

AdvancedWebDevelopment- JEvans-p151713

by Jonathan Ieuan Evans

FILE	4766_JONATHAN_IEUAN_EVANS_ADVANCEDWEBDEVELOPMENT-JEVANS-P151713_178145_290381486.DOCX (236.65K)		
TIME SUBMITTED	08-APR-2019 10:10PM (UTC+0100)	WORD COUNT	1461
SUBMISSION ID	104405553	CHARACTER COUNT	9338

Advanced Web Development

Assignment 2

Jonathan Evans P151713

8th April 2019

Contents

Contents.....	2
1. Introduction.....	3
2. Front controller and page controllers in MVC.....	4
2.1 Front Controller.....	4
2.2 Page Controller	6
3. Template Presentation Pattern	7
4. Usage of Symfony	8
5. References.....	11

1. Introduction

This report will analyse the functionalities of a front and back controller in an MVC (Model-View-Controller) framework. In addition to this, the templating pattern will be investigated. Following this, details on how Symfony implements certain functionalities in their MVC framework will be detailed and analysed.

2. Front controller and page controllers in MVC

An MVC (Model-View-Controller) framework is responsible for handling common website back-end operations that involve tasks such as getting data from a database model and displaying it on a page. It neatly separates data models, view logic, and controller logic into their own areas. [1]

This section of the report will investigate two core components of an MVC framework, the front controller and a page controller.

The functionalities of these controllers are ultimately what is responsible for serving pages to the user.

2.1 Front Controller

The front controller is the component that is responsible for serving the user with any given page. [2, p. 263] It includes operations that handle the request of every page.

This can allow a developer to implement procedures that may be desirable on every page, such as logging, internationalization, and security matters. With this, such behaviour is only required to be implemented once. Resulting in code that can be shared across pages, rather than including more operations on individual page controllers.

An example of a function provided in a front controller is shown on the following page. This function in particular handles the details on how a request is handled.

```
public function HandleRequest (Request $request)
{
    // 1. Identify required command for request
    $property = $request->GetProperty();

    switch ($property)
    {
        case "customer":
            $command = new CustomerCommand();

        default:
            $command = new DefaultCommand();
    }

    // 2. Execute command to get response
    $response = $command->Execute($request);

    // 3. Get the corresponding view
    $view = ApplicationController::GetView($response);

    // 4. Render the view
    $view->Render();
}
```

Figure 1 Example of a Front Controller in PHP (borrowed from University Moodle)

2.2 Page Controller

The page controller, like the front controller, is responsible for dealing with how a request is dealt with, however, rather than every page, a page controller is concerned with how individual pages are implemented. [2, p. 254] This is useful when the developer wants to implement functionality on a page that is not applicable for every page, such as calculations and displaying data objects that have been retrieved from a database and displayed in a given template.

An example of a page controller is shown below.

```
/**
 * @Route("/course")
 */
class CourseController extends AbstractController
{
    /**
     * @Route("/", name="course_index", methods={"GET"})
     */
    public function index(CourseRepository $courseRepository): Response
    {
        return $this->render('course/index.html.twig', [
            'courses' => $courseRepository->findAll(),
        ]);
    }

    /**
     * @Route("/new", name="course_new", methods={"GET", "POST"})
     */
    public function new(Request $request): Response
    {
        $course = new Course();

        $form = $this->createFormBuilder($course)
            ->add('title', TextType::class)
            ->add('Lecturer', EntityType::class, [
                'class' => Lecturer::class,
                'choice_label' => 'name',
            ])
            ->add('slug', TextType::class)
            ->add('save', SubmitType::class, [
                'label' => 'Create Course'
            ])
            ->getForm();

        $form->handleRequest($request);
    }
}
```

Figure 2 Example of a page controller

The page controller shown above is responsible for displaying a form within a page. It uses a built-in function of Symfony that allows developers to easily create and format a form.

3. Template Presentation Pattern

A template is a file that “renders information into HTML by embedding markers in an HTML page” as described by Fowler. [3] This allows the developer to create different templates that can embed database models on a page where desired. As a result, the developer does not need to duplicate code that makes database calls, rather, incorporate easy-to-read tags.

The picture below demonstrates a for loop that iterates over course objects and provides the corresponding column data per iteration. This utilises the twig templating engine in Symfony).

```
{% for courseItem in courses %}
<div class="course-list__course">
  <p class="course-list__course__name"><a href="#">{{ courseItem.name }}</a>
<div class="course-list__course__contact">
  <p>{{ courseItem.Lecturer.name }} <a href="#">{{ courseItem.Lecturer.email }}</a>
</div>
<div class="course-list__course__action">
  <a href="#" onClick="alert('Course {{ courseItem.name }}')">View</a>
</div>
</div>
{% endfor %}
```

Figure 3 Example of a Twig template in Symfony

More information on the benefits of templating is included in the following section where Symfony’s use of templating is investigated.

4. Usage of Symfony

For this report, a website has been created using the Symfony framework. Using Symfony allows for a developer to make use of tools such as ORM functionality, REST, and templating.

ORM

ORM (or Object Relational Mapping) allows a developer to create a database by specifying relationship entities in object form. An ORM tool will use these objects to create corresponding data entities in a database.

This functionality prevents the need for the developer to handle database-level operations (through using SQL to directly manipulate the database). This is useful as it grants the developer a few benefits – such as allowing the developer to change the database to another database supported by the ORM software (the developer is not required to change any database-specific code as they have not have written any code that is database specific).

