# PHP & MYSQL

# Kết nối đến MySQL dùng PHP

1. *MySQLi* extension: là một module mở rộng, cho phép truy cập đến các tính năng được cung cấp bởi MySQL 4.1 trở lên, và dùng thay thế cho *mysql Extension*
   - Giao diện lập trình hướng đối tượng, và cả kiểu thủ tục
   - Hỗ trợ đa truy vấn
   - Hỗ trợ Transactions
   - Tăng cường khả năng debug
   - Chỉ dùng để làm việc với MySQL
2. *PDO*: PHP Data Objects
   - Hỗ trợ làm việc với 11 hệ cơ sở dữ liệu khác ngoài MySQL
   - Dễ dàng chuyển đổi cơ sở dữ liệu khi cần mà không phải thay đổi mã PHP quá nhiều
   - Giao diện lập trình hướng đối tượng

# Kết nối đến MySQL dùng PHP (tt)

- Các bước kết nối và truy vấn cơ sở dữ liệu MySQL dùng ***mysqli***:
  1. Tạo kết nối:
     a) $conn = mysqli_connect($hostname, $username, $password);
     b) $mysqli = new mysqli($hostname, $username, $password);
  2. Chọn cơ sở dữ liệu làm việc:
     a) mysqli_select_db($conn, $db_name);
     b) $mysqli->select_db($db_name);
  3. Thực hiện câu truy vấn SQL:
     a) $result = mysqli_query ( $conn, $query);
     b) $result = $mysqli->query($query);
  4. Lấy 1 dòng kết quả trả về:
     a) $row = mysqli_fetch_assoc ($result);
     b) $row = $ result->fetch_assoc ();
  5. Đọc giá trị một trường:
     $val = $row["col-name"];

# Kết nối đến MySQL dùng PHP (tt)

- Giải phóng tài nguyên kết quả:
  a) mysqli_free_result($result);
  b) $result->close();
- Đóng kết nối:
  a) mysqli_close($conn );
  b) $mysqli->close();

# Kết nối đến MySQL dùng PHP - VD

- Tài khoản kết nối hệ cơ sở dữ liệu MySQL: *user name: root, password: admin*, được cài đặt cục bộ tại máy tính người dùng
- Tạo một kết nối đến hệ cơ sở dữ liệu trên:

```php
<?php
 $mysqli = new mysqli("127.0.0.1", "root", "admin");

 if ($mysqli->connect_errno) {
     printf("Connect failed: %s\n", $mysqli->connect_error);
    exit();
 } else {
     printf("Host information: %s\n", $mysqli->host_info);
     $mysqli->close();
 }
?>
```

# Kết nối đến MySQL dùng PHP - VD

- Cho một cơ sở dữ liệu **addressbook** với các bảng sau:

| Tên bảng | Tên trường |
|---|---|
| master_name | id, date_added, date_modified, f_name, l_name |
| Address | id, master_id, date_added, date_modified, address, city, state, zipcode, type |
| Telephone | id, master_id, date_added, date_modified, tel_number, type |
| Fax | id, master_id, date_added, date_modified, fax_number, type |
| Email | id, master_id, date_added, date_modified, email, type |
| personal_notes | id, master_id, date_added, date_modified, note |

# Kết nối đến MySQL dùng PHP - VD

- Script tạo bảng **master_name**:

```
<?php
$mysqli = new mysqli("127.0.0.1", "root", "admin","addressbook");
if ($mysqli->connect_errno) {
      printf("Connect failed: %s\n", $mysqli->connect_error);
     exit();
 } else {
      $sql = "CREATE TABLE master_name (
                id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
                date_added DATETIME,
                date_modified DATETIME,
                f_name VARCHAR (75),
                l_name VARCHAR (75));";
     $res = $mysqli->query($sql);
     if ($res === TRUE) {
             echo "Table master_name successfully created.";
     } else {
         printf("Could not create table: %s\n",$mysqli->error);
     }
     $mysqli->close();
}
?>
```

# Kết nối đến MySQL dùng PHP - VD

- Script thêm dữ liệu vào bảng **master_name**:

```php
<?php
$mysqli = new mysqli("127.0.0.1", "root", "admin","addressbook");
if ($mysqli->connect_errno) {
     printf("Connect failed: %s\n", $mysqli->connect_error);
    exit();
 } else {
     $sql = "INSERT INTO master_name (date_added, date_modified,
                    f_name, l_name) VALUES (now(), now(),
                    'Diane', 'Murphy')";
     $res = $mysqli->query($sql);

     if ($res === TRUE) {
            echo "A record has been inserted.";
     } else {
         printf("Could not insert record: %s\n",$mysqli->error);
     }
     $mysqli->close();
 }
?>
```

