LEARNING pdo

#pdo

Table of Contents

About

Chapter 1: Getting started with pdo

Remarks

This section provides an overview of what pdo is, and why a developer might want to use it.

It should also mention any large subjects within pdo, and link out to the related topics. Since the Documentation for pdo is new, you may need to create initial versions of those related topics.

Examples

Installation or Setup

PDO is a universal database connection command in PHP, it support 12 different database type e.g *MySQL, MongoDB, NoSQL*. A big bonus about PDO is that it calculate your code to support the database type, so you don't need to make any chance when moving over to another database system.

Summary

	PDO	MySQLi
Database support	12 different drivers	MySQLi
API	OOP	OOP + procedural
Connection	Easy	Easy
Named parameters	Yes	No
Object mapping	Yes	Yes
Prepared statements (client side)	Yes	No
Performance	Fast	Fast
Stored procedures	Yes	Yes

Connection to the database with PDO:

The connection part looks awkward but that we need to deal with. The rest of the PDO is simple and useful, it's also help to make the secure part even easier.

```
$connection = new PDO("mysql:host=localhost;dbname=myDatabase, username, password);
```

The PDO connect is order by following:

PDO(`database type`**:host=**`host`**;dbname=**`database name`**,** `root`**,** `password`**);**

Secure selection from the database using PDO:

```
// We use a array to hold the data about whats the :var is in normal $var
$params = array(
    ':username' => '$username',
    ':email' => $mail,
);

// Prepare the SQL and using named secure parameters ":username"
$pdo->prepare('SELECT * FROM users WHERE username = :username AND email = :email');

// Execute the $params and send them to the $pdo->prepare
$pdo->execute($params);
```

The code you just read, is protected agents SQL injection

Tutorials:

Install:

How to install PDO if you doesn't have it

Guides:

W3Schools tutorial
Tuts+ Tutorial (Recommended)

PDO Setting Attribute Errormode

PDO::setAttribute sets an attribute on the database handle. Desctiption of setAttribute is:

```
public bool PDO::setAttribute ( int $attribute , mixed $value )
```

PDO::ATTR_ERRMODE: This attribute is used for error reporting. It can have one of the following values.

- `PDO::ERRMODE_SILENT`: If the ATTR_ERRMODE is not set in the code, ERRMODE_SILENT is the default value of ATTR_ERRMODE attribute. It sets error codes. In silent mode, if there is an error in SQL, PDO will throw no exceptions; PDO will issue no warnings; it will simply return false. Value of PDO::ERRMODE_SILENT is 0. The script will run without generating any error or warning.

- `PDO::ERRMODE_WARNING`: This value raises E_WARNING. In warning mode, if there is an error in SQL, PDO will issue warnings but script will continue running. Value of PDO::ERRMODE_WARNING is 1. The script will run with generating warning about the error.
- `PDO::ERRMODE_EXCEPTION`: This value throws exceptions. In exception mode, if there is an error in SQL, PDO will throw exceptions and script will stop running. Value of PDO::ERRMODE_EXCEPTION is 2. The script will stop executing generating the error which throws the exception.

Example: Now we are going to see the various values of the attribute ATTR_ERRMODE with some examples. To do so, create a database called `learn_project_db` and insert a table called `user_table` inside it. The following SQL snippet can be used to achieve so:

```
DROP DATABASE IF EXISTS `learn_project_db`;
CREATE DATABASE `learn_project_db`;
USE `learn_project_db`;
CREATE TABLE `user_table` (
  `user_email` varchar(50) PRIMARY KEY,
  `user_password` varchar(50) NOT NULL
);
INSERT INTO `user_table` (`user_email`, `user_password`) VALUES
('test1@example.com', '123'),
('test2@example.com', '1234'),
('test3@example.com', '12345');
```

At first we are going to check what will happen if we do not set the `ATTR_ERRMODE` and we will have error in the SQL query. Create a PHP file called `default.php` and try this:

```
<?php
    $server = "localhost";
    $db_username = "root";
    $db_password = "";
    $db_name = "learn_project_db";
    $conn = new PDO("mysql:host=$server;dbname=$db_name",$db_username,$db_password);
    $sql_query = "SELECT * FROM wrong_user_table";
    $stmt = $conn->prepare($sql_query);
    $stmt->execute();
    $result_set = $stmt->fetchAll();
    var_dump($result_set);
    /*Get the current error mode of PDO*/
    $current_error_mode = $conn->getAttribute(PDO::ATTR_ERRMODE);
    echo "<br>";
    echo "Value of PDO::ATTR_ERRMODE: ".$current_error_mode;
?>
```

Notice that, the table name is `wrong_user_table` in the query which is not defined in the database we created earlier. But, as we did not set the `ATTR_ERRMODE` it will run the script without throwing any exception or issuing any warning. It will output an empty array as result set. We should also notice that, the value of `PDO::ATTR_ERRMODE` is 0.

