5.1 PDO class - MVC

In de vorige lessen hebben we een calculator gemaakt en hebben hier een cookie als mini-database-je gebruikt. We gaan nu een 'echte' database verbinding maken. Als het goed is hebben we al eerder met PDO gewerkt. We gaan even kort herhalen hoe het ook alweer zat met PDO en dan gaan we een eigen class maken met PDO. Met die class kunnen we dan eenvoudig gegevens opslaan en opvragen.

We gaan dus:

1. een stukje PDO herhalen
2. en een PDO-database class maken

PDO 'old style'

PDO wordt gebruikt om een verbinding met een database te maken en om via deze verbinding gegevens op te slaan, te wijzigen, te verwijderen of op te vragen (insert, update, delete, select).

Vorig jaar heb je PDO gebruikt op de (ongeveer) de volgende manier:

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

try {
    $conn = new PDO("mysql:host=$servername;dbname=myDB", $username, $password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    echo "Connected successfully";
    }
catch(PDOException $e)
    {
    echo "Connection failed: " . $e->getMessage();
    }
?>
```

Gebruik deze code om te controleren of PDO goed werkt en of je de username en password van je database nog weet.

> Gebruik je root als username? Waarom is dit onverstandig?

In PDO heb je een aantal stappen die je moet doorlopen om een query uit te voeren:

1. De verbinding maken; dit hoeft maar één keer.
2. De query klaarzetten (prepare).
3. De query uitvoeren (execute).
4. De resultaten uitlezen (fetch).

Sommigen queries hebben geen resultaten, dan is stap 4 niet nodig.

> Vraag: welk soort queries hebben geen resultaat zoals in stap 4 wordt bedoeld?

PDO Object-style

We gaan nu een PDO-object maken waarin we deze stappen proberen samen te voegen.

Omdat het maken van de verbinding maar één keer nodig is, zetten we dit in de __construct van de class.

> Vraag: weet je nog wat de __construct method (functie) doet?

In

https://www.youtube.com/watch?v=7MVOpcKVr_g

kun je nog een keer bekijken hoe de __construct ook alweer werkt.

De PDO class ziet er uit zoals hieronder. Bekijk de code en zorg dat je alles begrijpt. Vul de code aan op de plaats waar ... staan.

```
<?php

// Define DB Params
define("DB_HOST", "localhost");
define("DB_USER", "");
define("DB_PASS", "");
define("DB_NAME", "");

class DB{
```

```
    protected $dbh;       // dit is de verbinding met de databas
    protected $stmt;      // dit is het huidige statement
    protected $resultSet; // hierin komen de resutlaten te staan van een query

    public function __construct(){
        $this->dbh = new PDO(....); // zoek op hoe je de verbinding maakt en vul de juiste code op de plaats van de puntjes
        $this->resultSet=[];
    }

    public function execute($query){
        $this->stmt = $this->dbh->prepare($query);
        // haal nu het resultaat binnen in $result door de query nu uit te voeren (je moet dus de method execute uitvoeren op $this->stmt)
        .....
        if (! $result) {
            die('<pre>Oops, Error execute query '.$query.'</pre><br><pre>'.'Result code: '.$result.'</pre>');
        }
        $this->resultSet = $this->stmt->fetchAll(PDO::FETCH_ASSOC);
        // return nu het het aantal rows dat de query terug geeft. Dit is dus het aantal array elementen dat je terug krijgt.
        .....
    }

    public function getRow(){
        // lees het array waarin de resultaten staan ($this->resultSet) uit, regel voor regel.
        // Elke keer als deze functie wordt aangeroepen return je de volgende regel uit de resultSet.
        // Gebruik de array_shift functie en zoek op hoe je deze moet gebruiken.
        // Let op wat er gebeurt als je een lege resultSet hebt.
        .....
        .....
        .....
        .....
        .....
    }
}
?>
```

We gaan onze PDO class testen maar daarvoor hebben we wat testdata nodig. We gaan dus testdata maken.

Aanmaken testdata

Maak nu een test-database en een testtabel.

Creëer een database 'flights' en maak een tabel met testdata met het volgende script via PHPMyAdmin

```
CREATE TABLE `airlines` (
  `code` char(2) NOT NULL,
  `airline` varchar(40) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

INSERT INTO `airlines` (`code`, `airline`) VALUES
('AA', 'American Airlines Inc.'),
('AS', 'Alaska Airlines Inc.'),
('B6', 'JetBlue Airways'),
('DL', 'Delta Air Lines Inc.'),
('EV', 'Atlantic Southeast Airlines'),
('F9', 'Frontier Airlines Inc.'),
('HA', 'Hawaiian Airlines Inc.'),
('IA', 'AIRLINE'),
('MQ', 'American Eagle Airlines Inc.'),
('NK', 'Spirit Air Lines'),
('OO', 'Skywest Airlines Inc.'),
('UA', 'United Air Lines Inc.'),
('US', 'US Airways Inc.'),
('VX', 'Virgin America'),
('WN', 'Southwest Airlines Co.');
```

MVC, stap 1

Maak nu een nieuw project met de volgende files en folder structuur:

```
/includes/db.php
crud.php
view.html
```

De db.php bevat de class DB zoals je die hierboven heb gemaakt. Met de crud.php file gaan we een CRUD-pagina maken en met de view.html gaan de data aan de gebruiker laten zien.

De /includes/db.php regelt dus de connectie naar de database, de crud.php is de pagina waar de 'controle' plaats vind; wat gebeurt er met de data. Ten slotte wordt er in de view.html pagina de

data getoond. We hebben hier dus te maken met vier functies Model (de connectie naar de database), View (de html), en Control (de PHP-code die ales 'regelt'). We hebben het hier over het MVC-model.

In de volgende les gaan we onze CRUD-pagina maken in deze MVC-structuur.

Handige resource: https://www.webcarpenter.com/blog/52-PDO-Tutorial-for-MySQL-Developers

---

Revision #11
Created 10 September 2019 18:50:22 by Admin
Updated 5 October 2020 11:18:07 by Max