LÀM VIỆC VỚI PDO TRONG PHP

Giảng viên: Bùi Quang Đăng

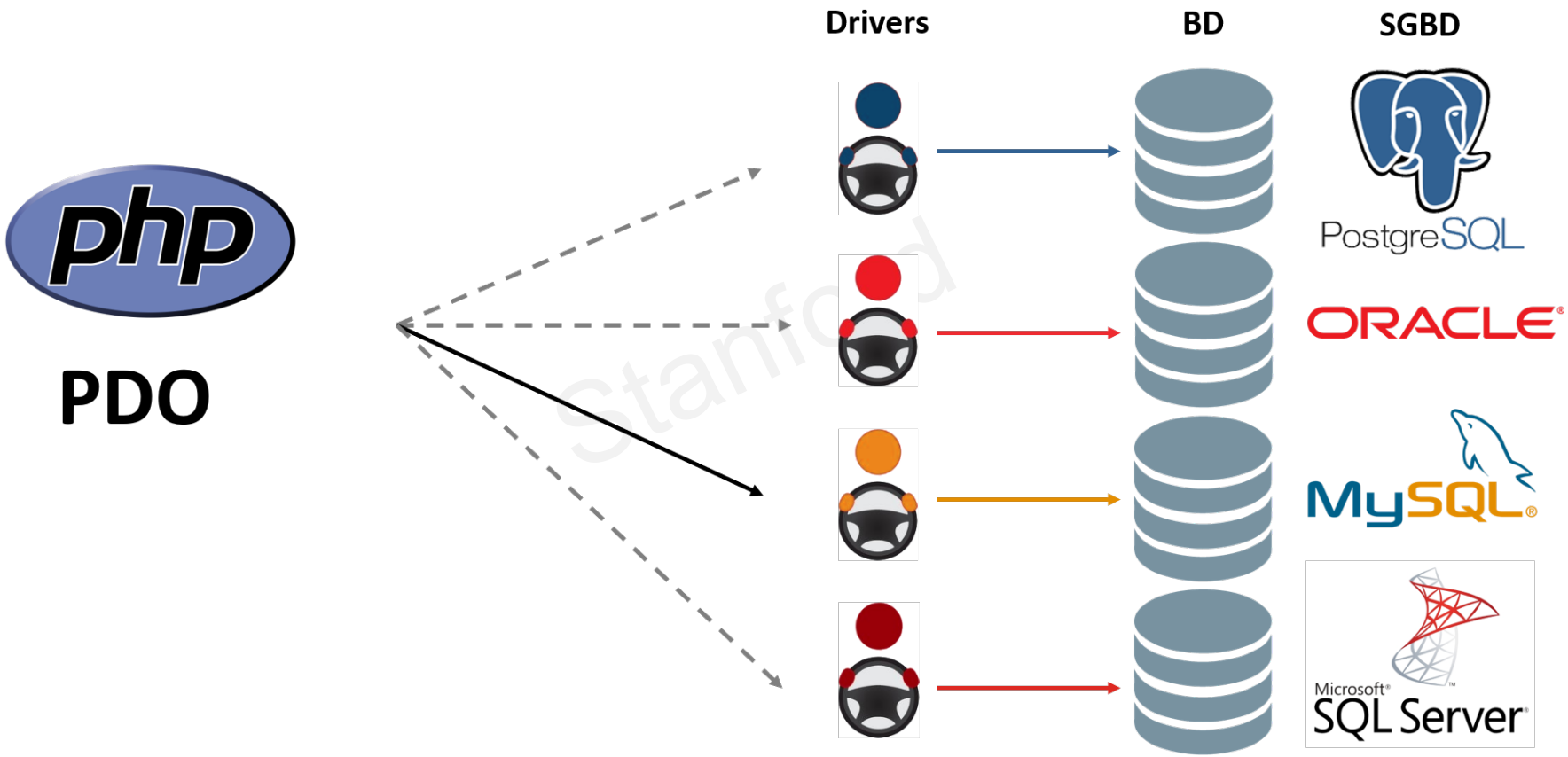
Contents

PHP for Developer

GIỚI THIỆU VỀ PDO

Làm việc với PDO trong PHP

Giới thiệu về PDO trong PHP



Làm việc với PDO trong PHP

Giới thiệu về PDO trong PHP

- **PDO** viết tắt của từ PHP Data Objects là một lớp truy xuất cơ sở dữ liệu cung cấp một phương pháp thống nhất để làm việc với nhiều loại cơ sở dữ liệu khác nhau.
- PDO được giới thiệu từ PHP 5.1 trở đi
- PDO cung cấp các cơ chế Prepared Statements, Stored Procedures và giúp bạn thao tác với database thông qua các Object (đối tượng) làm cho công việc trở nên hiệu quả, dễ dàng hơn.

