BÀI 10:
Thao tác dữ liệu với PDO

Mục tiêu bài học

- Truy vấn và xử lý dữ liệu với PDO
- Gọi thủ tục lưu với PDO
- Thực hành thao tác dữ liệu với PDO
- Thực hành chức năng phân trang – đăng nhập với PDO

Truy vấn dữ liệu

- Trong quá trình thao tác dữ liệu với database, dữ liệu được đọc thông qua việc gọi phương thức fetch() của đối tượng statement. Trước khi gọi phương thức fetch(), ta cần biết cách thức mà PDO xử lý dữ liệu từ database.

- Có 3 chế độ fetch được dùng nhiều nhất: FETCH_ASSOC, FETCH_CLASS, and FETCH_OBJ. Để thiết lập chế độ fetch, ta sử dụng phương thức setFetchMode() như sau:

```
$STH->setFetchMode(PDO::FETCH_ASSOC);
```

Truy vấn dữ liệu

- Danh sách các hằng số để thiết lập chế độ fetch:
  - **PDO::FETCH_ASSOC:** Trả về một mảng được đánh chỉ số bằng tên cột.
  - **PDO::FETCH_BOTH (default):** Trả về một mảng được đánh chỉ số bằng cả tên cột và số thứ tự.
  - **PDO::FETCH_BOUND:** Cho phép gắn kết giá trị của các cột với các biến thông qua phương thức bindColumn().
  - **PDO::FETCH_CLASS:** Gán giá trị của các cột với các thuộc tính của class.
  - **PDO::FETCH_INTO:** Cập nhật một thể hiện đã tồn tại của một class.
  - **PDO::FETCH_LAZY**: Kết hợp giữa các chế độ PDO::FETCH_BOTH/PDO::FETCH_OBJ, tạo ra các tên biến đối tượng
  - **PDO::FETCH_NUM:** Trả về một mảng được đánh chỉ số bằng số cột.
  - **PDO::FETCH_OBJ:** Trả về một anonymous object với các tên thuộc tính ứng với tên các cột trong bảng.

FETCH_ASSOC

- Kiểu fetch này sẽ tạo nên một associative array, được đánh chỉ mục ở tên cột. Ta nên sử dụng kiểu fetch này, vì nó quen thuộc với những người đã từng sử dụng thư viện mysql/mysqli extensions.
- Ví dụ:

```
# using the shortcut ->query() method here since there are no variable
# values in the select statement.
$STH = $DBH->query('SELECT name, addr, city from folks');

# setting the fetch mode
$STH->setFetchMode(PDO::FETCH_ASSOC);

while($row = $STH->fetch()) {
    echo $row['name'] . "\n";
    echo $row['addr'] . "\n";
    echo $row['city'] . "\n";
}
```

- Trong đoạn code trên, vòng lặp while sẽ thực thi và mỗi lần lặp sẽ đọc dòng tiếp theo trong result set cho đến khi kết thúc.

FETCH_OBJ

- Kiểu fetch này sẽ tạo ra một đối tượng của class khi đọc từng dòng từ result-set.
- Ví dụ:

```
# creating the statement
$STH = $DBH->query('SELECT name, addr, city from folks');

# setting the fetch mode
$STH->setFetchMode(PDO::FETCH_OBJ);

# showing the results
while($row = $STH->fetch()) {
    echo $row->name . "\n";
    echo $row->addr . "\n";
    echo $row->city . "\n";
}
```

FETCH NUM

- PDO::FETCH_NUM cung cấp một chỉ số cho từng cột thay cho tên cột.
- Ví dụ:

```php
/*** The SQL SELECT statement ***/
$sql = "SELECT * FROM animals";

/*** fetch into an PDOStatement object ***/
$stmt = $dbh->query($sql);

/*** echo number of columns ***/
$result = $stmt->fetch(PDO::FETCH_NUM);

/*** loop over the object directly ***/
foreach($result as $key=>$val)
{
echo $key.' - '.$val.'<br />';
}

/*** close the database connection ***/
$dbh = null;
```

