middle'>{$row['price']} VND</td>";
    echo "<td class='align-middle'>{$row['description']}</td>";
    echo "<td class='align-middle'>";
    echo "<button class='btn btn-sm btn-primary mr-1 edit-btn'>
            <i class='fas fa-pen'></i></button>";
    echo "<button class='btn btn-sm btn-danger delete-btn'>
            <i class='fas fa-trash-alt'></i></button>";
    echo "</td>";
    echo "</tr>";
}
```

add_product.php

```php
require "db.php";

$error = '';
$name = '';
$price = '';
$desc = '';


if (isset($_POST['name']) && isset($_POST['price']) && ...) {
    $name = $_POST['name'];
    $price = $_POST['price'];
    $desc = $_POST['desc'];


    if (empty($name)) {
        $error = 'Hãy nhập tên sản phẩm';
    } else if (intval($price) <= 0) {
        $error = 'Giá của sản phẩm không hợp lệ';
    } else if (intval($price) < 1000000
                    || intval($price) % 10000 != 0) {
        $error = 'Giá phải > 1,000,000đ và là bội số của 10,000 đ';
    } else if (empty($desc)) {
        $error = 'Hãy nhập mô tả của sản phẩm';
    } else if ($_FILES['image']['error'] != UPLOAD_ERR_OK) {
        $error = 'Vui lòng upload ảnh của sản phẩm';
    } else {
        $upload_dir = 'images/';
        $tmp_name = $_FILES['image']['tmp_name'];
        $image_name = uniqid() . '_' . $_FILES['image']['name'];
        move_uploaded_file($tmp_name, $upload_dir . $image_name);

        $sql = "INSERT INTO product (name, price, description, image)
                    VALUES ('$name', $price, '$desc', '$image_name')";
        $result = mysqli_query($conn, $sql);

        header("Location: index.php");
    }
}
```

References

https://www.w3schools.com/php/default.asp