Webbutveckling med PHP

DB, PDO

Utbildare: Mikael Olsson

mikael.olsson@emmio.se

076-174 90 43

NACKADEMIN

PDO

- PHP Database Objects
    - Database Access Abstraction Layer
        - A database abstraction layer is an API (application programming interface) which unifies the communication between a computer application and various databases.

          https://en.wikipedia.org/wiki/Database_abstraction_layer

PDO

- Benefits include:
  - *security* (usable prepared statements)
  - *usability* (many helper functions to automate routine operations)
  - *reusability* (unified API to access multitude of databases, from SQLite to Oracle)

PDO - ORM

- Note that although PDO is the best out of native db drivers, for a modern web-application consider to use an ORM with a Query Builder, or any other higher level abstraction library, with only occasional fallback to vanilla PDO.
    - Doctrine, Eloquent, RedBean, Yii::AR, Aura.SQL

- Object-relational mapping is a programming technique for converting data type systems using object-oriented programming languages. https://en.wikipedia.org/wiki/Object-relational_mapping

```
var sql = "SELECT id, first_name, last_name, phone, birth_date, sex, age FROM persons WHERE id = 10";
var result = context.Persons.FromSqlRaw(sql).ToList();
var name = result[0]["first_name"];
```

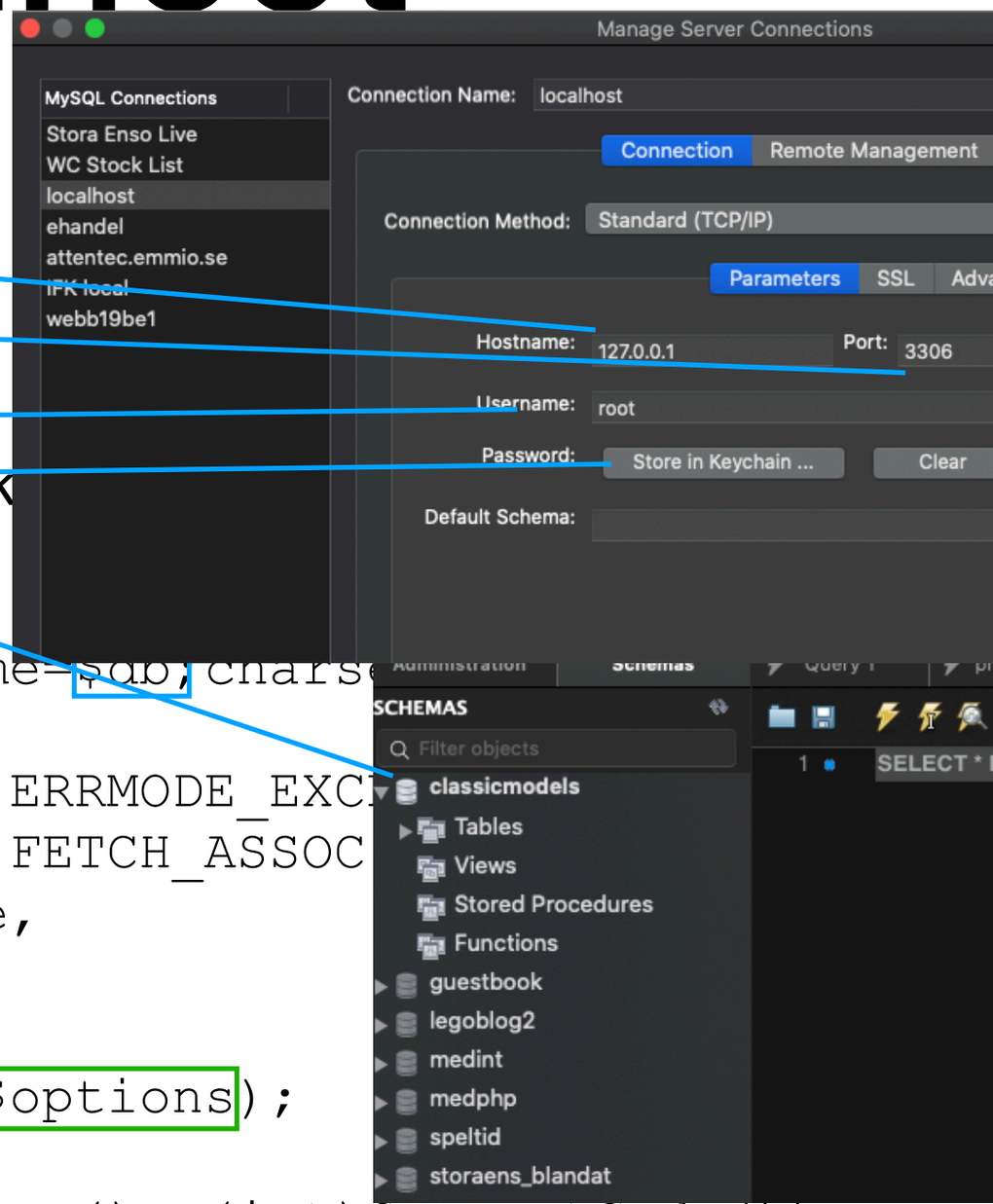
```
var person = repository.GetPerson(10);
var firstName = person.GetFirstName();
```

