# Kết nối MySQL

# Nội Dung Khóa Học

1

## Cơ Bản

Tổng quan MySQL

2

## Kết Nối

Thiết lập kết nối PHP-MySQL

3

## Truy Vấn

Các thao tác CRUD cơ bản

4

## Nâng Cao

Bảo mật và tối ưu hóa

5

## Dự Án

Xây dựng ứng dụng hoàn chỉnh

# MySQL Là Gì?

MySQL là hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) phổ biến nhất thế giới:

- Phát triển bởi Oracle Corporation
- Mã nguồn mở và miễn phí (phiên bản Community)
- Tốc độ cao, bảo mật và đáng tin cậy
- Hỗ trợ SQL - ngôn ngữ truy vấn chuẩn
- Hoạt động trên nhiều nền tảng (Windows, Linux, macOS)
- Được sử dụng bởi Facebook, Twitter, YouTube, v.v.

# Tại Sao Kết Hợp PHP & MySQL?

## Tương Thích Cao

PHP và MySQL được phát triển để hoạt động tốt cùng nhau, với nhiều hàm và phương thức được tối ưu sẵn.

## Hiệu Suất Tốt

Cả hai đều được thiết kế để xử lý nhanh, tối ưu tài nguyên và đáp ứng tốt với khối lượng truy cập lớn.

## Cộng Đồng Lớn

Dễ dàng tìm kiếm tài liệu, hướng dẫn và giải pháp cho hầu hết các vấn đề nhờ cộng đồng đông đảo người dùng.

# Cài Đặt Laragon

1

## Bước 1: Tải Laragon

Truy cập laragon.org và tải bản Laragon phù hợp với hệ điều hành của bạn (Laragon Full hoặc Laragon Lite).

2

## Bước 2: Cài đặt

Chạy file cài đặt và làm theo hướng dẫn. Laragon đã tích hợp sẵn Apache, Nginx, MySQL, PHP và các công cụ khác.

3

## Bước 3: Khởi động dịch vụ

Mở Laragon, nhấn nút "Start All" để khởi động tất cả các dịch vụ hoặc khởi động từng dịch vụ riêng lẻ.

4

## Bước 4: Kiểm tra cài đặt

Mở trình duyệt và truy cập http://localhost. Nếu thấy trang chào mừng của Laragon, bạn đã cài đặt thành công.

# Tạo Cơ Sở Dữ Liệu MySQL

1. Mở trình duyệt và truy cập **http://localhost/phpmyadmin**
2. Đăng nhập với tên người dùng "root" (mặc định không có mật khẩu)
3. Nhấp vào tab "Databases" (Cơ sở dữ liệu)
4. Nhập tên cơ sở dữ liệu mới (ví dụ: "shop_db")
5. Chọn bảng mã "utf8mb4_unicode_ci" để hỗ trợ tiếng Việt
6. Nhấp "Create" (Tạo) để hoàn tất

# Tạo Bảng Trong MySQL

Sau khi tạo cơ sở dữ liệu, chúng ta cần tạo các bảng để lưu trữ dữ liệu:

1. Chọn cơ sở dữ liệu vừa tạo từ menu bên trái
2. Nhấp tab "SQL" để nhập lệnh SQL
3. Nhập lệnh CREATE TABLE và thực thi

```
CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL,
    email VARCHAR(100) NOT NULL UNIQUE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Lệnh trên tạo bảng "users" với các trường: id, username, password, email và created_at.

# Kết Nối PHP với MySQL

PHP cung cấp nhiều cách khác nhau để kết nối với MySQL:

## MySQLi (Cải tiến)

Là phiên bản cải tiến của thư viện MySQL cũ, hỗ trợ cả hướng thủ tục và hướng đối tượng.

Được khuyên dùng cho các dự án mới làm việc với MySQL.

## PDO (PHP Data Objects)

Giao diện truy cập dữ liệu nhất quán cho nhiều loại cơ sở dữ liệu khác nhau.

Linh hoạt hơn vì có thể làm việc với 12+ hệ cơ sở dữ liệu khác nhau.

# Kết Nối MySQL với MySQLi

## Cách thức hướng thủ tục:

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

// Tạo kết nối
$conn = mysqli_connect($servername,
        $username, $password, $dbname);

// Kiểm tra kết nối
if (!$conn) {
    die("Kết nối thất bại: " .
        mysqli_connect_error());
}
echo "Kết nối thành công!";
?>
```

