Volume 6, Issue 2, February 2016 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Comparison of Procedural PHP with Codeigniter Framework

Ripunjit Das
BTech Student, CSE Dept.
Assam Downtown University, Assam, India

Dr. Lakshmi Prasad Saikia
Professor & HOD, CSE Department
Assam Downtown University, Assam, India

Abstract— The objective of this study is to evaluate performance of codeigniter with the plain PHP. The decision of making this study was based on the fact that there is a lack of comparison tests between the most popular PHP framework codeigniter and plain php. Many studies have been conducted between the frameworks but none have been done between a framework and procedural php. Visitors nowadays have less patience to wait for a website to load. Meanwhile, PHP Framework has become known amongdevelopers, but the part of the performance that reduces the load time even more so visitors can surf without any problems, is missing. Therefore, it is a good idea to try to discover how the performance of the PHP Framework can change and improve the visitor experience. The result outcome of these experiments has been analyzed and interpreted in order to expose the performance of the targeted frameworks.

Keywords— MVC, Procedural php, Codeigniter, Memory usage, Execution time

I. INTRODUCTION

Originally, the World Wide Web was imagined as a means for sharing information where documents are linked together in an inter-network (Berners-Lee & Cailliau, 1990). These documents were basically static. As growing, forms were introduced, helping users to interact with servers. Soon after, the first Web application was created with the born of server-side scripting language making dynamic generation of HTML documents possible. Until now, the web technology has reached to a new standard where web applications have full abilities of desktop applications. A group of dynamically-generated web pages is the most common way to represent web applications. The online market is emerging required developers to produce better featured application within less amount of time. Web application frameworks are created with this purpose in mind. As companies have applied and succeeded in saving time and boosting applications' features, frameworks have proven its ability to cope with the market demands.

PHP: Hypertext Preprocessor is well known scripting language often associated with web development even though it has other areas of usages. According to w3techs.com PHP is the most common used scripting languages on the Internet with 82% coverage. Many frameworks based on PHP popped up during the last decade. Frameworks like CodeIgniter, Symfony, Phalcon and Laravel are widely used and according to sitepoint.com they are four of the most promising frameworks in 2014. The first thing to recognize when visiting the webpages of these frameworks is the promises, which the people behind will guarantee us.

When visiting Laravel's website, the first thing are slogans like beautiful code, rapidity and speed. Phalcon's Team claims that their framework is the fastest. Symfony's website title says "High performance PHP framework for web development". CodeIgniter's team states that their framework is "powerful with a very small print".

A developer aiming to develop a project where PHP is required as the development language and high performance is one of the highly prioritized requirements in the requirements specification, how would the decisions be made on whether choosing the plain PHP stack or a PHP framework in the development? In the case of using the framework, which one of these above mentioned would be suitable bearing in mind the performance requirements? To answer these questions this study has been conducted in which two functionally equivalent blogging web applications have been developed and subjected to an experiment in which the performance of each of the mentioned frameworks is measured and evaluated. The first web application will be developed in plain PHP. The other one will be developed in the PHP framework CodeIgniter. After implementing the applications, performance metrics are measured on all these versions of the web application. These measurements will consist of (i) Execution time of CRUD-functionality (ii) Memory usage for each of them. The results from the experiments are analyzed and interpreted in order to become the basis of the decisions and conclusions of this study.

II. LITERATURE SURVEY

Our study included more or less different types of strings like "web development evaluation" and "web development performance". In the findings, most of the literature are about the evaluation in general, not about performance in web development. The search for data was about performance evaluation and the selection of literature research in this field was not plentiful but easy to find. The important is a good way on how they did the research to find out the information in the data, and this was often found in the abstract but sometimes in the background.

In 2001, Samisa Abeysingha, described the focus is mainly on how the MVC and framework techniques make a difference in the web development. This book has MVC as a big part of PHP development and is about what it does for web development. The book helps the developers in learning the MVC structure more easily

In 2001, Mohammed Musthafa, defines **Analysis of Model-based MVC Framework for PHP Development CodeIgniter** is an analysis of how the MVC based framework Codeigniter is doing in the development of performance and of user ability for developers. Here it is explained how MVC structure helps in the development of a web application more easily, how the developers can develop more efficient web application .

In 2001, Peter Sevcik and **John Bartlett** described **Understanding Web Performance**. They described present how performance can be used and its importance in the web business. It also brings up how performance can be tested and what can impact the tests. This paper gives the developers how cost and time can be managed in the development of a web application.