PDO - connect

```
$host    = '127.0.0.1';
$port    = '3306';
$db      = 'classicmodels';
$user    = 'root';
$pass    = '';
$charset = 'utf8mb4';

$dsn = "mysql:host=$host;port=$port;dbname=$db;charse
$options = [
    PDO::ATTR_ERRMODE            => PDO::ERRMODE_EXCI
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC
    PDO::ATTR_EMULATE_PREPARES   => false,
];
try {
     $pdo = new PDO($dsn, $user, $pass, $options);
} catch (\PDOException $e) {
     throw new \PDOException($e->getMessage(), (int)$e->getCode());
}
```

Vilken typ av db sk



`$pdo` innehåller nu objektet som vi använder för att interagera med databasen.

Uppgift

- Skapa ett skript som kopplar upp din applikation mot din databas.

- Vi ska använda databasen ClassicModels som vi använde på förra kursen, så installera om den om du inte har kvar den.
  https://www.mysqltutorial.org/mysql-sample-database.aspx

- Ditt skript behöver inte göra något mer än att inte få fel, men du kan skriva ut variabeln med PDO-objektet för att se att du har fått ett fungerande objekt.

PDO - running queries

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

```
$statement = $pdo->query('SELECT name FROM users');

while ($row = $statement->fetch()) {
    print_r($row);
    echo $row['name'] . PHP_EOL;
}
```

- `PDO::query()` executes an SQL statement in a single function call, returning the result set (if any) returned by the statement as a PDOStatement object.
https://www.php.net/manual/en/pdo.query.php

- `PDOStatement::fetch()` — Fetches the next row from a result set.
https://www.php.net/manual/en/pdostatement.fetch

Uppgift

- Skriv ut namn och stad för alla kunder med hjälp av ditt php-skript, en kund per rad.

- Extra uppgift

    - Gör en webbsida som skriver ut kunderna i en tabell.

SQL injections

```
    <form method="post">
        <input type="text" name="username">
        <input type="password" name="password">
        <input type="submit" name="action" value="Logga in">
    </form>
```

```
    $username = $_POST['username'];
    $password = $_POST['password'];

    $query = "SELECT * FROM users WHERE user='$name' AND password =
'$password'";
```

```
$username = "Bobby';DROP TABLE users; -- ";
```

```
$query = "SELECT * FROM users WHERE user='$name' AND password =
'Bobby';DROP TABLE users; -- '";
```

Prepared statements

- Prepared statement is the only proper way to run a query, if any variable is going to be used in it.

- If at least one variable is going to be used, you have to substitute it with a placeholder, then *prepare* your query, and then *execute* it, passing variables separately.

SQL injections

```php
$stmt = $pdo->prepare('SELECT * FROM users WHERE email = ? AND status=?');
$stmt->execute([$email, $status]);
$user = $stmt->fetch();

// or

$stmt = $pdo->prepare('SELECT * FROM users WHERE email = :email AND status=:status');
$stmt->execute(['email' => $email, 'status' => $status]);
$user = $stmt->fetch();
```

När vi binder med `execute` kommer alla värden att bindas som strängar.

SQL injections

```php
/* Execute a prepared statement by binding PHP variables */
$calories = 150;
$colour = 'red';
$sth = $dbh->prepare('SELECT name, colour, calories
    FROM fruit
    WHERE calories < :calories AND colour = :colour');
$sth->bindValue(':calories', $calories, PDO::PARAM_INT);
$sth->bindValue(':colour', $colour, PDO::PARAM_STR);
$sth->execute();
```

Vi kan välja vilken datatyp vi vill binda mot genom att använda `bindValue()`.