Làm việc với PDO trong PHP

Giới thiệu về PDO trong PHP

So sánh PDO và MySQLi

	PDO	MySQLi
Database hỗ trợ	Hơn 12 loại	Chỉ hỗ trợ MySQL
API	Hướng đối tượng (OOP)	Hướng đối tượng (OOP) - Hướng thủ tục (Procedural)
Kết nối Database	Dễ dàng	Dễ dàng
Đặt tên tham số	Có	Không

Làm việc với PDO trong PHP

Giới thiệu về PDO trong PHP

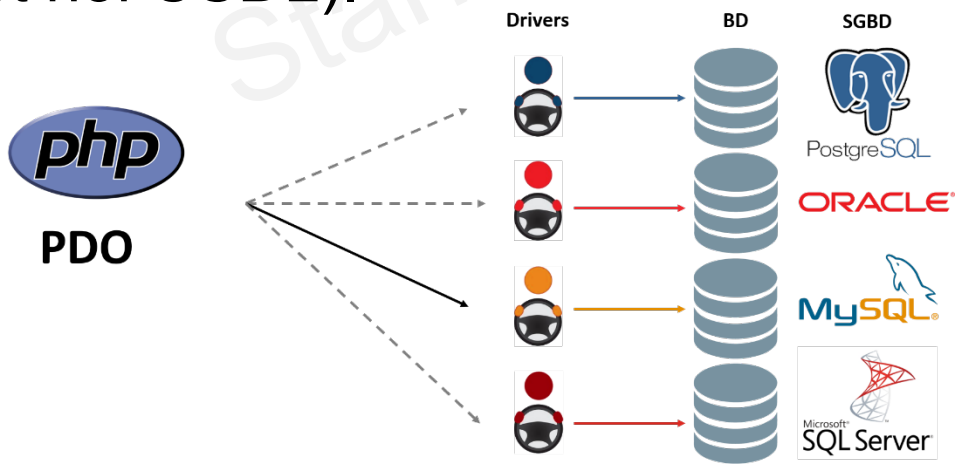
So sánh PDO và MySQLi

	PDO	MySQLi
Object Mapping	Có	Có
Prepared Statements	Có	Không
Hiệu năng	Cao	Cao
Stored Procedures	Có	Có

Làm việc với PDO trong PHP

Giới thiệu về PDO trong PHP

- Khi làm việc với PDO bạn sẽ không cần phải viết các câu lệnh SQL cụ thể mà chỉ sử dụng các phương thức mà PDO cung cấp, giúp tiết kiệm thời gian và làm cho việc chuyển đổi Hệ quản trị cơ sở dữ liệu trở nên dễ dàng hơn, chỉ đơn giản là thay đổi Connection String (chuỗi kết nối CSDL).



Làm việc với PDO trong PHP

Giới thiệu về PDO trong PHP

- Người lập trình chỉ cần nắm rõ API mà PDO cung cấp là có thể làm việc được với nhiều Hệ quản trị cơ sở dữ liệu khác nhau như MySQL, SQLite, PostgreSQL, Microsoft SQL Server,... và có thể dễ dàng chuyển đổi chúng.

Làm việc với PDO trong PHP

Giới thiệu về PDO trong PHP

- Các hệ quản trị cơ sở dữ liệu mà PDO hỗ trợ

Tên driver	DBMS
PDO_CUBRID	Cubrid
PDO_DBLIB FreeTDS	Microsoft SQL Server / Sybase
PDO_FIREBIRD	Firebird
PDO_IBM	IBM DB2
PDO_INFORMIX	IBM Informix Dynamic Server
PDO_MYSQL	MySQL 3.x/4.x/5.x
PDO_OCI	Oracle Call Interface

Làm việc với PDO trong PHP

Giới thiệu về PDO trong PHP

- Các hệ quản trị cơ sở dữ liệu mà PDO hỗ trợ

Tên driver	DBMS
PDO_ODBC	ODBC v3 (IBM DB2, unixODBC and win32 ODBC)
PDO_PGSQL	PostgreSQL
PDO_SQLITE	SQLite 3 and SQLite 2
PDO_SQLSRV	Microsoft SQL Server / SQL Azure
PDO_4D	4D