In 2003, Lance Titchkosky et. al., published the paper named **A Performance Comparison of Dynamic Web Technologies,** in which there is a description of the different impacts on performance, comparative analysis of different performance types and how they can be done.

In 2012, Nguyen Minh Thanh wrote about **Adopting Web Framework in Web Application Development**, where he had discussed about the advantages of web frameworks in web application development.

In 2015, David Díaz Clavijo wrote about **practical comparison of Agile Web Frameworks,** in which he compared among the codeigniter, cakephp and laravel framework. He also made a review about the different web applications developed by using this framework. There it also explained which framework should be chosen by a developer for a efficient website development.

III. METHODOLOGY

In this study an experiment is conducted in which five functionally equivalent web applications representing a simple blog are developed. The first web application is developed using plain PHP and the second one is developed using CODEIGNITER framework.

In the experiments, execution time and the consumed memory usage when executing the CREATE, READ, UPDATE and DELETE (CRUD) implementation in each web application of the experiment will be measured. The memory usage and the execution time is measured in plain php by using the built in functions in php scripts. The codeigniter framework also has the built in functions for memory usage and execution time measurement.

The research answers the basic question:

How does memory usage and execution time differs for a simple blog web application developed in plain php and the one developed with codeigniter framework.

IV. SIMULATION

The two web applications developed will be tested in terms of their execution time and memory usage. The two measurements have been done on the basis of CRUD i.e. create, read, update, delete. These are the basic function of any web application.

Create action- The measurement starts in the create-blog-entry view when the user has pressed the create-blog-entry button after the form has been filled in. Once done, the newly created blog entry is presented in view-blog-entry view and the measurement is stopped.

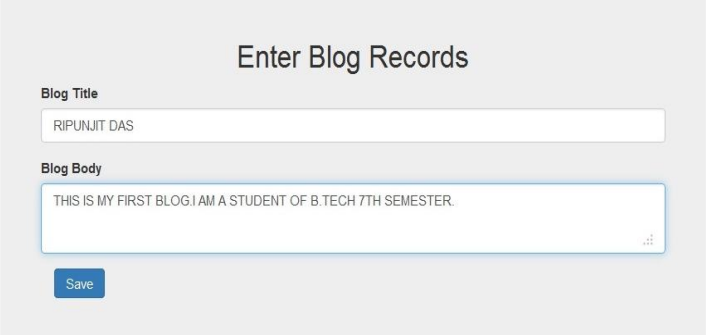

Fig.1: Create Blog Interface

View action- The measurement starts when the link responsible to show the blog entry is clicked and ends when the controller views the blog entry. This happens in the index page where only one blog entry is created. Once the view is rendered, the measurement is stopped.

Fig.2: View Blog Interface

Update action- The measurement starts when the updating input data is posted from the update-blog-entry view using a submission button. The measurement ends when the updated blog entry is shown in a view-blog-entry view.

Fig.3: Update Action Interface

Delete action- The measurement starts when the delete button in a view-blog-entry view is pressed and ends after the index page has been rendered completely with the information confirming that the deletion was successful.

Fig.4. Delete Action Interface

V. RESULT AND ANALYSIS

T he results from the conducted experiments are presented and analyzed. The presentation is outlined in a tabular form in which the horizontal columns represent the tested actions Create, Read, Update and Delete. For each tabular form there is a corresponding visual presentation in a form of a graph.

Execution time experiment result- The results show that codeigniter has more execution time during the create and update action but it takes less time during view and delete action. Although plain php is only fast by a few milliseconds which will be almost ignored by the user. The results of the ten inputs as a mean for the four actions are presented below:

Table 1: Execution time experiment result

TESTED TARGETS	CREATE	READ	UPDATE	DELETE
PLAIN PHP	0.066	0.063	0.0925	0.065
CODEIGNITER	0.0634	0.0747	0.0825	0.616

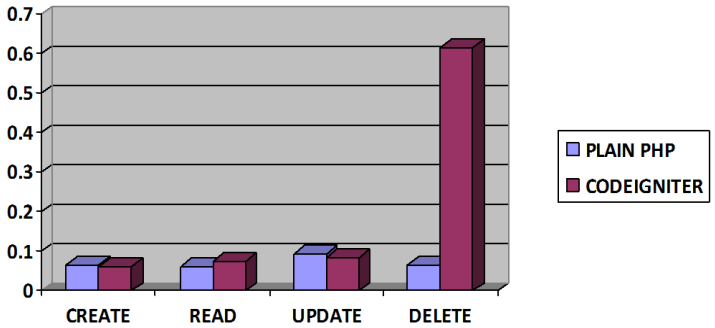

FIG 5: Execution Time experiment Representation Graph

Memory Usage Experiment Result- The results of memory usage experiment is such that codeigniter uses a fixed memory for all the four actions of create, read, update and delete, but the memory used by plain php blog varies everytime for all the actions. The values have been plotted in the tabular form. The tabular form is also represented in visual form as a graph.

