An Empirical Study of Three PHP Frameworks

Xiaosong Li, Sai Karnan
Computer Science Practice Pathway
Unitec Institute of Technology
Auckland, New Zealand

Jahanzaib Ali Chishti
Department of Electrical and Computer Engineering
University of Auckland
Auckland, New Zealand

Abstract— **In recent years, there are many PHP frameworks for developers to choose. The developers should know how to choose the most suitable framework with the best support features to use in their project. For that reason, good understanding of different frameworks becomes an essential requirement for web developers nowadays. This study conducted a series of experiments to compare the performance and reusability of three selected frameworks by implementing the same token web application using three PHP frameworks: CakePHP, Laravel and CodeIgniter respectively. In terms of performance, CodeIgniter performed the best for the three tasks. Laravel performed the best for one of the tasks and performed the worst for one of the other tasks. In general, CodeIgniter has the best performance and Cake PHP has the worst performance. In terms of reusability, CakePHP has the highest score. This suggested that the performance and the reusability are not in proportional relationship. There could be a number of reasons for this. Regular expressions were used to detect reusability data from the code. These regular expressions cannot deal with repeat variable occurrences which may result in inaccuracy of the reusability score. Further research is required. In the future, more frameworks should be included to get the general guidelines for PHP framework evaluation. More factors should be considered in the PHP framework measurements.**

Keywords-PHP frameworks; experiment; measurement; evaluation; comparison

I. INTRODUCTION

PHP is considered as one of the most popular scripting languages to be used in developing web applications due to the fact that it is dynamic, provides a large amount of flexibility, it's easy to use and is easy to learn [1]. However, sometimes coding in PHP become a repetitive task. Moreover, the easiness to apply PHP into projects also leads inexperience developers to unorganized code process that they do not even notice.

PHP frameworks come into the picture to help developing web application smoother and faster by providing a template basic structure, along with a completed set of built-in APIs, libraries and numerous ~~of~~ extensions (which are developed by framework creators or the community). In other words, frameworks can speed up developing process, cut off developing time, help developers be more productive by reducing repetitive code in the project. The final application created by frameworks will be more stable and secure.

In recent years, there are many PHP frameworks for developers to choose from such as ***Laravel***, ***CakePHP*** and ***CodeIgniter*** etc. In fact, in real world working environment, developers will not write every single line of code while building applications, but instead they use frameworks to optimize their tasks. Therefore, developers should know how to choose the most suitable framework with the best support features to use in their project. For that reason, good understanding of different frameworks become an essential requirement for web developers nowadays.

This issue usually ends up with questions like: What is the best framework to use? Or, what are pros and cons of different PHP frameworks? How to know which framework is the most suitable for a project? And many more.

For that purpose, this study conducted a series of evaluations and experiments to compare the performance, efficiency and reusability on three selected frameworks that were used in workplaces popularly in 2013 to help web developers understand PHP frameworks and making decisions in selecting PHP frameworks for their projects. [2]. Most comments regarding PHP frameworks are based on the subjective observations and qualitative analysis. Recently, there were a few studies on PHP frameworks evaluations and comparisons. The study of [3] explored the relationship between a framework and the effort it takes to develop a web application. The study of [4] compared and analyzed the impact of data abstraction layer (ORM) on the performance of two PHP frameworks CakePHP and CodeIgniter.

The intention of this research was to work out a complete set of measurements for the evaluation of six popular frameworks, which can be further refined into a PHP framework evaluation formal procedure. Ultimately, a guideline for developers in choosing and applying frameworks in their workplace or personal projects will be provided. This paper presents the first phase of the research, where three frameworks were considered, and the measurements were focusing on the performance and reusability. The results should be useful for PHP developers and could be improved in the next phase of the research.

By implementing the same token web application using three selected PHP frameworks: CakePHP, Laravel and CodeIgniter respectively, the research provided understanding on how each framework was used in process and the technical details to evaluate the performance of each framework using time measurements for different tasks. The results were analyzed qualitatively.