PHP for Developer

LÀM VIỆC VỚI PDO TRONG PHP

Làm việc với PDO trong PHP

Làm việc với MySQL bằng PDO

Khai báo kết nối làm việc với MySQL bằng PDO

```
$conn = new
PDO('mysql:host=localhost;dbname=stanford',
$username, $password);
```

- Trong đó:
  - **localhost** là máy chủ cần làm việc
  - **stanford** là tên cơ sở dữ liệu cần làm việc
  - Gán $conn = null khi muốn đóng kết nối

Làm việc với PDO trong PHP

Làm việc với MySQL bằng PDO

- **Sử dụng lệnh thực thi trong PDO:** Với PDO, mỗi hoạt động insert hay update được thực hiện qua 3 quá trình sử dụng cơ chế Prepared Statement
  - **Prepare statement**: Chuẩn bị một câu lệnh SQL làm khung/mẫu được gọi là Prepared Statement với các tham số của câu lệnh (placeholder).
  - **Bind params**: Gắn giá trị thực vào các tham số, nhận giá trị trước và sau bindParam trong khi bind value chỉ nhận giá trị trước bindValue.
  - **Execute**: Thực thi câu lệnh.

Làm việc với PDO trong PHP

Làm việc với MySQL bằng PDO

- **Sử dụng placeholder không định danh:** Sử dụng dấu **?** để thực hiện, **bindParam** nhận giá trị trước sau lệnh gán.

```
//Khởi tạo Prepared Statement từ biến $conn ở phần trước
$stmt = $conn->prepare('INSERT INTO users (username, password) values
(?, ?)');
//Gán các biến (lúc này chưa mang giá trị) vào các placeholder theo thứ tự
tương ứng
$stmt->bindParam(1, $username);
$stmt->bindParam(2, $password);
//Gán giá trị và thực thi
$username = ”stanford”;
$password = ”2012";
$stmt->execute();
```

Làm việc với PDO trong PHP

Làm việc với MySQL bằng PDO

- **Sử dụng placeholder định danh:** Sử dụng dấu **:** trước tên tham số trong câu lệnh.

```
//Khởi tạo Prepared Statement từ biến $conn ở phần trước
$stmt = $conn->prepare('INSERT INTO users (username, password) values
(:name, :pass)');
//Gán các biến (lúc này chưa mang giá trị) vào các placeholder theo thứ tự
tương ứng
$stmt->bindParam(':name', $name);
$stmt->bindParam(':pass', $pass);
//Gán giá trị và thực thi
$name = "stanford";
$pass = "2012";
$stmt->execute();
```

Làm việc với PDO trong PHP

Làm việc với MySQL bằng PDO

- **Sử dụng placeholder định danh:** Sử dụng dấu **:** trước tên tham số trong câu lệnh.

```
//Khởi tạo Prepared Statement từ biến $conn ở phần trước
$stmt = $conn->prepare('INSERT INTO users (username, password) values
(:name, :pass)');

$data = array('stanford', '2012');
//Phương thức execute() dưới đây sẽ gán lần lượt giá trị trong mảng vào
các Placeholder theo thứ tự

$stmt->execute($data);
```

Làm việc với PDO trong PHP

Làm việc với MySQL bằng PDO

- **Sử dụng placeholder định danh:** Có thể không dùng dấu : cũng được khi gán giá trị.

```
class $user {
public $username;
public $pass;
}
$u = new $user();
$u->username = ‘stanford’;
$u->pass = ‘2012’;

$stmt = $conn->prepare('INSERT INTO users (username, pass) values
(:username, :pass)’);

$stmt->execute((array)$u);
```

Làm việc với PDO trong PHP

Làm việc với MySQL bằng PDO

- **Sử dụng placeholder định danh:** Sử dụng **bindValue** để gán giá trị

```
//Lấy thông tin đăng nhập
$username = stanford';

$stmt = $db->prepare("SELECT * FROM users WHERE user = :username");

// Use bindValue function
$stmt->bindValue(':username', $username);

$username = ‘admin';

$stmt->execute(); //Giá trị của $username sẽ được gán vào câu lệnh khi dùng bindValue
```

Làm việc với PDO trong PHP

Đọc dữ liệu trong cơ sở dữ liệu

- Khi đọc dữ liệu từ database, PDO sẽ trả về dữ liệu theo cấu trúc mảng (array) hoặc đối tượng (object) thông qua phương thức fetch(). Thiết lập trước cấu trúc dữ liệu trước khi gọi phương thức này, PDO hỗ trợ các tuỳ chọn sau:

