CHT2520 Advanced Web Programming

Matthew Mantle m.e.mantle@hud.ac.uk

Today's Session - Object Relational Mapping (ORM)

Working with a Database

- In previous examples data from the database has been structured as associative arrays:

```
echo "<h1>{$film['title']}</h1>";
echo "<p>Year: {$film['year']}</p>";
echo "<p>Duration: {$film['duration']}</p>";
```

- It would be nice if it were structured as objects:

```
echo "<h1>{$film->title}</h1>";
echo "<p>Year:{$film->year} ({$film->getAge()} years old)</p>";
echo "<p>Duration:{$film->duration}</p>";
```

Why objects?

- Objects can have behaviour

```
$film->getAge();
```

- Easier to pass an object compared to lots of individual variables

```
save($title, $year, $duration);
save($film);
```

- We can use encapsulation

```
$film->year = 1800; //Invalid Argument Exception
```

Object Relational Mapping (ORM)

- Converting data from a database row to an object

```php
class Film {
    public $id;
    public $title;
    public $year;
    function __construct($title, $year){
        $this->title = $title;
        $this->year = $year;
    }
    function getAge(){
        return date("Y") - $this->year;
    }
}
```

```php
$resultset = $conn->query("SELECT * FROM films WHERE films.id = 3");
$row = $resultset->fetch();
$film = new Film($row["title"], $row["year"]);
echo "<p>The film {$film->title} is {$film->getAge()} years old.</p>";
```

ORM Design Patterns - Active Record

- See https://www.martinfowler.com/eaaCatalog/activeRecord.html
- The domain object is responsible for working with the database e.g. adding a new film to the database

```php
//get the data from the form
$title = $_POST['title'];
$year = $_POST['year'];
$duration = $_POST['duration'];
//Create a new instance of the Film class
$film = new Film($title, $year, $duration);
//Save the film object in the database
$film->save();
```

- Static methods for reading data

```php
$film = Film::find(3);
echo "<p>The film {$film->title} is {$film->getAge()} years old.</p>";
```

ORM Design Patterns - Active Record

- Laravel uses the Active Record pattern through Eloquent
- Advantage
  - All the complexity for working with the database is encapsulated in the domain class.
  - Intuitive to call methods such as `save()` and `delete()` on the actual object.
- Disadvantage
  - It tends to make our domain classes large and overly complex.
  - Breaks the 'single responsibility' principle.

ORM Design Patterns - Data Mapper

- See https://martinfowler.com/eaaCatalog/dataMapper.html
- A mapper object moves data between the database and objects, keeping them separate

```php
//get the data from the form
$title = $_POST['title'];
$year = $_POST['year'];
$duration = $_POST['duration'];
//Create a new Film object
$film = new Film($title, $year, $duration);
//Create a FilmMapper
$filmMapper = new FilmMapper();
//Ask the mapper to save the film in the database
$filmMapper->persist($film);
```

ORM Design Patterns - Data Mapper

- The Doctrine project uses the Data Mapper pattern
  - https://www.doctrine-project.org/
- Advantage
  - Keeps the database related code separate
- Disadvantage
  - More complex to use?
    - Have to coordinate the domain object with the mapper

Practical Work

- Adding ORM to the OO MVC Example
  - First using Active Record
  - Then using Data Mapper
- These are really simple examples
  - In reality ORM is much more challenging
    - When working with multiple tables and different relationships

Next Week

- Intro to Laravel
  - Clearly important
  - You must use Laravel for the Assessment
- Go through Assignment 1
  - In preparation have a look at the specifications for Assignment 1 and Assignment 2 (under Assessment on Brightspace)
  - There are very specific requirements for Assignment 1 which you must follow!