## Cách thức hướng đối tượng:

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

// Tạo kết nối
$conn = new mysqli($servername,
 $username, $password, $dbname);

// Kiểm tra kết nối
if ($conn->connect_error) {
 die("Kết nối thất bại: " .
 $conn->connect_error);
}
echo "Kết nối thành công!";
?>
```

# Kết Nối MySQL với PDO

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

try {
    // Tạo kết nối PDO
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);

    // Thiết lập chế độ lỗi PDO
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    echo "Kết nối thành công!";
} catch(PDOException $e) {
    echo "Kết nối thất bại: " . $e->getMessage();
}
?>
```

PDO sử dụng cơ chế xử lý ngoại lệ (exception) giúp quản lý lỗi tốt hơn.

# Tối Ưu Hóa Kết Nối Database

## 1 Sử Dụng Kết Nối Duy Nhất

Tạo một file riêng cho kết nối (ví dụ: db_connect.php) và include vào các file PHP khác khi cần, tránh tạo nhiều kết nối không cần thiết.

## 2 Đóng Kết Nối Khi Không Sử Dụng

Sử dụng $conn = null; (với PDO) hoặc $conn->close(); (với MySQLi) khi hoàn thành công việc để giải phóng tài nguyên.

## 3 Sử Dụng Prepared Statements

Giúp tăng hiệu suất khi thực hiện các truy vấn lặp lại và bảo vệ ứng dụng khỏi SQL Injection.

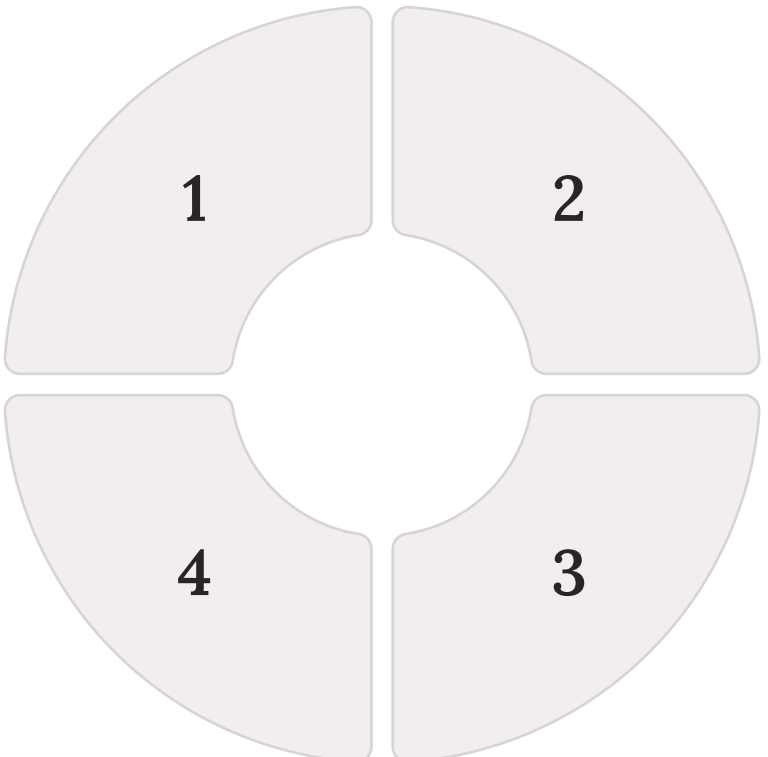
## 4 Xử Lý Lỗi Hiệu Quả

Sử dụng try-catch với PDO hoặc kiểm tra lỗi với MySQLi để phát hiện và xử lý vấn đề kết nối kịp thời.