Symfony incorporates ORM functionality through a tool known as Doctrine. [4] This tool allows the developer to read and write objects to and from a database without ever needing to specify any SQL, rather, entities can be specified as objects in a Symfony project, or created using a command line tool for easy creation of entities.

REST API

A REST API allows a developer to create a stateless API that allows other developers to query data objects within a web application. [5]

Many web platforms such as Twitter and Facebook allow developers to create widgets that can be displayed on their website. This can be code fully supplied by such platforms, or an API that allows the developer to query specific data from their website (in order to make their own widgets).

The benefit of using a REST API for the purpose of querying a website for data is that it is stateless and it provides a standard method for connectivity.

A stateless API implies that no prior information is required to retrieve objects from the API, rather, the developer can request any objects at any time (assuming that there are no restrictions of the use of the API).

Symfony (through use of the FosUserBundle extension) can allow a developer to easily create a REST API on database objects. [6]

Templates

Twig is a template engine that can be used for Symfony. With this, developers can create templates that allow data objects to be displayed within a page, allowing developers to create different layouts while keeping operations such as database interactions separate.

In addition to the convenience of creating easy to understand templates to insert data objects into a page, a templating engine also grants the developer two noteworthy advantages: page caching and security. [7]

Page caching and security with templates

Templates allow a developer to create a base .html.twig file. This is essentially a HTML file that allows developers to incorporate template functionality. With this, developers can extend another template file, which allows developers to create a base html file that can be used to extend pages on the website. Not only does this save the developer time, it allows Symfony to cache re-used parts of the HTML to create a new file to send to the user quickly. However, the benefit from this may only be observable on larger websites which gain a lot of traffic.

Not only does templating bring caching advantages, it also allows developers to escape output. This may be desirable to prevent malicious users from exploiting code on-page to execute an XSS (cross-site scripting) attack (this is an attack where a hacker can exploit functionality of a website to serve unsuspecting users with malicious scripts).



Convention over configuration

Symfony follows the paradigm “Convention over Configuration”. This paradigm means that the developer is ultimately required to code more and configure less. For example, Symfony does a lot of processes on the developer’s behalf, such as database manipulation and cross site scripting protection. As a result, the developer is only required to develop a website following the conventional procedures used in the Symfony framework – meaning that the developer does not have to worry about somewhat trivial matters such as how they are going to set something up. This will save the developer time as they can get their tasks done much quicker as less thinking of the website’s core procedures are required.

Testing

Unit testing is crucial when developing an application. A unit test consists of testing individual components of an application. For example, a Test Harness can be created to test if individual methods of an object can store given values (and may include other more complicated tests).

Symfony provides a library called PHPUnit that allows developers to unit test their code. This library allows the developer to easily create unit tests for their classes created in PHP, which is done through extending the TestCase class (which will allow the developer to validate their code by testing different cases against an expected result). An example of this is shown below:

```
1  // tests/Util/CalculatorTest.php
2  namespace App\Tests\Util;
3
4  use App\Util\Calculator;
5  use PHPUnit\Framework\TestCase;
6
7  class CalculatorTest extends TestCase
8  {
9      public function testAdd()
10     {
11         $calculator = new Calculator();
12         $result = $calculator->add(30, 12);
13
14         // assert that your calculator added the numbers correctly!
15         $this->assertEquals(42, $result);
16     }
17 }
```

Figure 4 Example of unit testing (taken from <https://symfony.com/doc/current/testing.html>)

5. References

- [1] G. E. Krasner, S. T. Pope, and P. Systems, 'A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System', p. 34.
- [2] M. Fowler, *Patterns of enterprise application architecture*. Boston: Addison-Wesley, 2003.
- [3] 'P of EAA: Template View'. [Online]. Available: <https://martinfowler.com/eaCatalog/templateView.html>. [Accessed: 08-Apr-2019].
- [4] 'Databases and the Doctrine ORM (Symfony Docs)'. [Online]. Available: <https://symfony.com/doc/current/doctrine.html>. [Accessed: 08-Apr-2019].
- [5] 'Understanding And Using REST APIs — Smashing Magazine'. [Online]. Available: <https://www.smashingmagazine.com/2018/01/understanding-using-rest-api/>. [Accessed: 08-Apr-2019].
- [6] 'Getting Started With FOSRestBundle (Symfony Bundles Docs)'. [Online]. Available: <https://symfony.com/doc/master/bundles/FOSRestBundle/index.html>. [Accessed: 08-Apr-2019].
- [7] 'Creating and Using Templates (Symfony Docs)'. [Online]. Available: <https://symfony.com/doc/current/templating.html>. [Accessed: 08-Apr-2019].
- [2] 'P of EAA: Template View'. [Online]. Available: <https://martinfowler.com/eaCatalog/templateView.html>. [Accessed: 08-Apr-2019].

FINAL GRADE

72/100

GENERAL COMMENTS

Instructor

Investigation of Fowler (30) 18

Investigation of Symfony (20) 15

Quality (20) 16

View handling (10) 9

Database handling (15) 13

Software testing (5) 1

The discussion of the Fowler patterns was very brief and lacked any evaluation. You did provide some examples, which was a good idea but you did not describe the workings of the pattern in any detail.

The Symfony discussion was a lot better, but it would have been clearer if you had provided more implementation examples or evidence of the benefits of the Symfony approach.

Code is built in Symfony ;and demonstrates its usage in a number of controllers. I would like to have seen a little more database searching. There is some discussion of testing but not much evidence of testing being performed.

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8



Comment 1

You need to expand on this point.



Comment 2

Hmm, OK. But how does this work?

PAGE 9



Comment 3

How about an example to illustrate this concept.

PAGE 10

PAGE 11
