CHT2520 Advanced Web Programming

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

Today's Session - Introduction to Laravel

- 2nd session this week - go through the Assignment 1

The Examples From Weeks 1-4 Are Very Simple

- Good for understanding key principles, design patterns etc.
- Not good for real world use e.g.:
  - No user input validation
  - No security
  - No error checking

Most Web Applications Have Things in Common

- There are key features we expect most web applications to have:
    - MVC Architecture, Routing, Database Integration (ORM), User Input validation, Templating etc.
- A framework (e.g. Laravel) provides all these things for us
    - Don't re-invent the wheel

Third-Party Code

- Third-party code is code someone else has written
  - Saves times
  - Code has been tested
  - Code has been used in live projects
- We can categorise third-party code as a library or framework
  - Library - used for a specific tasks
  - Framework (e.g. Laravel) - a 'skeleton' we develop within
    - Inversion of control

Using Third Party Code is Not Strightforward

- Web projects often require multiple libraries (dependencies)
  - These libraries might be dependent on other libraries
- We need a way to manage and organise third party code
  - Composer - A dependency manager for PHP
  - You can view available libraries (Packages) at https://packagist.org/

Composer

- Composer is a dependency manager.
  - Composer is used to load and manage third-party code and libraries.
  - We need to install Composer before using Laravel.
- It has a Command Line Interface (CLI) e.g. the following command...

```
composer create-project laravel/laravel film-app
```

- Instructs Composer to download Laravel and all the needed dependencies into a folder called *film-app*.

Laravel

- Laravel is a free, open-source PHP MVC Framework.
  - Widely used to build real world projects.
- There are alternatives
  - In PHP e.g. Symfony, CodeIgniter.
  - In other languages ASP.NET MVC, Spring (Java), Django (Python), Ruby on Rails.

Using Laravel

- 'Convention over Configuration'
  - Pay careful attention to the location and naming of classes and files.
    - e.g. if we create a model class called *Film*, Laravel assumes there is a database table called *films* and sets up ORM for us automatically.
- The framework is in charge
  - Don't resort to plain 'vanilla' PHP unless you really have to
  - Ask '*how does the framework want me to solve the problem?*'

Practical Work

- The Basic CRUD example using Laravel.