- Kết quả:

```
0 - 1
1 - emu
2 - bruce
```

FETCH BOTH

- Đây là kiểu fetch trong đó ta có thể truy cập đến các cột trong resultset theo chỉ số cột hoặc tên cột.
- Ví dụ:

```php
$dbh = new PDO("mysql:host=$hostname;dbname=animals", $username, $password);
/*** echo a message saying we have connected ***/
echo 'Connected to database<br />';

/*** The SQL SELECT statement ***/
$sql = "SELECT * FROM animals";

/*** fetch into an PDOStatement object ***/
$stmt = $dbh->query($sql);

/*** echo number of columns ***/
$result = $stmt->fetch(PDO::FETCH_BOTH);

/*** loop over the object directly ***/
foreach($result as $key=>$val)
{
echo $key.' - '.$val.'<br />';
}
```

- Kết quả:

```
Connected to database
animal_id - 1
0 - 1
animal_type - emu
1 - emu
animal_name - bruce
2 - bruce
```

FETCH CLASS

- Chế độ PDO::FETCH_CLASS sẽ khởi tạo một thể hiện mới của class. Các tên trường sẽ được map với các thuộc tính của class. Điều này sẽ giúp giảm bớt code và tốc độ sẽ được cải thiện khi quá trình mapping sẽ được xử lý ngầm.
- Ví dụ:

```php
$dbh = new PDO("mysql:host=$hostname;dbname=animals", $username, $password);

/*** echo a message saying we have connected ***/

echo 'Connected to database<br />';


/*** The SQL SELECT statement ***/

$sql = "SELECT * FROM animals";


/*** fetch into an PDOStatement object ***/

$stmt = $dbh->query($sql);


/*** fetch into the animals class ***/

$obj = $stmt->fetchALL(PDO::FETCH_CLASS, 'animals');
```

FETCH LAZY

- Chế độ PDO::FETCH_LAZY là sự kết hợp của 2 chế độ PDO::FETCH_BOTH và PDO::FETCH_OBJ.
- Ví dụ:

```php
$dbh = new PDO("mysql:host=$hostname;dbname=animals", $username, $password);
/*** echo a message saying we have connected ***/
echo 'Connected to database<br />';

/*** The SQL SELECT statement ***/
$sql = "SELECT * FROM animals";

/*** fetch into an PDOStatement object ***/
$stmt = $dbh->query($sql);

/*** echo number of columns ***/
$result = $stmt->fetch(PDO::FETCH_BOTH);

/*** loop over the object directly ***/
foreach($result as $key=>$val)
{
echo $key.' - '.$val.'<br />';
}
```

Giao dịch (Transactions)

- Một PDO transaction sẽ bắt đầu với lời gọi phương thức PDO::beginTransaction(). Phương thức này sẽ tắt chế độ auto-commit và bất cứ câu lệnh hay câu query database nào cũng sẽ không được committed đến database cho đến khi transaction được committed với lời gọi PDO::commit.
- Khi phương thức PDO::commit được gọi is called, tất cả các câu statements/queries được committed, và kết nối đến CSDL sẽ được trả lại chế độ auto-commit.

Giao dịch (Transactions)

- Ví dụ:

```php
<?php
/*** mysql hostname ***/
$hostname = 'localhost';

/*** mysql username ***/
$username = 'username';

/*** mysql password ***/
$password = 'password';

/*** database name ***/
$dbname = 'animals';

try {
    $dbh = new PDO("mysql:host=$hostname;dbname=$dbname", $username, $password);
    /*** echo a message saying we have connected ***/
    echo 'Connected to database<br />';

    /*** set the PDO error mode to exception ***/
    $dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    /*** begin the transaction ***/
    $dbh->beginTransaction();
```