In the rest of this paper, the related research is discussed first, which is followed by the experiment design, data collection and analysis. Finally, the summary and future work are given.

II. RELATED RESEARCH

Five general criteria that developers use to select a framework were indicated by [3]. They are architecture, documentation, community support, flexibility or whether they

978-1-5386-1107-4/17/$31.00 ©2017 IEEE

possess a list of features. While these criteria are essential, they are not task specific and lack ~~of~~ implementation details. They don't provide deeper help on making decision for a concrete web application. In [3], a method using source lines of code as a comparative metric to evaluate PHP MVC frameworks and determine the differences in effort required to implement the fundamental components of a web application was developed.

According to [4], PHP frameworks can considerably improve the performance of an application. These frameworks usually are based on Model, View, and Controller design pattern. These frameworks provide, different common functionalities and classes in the form of helpers, components, and plug-in to reduce the development time. Due to th~~ese~~ features such as robustness, scalability, maintainability and performance, these frameworks are mostly used for web development in PHP, with performance being the most important factor. [4] reported the research on comparing and analyzing the impact of data abstraction layer (ORM) on the performance of CakePHP and CodeIgniter. Load testing and stress testing were used to measure the performance. The results of [4] showed that there is not much of a difference, as per as the performance of CakePHP and CodeIgniter was concerned, with the respect to response time on a live server. CodeIgniter had better performance with respect to throughput on live server.

[5] indicated that although PHP dynamic features can give the developers great flexibility, they could have a negative impact on performance.

According to [6], database connection is very important for a PHP framework. What happens when multiple parts of your application need to interact with the database? Surely that database-related code shouldn't need to be duplicated in each and every PHP script. The prudent thing to do would be to refactor all database code into a shared PHP file. A developer should really need to worry about opening and closing database connections. The database-related tasks should be included in the PHP framework evaluation experiments.

In software engineering, software metrics are the only tools to control the quality of software [7]. Most of the metrics are developed covering generally programming languages such as C, C++, and Java etc. Some metrics may not be suitable for some programming or scripting languages. There is lack of metric in the literature, which measures the quality of PHP language [7]. In this respect, specific metrics should be developed, and we develop a metric for PHP, which is capable to calculate reusability quality of PHP code. A reusability metric to measure quality of PHP script language (REUphp) was proposed by [7]. Since PHP is an object oriented language, the present metric is capable to evaluate any object-oriented.

Software reuse is the process of implementing or updating software systems using existing software components. Reusability of Object Oriented Programming is the use of previous classes or functions or methods in the present class but no problem of previous classes. The goal of Reusability quality of PHP code (REUphp) is to evaluate the current state of the art on the reuse metrics area with special emphasis on code-based metrics. It includes the following factors:

(a) Number of include and Number of require ($NIRphp$)

$$NIRphp = \sum NIphp + \sum NRphp \quad (1)$$

where $NIphp$ = number of include files and $NRphp$ = number of require files.

(b) Number of object inheritance elements ($NOIphp$)

$$NOIphp = \sum NTphp + \sum NEphp + \sum NCCphp \quad (2)$$

where $NTphp$ = number of include files; $NEphp$ = number of extends and $NCCphp$ = number of concrete class.

(c) Class Interface Size ($CISphp$)

$$CISphp = \sum public(NOM) + \sum public(VARS) \quad (3)$$

where $public(NOM)$ = number of public methods and $public(VARS)$ = number of public variables.

(d) The values of the above defined metrics will become base attributes to evaluate reusability ($REUphp$) of PHP codes.

$$REUphp = \sum NIRphp + \sum NOIphp + CISphp \quad (4)$$

The above formula had been used to calculate the reusability measurements of the PHP framework applications in this study.