Fetch

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

- `PDO::FETCH_NUM` returns enumerated array
- `PDO::FETCH_ASSOC` returns associative array
- `PDO::FETCH_BOTH` - both of the above
- `PDO::FETCH_OBJ` returns object
- `PDO::FETCH_LAZY` allows all three (numeric associative and object) methods without memory overhead.

Uppgift

- Lista ordernummer och det sammanlagda ordervärdet för varje order för alla kunder i Frankrike som har en kredit på minst $80.000.

IN-frågor

```php
$arr = ['S10_1678', 'S10_1949', 'S10_2016'];
$in  = str_repeat('?,', count($arr) - 1) . '?';
$sql = "SELECT * FROM products WHERE productCode IN ($in)";
$stm = $db->prepare($sql);
$stm->execute($arr);
$data = $stm->fetchAll();
```

```php
$arr = [1,2,3];
$in  = str_repeat('?,', count($arr) - 1) . '?';
$sql = "SELECT * FROM table WHERE foo=? AND column IN ($in) AND bar=? AND baz=?";
$stm = $db->prepare($sql);
$params = array_merge([$foo], $arr, [$bar, $baz]);
$stm->execute($params);
$data = $stm->fetchAll();
```

Sortera

```php
$orders  = ["name","price","qty"]; //field names
$key     = array_search($_GET['sort'],$orders); // see if we have such a name
$orderby = $orders[$key]; //if not, first one will be set automatically.
$query   = "SELECT * FROM `table` ORDER BY $orderby"; //value is safe
```

array_search

(PHP 4 >= 4.0.5, PHP 5, PHP 7)

array_search — Searches the array for a given value and returns the first corresponding key if successful

Description

```
array_search ( mixed $needle , array $haystack [, bool $strict = FALSE ] ) : mixed
```

Searches **haystack** for **needle**.

Return Values

Returns the key for **needle** if it is found in the array, **FALSE** otherwise.

Kombinera

```php
$sql = "SELECT productLine, productCode, productName FROM products ";

$orders  = ["productName","productCode","qty"]; //field names
$key     = array_search($_GET['sort'] ?? null, $orders); // see if we have such a name
$orderby = $orders[$key]; //if not, first one will be set automatically. smart enuf :)

if (isset($_GET['productLine'])) {
    $sql .= " WHERE productLine = '" . filter_input(INPUT_GET, 'productLine', FILTER_SANITIZE_STRING) . "' ";
}

$sql .= " ORDER BY $orderby ";
```

Stored Procedure

```php
$stmt = $pdo->prepare("CALL foo()");
$stmt->execute();
do {
    $data = $stmt->fetchAll();
    var_dump($data);
} while ($stmt->nextRowset() && $stmt->columnCount());
```

Password

- Sedan ett tag tillbaka har PHP ett par nya lösenordsfunktioner.
  - `password_hash()` – hashar lösenordet
  - `password_verify()` – verifierar ett lösenord mot dess hash
  - `password_needs_rehash()` – används för omhashning
  - `password_get_info()` - ger information om hashningen

password_hash(string $password , int $algo [, array $options])

- PASSWORD_DEFAULT
- PASSWORD_BCRYPT

```php
echo password_hash("rasmuslerdorf", PASSWORD_DEFAULT);
```

```
$2y$10$.vGA1O9wmRjrwAVXD98HNOgsNpDczlqm3Jq7KnEd1rVAGv3Fykk1a
```

password_verify (string $password , string $hash)

```php
$hash = '$2y$07$BCryptRequires22Chrcte/VlQH0piJtjXl.0t1XkA8pw9dMXTpOq';

if (password_verify('rasmuslerdorf', $hash)) {
    echo 'Password is valid!';
} else {
    echo 'Invalid password.';
}
```

Varför inte spara i klartext?

- Vad händer om någon kommer åt databasen?
  - Användaren kan läsa ditt lösenord.
  - Många användare använder dessutom samma lösenord på flera ställen.

Projekt

- Vi ska ägna resten av dagen åt att sätta ihop ett litet projekt med hjälp av PHP och MySQL.

- I grupper om 2-4 person ska vi bygga en ToDo-applikation.

  - Användare ska kunna skapa, redigera, stryka och ta bort todo-items från sin lista.

  - Om ni får tid över eller behöver en extra utmaning, låt användaren kunna lägga sina items i kategorier, sätta deadlines och kunna sortera efter dessa.

Utvärdering

- Prata i grupper om 2-3 personer i två minuter.

- Vad har varit bra idag?

- Vad skulle kunna förbättras?