# Thao Tác CRUD Trong PHP & MySQL

### Create (Tạo)

Thêm dữ liệu mới vào bảng bằng lệnh INSERT INTO




### Read (Đọc)

Truy xuất dữ liệu từ bảng bằng lệnh SELECT

### Delete (Xóa)

Loại bỏ dữ liệu khỏi bảng bằng lệnh DELETE

### Update (Cập nhật)

Sửa đổi dữ liệu đã có bằng lệnh UPDATE

# Create: Thêm Dữ Liệu vào MySQL

### Sử dụng MySQLi:

```
<?php
// Kết nối với database
include 'db_connect.php';

// Chuẩn bị dữ liệu
$username = "nguyenvan";
$password = password_hash("matkhau123",
            PASSWORD_DEFAULT);
$email = "nguyenvan@example.com";

// Câu lệnh SQL
$sql = "INSERT INTO users (username, password,
        email) VALUES ('$username', '$password',
        '$email')";

// Thực thi truy vấn
if (mysqli_query($conn, $sql)) {
    echo "Thêm thành công!";
} else {
    echo "Lỗi: " . mysqli_error($conn);
}

// Đóng kết nối
mysqli_close($conn);
?>
```

### Sử dụng PDO:

```
<?php
// Kết nối với database
include 'db_connect.php';

try {
 // Chuẩn bị dữ liệu
 $username = "nguyenvan";
 $password = password_hash("matkhau123",
 PASSWORD_DEFAULT);
 $email = "nguyenvan@example.com";

 // Sử dụng prepared statement
 $stmt = $conn->prepare("INSERT INTO users
 (username, password, email)
 VALUES (:username, :password, :email)");

 // Gắn giá trị cho các tham số
 $stmt->bindParam(':username', $username);
 $stmt->bindParam(':password', $password);
 $stmt->bindParam(':email', $email);

 // Thực thi truy vấn
 $stmt->execute();

 echo "Thêm thành công!";
} catch(PDOException $e) {
 echo "Lỗi: " . $e->getMessage();
}

// Đóng kết nối
$conn = null;
?>
```

# Read: Đọc Dữ Liệu từ MySQL

## Sử dụng MySQLi:

```
<?php
// Kết nối với database
include 'db_connect.php';

// Câu lệnh SQL
$sql = "SELECT id, username, email, created_at
        FROM users";
$result = mysqli_query($conn, $sql);

// Kiểm tra và hiển thị dữ liệu
if (mysqli_num_rows($result) > 0) {
    // Lặp qua từng bản ghi
    while($row = mysqli_fetch_assoc($result)) {
        echo "ID: " . $row["id"] . " - Tên: "
          . $row["username"] . " - Email: "
          . $row["email"] . "
";
    }
} else {
    echo "Không tìm thấy dữ liệu";
}

// Đóng kết nối
mysqli_close($conn);
?>
```

## Sử dụng PDO:

```
<?php
// Kết nối với database
include 'db_connect.php';

try {
 // Chuẩn bị và thực thi truy vấn
 $stmt = $conn->prepare("SELECT id, username,
 email, created_at FROM users");
 $stmt->execute();

 // Thiết lập chế độ lấy dữ liệu
 $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);

 // Lặp qua kết quả
 foreach($stmt->fetchAll() as $row) {
 echo "ID: " . $row["id"] . " - Tên: "
 . $row["username"] . " - Email: "
 . $row["email"] . "
";
 }
} catch(PDOException $e) {
 echo "Lỗi: " . $e->getMessage();
}

// Đóng kết nối
$conn = null;
?>
```

# Update: Cập Nhật Dữ Liệu trong MySQL

### Sử dụng MySQLi:

```
<?php
// Kết nối với database
include 'db_connect.php';

// Chuẩn bị dữ liệu
$id = 1;
$new_email = "email_moi@example.com";

// Câu lệnh SQL
$sql = "UPDATE users SET email='$new_email'
        WHERE id=$id";

// Thực thi truy vấn
if (mysqli_query($conn, $sql)) {
    echo "Cập nhật thành công!";
} else {
    echo "Lỗi: " . mysqli_error($conn);
}

// Đóng kết nối
mysqli_close($conn);
?>
```

