

Mircea-Radu-Serban Dobrau

Advanced Web Development

Report Sem 2

by Mircea-Radu-Serban Dobrau

FILE	2787_MIRCEA-RADU-SERBAN_DOBRAU_MIRCEA-RADU-SERBAN_DOBRAU_ADVANCED_WEB_DEVELOPMENT_REPORT_SEM_2_93284_1762281040.DOCX		
TIME SUBMITTED	23-MAR-2018 10:01PM (UTC+0000)	WORD COUNT	2354
SUBMISSION ID	85491191	CHARACTER COUNT	13538

Mircea-Radu-Şerban Dobrău

P147987

Web Development (BSc)

Advanced Web Development (M6X01086)

Michael Dacey

Abstract

Whilst discussing the Front Controller, Martin Fowler, in his book titled "Patterns of Enterprise Application Architecture" [1], states that in a complex website security, internationalization, and providing particular views to users are all similar processes which the developer must do when handling a request. The report starts with an evaluation upon the scope and use of the Page and Front controllers and their integration in the MVC pattern, as well as a description of Template Views. Next, a brief rationale for using PHP frameworks in building complex web applications is presented. Following this, a brief investigation of 5 notable PHP MVC frameworks is presented, in chronological order: CakePHP, Symfony, Zend Framework, Yii and Laravel, presenting several notable features of each framework. Finally, the report ends with an outline of benefits and drawbacks of 3 PHP frameworks.

Table of Contents

Fowler and MVC	4
Page Controller	4
Front Controller	5
Template Views	6
Page Controllers, Template Views and MVC	7
Investigation of PHP Frameworks	8
Laravel, Symfony and CakePHP compared	10
References	12

Fowler and MVC

Page Controller [1, pp. 333–343]

The Page Controller consists of one single input controller for each logical Web page (or document file) of the Web site. The page controller can either be the page itself or a separate object that corresponds to the page. The basic implementation requires one “module” per document/page, however a more strict way of approach page controllers is to tie each one of them to a specific action such as clicking a link or a button.

It can either be made as a script or as a server page. The server page most commonly consists of the controller alongside with the document's view in the same file. This implementation, however, can result in unstructured script code if it involves more complex logic such as pulling data out of the request or deciding which view to display.

The basic responsibilities of the page controller consist of the following:

- Decode URL for form data required for the specific action.
- Create and invoke the model objects required for processing the data. The relevant data found in the HTML request must be passed to the model so that the model object's do not require any connection to the HTML request.
- Determine which of the views should display the result page and forward the model information to it.

The page controller doesn't necessarily have to be a single class. Alternatively, it can invoke helper objects. The server page can alternatively invoke a helper object for handling all of the logic required. Once done, the helper may return control to the original page or forward to a different server page which acts as the view. In this case, the server page is the request handler whilst the controller logic lies in the helper.

Furthermore, another approach to handling script code is by making a script the handler and the controller, in which case the script carries out the controller's responsibilities after which it goes to the appropriate view for displaying results.

Page controllers' best use case scenario involve simple web pages in which controller logic is very simple. In such cases, most URLs are handled with server pages and more complicated situations with object helpers. Additionally, page controllers do not pose much conflict when intermixed with Front Controllers. [1, pp. 333–335]

Front Controller [1, pp. 344–349]

The Front Controller consists of two parts which are the Web handler and a command hierarchy. The Web handler is an object whose scope is to receive POST or GET requests from the Web server. In other words, it handles all of a Web site's calls. It handles URLs in a minimal fashion, retrieving enough information from it and its subsequent request so that it can afterwards decide the action that needs to be initiated, after which it delegates to a command which carries out the required action.

As for its structure, it is very commonly implemented as a class rather than as a server page, for it doesn't produce any **response** back to the caller. Furthermore, the commands in turn are classes as well as they also do not require any information in regard to the Web environment, even though they get passed the HTTP information.

The web handler has two modes of deciding which command should be run: either statically or dynamically. The static approach involves a simple parsing of the URL combined with conditional logic. The dynamic approach, alternatively, takes a standard piece of the URL and uses dynamic instantiation for making a command class.

The static approach has a few advantages, namely compile time errors checks on dispatch, URL flexibility, as well as an explicit logic. On the other hand, dynamic invocation allows the developer to put the name of the command class inside the URL or bind URLs to command class names, which makes it easier to change class names without searching through one's Web pages.

A useful way of handling issues such as login, authentication and internationalization is by using the Front Controller in conjunction with the "Intercepting Filter" pattern. This pattern is a decorator which wraps the handler of the front controller, allowing the developer to build a pipeline of filters which in turn are setup up dynamically at runtime.

As there only one Front Controller must be configured into the Web server and newer commands can be added dynamically, the problem of duplicate code has been solved.