Giao dịch (Transactions)

```
/*** CREATE table statements ***/
$table = "CREATE TABLE animals ( animal_id MEDIUMINT(8) NOT NULL AUTO_INCREMENT PRIMARY KEY
animal_type VARCHAR(25) NOT NULL,
animal_name VARCHAR(25) NOT NULL
)";
$dbh->exec($table);
/***  INSERT statements ***/
$dbh->exec("INSERT INTO animals (animal_type, animal_name) VALUES ('emu', 'bruce')");
$dbh->exec("INSERT INTO animals (animal_type, animal_name) VALUES ('funnel web', 'bruce')")
$dbh->exec("INSERT INTO animals (animal_type, animal_name) VALUES ('lizard', 'bruce')");
$dbh->exec("INSERT INTO animals (animal_type, animal_name) VALUES ('dingo', 'bruce')");
$dbh->exec("INSERT INTO animals (animal_type, animal_name) VALUES ('kangaroo', 'bruce')");
$dbh->exec("INSERT INTO animals (animal_type, animal_name) VALUES ('wallaby', 'bruce')");
$dbh->exec("INSERT INTO animals (animal_type, animal_name) VALUES ('wombat', 'bruce')");
$dbh->exec("INSERT INTO animals (animal_type, animal_name) VALUES ('koala', 'bruce')");
$dbh->exec("INSERT INTO animals (animal_type, animal_name) VALUES ('kiwi', 'bruce')");

/*** commit the transaction ***/
$dbh->commit();

    /*** echo a message to say the database was created ***/
    echo 'Data entered successfully<br />';
}
catch(PDOException $e)
    {
    /*** roll back the transaction if we fail ***/
    $dbh->rollback();

    /*** echo the sql statement and error message ***/
    echo $sql . '<br />' . $e->getMessage();
    }
?>
```

Gọi thủ tục (Stored Procedures)

- Stored procedure: Là một tập hợp các câu lệnh SQL **được đóng gói thành một đơn vị thực thi, để thực hiện một tác vụ độc lập cụ thể.**
- Các bước trong quá trình gọi một stored procedure mà có trả về một result set bằng cách sử dụng PHP PDO tương tự như quá trình truy vấn dữ liệu từ bảng trong CSDL MySQL bằng câu lệnh SELECT.
- Thay cho việc gửi câu lệnh SELECT cho CSDL MySQL, ta gửi một lời gọi thủ tục.
- Ví dụ:

```
CREATE PROCEDURE GetCustomers()
BEGIN
 SELECT customerName, creditlimit
 FROM customers;
    END$$
```

Gọi thủ tục (Stored Procedures)

```php
<html>
    <head>
        <title>PHP MySQL Stored Procedure Demo 1</title>
        <link rel="stylesheet" href="css/table.css" type="text/css" />
    </head>
    <body>
        <?php
        require_once 'dbconfig.php';
        try {
            $pdo = new PDO("mysql:host=$host;dbname=$dbname", $username, $password
);
            // execute the stored procedure
            $sql = 'CALL GetCustomers()';
            // call the stored procedure
            $q = $pdo->query($sql);
            $q->setFetchMode(PDO::FETCH_ASSOC);
        } catch (PDOException $e) {
            die("Error occurred:" . $e->getMessage());
        }
        ?>
         <table>
             <tr>
                 <th>Customer Name</th>
                 <th>Credit Limit</th>
             </tr>
             <?php while ($r = $q->fetch()): ?>
                 <tr>
                     <td><?php echo $r['customerName'] ?></td>
                     <td><?php echo '$' . number_format($r['creditlimit'], 2) ?>
                     </td>
                 </tr>
             <?php endwhile; ?>
         </table>
     </body>
```

Gọi thủ tục (Stored Procedures)

- Kết quả:

Customer Name	Credit Limit
Atelier graphique	$21,000.00
Signal Gift Stores	$71,800.00
Australian Collectors, Co.	$117,300.00
La Rochelle Gifts	$118,200.00
Baane Mini Imports	$81,700.00
Mini Gifts Distributors Ltd.	$210,500.00
Havel & Zbyszek Co	$0.00

Demo

- Giáo viên demo các thao tác PDO gọi thủ tục, transaction …

Thực hành

- Sinh viên thực hành bài toán phân trang – đăng nhập sử dụng PDO

Tổng kết

- Truy vấn và xử lý dữ liệu với PDO
- Gọi thủ tục lưu với PDO
- Thực hành thao tác dữ liệu với PDO
- Thực hành chức năng phân trang – đăng nhập với PDO