

JenAdaAdvancedWeb2

by Jennifer Ada

FILE	3017_JENNIFER_ADA_JENADAADVANCEDWEB2_93284_1889845722.DOCX		
TIME SUBMITTED	24-MAR-2018 12:00AM (UTC+0000)	WORD COUNT	937
SUBMISSION ID	85500288	CHARACTER COUNT	5012

Investigation of Fowler

Front Controllers

In his book "Patterns of Enterprise Application Architecture" [1], Fowler says that there are similar tasks such as "security, internationalization, and providing particular views for certain users." [1], if not handled properly will result in input controller behaviour "scattered across multiple objects"[1] and behaviour duplicated. This necessitates a pattern such as Front Controllers.

Evaluation of Front Controllers in web applications.

Front Controller

It handles all calls for a Web site, and consists of a Web Handler which receives requests from the Web server. It reads information from the URL and the request passed to decide what to do then it sends the a command to carry out that action[1].

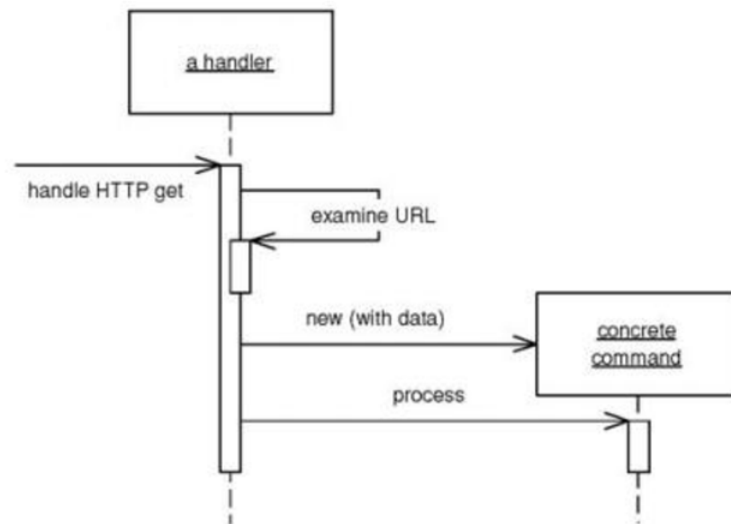


Figure 1 How the front controller works.[1]

Investigation of MVC frameworks

Argument for using PHP MVC frameworks in the development of complex applications.

One of the primary benefits to using the MVC design pattern properly is the adhesion to the DRY principle. DRY is an acronym for "Don't Repeat Yourself" and encourages code reuse where feasible, barring edge cases where optimisation necessitates copies of code be in more than one place. This is not exclusive to PHP and also appears in Ruby on Rails amongst countless other web languages[2].

The DRY principle is also not exclusive to MVC, inheritance in object-oriented-programming (OOP) for example also adhere to DRY. This is most explicit in the MVC framework in the way in which inheritance is used to create instances of pairs of views and controller which can use the same model.

An example of this would be an authorisation interface for the critic rating website. Administrators that log in could be presented with a view and controller which allow them to add new critics to the database. In contrast, non-administrators would be presented with a view and controller which allows them to rate critics, effectively updating the critic table or associated table in the same model.

In addition and complementary to the DRY principle, orthogonality is also identified as an important benefit of the MVC pattern for web applications[3]. Whereas DRY avoids duplication of code, orthogonality avoids interdependency among code. These two principles combined make MVC “flexible, more understandable, and easier to debug, test, and maintain.”[3]

Unfortunately, MVC has been a victim of its own popularity[3]. Many frameworks which claim to use MVC or similar architectural patterns do not adhere to design principles the perusal of qualities espoused by MVC can also lead to unnecessary decoupling and overcomplication of the solution when abused[4].

Architectural patterns such as MVC can be used to solve problems inherent to web applications. The use of architectural patterns designed for handling the demands of interactive and multifunctional programs make much of the functionality possible to realise through modularisation and communication as part of their inherent design.

As more distributed, multi-core and asymmetric processing architectures are developed, further challenges will present themselves in terms of both programming these applications to mitigate wasted processing resources, and assurance that threads etc. do not fail due to concurrency issues. As web application functionality in terms of complexity and sophistication continue to increase, it only makes sense to further develop practices for design patterns to ease the development and use of web applications. There are variations even among MVC but it is prudent to examine whether a solution is what it claims to be.

Comparison of the main benefits and drawbacks of three MVC frameworks.

'Analysis and practical Application of PHP Frameworks in Development of Web Information Systems' [5], looked at CakePHP2, CodeIgniter, Symfony2, Yii and Phalcon PHP frameworks. All supported MVC with the exception of Symfony2, were "except Symfony2 are based on the MVC architecture style" [5].

Phalcon was the MVC framework which was studied most in depth, it is written as a PHP extension using C based on the principles of MVC. It is supposed to be "high performance and low server resource expenses" (sic) [5] so much so that testing of Phalcon has shown it to be one of the fastest performing frameworks beating Symfony2 (a popular PHP framework) by executing ~230 requests in the time it takes Symfony2 to do ~49 requests. It has ORM, flexible routing with multiple options based on URL mapping. Finally, regular expressions can be used to validate input.

CakePHP is examined in more depth in An Empirical Study of the Evolution of PHP MVC Framework[6]. It notes that CakePHP is based on concepts from Ruby on Rails with MVC being implemented in PHP. It uses "Convention over configuration, Active Record, Association Data Mapping, and Front Controller" [6].

It examines Laravel in more depth, which is another MVC framework written in PHP[6]. Laravel managed approximately 3000 requests a second compared to Cake PHP which achieved less than 1000. Laravel also supports ORM with active record[6].

- [1] M. Fowler, D. Rice, M. Foemmel, E. Hieatt, R. Mee, and R. Stafford, *Patterns of Enterprise Application Architecture*. 2002.
- [2] S. Meenakshi, "Ruby on Rails – An Agile Developer's Framework," *Int. J. Comput. Appl.*, vol. 112, no. 1, pp. 975–8887, 2015.
- [3] A. Hunt and D. Thomas, *The Pragmatic Programmer: From Journeyman to Master*, 1st ed. Addison Wesley, 1999.
- [4] M. Hameed, M. Abrar, A. Siddiq, and T. Javeed, "MVC Software Design Pattern in Web Application Development," *Int. J. Sci. Eng. Res.*, vol. 5, no. 5, 2014.
- [5] N. Prokofyeva and V. Boltunova, "ScienceDirect Analysis and Practical Application of PHP Frameworks in Development of Web Information Systems," *Procedia Comput. Sci.*, vol. 104, pp. 51–56, 2017.
- [6] R. F. Olanrewaju, T. Islam, and N. Ali, "An Empirical Study of the Evolution of PHP MVC Framework," Springer, Cham, 2015, pp. 399–410.

FINAL GRADE

48/100

GENERAL COMMENTS

Instructor

Investigation of Fowler (30) 6

Investigation of MVC frameworks (20) 13

Quality of website and techniques used (25) 16

View handling (10) 5

Database handling (15) 8

The discussion of the Front controller is very brief and lacks detailed analysis.

I liked your initial arguments for/against the use of MVC. Unfortunately your analysis of the frameworks doesn't provide enough details to determine how they compare against each other and their approaches to MVC.

For the code, you have used a simple MVC framework obtained online, which you have referenced and have created a small application.

PAGE 1

PAGE 2

PAGE 3

PAGE 4