# Kết nối đến MySQL dùng PHP - VD

- Script hiển thị dữ liệu trong bảng **master_name**:

```
$mysqli = new mysqli("127.0.0.1","root","admin","addressbook");
if ($mysqli->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}else{
    $sql = "select * from master_name";
    $res = $mysqli->query($sql);
    if($res){
        $columns = array("id","date_added","date_modified","f_name","l_name");
        echo "<table>";
        echo "<tr>";
        foreach($columns as $column)  echo "<th>".$column."</th>";
        echo "</tr>";
        while($row = $res->fetch_assoc()){
            echo "<tr>";
            foreach($columns as $col)  echo "<td>".$row[$col]."</td>";
            echo "</tr>";}
        echo "</table>";}}
    $mysqli->close();
```

# Hiển thị thông báo lỗi

- Hai cách để hiển thị các thông báo lỗi khi thực thi PHP script
  - Thêm vào đoạn mã sau:

```
ini_set('display_errors', 1);
```

  - Hoặc thay đổi cấu hình trong file PHP.ini với dòng chỉ thị sau:

```
display_errors=on
```

# SQL Injection

- Là một kỹ thuật chèn mã có thể phá hủy cơ sở dữ liệu
- Là một trong những kỹ thuật hack web phổ biến nhất.
- Là việc đặt mã độc vào câu lệnh SQL thông qua đầu vào trang web.

# SQL Injection

- Việc chèn SQL thường xảy ra khi yêu cầu người dùng nhập dữ liệu, chẳng hạn như tên/userid của họ và thay vì tên/id, người dùng sẽ cung cấp một câu lệnh SQL mà sẽ vô tình chạy trên cơ sở dữ liệu của ứng dụng
- Một số kỹ thuật SQL Injection phổ biến:
  - SQL Injection dựa trên 1=1 luôn đúng
  - SQL Injection dựa trên ""="" luôn đúng
  - SQL Injection dựa trên các câu lệnh SQL hàng loạt (Batched SQL Statements)

# SQL Injection dựa trên 1=1 luôn đúng

- **Ví dụ:**
  - Đoạn mã PHP sau lấy ra thông tin của người dùng tương ứng với một userId cụ thể được người dùng nhập vào từ html form

```
$txtUserId = $_POST["UserId"];
$txtSQL = "SELECT * FROM Users WHERE UserId = ".$txtUserId;
```

  - Tuy nhiên người dùng có thể nhập vào chuỗi sau: 105 OR 1=1
  - Hệ quả ta có câu lệnh SQL sau sẽ thực thi trên hệ thống:

```
SELECT * FROM Users WHERE UserId = 105 OR 1=1;
```

  - Câu lệnh này hợp lệ và sẽ lấy về tất cả dữ liệu từ bảng Users, nếu bảng Users có chứa các thông tin nhạy cảm như usernames và passwords sẽ dẫn đến các dữ liệu này bị rò rỉ.

# SQL Injection dựa trên ""="" luôn đúng

- **Ví dụ:**
  - Đoạn mã PHP sau lấy ra thông tin của người dùng tương ứng điều kiện được tạo thành từ hai tham số được người dùng nhập vào từ html form

```
$uName = $_POST["username"];
$uPass = $_POST["userpassword"];
$sql = 'SELECT * FROM Users WHERE Name ="'.$uName.'" AND Pass ="'.$uPass.'"'
```

  - Tuy nhiên người dùng có thể nhập vào chuỗi sau cho cả hai trường username và userpassword trên html form: " or ""="
  - Hệ quả ta có câu lệnh SQL sau sẽ thực thi trên hệ thống:

```
SELECT * FROM Users WHERE Name ="" or ""="" AND Pass ="" or ""=""
```

  - Câu lệnh này hợp lệ và sẽ lấy về tất cả dữ liệu từ bảng Users

# SQL Injection dựa trên các câu lệnh SQL hàng loạt

- Hầu hết các cơ sở dữ liệu đều hỗ trợ thực thi nhiều câu lệnh SQL trên cùng một dòng
  - Một loạt câu lệnh SQL là một nhóm gồm hai câu lệnh SQL trở lên, được phân tách bằng dấu chấm phẩy.
- **Ví dụ:**
  - Ta có đoạn mã PHP sau:

```
$txtUserId = $_POST["UserId"];
$txtSQL = "SELECT * FROM Users WHERE UserId = " + $txtUserId;
```