### Sử dụng PDO:

```
<?php
// Kết nối với database
include 'db_connect.php';

try {
 // Chuẩn bị dữ liệu
 $id = 1;
 $new_email = "email_moi@example.com";

 // Sử dụng prepared statement
 $sql = "UPDATE users SET email = :email
 WHERE id = :id";
 $stmt = $conn->prepare($sql);

 // Gắn giá trị cho các tham số
 $stmt->bindParam(':email', $new_email);
 $stmt->bindParam(':id', $id);

 // Thực thi truy vấn
 $stmt->execute();

 echo "Cập nhật " . $stmt->rowCount() . " bản ghi";
} catch(PDOException $e) {
 echo "Lỗi: " . $e->getMessage();
}

// Đóng kết nối
$conn = null;
?>
```

# Delete: Xóa Dữ Liệu từ MySQL

### Sử dụng MySQLi:

```
<?php
// Kết nối với database
include 'db_connect.php';

// Chuẩn bị dữ liệu
$id = 1;

// Câu lệnh SQL
$sql = "DELETE FROM users WHERE id=$id";

// Thực thi truy vấn
if (mysqli_query($conn, $sql)) {
    echo "Xóa thành công!";
} else {
    echo "Lỗi: " . mysqli_error($conn);
}

// Đóng kết nối
mysqli_close($conn);
?>
```

### Sử dụng PDO:

```
<?php
// Kết nối với database
include 'db_connect.php';

try {
 // Chuẩn bị dữ liệu
 $id = 1;

 // Sử dụng prepared statement
 $sql = "DELETE FROM users WHERE id = :id";
 $stmt = $conn->prepare($sql);

 // Gắn giá trị cho tham số
 $stmt->bindParam(':id', $id);

 // Thực thi truy vấn
 $stmt->execute();

 echo "Xóa " . $stmt->rowCount() . " bản ghi";
} catch(PDOException $e) {
 echo "Lỗi: " . $e->getMessage();
}

// Đóng kết nối
$conn = null;
?>
```

# Nguy Cơ SQL Injection và Cách Phòng Tránh

## SQL Injection là gì?

SQL Injection là kỹ thuật tấn công phổ biến, khi kẻ tấn công chèn mã SQL độc hại vào các trường nhập liệu để thao túng cơ sở dữ liệu.

## Ví dụ SQL Injection:

```
// Code không an toàn
$username = $_POST['username']; // Giả sử = "' OR '1'='1";
$password = $_POST['password'];

$sql = "SELECT * FROM users WHERE username='$username'
        AND password='$password'";
// Kết quả: SELECT * FROM users WHERE username='' OR '1'='1'
// AND password='...'
// Điều này sẽ trả về TẤT CẢ người dùng!
```

## Cách phòng tránh:

1. Sử dụng Prepared Statements (cách tốt nhất)
2. Lọc dữ liệu đầu vào (mysqli_real_escape_string)
3. Kiểm tra kiểu dữ liệu và độ dài
4. Giới hạn quyền của tài khoản database
5. Bảo Mật Khi Truy Vấn

Prepared Statements là cách hiệu quả nhất để bảo vệ ứng dụng khỏi SQL Injection:

## Nguyên lý hoạt động:

1. Tách biệt lệnh SQL và dữ liệu người dùng
2. Chuẩn bị câu lệnh SQL với các tham số
3. Gắn giá trị vào các tham số
4. Thực thi câu lệnh đã chuẩn bị

## Lợi ích:

- Ngăn chặn SQL Injection hiệu quả
- Tăng hiệu suất khi thực hiện nhiều truy vấn tương tự
- Dễ dàng xử lý các kiểu dữ liệu khác nhau
- Mã nguồn rõ ràng, dễ bảo trì

# Xử Lý Form với PHP & MySQL

## Form HTML:

```
<!DOCTYPE html>
<html>
<head>
    <title>Đăng ký thành viên</title>
    <meta charset="UTF-8">
</head>
<body>
    <h2>Đăng ký thành viên mới</h2>
    <form action="register.php" method="post">
        <label for="username">Tên đăng nhập:</label>
        <input type="text" id="username"
            name="username" required><br><br>

        <label for="email">Email:</label>
        <input type="email" id="email"
            name="email" required><br><br>

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

        <input type="submit" value="Đăng ký">
    </form>
</body>
</html>
```