Now, we are going to check what will happen if we set the `ATTR_ERRMODE` with value `PDO::ERRMODE_WARNING` and we will have error in the SQL query. Create a PHP file called `warning.php` and try this:

```
<?php
    $server = "localhost";
    $db_username = "root";
    $db_password = "";
    $db_name = "learn_project_db";
    $conn = new PDO("mysql:host=$server;dbname=$db_name",$db_username,$db_password);
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_WARNING);
    $sql_query = "SELECT * FROM wrong_user_table";
    $stmt = $conn->prepare($sql_query);
    $stmt->execute();
    $result_set = $stmt->fetchAll();
    var_dump($result_set);
    /*Get the current error mode of PDO*/
    $current_error_mode = $conn->getAttribute(PDO::ATTR_ERRMODE);
    echo "<br>";
    echo "Value of PDO::ATTR_ERRMODE: ".$current_error_mode;
?>
```

Output of `warning.php` is:

```
Warning: PDOStatement::execute(): SQLSTATE[42S02]: Base table or view not found: 1146 Table
'learn_project_db.wrong_user_table' doesn't exist in E:\xampp\htdocs\oop\db.php on line 10
array(0) { }
Value of PDO::ATTR_ERRMODE: 1
```

This time as we set the `ATTR_ERRMODE` with `PDO::ERRMODE_WARNING` value it will show a warning message. The script runs successfully and shows an empty array as output with the value of `PDO::ATTR_ERRMODE` of 1.

At last, we are going to check what will happen if we set the `ATTR_ERRMODE` with value `PDO::ERRMODE_EXCEPTION` and we will have error in the SQL query. Create a PHP file called `error.php` and try this:

```
<?php
    $server = "localhost";
    $db_username = "root";
    $db_password = "";
    $db_name = "learn_project_db";
    $conn = new PDO("mysql:host=$server;dbname=$db_name",$db_username,$db_password);
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $sql_query = "SELECT * FROM wrong_user_table";
    $stmt = $conn->prepare($sql_query);
    $stmt->execute();
    $result_set = $stmt->fetchAll();
    var_dump($result_set);
    /*Get the current error mode of PDO*/
    $current_error_mode = $conn->getAttribute(PDO::ATTR_ERRMODE);
    echo "<br>";
    echo "Value of PDO::ATTR_ERRMODE: ".$current_error_mode;
?>
```

Output of `error.php` is:

```
Fatal error: Uncaught exception 'PDOException' with message 'SQLSTATE[42S02]: Base table or
view not found: 1146 Table 'learn_project_db.wrong_user_table' doesn't exist' in
E:\xampp\htdocs\oop\db.php:10 Stack trace: #0 E:\xampp\htdocs\oop\db.php(10): PDOStatement-
>execute() #1 {main} thrown in E:\xampp\htdocs\oop\db.php on line 10
```

This time as we set the `ATTR_ERRMODE` with `PDO::ERRMODE_EXCEPTION` value it will throw `PDOException` which generates a fatal error. The script will stop executing after the exception is thrown. This is most common approach to handle database query related errors. In most of the time, we set the attribute `ATTR_ERRMODE` with this value to handle any exception that might resides in SQL query.

Read Getting started with pdo online: https://riptutorial.com/pdo/topic/4393/getting-started-with-pdo

Chapter 2: Prepare and Execute your SQL statements

Remarks

Warning

Prepared statement cannot care a wild parameter for the table names. For exemple this following statement is not correct :

```
$query = "SELECT name, city FROM ? WHERE id = ? AND country = ?";
```

The correct prepared query would be :

```
$query = "SELECT name, city FROM users WHERE id = ? AND country = ?";
```

Examples

Usage

```
// 1. Connect to the database (this example with MySQL)
$host = 'localhost';
$database = 'users';
$user = 'root';
$password = '';
$dsn = "mysql:host=$host;dbname=$database";
$pdo = new PDO($dsn, $user, $password);

// 2. Prepare your query

// 2.1 First way
$query_1 = "SELECT name, city FROM users WHERE id = ? AND country = ?";
$statement_1 = $pdo->prepare($query_1);

// 2.2 Second way
$query_2 = "SELECT name, city FROM users WHERE id = :id AND country = :country";
$statement_2 = $pdo->prepare($query_2);

// 3. Execute your query

// 3.1 With the first way
$statement_1->execute([1, 'US']);

// 3.2 With the second way
$statement_2->execute([
    ':id' => 1,
    ':country' => 'US'
]);

// 4. Fetch your data
```

```
$data_1 = $statement_1->fetchAll();
$data_2 = $statement_2->fetchAll();
```

Read Prepare and Execute your SQL statements online:
https://riptutorial.com/pdo/topic/6511/prepare-and-execute-your-sql-statements

Credits

S. No	Chapters	Contributors
1	Getting started with pdo	arsho, Community, hjpotter92, Ilyas Mimouni, manian, TheCrazyProfessor, Your Common Sense
2	Prepare and Execute your SQL statements	Anwar Nairi, Your Common Sense