[8, 9] indicated that web application frameworks usually implement the Model View Controller (MVC) design pattern. The MVC pattern is a proven and effective way for the generation of organized modular applications. The MVC pattern breaks an application into three modules: model, view and controller. The model module contains the underlying classes whose instances are to be used for manipulating the database, templating frameworks and session management, and they often promote code reuse. All the three PHP frameworks selected for this study implement MVC design pattern.

III. EXPERIMENT DESIGN

Empirical study is based on observed and measured phenomena and derives knowledge from actual experience rather than from theory or belief [10]. A PHP system was developed to conduct the experiments. It consists of a control panel and three PHP applications. The PHP applications are the re-implementation of the same token application by using CakePHP, Laravel and CodeIgniter frameworks respectively. The control panel allows the users to conduct the experiments and to collect data. The same set of measurement points were integrated with each of the applications. The databases for each application were identical as well. Fig. 1 shows an overview of the experiment system architecture. The system was implemented by PHP 5.6 on windows operating system and IIS, and ***Database 1***, ***Database 2*** and ***Database 3*** were implemented by MySQL server. The management system included an UI allowing access to the applications implemented by CakePHP, Laravel and CodeIgniter respectively.

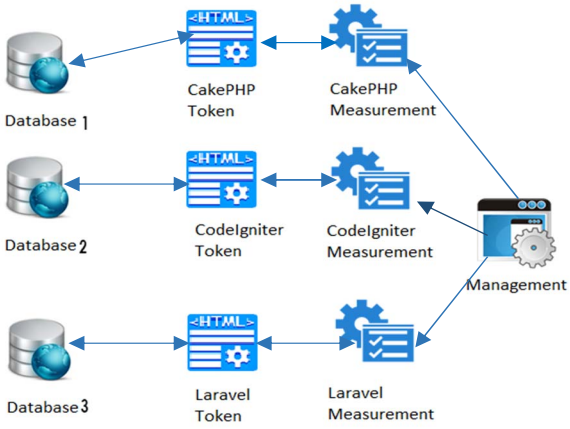

Figure 1. An overview of the experiment system architecture.

The token application integrated minimal necessary features of a web application. These features including: uploading of files, connecting with database, populating data from database, data validation, inserting data into a database, using master page as a template etc.

The experiments could help us to investigate benchmark tasks for next step of research. The following are the tasks used for performance measurements in this study:

- Read/write text files.
- Upload and save images to the web server.
- Retrieve large amount of data from the database and display them in a table.
- Database CRUD operations (connecting to/insert into/delete from/update the database).

Fig. 2 shows the web page from the token application used for large amount data (complex data) processing measurements.

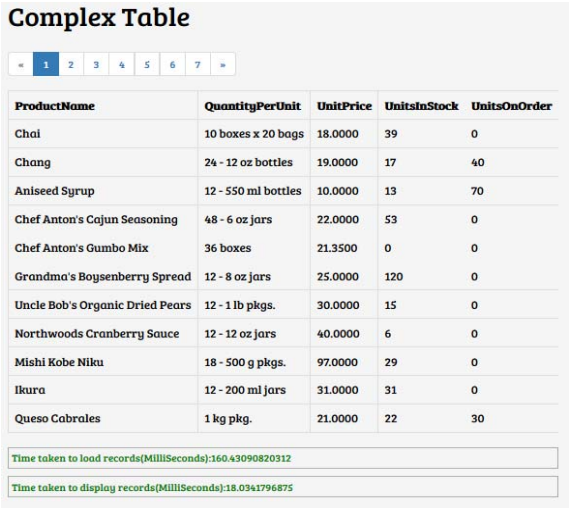

Figure 2. A web page from the token application.

Another piece of software called RIPS (http://rips-scanner.sourceforge.net/) was used to extract and calculate the reusability measurements. After the URL of the PHP application is provided, the measurements defined in formula (1), (2), (3) and (4) of [7] will be produced. Regular expressions defined in Table I were implemented to help extract the measurements.