Kiểu	Mô tả
PDO::FETCH_ASSOC	Trả về dữ liệu dạng mảng với key là tên của column (column của các table trong database)
PDO::FETCH_BOTH (default):	Trả về dữ liệu dạng mảng với key là tên của column và cả số thứ tự của column
PDO::FETCH_BOUND:	Gán giá trị của từng column cho từng biến đã khởi tạo trước đó qua phương thức bindColumn()

Làm việc với PDO trong PHP

Đọc dữ liệu trong cơ sở dữ liệu

- Thiết lập trước cấu trúc dữ liệu trước khi gọi phương thức này, PDO hỗ trợ các tuỳ chọn sau:

Kiểu	Mô tả
PDO::FETCH_CLASS:	Gán giá trị của từng column cho từng thuộc tính
PDO::FETCH_INTO:	Gán giá trị của từng column cho từng thuộc tính của một Class Instance (thể hiện của một lớp)
PDO::FETCH_LAZY:	Gộp chung PDO::FETCH_BOTH/PDO::FETCH_OBJ
PDO::FETCH_NUM:	Trả về dữ liệu dạng mảng với key là số thứ tự của column
PDO::FETCH_OBJ:	Trả về một Object của stdClass (link is external) với tên thuộc tính của Object là tên của column.

Làm việc với PDO trong PHP

Làm việc với MySQL bằng PDO

- **Sử dụng thiết lập cấu trúc dữ liệu:** Có thể dùng theo các cách sau:

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

Hoặc

$stmt->fetch(PDO::FETCH_ASSOC);
```

Làm việc với PDO trong PHP

Làm việc với MySQL bằng PDO

- **Sử dụng FETCH_ASSOC:** Đưa về mảng và lấy chỉ mục theo tên

```
$stmt = $conn->prepare('SELECT username, pass from users WHERE
name = :username');
//Thiết lập kiểu dữ liệu trả về
$stmt->setFetchMode(PDO::FETCH_ASSOC);
//Gán giá trị và thực thi
$stmt->execute(array('username' => 'a'));
//Hiển thị kết quả, vòng lặp sau đây sẽ dừng lại khi đã duyệt qua toàn bộ kết
quả
while($row = $stmt->fetch()) {
echo $row['username'] , '\n';
echo $row['passl'] , '\n';
}
```

Làm việc với PDO trong PHP

Làm việc với MySQL bằng PDO

- **Sử dụng FETCH_OBJ:** Kiểu fetch này trả về một Object với tên thuộc tính là tên cột trả về cho mỗi row của kết quả

```
$stmt = $conn->prepare('SELECT username, pass from users WHERE
name = :username');
//Thiết lập kiểu dữ liệu trả về
$stmt->setFetchMode(PDO::FETCH_OBJ);
//Gán giá trị và thực thi
$stmt->execute(array('username' => 'a'));

while($row = $stmt->fetch()) {
  echo $row->username , '\n';
  echo $row->pass, '\n';
}
```

Làm việc với PDO trong PHP

Làm việc với MySQL bằng PDO

Sử dụng FETCH_CLASS:

- Kiểu fetch này cho phép bạn đưa kết quả vào Object của một lớp (class) mà bạn chỉ định.
- Khi sử dụng FETCH_CLASS, thuộc tính của class sẽ được gán giá trị trước khi constructor của class đó được gọi (phải chú ý vì điều này rất quan trọng). Nếu không có thuộc tính khớp với tên của một column bất kỳ thì thuộc tính đó sẽ được tự động tạo ra (public).

Làm việc với PDO trong PHP

Làm việc với MySQL bằng PDO

- **Sử dụng FETCH_CLASS:** Kiểu fetch này cho phép bạn đưa kết quả vào Object của một lớp (class) mà bạn chỉ định.

```
class User {
public $name;
public $email;
public $isAdmin = 'No';
function __construct() {
if ($this->name == 'Stanford')
$this->isAdmin = 'Yes';
} }
```

Làm việc với PDO trong PHP

Làm việc với MySQL bằng PDO

- **Sử dụng FETCH_CLASS:** Kiểu fetch này cho phép bạn đưa kết quả vào Object của một lớp (class) mà bạn chỉ định. Constructor được gọi sau khi thuộc tính được gán.