## Xử lý Form (register.php):

```
<?php
// Kết nối với database
include 'db_connect.php';

// Kiểm tra nếu form được submit
if ($_SERVER["REQUEST_METHOD"] == "POST") {
 // Lấy và làm sạch dữ liệu
 $username = htmlspecialchars($_POST['username']);
 $email = filter_var($_POST['email'],
 FILTER_SANITIZE_EMAIL);
 $password = password_hash($_POST['password'],
 PASSWORD_DEFAULT);

 try {
 // Chuẩn bị câu lệnh SQL
 $stmt = $conn->prepare("INSERT INTO users
 (username, email, password)
 VALUES (:username, :email, :password)");

 // Gắn giá trị
 $stmt->bindParam(':username', $username);
 $stmt->bindParam(':email', $email);
 $stmt->bindParam(':password', $password);

 // Thực thi
 $stmt->execute();

 echo "Đăng ký thành công!";
 } catch(PDOException $e) {
 echo "Lỗi: " . $e->getMessage();
 }

 // Đóng kết nối
 $conn = null;
}
?>
```

# Hiển Thị Dữ Liệu từ MySQL trong Bảng HTML

```
<?php
// Kết nối với database
include 'db_connect.php';

// Truy vấn dữ liệu
try {
 $stmt = $conn->prepare("SELECT id, username, email, created_at FROM users ORDER BY created_at
DESC");
 $stmt->execute();

 // Kiểm tra có dữ liệu không
 if ($stmt->rowCount() > 0) {
 echo "<table border='1'>";
 echo "<tr><th>ID</th><th>Tên đăng nhập</th><th>Email</th><th>Ngày tạo</th></tr>";

 // Hiển thị dữ liệu từng dòng
 while ($row = $stmt->fetch(PDO::FETCH_ASSOC)) {
 echo "<tr>";
 echo "<td>" . $row['id'] . "</td>";
 echo "<td>" . htmlspecialchars($row['username']) . "</td>";
 echo "<td>" . htmlspecialchars($row['email']) . "</td>";
 echo "<td>" . $row['created_at'] . "</td>";
 echo "</tr>";
 }

 echo "</table>";
 } else {
 echo "Không có dữ liệu để hiển thị";
 }
} catch(PDOException $e) {
 echo "Lỗi: " . $e->getMessage();
}

// Đóng kết nối
$conn = null;
?>
```

# Tìm Kiếm Dữ Liệu với PHP & MySQL

### Form tìm kiếm:

```
<!DOCTYPE html>
<html>
<head>
    <title>Tìm kiếm người dùng</title>
    <meta charset="UTF-8">
</head>
<body>
    <h2>Tìm kiếm người dùng</h2>
    <form action="" method="get">
        <input type="text" name="search"
               placeholder="Nhập tên hoặc email...">
        <input type="submit" value="Tìm kiếm">
    </form>

    <?php
    // Đoạn code PHP xử lý tìm kiếm sẽ đặt ở đây
    ?>
</body>
</html>
```

### Mã PHP xử lý tìm kiếm:

```
<?php
// Kết nối với database
include 'db_connect.php';

// Kiểm tra nếu có từ khóa tìm kiếm
if (isset($_GET['search']) && !empty($_GET['search'])) {
 $search = '%' . $_GET['search'] . '%';

 try {
 // Chuẩn bị câu lệnh SQL tìm kiếm
 $stmt = $conn->prepare("SELECT id, username, email
 FROM users
 WHERE username LIKE :search
 OR email LIKE :search");

 // Gắn giá trị tìm kiếm
 $stmt->bindParam(':search', $search);

// Thực thi truy vấn
$stmt->execute();

// Hiển thị kết quả
echo "<h3>Kết quả tìm kiếm:</h3>";
if ($stmt->rowCount() > 0) {
echo "<table border='1'>";
echo "<tr><th>ID</th><th>Tên</th><th>Email</th></tr>";

while ($row = $stmt->fetch(PDO::FETCH_ASSOC)) {
echo "<tr>";
echo "<td>" . $row['id'] . "</td>";
echo "<td>" . htmlspecialchars($row['username']) . "</td>";
echo "<td>" . htmlspecialchars($row['email']) . "</td>";
echo "</tr>";
}

 echo "</table>";
 } else {
 echo "Không tìm thấy kết quả nào.";
 }
 } catch(PDOException $e) {
 echo "Lỗi: " . $e->getMessage();
 }
}

// Đóng kết nối
$conn = null;
?>
```

