

# Comparison of Procedural PHP with Codeigniter and Laravel Framework

Ripunjit Das<sup>1</sup>, Dr.Lakshmi Prasad Saikia<sup>2</sup>

<sup>1</sup> B.Tech Student, Computer Science & Engineering, Assam Downtown University, [ripunjitdas@gmail.com](mailto:ripunjitdas@gmail.com)

<sup>2</sup> Professor & HoD, Computer Science & Engineering, Assam Downtown University, [lp\\_saikia@yahoo.co.in](mailto:lp_saikia@yahoo.co.in)

---

**Abstract**—The objective of my study is to evaluate the performance of procedural PHP with codeigniter and laravel framework. The decision of making this study is based on the fact that there is a lack of comparison tests between the popular PHP frameworks like laravel, codeigniter with plain php. The result outcome of this experiment have been analyzed and interpreted in order to expose the performance of the targeted frameworks.

**Keywords**—*Procedural PHP, Laravel, Codeigniter, Execution time, Memory Usage*

---

## 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 the development of the frameworks promises.

When visiting Laravel's website the first thing we saw 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

The literature research included more or less different types of strings like “web development evaluation” and “web development performance”. In the findings there were 30 papers and of those 6 were relevant because the others were written 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. There were often over 300 results in the search, but only up to 1-3 relevant pages in each string, and the same papers were often found in many of the strings.

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 2006 **Martin Bean** wrote the book **Laravel Essentials** where we get the basic idea of the laravel framework. He also gave us an idea about the essential commands needed to develop a laravel web application.

In 2008 **Arda Kilicdagi** describes in the book **Laravel Design Pattens and Best practices** how the MVC concept is used in Laravel

In 2014 Taylor Otwell, the creator of laravel framework wrote **Laravel Starter** where he provides the basic ideas and helps to the beginners how to get started with the laravel framework.

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.

In 2016, **Dr. Saikia** and **Das** wrote **Comparison of Procedural PHP with Codeigniter Framework** in which they have compared the procedural php with codeigniter framework by developing a blog web application.

### 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 and last one is developed using Laravel 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 and also codes are available for the laravel framework to measure the memory usage and execution time for the four functions.

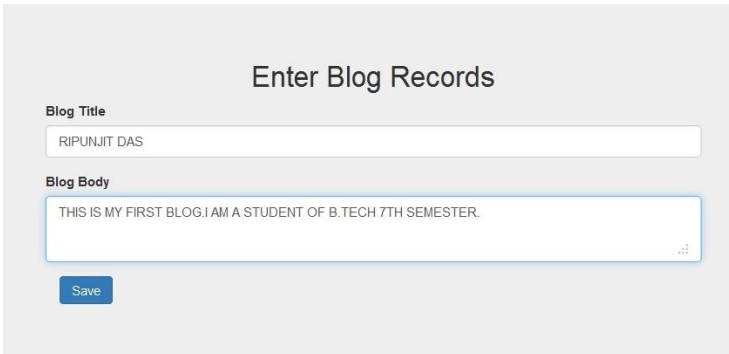
Our research will answer the following basic question:

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.



**FIG1: 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

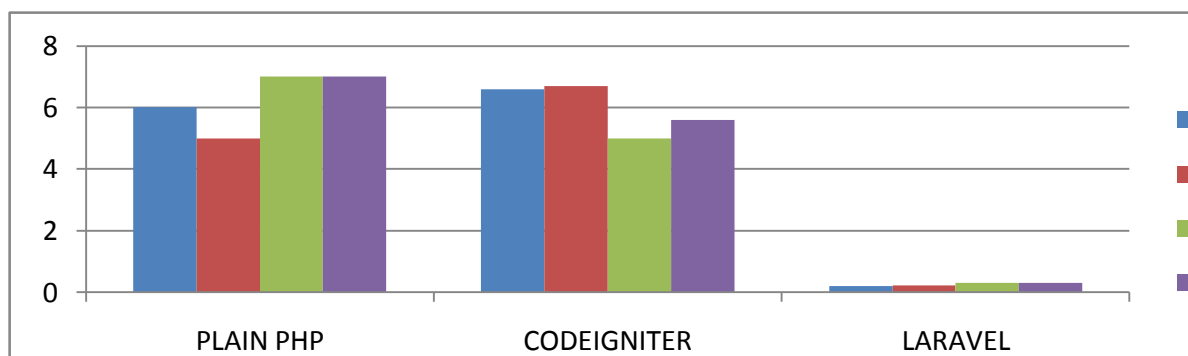
The results from the conducted experiments are presented and analyzed. The presentation is outlined in a tabular form in which the horizontal columns represent the tested actions Create, Read, Update and Delete. For each tabular form there is a corresponding visual presentation in a form of a graph.