Template Views [1, pp. 350–360]

Template views represent HTML pages containing some type of markers, which in turn output some type of computation such as a database query. There exist multiple ways of embedding markers inside an HTML file, such as the server page approach which lets the programmer embed programming logic into the page, although it isn't recommended for the following reasons:



- It hinders web designers or non-programmers from editing the page properly
- Different layers of functionality are mingled together in a single page

Additionally, conditional page behaviour is also possible within template views although still not recommended for the reasons above.

A way through which the presentation layer of the page and its underlying functionality can be decoupled is by providing each page with a helper object. The page's only reach to this object is through calls. The helper object in turn contains the real programming logic of the webpage, including conditional page behaviour.

Page Controllers, Template Views and MVC [1, pp. 330–333]

The MVC pattern is a software design pattern whose purpose is to split user face interaction into three distinct roles: The Model, the View, and the Controller.

The Model is a non-visual object which contains all the data other than that which is used in the UI layer.

The View is the UI display of the model.

Finally, the controller handles changes to the information; For example, the controller handles model manipulation upon user input. After changes to information have been received the controller updates the view.

The separation of the model from the view brings several benefits:

- Easier development of multiple views/presentations relying on the same model as users might want to see model information in different ways.
- Test domain logic easily without resorting to view manipulation or scripting.
- Clear separations of concerns leading to each team member having better focus on a single layer of the application of which they are more specialized.

Investigation of PHP Frameworks

The rationale behind using PHP frameworks is that they provide common functionalities and classes in the form of helpers, thus reducing development time. Additionally, they provide robustness, scalability and especially maintainability to applications [2] as well as ease of use by way of providing usable libraries. [3]

Below is a brief investigation of 5 notable frameworks, sorted in chronological order by initial release date: CakePHP, Symfony, Zend Framework, Yii and Laravel.

CakePHP (2005) [4, p. 1044]: Offers a solution for Object-Relational Mapping (ORM), a class for sanitizing user input, salts against Cross-Site-Request Forgery (CSRF) attacks, as well as some solutions for Content Management Systems (CMSs).

Additionally, it is argued that it is very intuitive and fast for beginner developers with easily codable forms and an easily extendable model class AppModel.

Symfony (2005) [5, pp. 32–33]: A highlight of Symfony is its different approach to release support in comparison to other PHP frameworks: Symfony's standard release version is maintained for eight months, whilst its Long-term Support (LTS) versions are supported for three years.

Symfony embraces the "Don't reinvent the wheel" philosophy, making heavy use of existing PHP open-source project such as:

- **Doctrine or Propel** as the ORM layer and PDO database abstraction layer
- **PHPUnit** for unit testing
- **Twig** as the templating engine
- **Swift Mailer** as the e-mail library

As well as automatic generation of complex service Web Services Description Language code.

Finally, another highlight of the Symfony framework is its interoperability; One can choose to use the full-stack Symfony framework, the Silex micro-framework or individually selected components, thus enabling each developer to pick only what is required for his specific needs.

Zend Framework (2006) [4, p. 1045]: Arguably a library rather than a framework, Zend is loosely coupled. Additionally, it has no support for ORM. It is however easily flexible and extensible with the use of plugins and Zend_Plugin; It encapsulates form elements as decorators whilst also providing a few functions and filters for sanitizing user input. Next, it comes with PHPUnit integrated for testing and provides support for a few enterprise-level CMSes.

Yii (2008): Comes with tightly integrated support for the PHPUnit and Selenium testing frameworks [6], as well as data, fragment, page, and HTTP caching [5, p. 37]. Furthermore, it also comes with localization, translation, and locale-sensitive formatting of dates, time, and numbers.

Next, it provides developers with security measures against SQL injection, Cross-site scripting (XSS), Cross-site request forgery (CSRF) and cookie tampering.

Finally, some other defining features of the Yii framework include a GUI tool called "Gii" which provides a web-based code generator for fast setup of models, forms, modules, CRUD operations etc. as well as its extensive collection of available extensions [7] [5, p. 37]

Laravel (2011) [8, pp. 1–6]: Based on Symfony, Laravel is built on top of over 20 different libraries and is itself split into individual modules. As it is tightly integrated with PHP's "Composer" dependency manager, these components can be updated with relative ease.

Furthermore, it is built from the ground up with the scope of easing testing; For example, it can visit routes defined in tests, crawl HTML, ensure that methods are called on certain classes and impersonate authenticated users.

Next, it has ORM and ActiveRecord for defining models and model relationships, as well as a default implementation for authentication and e-mail.

Other features include Redis, Queues, Schema builder and PHP 5.4+ feature support such as namespaces, interfaces, overloading and anonymous functions, as well as a command line tool called Artisan [5, p. 31].