# Phân Trang Dữ Liệu với PHP & MySQL

```
<?php
// Kết nối với database
include 'db_connect.php';

// Thiết lập thông số phân trang
$records_per_page = 10; // Số bản ghi mỗi trang

// Xác định trang hiện tại
$page = isset($_GET['page']) ? (int)$_GET['page'] : 1;
$offset = ($page - 1) * $records_per_page;

try {
 // Đếm tổng số bản ghi
 $stmt_count = $conn->prepare("SELECT COUNT(*) FROM users");
 $stmt_count->execute();
 $total_records = $stmt_count->fetchColumn();
 $total_pages = ceil($total_records / $records_per_page);

 // Lấy dữ liệu cho trang hiện tại
 $stmt = $conn->prepare("SELECT id, username, email, created_at
 FROM users ORDER BY id DESC LIMIT :offset, :records_per_page");
 $stmt->bindParam(':offset', $offset, PDO::PARAM_INT);
 $stmt->bindParam(':records_per_page', $records_per_page, PDO::PARAM_INT);
 $stmt->execute();

 // Hiển thị dữ liệu
 if ($stmt->rowCount() > 0) {
 echo "<table border='1'>";
 echo "<tr><th>ID</th><th>Tên</th><th>Email</th><th>Ngày tạo</th></tr>";

while ($row = $stmt->fetch(PDO::FETCH_ASSOC)) {
echo "<tr>";
echo "<td>" . $row['id'] . "</td>";
echo "<td>" . htmlspecialchars($row['username']) . "</td>";
echo "<td>" . htmlspecialchars($row['email']) . "</td>";
echo "<td>" . $row['created_at'] . "</td>";
echo "</tr>";
}

echo "</table>";

// Hiển thị các liên kết phân trang
echo "<div>";
for ($i = 1; $i <= $total_pages; $i++) {
if ($i == $page) {
echo "<span>[$i]</span> ";
} else {
 echo "<a href='?page=$i'>$i</a> ";
 }
 }
 echo "</div>";
 } else {
 echo "Không có dữ liệu để hiển thị";
 }
} catch(PDOException $e) {
 echo "Lỗi: " . $e->getMessage();
}

// Đóng kết nối
$conn = null;
?>
```

# Xây Dựng Hệ Thống Đăng Nhập

1

## Bước 1: Tạo Form Đăng Nhập

Thiết kế form HTML với trường username/email và password, kèm theo action trỏ đến trang xử lý PHP.

2

## Bước 2: Xác Thực Người Dùng

Kiểm tra thông tin đăng nhập từ form, so sánh với dữ liệu trong cơ sở dữ liệu sử dụng prepared statements.

3

## Bước 3: Tạo Phiên Đăng Nhập

Sử dụng SESSION để lưu trữ thông tin người dùng đã đăng nhập, thiết lập các biến phiên cần thiết.

4

## Bước 4: Kiểm Tra Trạng Thái Đăng Nhập

Thêm mã kiểm tra vào các trang cần bảo vệ để xác minh người dùng đã đăng nhập hay chưa.

5

## Bước 5: Tạo Chức Năng Đăng Xuất

Xây dựng trang đăng xuất để hủy phiên làm việc và đưa người dùng về trang đăng nhập.