```
//Tạo Prepared Statement
$stmt = $conn->prepare('SELECT username, pass from users WHERE
username = :name');
//Thiết lập kiểu dữ liệu trả về, chỉ định dữ liệu được đưa vào object của
class User
$stmt->setFetchMode(PDO::FETCH_CLASS, 'User');
//Gán giá trị và thực thi
$stmt->execute(array('name' => 'a'));
while($obj = $stmt->fetch()) {
echo $obj->username; echo $obj->pass;
}
```

Làm việc với PDO trong PHP

Làm việc với MySQL bằng PDO

- **Sử dụng FETCH_CLASS:** Nếu muốn constructor của class được gọi trước khi các thuộc tính được gán giá trị, bạn phải sử dụng thêm PDO::FETCH_PROPS_LATE. Cách sử dụng như sau:

```
//Hãy thử sử dụng kiểu fetch trên và so sánh kết quả hiển thị
$stmt->setFetchMode(PDO::FETCH_CLASS |
PDO::FETCH_PROPS_LATE, 'User');
```

Làm việc với PDO trong PHP

Làm việc với MySQL bằng PDO

- **Sử dụng FETCH_CLASS:** Nếu cần truyền các tham số cho constructor của class thông qua phương thức fetch(), các bạn có thể đặt chúng trong một array theo thứ tự tương ứng cụ thể như sau:

```
$stmt->setFetchMode(PDO::FETCH_CLASS, 'User', array('par1', 'par2', 'par3'));
```

Làm việc với PDO trong PHP

Làm việc với MySQL bằng PDO

- **Exceptions:** PDO dùng các Exceptions để xử lý các lỗi phát sinh khi làm việc với database, sử dụng khối try/catch để bắt lỗi. PDO cung cấp 3 chế độ xử lý lỗi (Error Mode) được thiết lập thông qua phương thức setAttribute():

```
$conn->setAttribute( PDO::ATTR_ERRMODE,
PDO::ERRMODE_SILENT );

$conn->setAttribute( PDO::ATTR_ERRMODE,
PDO::ERRMODE_WARNING );

$conn->setAttribute( PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION );
```

Làm việc với PDO trong PHP

Làm việc với MySQL bằng PDO

- **PDO::ERRMODE_SILENT:** Đây là chế độ xử lý lỗi mặc định của PDO, khi gặp mốt lỗi bất kỳ, PDO sẽ im lặng (silent) và chương trình vẫn tiếp tục chạy. Bạn có thể lấy mã lỗi và thông tin về các lỗi đã xảy ra qua PDO::errorCode() và PDO::errorInfo()

Làm việc với PDO trong PHP

Làm việc với MySQL bằng PDO

- **PDO::ERRMODE_WARNING:** Ở chế độ này khi gặp phải lỗi PDO sẽ ném ra một PHP Warning, chương trình sẽ tiếp tục chạy.
- **PDO::ERRMODE_EXCEPTION:** Đây là mode nên sử dụng nhất, khi đặt trong một try/catch block sẽ giúp chúng ta kiểm soát các lỗi phát sinh một cách uyển chuyển và giấu các thông báo lỗi có thể khiến Attacker khai thác hệ thống.

Làm việc với PDO trong PHP

Làm việc với MySQL bằng PDO

Một số phương thức quan trọng khác:

- Phương thức trên trả về Auto Incremented ID của dòng được thêm gần nhất.

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

Làm việc với PDO trong PHP

Làm việc với MySQL bằng PDO

Một số phương thức quan trọng khác:

- Đối với các lệnh SQL không có dữ liệu trả về, và không cần thiết phải truyền tham số thì có thể sử dụng phương thức exec(). Phương thức này sẽ trả về số lượng row bị tác động sau khi thực hiện câu lệnh.

```
$conn->exec('DELETE FROM users WHERE id = 1');
```

Làm việc với PDO trong PHP

Làm việc với MySQL bằng PDO

Một số phương thức quan trọng khác:

- Phương thức quote() sẽ giúp thêm dấu nháy cho một string câu lệnh để string đó an toàn khi sử dụng để truy vấn, nếu không muốn sử dụng Prepared Statement.

```
$conn = $DBH->quote($foo);
```

Làm việc với PDO trong PHP

Làm việc với MySQL bằng PDO

Một số phương thức quan trọng khác:

- Phương thức rowCount() trả về số lượng row bị tác động sau khi thực hiện các thao tác DELETE, INSERT và UPDATE.

```
$stmt->rowCount();
```

PHP for Developer

Exercises

Thank You !

www.stanford.com.vn