  - Người dùng có thể nhập vào chuỗi sau: 105; DROP TABLE Suppliers
  - Hệ quả ta có câu lệnh SQL sau sẽ thực thi trên hệ thống:

```
SELECT * FROM Users WHERE UserId = 105; DROP TABLE Suppliers;
```

  - Câu lệnh này hợp lệ và sẽ xoá đi bảng Suppliers

# Giải pháp ngăn ngừa SQL Injection

- Để ngăn ngừa SQL injection ta dùng các câu lệnh SQL được chuẩn bị trước (Prepared statements)
  - Cả MySQLi và PDO đều hỗ trợ kỹ thuật này

# Prepared statements

- Câu lệnh được chuẩn bị là một tính năng được sử dụng để thực thi lặp đi lặp lại các câu lệnh SQL giống nhau (hoặc tương tự) với hiệu suất cao.
- Các câu lệnh được chuẩn bị về cơ bản hoạt động như sau:
  1. **Bước chuẩn bị**: Một mẫu câu lệnh SQL được tạo và gửi đến cơ sở dữ liệu. Một số giá trị nhất định không được chỉ định, được gọi là tham số (được gắn nhãn "?"). Ví dụ: INSERT INTO MyGuests VALUES(?, ?, ?)
  2. Cơ sở dữ liệu phân tích cú pháp, biên dịch và thực hiện tối ưu hóa truy vấn trên mẫu câu lệnh SQL và lưu trữ kết quả mà không cần thực thi nó
  3. **Thực thi**: Sau đó, ứng dụng liên kết các giá trị với các tham số và cơ sở dữ liệu sẽ thực thi câu lệnh. Ứng dụng có thể thực thi câu lệnh bao nhiêu lần tùy thích với các giá trị khác nhau

# Prepared statements

- So với việc thực thi trực tiếp các câu lệnh SQL, các câu lệnh được chuẩn bị sẵn có ba ưu điểm chính:
  1. Các câu lệnh được chuẩn bị sẵn giúp giảm thời gian phân tích cú pháp vì việc chuẩn bị truy vấn chỉ được thực hiện một lần
  2. Các tham số bị ràng buộc sẽ giảm thiểu băng thông đến máy chủ vì bạn chỉ cần gửi các tham số mỗi lần chứ không phải toàn bộ truy vấn
  3. Các câu lệnh được chuẩn bị sẵn rất hữu ích trong việc chống lại việc chèn SQL, bởi vì các giá trị tham số được truyền sau bằng một giao thức khác, không cần phải được "escape" một cách chính xác. Nếu mẫu câu lệnh gốc không được lấy từ đầu vào bên ngoài thì việc chèn SQL không thể xảy ra.

# Prepared statements

- Cú pháp sử dụng Prepared statements trong PHP với MySQLi:
  - Chèn một dấu chấm hỏi (?) vào nơi muốn thay thế bằng một giá trị integer, string, double hoặc blob. VD:

    "INSERT INTO MyGuests (firstname, lastname, email) VALUES (?, ?, ?)"
  - Sau đó, dùng hàm bind_param() để liên kết các tham số vào câu lệnh SQL đã được chuẩn bị và báo cho cơ sở dữ liệu biết các tham số đó là gì. VD:

    $stmt->bind_param("sss", $firstname, $lastname, $email);
    - Đối số "sss" liệt kê các loại dữ liệu chứa tham số. Ký tự s cho mysql biết rằng tham số là một chuỗi.
    - Đối số có thể là một trong bốn loại:
      - i - integer
      - d - double
      - s - string
      - b - BLOB

# VD sử dụng Prepared statements

```php
$mysqli = new mysqli("127.0.0.1", "root", "", "addressbook");
if ($mysqli->connect_error) {
    die("Connection failed: " . $mysqli->connect_error);
  }else{
    $stmt = $mysqli->prepare("INSERT INTO master_name (date_added, date_modified, f_name, l_name)
            VALUES (?,?,?,?)");
    $stmt->bind_param("ssss", $date_value, $date_value, $f_name, $l_name);

    $date_value = date("Y/m/d H/i/s");
    $f_name = 'A';
    $l_name = 'Nguyen';
    $stmt->execute();

    $date_value = date("Y/m/d H/i/s");
    $f_name = 'B';
    $l_name = 'Tran';
    $stmt->execute();

    echo "New records created successfully";
    $stmt->close();
    $mysqli->close();
}
```

# Question ?

# THANK YOU !