Class 2

PHP Data Object

BY:

BRUNO ANGELO MEDEIROS
bruno.medeiros@sc.senai.br

In PHP 7 the **mysql_***functions are removed. Because of that now is necessary use MySQLi or PDO.

For WS the PDO is more useful to manipulate the database when we need develop without a framework.

CONNECTING

Every connection with database is made to create an instance of the PDO class. PDO has a method called DSN (Data Source Name), which is basically a string of options that tell which driver to use and the connections details.

- database driver, host, db (schema) name and charset;
- username and password go to constructor;
- all other options go into options array.

Where DSN is a semicolon-delimited string, consists of **parameter=value** pairs, that begins from the driver name and a colon:

```
mysql:host=localhost;dbname=russia;charset=utf8
```

Note: It's important to follow the proper format with **no spaces** or **quotes.**

An example for mysql:

```
<?php
$host = 'localhost';
$db   = 'russia';
$user = 'root';
$pass = '';
$charset = 'utf8';
$dsn = "mysql:host=$host;dbname=$db;charset=$charset";
$opt = [
    PDO::ATTR_ERRMODE            => PDO::ERRMODE_EXCEPTION, //throw exceptions

    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
// returns an array indexed by column name as returned in your result set
];
$pdo = new PDO($dsn, $user, $pass, $opt);
?>
```

PREPARED STATEMENTS

A prepared statement is a precompiled SQL statement. It has the added advantage of automatically making the data used in the placeholders safe from SQL injection attacks.

You need only two functions:

- prepare
- execute

First of all, you have to alter the common query, adding the placeholder mentioned before, like this:

```
1. $sql = "SELECT * FROM country WHERE name = '$name' AND abbreviation =
   '$abbreviation'";
```

Will become:

```
1. $sql = "SELECT * FROM country WHERE name = ? AND abbreviation = ?";
```

Or:

```
1. $sql = "SELECT * FROM country WHERE name = :name AND abbreviation =
   :abbreviation";
```

Note that PDO supports positional (?) and named (:name) placeholders.

Having a query with placeholders, you have to prepare it, using the **prepare()** method. Finally, to get the query executed, you must run **execute()** method of this object, passing variables in it, in the form of array. After that, you will be able to get the resulting data out of statement.

```
1. $stmt = $pdo->prepare('SELECT * FROM country WHERE name = ? AND abbreviation =
   ?');
2. $stmt->execute([$name, $abbreviation]);
3. $user = $stmt->fetch();
4. // or
5. $stmt = $pdo->prepare('SELECT * FROM country WHERE name = :name
   AND abbreviation = :abbreviation');
6. $stmt->execute([name => $name, 'abbreviation' => $abbreviation]);
7. $user = $stmt->fetch();
```

INSERT, UPDATE and DELETE

Prepared Statements for INSERT, UPDATE, and DELETE are not different than SELECT. But let's do some examples:

INSERT

```
1. // unnamed placeholders
2. $stmt = $pdo-
   >prepare("INSERT INTO country (name, abbreviation) values (?, ?)");
3. // named placeholders
4. $stmt = $pdo-
   >prepare("INSERT INTO country (name, abbreviation) value (:name, :abbreviation
   )");
```

UPDATE

```
$stmt = $db->prepare("UPDATE country SET name = ? WHERE abbreviation = ?");
$stmt->execute([$name, $abbreviation]);
$affected_rows = $stmt->rowCount();
```

DELETE

```
$stmt = $pdo->prepare("DELETE FROM client WHERE id = ?");
$stmt->execute([$id]);
$deleted = $stmt->rowCount();
```

GETTING DATA

A helper function that returns value of the single field of returned row. Very handy when we are selecting only one field:

```
// Getting the name based on id
$stmt = $pdo->prepare("SELECT name FROM client WHERE id = ?");
$stmt->execute([$id]);
$name = $stmt->fetchColumn();
```

In another way, when you need to get all the rows returned by the query, you can use the **fetchAll()** instead of **fetchColumn()**, like this:

```
$data = $pdo->query('SELECT name FROM client')->fetchAll();
```

Note: If no variables are going to be used in the query, you can use the **query()** method.

LIKE CLAUSE

To prevent an error from occurring, the method for using the LIKE clause with PDO is as follows:

```
$search = "%ia%";
$stmt = $pdo->prepare("SELECT * FROM country WHERE name LIKE ?");
$stmt->execute([$search]);
$data = $stmt->fetchAll();
```

It’s because a placeholder has to represent a complete data literal only.