Table 2: Memory Usage experiment results table

TESTED TARGETS	CREATE	READ	UPDATE	DELETE
PLAIN PHP	3.24	3.67	3.45	3.80
CODEIGNITER	3.78	3.78	3.78	3.78

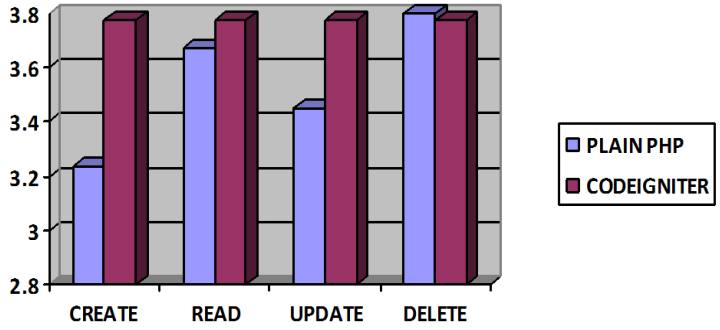

FIG 6: Memory usage experiment results visual representation

VI. CONCLUSION

From the study and the simulation we come to the conclusion where the study will be concluded based on two research questions:

(1) How does execution time and memory usage differ for a simple blog web application written in plain PHP and the Codeigniter framework?

(2) What will be more reliable for the development of web applications?

It would initially be stated that the following text would be considered as an answer for both research questions. The reason for that is that the questions are conditional, where the second research question(2) is depended on the outcome of the first research question(1). So therefore, the result analysis for the plain PHP and Codeigniter shows that the differences between them are relatively small to plain PHP's advantage. The plain PHP is only a dozens milliseconds faster and consumes a few bytes less memory than Codeigniter.If the plain PHP application was large, it would mean that the plain PHP would not have been performing better than the Codeigniter. That is because the growth of the stack trace in the plain PHP application would have caused a higher execution time and memory consumption. Plain PHP is mostly suited for smaller projects even if there are no obvious reasons to not use Codeigniter, but Codeigniter would be the preferred choice when developing larger projects.

REFERENCES

[1] Samisa Abeysingha , "Php Team Development", packt publishing, October 2001
[2] Mohammed Musthafa, "Analysis of Model based MVC framework for php development", Academics Education conference,2008
[3] Peter Sevcik and John Bartlett, "Understanding Web Performance", Business Communications Review, October 2001
[4] Martin Artlitt, "A performance Comparison of Dynamic Web Technologies", ACM sigmetrics,December 2003
[5] Nguyen Minh Thanh, "Adopting Web Framework in Web Application Development", Spring 2012
[6] David Díaz Clavijo, "A practical comparison of Agile Web Frameworks", 2014 [21-45].
[7] M. Alsadat Kazemi and S. Eskandari, "Web-based application for Collaborative Ethical Decision Making" Dep. of Information Technology, Uppsala Universitet, 2013.
[8] The Open Group Base Specifications Issue 6, IEEE Std 1003.1. "clock - report CPU time used" opengroup.org http://pubs.opengroup.org/onlinepubs/000095399/functions/clock html
[9] C. Wang, "Web Services Integration Method" Dep. of Computer Science and Engineering, Chalmers University of Technology, 2010.
[10] PHP Documentation. "PHP's memory_get_usage function." http://php.net/manual/en/function.memory-get-usage.php
[11] Xdebug documentation. "Xdebug's stack trace feature" xdebug.org. http://xdebug.org/docs/stack_trace
[12] PHP documentation. "History of PHP" php.net. http://php.net/manual/en/history.php.php
[13] D. Powers, "PHP Solutions: Dynamic Web Design Made Easy". Apress, 2006.
[14] W3techs. "W3Techs - World Wide Web Technology Surveys", w3techs.com.
[15] Packagist. "PHP's package manager" Packagist.org.

