BÀI 9:
Lập trình CSDL với PHP

Mục tiêu bài học

- Giới thiệu về PDO: Ý nghĩa, cú pháp, cách sử dụng
- Hướng dẫn các bước để kết nối đến CSDL MySQL sử dụng PDO
- Ý nghĩa của các lớp/interface trong PDO
- Sử dụng tham số với PDO
- Thực hành kết nối và thao tác với dữ liệu bằng PDO

Giới thiệu về PDO

- Từ phiên bản PHP 5 trở lên, ta có thể kết nối với CSDL MySQL bằng cách sử dụng một trong 2 thư viện sau:
  - **MySQLi extension** (chữ cái "i" đại diện cho improved)
  - **PDO (PHP Data Objects)**
- PDO là một thư viện PHP cho phép chuẩn hóa việc kết nối với CSDL bằng PHP. PDO cho phép các nhà phát triển có thể viết mã portable đối với nhiều DBMS khác nhau.
- *PHP Data Objects* (PDO) định nghĩa một giao diện nhất quán, lightweight cho việc truy cập vào các loại cơ sở dữ liệu trong PHP.
- PDO là thư viện truy cập cơ sở dữ liệu trong PHP, nó cung cấp một hệ thống API duy nhất giúp kết nối với nhiều cơ sở dữ liệu khác nhau như MySQL, SQL Server ...
- Điểm mạnh của của PDO so với việc sử dụng các extension khác là khi ta muốn chuyển đổi giữa các loại cơ sở dữ liệu khác nhau thì ta không phải sửa đổi code mà ứng dụng vẫn có thể hoạt động bình thường.

Giới thiệu về PDO

- So sánh giữa MySQLi và PDO:
  - PDO cho phép làm việc với 12 DBMS, trong khi MySQLi chỉ làm việc với cơ sở dữ liệu MySQL.
  - Vì vậy, khi ta có nhu cầu muốn chuyển dự án sang làm việc với một CSDL khác, với PDO ta có thể chuyển đổi một cách dễ dàng. Ta chỉ cần thay đổi chuỗi kết nối và một số đoạn mã. Nhưng với MySQLi, ta cần viết lại toàn bộ mã truy vấn trong dự án.
  - Cả hai thư viện đều hướng đối tượng, nhưng MySQLi hỗ trợ cú pháp hướng thủ tục.
  - Cả hai thư viện đều hỗ trợ Prepared Statement.

Giới thiệu về PDO

- PDO được giới thiệu cùng với PHP 5, và sử dụng cú pháp hướng đối tượng. Vì vậy PDO sẽ không hoạt động đối với các phiên bản PHP trước đó.

Database Support

- PDO cho phép kết nối và thao tác với bất kỳ CSDL nào mà có PDO driver. Dưới đây là một số database drivers mà PDO hỗ trợ:
  - PDO_DBLIB (FreeTDS / Microsoft SQL Server / Sybase)
  - PDO_FIREBIRD (Firebird/Interbase 6)
  - PDO_IBM (IBM DB2)
  - PDO_INFORMIX (IBM Informix Dynamic Server)
  - PDO_MYSQL (MySQL 3.x/4.x/5.x)
  - PDO_OCI (Oracle Call Interface)
  - PDO_ODBC (ODBC v3 (IBM DB2, unixODBC and win32 ODBC))
  - PDO_PGSQL (PostgreSQL)
  - PDO_SQLITE (SQLite 3 and SQLite 2)
  - PDO_4D (4D)
- Để hiển thị danh sách các driver có sẵn, ta có thể sử dụng cú pháp như sau:

```
print_r(PDO::getAvailableDrivers());
```

Kết nối với MySQL

- Để thiết lập kết nối đến CSDL, ta khởi tạo một đối tượng của lớp PDO. Ta luôn sử dụng tên lớp PDO mà không quan trọng sử dụng driver nào.
- Constructor sẽ nhận các tham số thể hiện database source , ngoài ra còn có username và password.

```php
<?php
$dbh = new PDO('mysql:host=localhost;dbname=test', $user, $pass);
?>
```

- Nếu có lỗi xảy ra, một đối tượng *PDOException* sẽ được ném ra. Ta có thể bắt và xử lý lỗi xảy ra trong quá trình kết nối đến CSDL.