Laravel, Symfony and CakePHP compared

Laravel Benefits [9, p. 5]: One of the advantages of Laravel is that it is itself composed of individual modules. Next, it is built from the ground up for the purpose of easy testing, including features such as impersonating authenticated users and HTML crawling.

Finally, Laravel comes with features which abstract building queries and interconnection of modules.

Laravel Drawbacks: Average response times for select operations using Laravel's Eloquent ORM is much higher than raw SQL. [10, p. 19] As can be deduced, raw SQL is ideal for working with huge databases or performance-critical databases

Symfony Benefits [11]: Firstly, Symfony can recognize different environments in which an application may function. It allows the developer to accommodate the application to each type of environment (development, test, staging and production) easily.

Next, Symfony supports a large amount of debugging features such as `php.config` information, HTTP headers, parameters passed to the current view template, as well as extensive logging. Additionally, with Symfony the developer can also use placeholders or “slots” inside views for extensive testing.

Symfony Drawbacks: One of the drawbacks of using Symfony is its large time required for the execution of code as well as its memory usage. [12]

Secondly, some plugins support only older Symfony versions and/or lack any documentation. [4, p. 1043]

CakePHP Benefits [4, p. 1044]: Some of the advantages include CLI tools for testing, built-in sanitizer, great documentation and easy coding of forms.

Arguably a benefit or a drawback depending on time constraints or the developer's taste is that CakePHP approaches the “convention over configuration” approach, claiming that CakePHP requires no configuration to setup as well as having very strict hierarchical and naming conventions. [13]

CakePHP Drawbacks [4, p. 1045]: Slow performance and coupling of controller and model. Additionally, the convention over configuration approach might present a significant drawback in scenarios which require the developer to have more freedom.

References

- [1] M. Fowler, *Patterns of Enterprise Application Architecture*. Addison-Wesley Professional, 2003.
- [2] A. R. Fayyaz, M. Munir, and B. Carlsson, "Performance Evaluation of PHP Frameworks (CakePHP and CodeIgniter) in relation to the Object-Relational Mapping, with respect to Load Testing," 2014.
- [3] H. Nylén and N. D. Fogelström, "PHP Framework Performance for Web Development," 2012.
- [4] B. Porebski, K. Przystaliski, and L. Nowak, *Building PHP Applications with Symfony, CakePHP, and Zend Framework*. John Wiley and Sons, 2011.
- [5] B. Ajzele, *Modular Programming with PHP 7*. Packt Publishing Ltd, 2016.
- [6] J. Winesett, *Web Application Development with Yii and PHP*. Packt Publishing Ltd, 2012.
- [7] "Extensions | Yii PHP Framework." [Online]. Available: <http://www.yiiframework.com/extensions/>. [Accessed: 27-Jan-2018].
- [8] M. Stauffer, *Laravel: Up and Running: A Framework for Building Modern PHP Apps*. O'Reilly Media, Inc., 2016.
- [9] M. Bean, *Laravel 5 Essentials*. Packt Publishing Ltd, 2015.
- [10] I. Jound, H. Halimi, S. Mikael, and J. Conny, "Comparison of performance between Raw SQL and Eloquent ORM in Laravel," 2016.
- [11] R. Grehan, "Fabulous PHP frameworks: Symfony," *InfoWorldcom San Mateo*, Jan. 2011.
- [12] J. Samra, J. Hagelbäck, and S. Pllana, "Comparing Performance of Plain PHP and Four of Its Popular Frameworks," 2015.
- [13] R. Grehan, "Fabulous PHP frameworks: CakePHP," *InfoWorldcom San Mateo*, Jan. 2011.

Mircea-Radu-Serban Dobrau Advanced Web Development Report Sem 2

GRADEMARK REPORT

FINAL GRADE

82/100

GENERAL COMMENTS

Instructor

Investigation of Fowler (30) 21

Investigation of MVC frameworks (20) 18

Quality of website and techniques used (25) 20

View handling (10) 10

Database handling (15) 13

Good description of Fowler, but you have missed the point when discussing Template Views by focusing on embedding program logic within the view.

Good evaluation of frameworks, you may have used the initial characteristics of MVC a little more to frame your comparison of the frameworks.

Good Symfony application containing an number of entities, views and controllers. I am satisfied that you have a good grasp of Symfony. Well done.

PAGE 1

PAGE 2

PAGE 3

PAGE 4



Comment 1

??

PAGE 5



Comment 2

*response

PAGE 6



Comment 3

This is the worst approach and fundamentally misses the point about Template Views. The "tags" should focus on presentation and avoid programming constructs e.g. Razor in ASP.Net or TWIG in Symfony.

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12