# Ví Dụ Mã Xử Lý Đăng Nhập

### Form đăng nhập (login.php):

```
<!DOCTYPE html>
<html>
<head>
   <title>Đăng nhập</title>
   <meta charset="UTF-8">
</head>
<body>
   <h2>Đăng nhập hệ thống</h2>

   <?php
   // Hiển thị thông báo lỗi nếu có
   if (isset($_GET['error'])) {
      echo "<p style='color: red;'>";
      echo htmlspecialchars($_GET['error']);
      echo "</p>";
   }
   ?>

   <form action="process_login.php" method="post">
      <label for="username">Tên đăng nhập:</label>
      <input type="text" id="username"
           name="username" required><br><br>

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

      <input type="submit" value="Đăng nhập">
   </form>
</body>
</html>
```

### Xử lý đăng nhập (process_login.php):

```
<?php
// Khởi động phiên làm việc
session_start();

// Kết nối với database
include 'db_connect.php';

// Kiểm tra nếu form được submit
if ($_SERVER["REQUEST_METHOD"] == "POST") {
 // Lấy dữ liệu từ form
 $username = $_POST['username'];
 $password = $_POST['password'];

 try {
 // Tìm người dùng trong CSDL
 $stmt = $conn->prepare("SELECT id, username, password
 FROM users WHERE username = :username");
 $stmt->bindParam(':username', $username);
 $stmt->execute();

 if ($stmt->rowCount() == 1) {
 // Lấy thông tin người dùng
 $user = $stmt->fetch(PDO::FETCH_ASSOC);

 // Xác thực mật khẩu
 if (password_verify($password, $user['password'])) {
 // Mật khẩu đúng, lưu thông tin vào session
 $_SESSION['loggedin'] = true;
 $_SESSION['id'] = $user['id'];
 $_SESSION['username'] = $user['username'];

 // Chuyển hướng đến trang chủ
 header("location: dashboard.php");
 exit;
 } else {
 // Mật khẩu không đúng
 header("location: login.php?error=Mật khẩu không đúng");
 exit;
 }
 } else {
 // Người dùng không tồn tại
 header("location: login.php?error=Tên đăng nhập không tồn tại");
 exit;
 }
 } catch(PDOException $e) {
 header("location: login.php?error=Lỗi hệ thống: " . $e->getMessage());
 exit;
 }
}
?>
```

# Bảo Vệ Trang Yêu Cầu Đăng Nhập

```
<?php
// Bắt đầu phiên làm việc (luôn đặt ở đầu file)
session_start();

// Kiểm tra người dùng đã đăng nhập chưa
if (!isset($_SESSION['loggedin']) || $_SESSION['loggedin'] !== true) {
 // Nếu chưa đăng nhập, chuyển hướng về trang đăng nhập
 header('location: login.php');
 exit;
}

// Nếu đã đăng nhập, hiển thị nội dung trang
?>

<!DOCTYPE html>
<html>
<head>
 <title>Trang Quản Trị</title>
 <meta charset="UTF-8">
</head>
<body>
 <h2>Chào mừng, <?php echo htmlspecialchars($_SESSION['username']); ?>!</h2>

 <div>
 <h3>Thông tin tài khoản của bạn:</h3>
 <p>ID: <?php echo $_SESSION['id']; ?></p>
 <p>Tên đăng nhập: <?php echo htmlspecialchars($_SESSION['username']); ?></p>
 </div>

 <p><a href="logout.php">Đăng xuất</a></p>
</body>
</html>
```

# Xử Lý Đăng Xuất

```
<?php
// Bắt đầu phiên làm việc
session_start();

// Hủy tất cả các biến session
$_SESSION = array();

// Hủy cookie phiên làm việc
if (ini_get("session.use_cookies")) {
    $params = session_get_cookie_params();
    setcookie(session_name(), '', time() - 42000,
        $params["path"], $params["domain"],
        $params["secure"], $params["httponly"]
    );
}

// Hủy phiên làm việc
session_destroy();

// Chuyển hướng về trang đăng nhập
header("location: login.php");
exit;
?>
```

Đăng xuất an toàn là việc không chỉ đơn giản xóa biến session mà còn cần hủy cookie phiên và hoàn toàn hủy phiên làm việc để ngăn ngừa các lỗ hổng bảo mật tiềm ẩn.