Kết nối với MySQL

- Ngay khi kết nối thành công đến CSDL, một đối tượng của lớp PDO sẽ được khởi tạo và trả về cho chương trình. Kết nối sẽ được duy trì trong suốt thời gian tồn tại của đối tượng PDO vừa được tạo.
- Ví dụ:

```php
<?php
try {
    $dbh = new PDO('mysql:host=localhost;dbname=test', $user, $pass);
    foreach($dbh->query('SELECT * from FOO') as $row) {
        print_r($row);
    }
    $dbh = null;
} catch (PDOException $e) {
    print "Error!: " . $e->getMessage() . "<br/>";
    die();
}
?>
```

Đóng kết nối

- Để đóng kết nối với CSDL, ta gán đối tượng PDO = null.
- Nếu ta không thực hiện việc gán này, PHP sẽ tự động đóng kết nối khi đoạn mã kịch bản kết thúc.
- Ví dụ:

```php
<?php
$dbh = new PDO('mysql:host=localhost;dbname=test', $user, $pass);
// use the connection here
$sth = $dbh->query('SELECT * FROM foo');

// and now we're done; close it
$sth = null;
$dbh = null;
?>
```

Prepared statements

- Là một class trong thư viện PDO, dùng để thực thi các câu lệnh SQL cho phép thao tác và cập nhật dữ liệu trong CSDL, cho phép sử dụng các parameter trong các câu SQL.
- Prepared statements cung cấp những lợi ích sau:
  - Câu truy vấn chỉ cần parsed (prepared) một lần duy nhất, nhưng có thể được thực thi nhiều lần với cùng hoặc tham số khác nhau. Khi câu truy vấn được prepared, CSDL sẽ phân tích, biên dịch và tối ưu hóa việc thực thi. Đối với những câu truy vấn phức tạp, quá trình này có thể sẽ mất nhiều thời gian, khiến ứng dụng bị chậm, nếu ta có sử dụng các tham số khác nhau cho cùng một câu truy vấn.
  - Bằng cách sử dụng một prepared statement, ứng dụng sẽ tránh được việc phải lặp lại quá trình analyze/compile/optimize nói trên. Prepared statements sẽ sử dụng ít tài nguyên hơn và thực thi nhanh hơn.
  - Các tham số được sử dụng trong prepared statements không cần phải đặt trong dấu '', driver sẽ tự động xử lý. Prepared statements sẽ giúp ứng dụng tránh bị lỗi SQL injection.

Prepared statements

- Để sử dụng tham số trong prepared statement, ta cần chèn các bộ giữ chỗ (placeholders) vào câu lệnh SQL.
- Có 2 loại placeholders
  - unnamed placeholders
  - named placeholders
- Có 3 cú pháp để truyền dữ liệu vào cho câu SQL trong prepared statement như sau:

```
# no placeholders - ripe for SQL Injection!
$STH = $DBH->("INSERT INTO folks (name, addr, city) values ($name, $addr, $city)");

# unnamed placeholders
$STH = $DBH->("INSERT INTO folks (name, addr, city) values (?, ?, ?);

# named placeholders
$STH = $DBH->("INSERT INTO folks (name, addr, city) value (:name, :addr, :city)");
```

Ví dụ

- Dưới đây là ví dụ thực thi câu lệnh INSERT bằng cách sử dụng named placeholders.

```php
<?php
$stmt = $dbh->prepare("INSERT INTO REGISTRY (name, value) VALUES (:name, :value)");
$stmt->bindParam(':name', $name);
$stmt->bindParam(':value', $value);

// insert one row
$name = 'one';
$value = 1;
$stmt->execute();

// insert another row with different values
$name = 'two';
$value = 2;
$stmt->execute();
?>
```

Ví dụ

- Dưới đây là ví dụ thực thi câu lệnh INSERT bằng cách sử dụng unnamed placeholders (?).

```php
<?php
$stmt = $dbh->prepare("INSERT INTO REGISTRY (name, value) VALUES (?, ?)");
$stmt->bindParam(1, $name);
$stmt->bindParam(2, $value);

// insert one row
$name = 'one';
$value = 1;
$stmt->execute();

// insert another row with different values
$name = 'two';
$value = 2;
$stmt->execute();
?>
```

Lớp PDOStatement

- Thể hiện một prepared statement, cho phép thực thi các câu lệnh SQL, và cho phép duyệt qua result set nếu có.
- Danh sách các phương thức của lớp PDOStatement:

Phương thức	Ý nghĩa
Execute()	Thực thi một prepared statement
Fetch()	Fetch dòng kế tiếp từ một result set
fetchAll()	Trả về một mảng chứa tất cả các dòng từ result set
fetchColumn()	Trả lại một column từ dòng kế tiếp trong result set
rowCount()	Trả về tổng số dòng được tác động bởi câu lệnh SQL cuối cùng
setFetchMode()	Thiết lập chế độ fetch của prepared statement
fetchObject()	Fetch dòng kế tiếp và trả lại dưới dạng một object
bindParam()	Gắn kết (bind) một tham số với một tên biến
bindValue()	Gắn kết (bind) một giá trị với một tham số
columnCount()	Trả về tổng số cột trong result set

Demo

- Giáo viên demo các bước kết nối với CSDL bằng PDO

Xử lý ngoại lệ với PDO

- PDO hỗ trợ cơ chế xử lý ngoại lệ trong quá trình kết nối và thao tác với CSDL, vì vậy các đoạn mã thao tác với CSDL nên được đặt trong khối try/catch.
- Ta có thể thiết lập thuộc tính chế độ xử lý lỗi trong PDP bằng cách gọi phương thức setAttribute() như hình bên dưới.
- Ví dụ:

```
$DBH->setAttribute( PDO::ATTR_ERRMODE, PDO::ERRMODE_SILENT );
$DBH->setAttribute( PDO::ATTR_ERRMODE, PDO::ERRMODE_WARNING );
$DBH->setAttribute( PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION );
```

Xử lý ngoại lệ với PDO

- **PDO::ERRMODE_SILENT**
  - Đây là chế độ lỗi mặc định.
- **PDO::ERRMODE_WARNING**
  - Chế độ này sẽ tung ra một cảnh báo PHP (PHP warning), và cho phép chương trình vẫn tiếp tục được thực thi. Chế độ này hữu ích cho quá trình gỡ lỗi.
- **PDO::ERRMODE_EXCEPTION**
  - Đây là chế độ được sử dụng trong hầu hết các tình huống. Chế độ này cho phép phát sinh ngoại lệ, xử lý ngoại lệ.

Xử lý ngoại lệ với PDO

- Ví dụ:

```
# connect to the database
try {
  $DBH = new PDO("mysql:host=$host;dbname=$dbname", $user, $pass);
  $DBH->setAttribute( PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION );

  # UH-OH! Typed DELECT instead of SELECT!
  $DBH->prepare('DELECT name FROM people');
}
catch(PDOException $e) {
    echo "I'm sorry, Dave. I'm afraid I can't do that.";
    file_put_contents('PDOErrors.txt', $e->getMessage(), FILE_APPEND);
}
```

Ví dụ tạo database

- Ví dụ về mã nguồn tạo database trong PDO:

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

try {
    $conn = new PDO("mysql:host=$servername", $username, $password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $sql = "CREATE DATABASE myDBPDO";
    // use exec() because no results are returned
    $conn->exec($sql);
    echo "Database created successfully<br>";
    }
catch(PDOException $e)
    {
    echo $sql . "<br>" . $e->getMessage();
    }

$conn = null;
?>
```

Tạo bảng

- Đoạn mã minh họa tạo bảng trong PDO:

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

try {
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    // sql to create table
    $sql = "CREATE TABLE MyGuests (
    id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    firstname VARCHAR(30) NOT NULL,
    lastname VARCHAR(30) NOT NULL,
    email VARCHAR(50),
    reg_date TIMESTAMP
    )";

    // use exec() because no results are returned
    $conn->exec($sql);
    echo "Table MyGuests created successfully";
    }
catch(PDOException $e)
    {
    echo $sql . "<br>" . $e->getMessage();
    }

$conn = null;
?>
```

Demo

- Giáo viên demo các thao tác với CSDL bằng PDO

Tổng kết

- Giới thiệu về PDO: Ý nghĩa, cú pháp, cách sử dụng
- Hướng dẫn các bước để kết nối đến CSDL MySQL sử dụng PDO
- Ý nghĩa của các lớp/interface trong PDO
- Sử dụng tham số với PDO
- Thực hành kết nối và thao tác với dữ liệu bằng PDO