PHP Data Objects - PDO

Fördelar

Ett sätt att kommunicera med olika databaser

Erbjuder större säkerhet i och med prepared statements - förhindrar SQL injection

Version

Finns i PHP 5.1 och uppåt

Drivers

Det finns PDO drivers för olika databaser. Du ställer in vilka som ska laddas in i filen php.ini (kika i katalogen **conf/aktuell_PHP_version**), och om du ändrar i den måste du starta om apache. MySQL och SQLite är aktiverade by default, så det krävs en ändring om du vill köra MSSQL.

Titta i PHPinfo (gå via MAMPs startsida) för att se vad som är laddat.

Enable MSSQL on MAMP

https://stackoverflow.com/questions/39889897/php-7-x-connection-with-mssql-server-with-mamp

https://docs.microsoft.com/en-us/sql/connect/php/loading-the-php-sql-driver?view=sql-server-ver15

PDO i koden

Alla exempel jobbar mot MySQL-databasen **godis** som innehåller tabellen **products**.

(Skapa en databas som heter godis, importera till den innehållet i sql-filen som du hittar på ITHS-distans, och skapa sedan en användare med rättigheter till den databasen.

1. *Gå till PHPMyAdmin (http://localhost:8088/phpMyAdmin/index.php) Du kan behöva justera portnumret.*
2. *Klicka på "Ny" i vänsterkolumnen/Välj fliken databaser i den horisontella menyn.*
3. *Skriv in namnet på din databas. Klicka Skapa*
4. *Välj fliken importera*
5. *Välj fil och klicka Kör - hoppas att allt går bra.*
6. *Välj fliken Privilegier och sedan "Lägg till användare"*
7. *Fyll i uppgifterna "godisAdmin", "localhost" och skriv lösenordet två gånger. Klicka sedan Kör längst ned på sidan.*

8. *Sidan laddar om och du får då ange databas-specifika privilegier. Klicka bara Kör längst ner.*
9. *Nu är din användare klar att använda.*

Skapa en uppkoppling

```
<?php
$dbh = new PDO('mysql:host=localhost;dbname=test', username, password);
```

Utför en query

```
$query = "SELECT name FROM products WHERE price < 10";

foreach ($dbh->query($query) as $godis) {
    echo $godis['name'] . "<br/>";
}
```

Vi har möjligheten att göra enkla queries (ställa frågor till databasen) i PDO, men det erbjuder ingen säkerhet, så vi bör avstå från den här metoden. Den duger om vi enbart har hårdkodade värden i vår query.

Bättre är att skriva ett prepared statment.

```
$query = "SELECT name FROM products WHERE price < :price";

$statement = $dbh->prepare($query, array(PDO::FETCH_ASSOC));
$statement->execute(array(':price' => 10));

$result = $statement->fetchAll();

foreach ($result as $godis) {
    echo $godis['name'] . "<br />";
}
```

Längre kod, men betydligt bättre.

:price är en s k placeholder, och i execute-satsen ersätts den med det värde vi vill lägga in. På det här sättet håller vi isär de statiska och de dynamiska delarna av queryn, PDO "escapar" de dynamiska värdena, vilket tar bort möjligheten till SQL injections, och vi kan därmed hämta in värdena direkt från ett inputfält eller liknande.

PHP-övningar PDO

1. Skapa en index.php där du hämtar ut alla godisprodukter som har weight mindre än 150. Skriv ut namn och pris i en table.
2. Utveckla övningen genom att skapa ett inputfält (maxpris) på sidan. Skriv kod så att du hämtar in det värdet och använder det i SQL-queryn.
3. Lägg till ett formulär för maxweight, så att du får två variabler att peta in i din SQL-query. Prova att skriva prepared statements både med kolon och frågetecken.
4. Varje gång sidan laddas om så får du upp de hårdkodade värdena i formulär-fälten. Försök ändra i koden så att du får med det du senast skrev in.
5. Utveckla ytterligare så att du skapar ett annat formulär, där du har inputfält för name, weight och price. Skriv sedan kod som lägger till en rad i databastabellen. Queryn ska se ut ungefär så här:
   `INSERT INTO products ("name", "weight", "price") VALUES (dina värden, fast med PDO-placeholders)`
   Använd därefter egenskapen **rowCount** för att se hur många rader som påverkades av queryn.
   https://www.php.net/manual/en/pdostatement.rowcount.php