*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. But we can see that LARAVEL performs well and has less execution time for all the four actions. The results of the ten inputs as a mean for the four actions are presented below:

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
LARAVEL	0.201	0.222	0.312	0.301

**TABLE 1: Execution time experiment result**

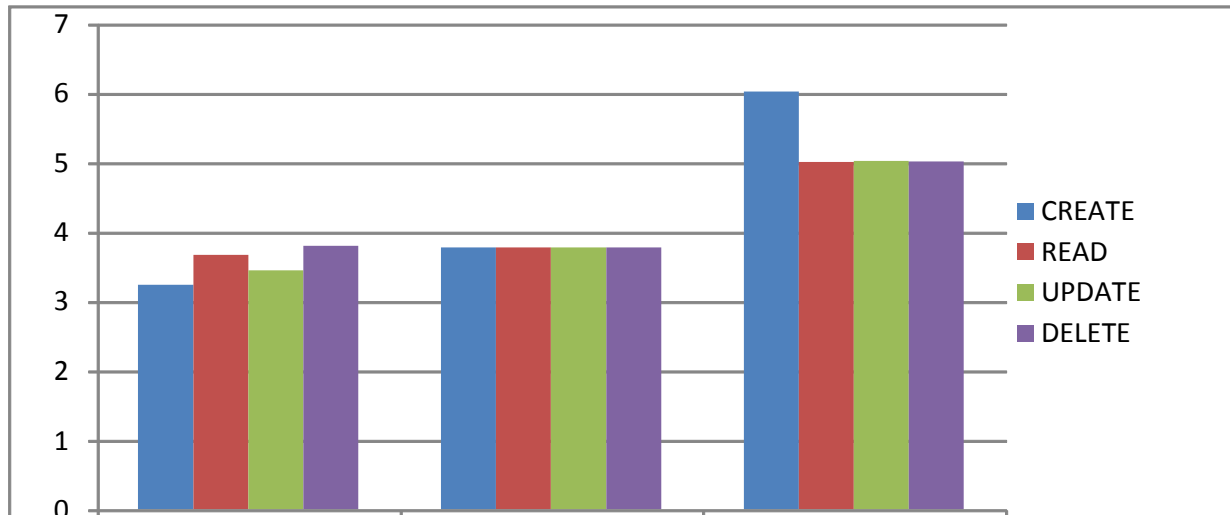


**FIG 4: Execution Time experiment Representation Graph**

**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 but laravel uses more memory in all the four functions. 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
LARAVEL	6.025	5.01	5.025	5.022



*Fig 5. Memory Usage experiment Representation Graph*

## VI. CONCLUSION

To answer the research questions defined below:

- (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 (RQ2) is depended on the outcome of the first research question (RQ1). 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 but if we see the LARAVEL version we can see that laravel takes minimum time than both codeigniter and procedural php. If the plain PHP application was large, it would mean that the plain PHP would not have been performing better than the Codeigniter and codeigniter also will not be able to perform as efficiently as LARAVEL because as the application will get bigger LARAVEL performance will be getting better as a result LARAVEL execution time measurement is better than both. In the case of memory usage laravel takes much larger time than both procedural php and codeigniter but as the application gets bigger it would be easy to effort memory but execution time should be of much importance. 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 or laravel, but Codeigniter and laravel would be the preferred choice when developing larger projects.

## REFERENCES

- [1] Samisa Abeysingha, "Php Team Development", packt publishing, October 2001
- [2] Peter Sevcik and John Bartlett, "Understanding Web Performance", Business Communications Review, October 2001
- [3] Martin Artlitt, "A performance Comparison of Dynamic Web Technologies", ACM sigmetrics, December 2003
- [4] D. Powers, "PHP Solutions: Dynamic Web Design Made Easy". Apress, 2006.
- [5] Mohammed Musthafa, "Analysis of Model based MVC framework for php development", Academics Education conference, 2008

- [6] C. Wang, "Web Services Integration Method" Dep. of Computer Science and Engineering, Chalmers University of Technology, 2010.
- [7] Nguyen Minh Thanh, "Adopting Web Framework in Web Application Development", Spring 2012
- [8] M. Alsadat Kazemi & S. Eskandari "Web-based application for Collaborative Ethical Decision Making" Dep. of Information Technology, Uppsala Universitet, 2013.
- [9] David Díaz Clavijo, "A practical comparison of Agile Web Frameworks", 2014
- [10] The Open Group Base Specifications Issue 6, IEEE Std 1003.1. "clock - report CPU time used" opengroup.org [Online; accessed 10-July-2015]
- [11] PHP Documentation. "PHP's memory\_get\_usage function." php.net [Online; accessed 19-Mars-2015] Available: <http://php.net/manual/en/function.memory-get-usage.php>
- [12] Xdebug documentation. "Xdebug's stack trace feature" xdebug.org. [Online; accessed 19-Mars-2015]. Available: [http://xdebug.org/docs/stack\\_trace](http://xdebug.org/docs/stack_trace)
- [13] PHP documentation. "History of PHP" php.net. [Online; accessed 19-Mars-2015]. Available:<http://php.net/manual/en/history.php.php>
- [14] W3techs. "W3Techs - World Wide Web Technology Surveys", w3techs.com. [Online; accessed 19-Mars-2015].
- [15] Packagist. "PHP's package manager" Packagist.org. Martin Bean, "Laravel Essentials",PACKT publishing.
- [16] Dr. Lakshmi Prasad Saikia and Ripunjit Das, "Comparison of Procedural PHP with Codeigniter Framework, International Journal of Advanced Research in Computer Science and Software Engineering, Issue 2, volume 6, February 2016.