TABLE I. REGULAR EXPERESSION USED IN THE EXPRIMENT

Purposes of the REGEX	Regular Expressions
include or require	^((include)|(require))
extends	extends
trait	^trait
functions without visibility modifiers	^\s*function[\s\n]+(\S+)[\s\n]*\(
functions with public modifiers	^\s*public\s*function[\s\n]+(\S+)[\s\n]*\(
ALL the variables	\$([a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]*)\s*=
variables with private or protected visibility modifiers	(protected\s*\$([a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]*)\s*=)|(private\s*\$([a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]*)\s*=)
concrete classes	^class\s*[\s\S]+

Table I shows the regular expressions used to measure the code reusability in the various cases of a PHP application as defined in [7], where ^ denotes the starting position of the line, hence can be used with 'include' and 'require' etc., but not with 'extends'. There are limitations for the regular expressions defined for variables; they count all the instances of the variable occurrences, e.g. $var1=x; $var2=m; $var1=p; will be counted as 3 not 2, as the regular expressions cannot deal with repeat variable occurrences (i.e. $var1 was counted twice). The measurements obtained based on these regular expressions contributed to the final reusability measurement for a PHP application by using the formula (1), (2), (3) and (4) defined in the [7].

IV. DATA COLLECTION AND ANALYSIS

In the experiments, identical data were used for all the three PHP applications. The resulting data were collected from the web pages or the resulting excel files. The results are presented in the tables in this section.

A. Reusability Results

Table II shows the final experiment results of reusability for three PHP application using three PHP frameworks CakePHP, Laravel and CodeIgniter respectively. ***REUphp*** indicates the reusability of a PHP application. ***NIRphp*** represents the number of ***include*** and the number of ***require*** used in the PHP code. ***NOIphp*** represents the number of object inheritance elements (traits, extends and concrete classes) in the PHP code. ***CISphp*** represents the ***Class Interface Size*** (public methods and variables) in the PHP code.

TABLE II. REUSABILITY RESULTS

Framework Used	*NIRphp*	*NOIphp*	*CISphp*	*REUphp*
CakePHP	97	5896	78331	84324
Laravel	116	7388	51874	59378
CodeIgniter	1	246	5466	5713

Table II shows that the PHP application implemented in CakePHP framework has the highest reusability score 84324 and the PHP application implemented in CodeIgniter framework has the lowest reusability score 5713.

TABLE III. REUSABILITY MEASUREMENTS

Measurements	*CakePHP*	*Laravel*	*CodeIgniter*
files scanned	3763	4700	212
include or require	97	116	1
extends	2480	3266	110
trait	115	110	0
functions without visibility modifiers	989	582	166
functions with public visibility modifiers	20729	19016	997
public functions	21718	19598	1163
ALL of the variables	57899	33926	4626
variables with private or protected visibility modifiers	1286	1650	323
public variables	56613	32276	4303
concrete classes	3301	4012	136

Table III shows the detail measurements for the PHP applications implemented in different frameworks. For CodeIgniter application, the number of files scanned were the smallest (212), as it is a simple PHP framework. However, it has ~~used~~ extremely small number of ***include or require*** and ***trait***, and these reduced the code reus~~a~~bility in CodeIgniter framework and their applications. The CakePHP application has fewer files scanned (3763) than the Laravel application (4700), however, the CakePHP application used more functions, private or protected modifiers and public variables, these made CakePHP framework and its applications have better code reusability than Laravel framework and its applications.

B. Complex Data

TABLE IV. COMPLEX DATA

Operations	*CakePHP Time (milliseconds)*	*Laravel Time (milliseconds)*	*CodeIgniter Time (milliseconds)*
Load	40.2	65.12	18.91
Display	1.7	7.32	0.022

Table IV shows the time for loading data from the database and displaying the data on an HTML table. Laravel application performed the worst in this case. CodeIgniter application performed the fastest of the three in this case. This could be due to that CodeIgniter is the simplest of the three PHP frameworks considered in this study, therefore, the framework data transferring overhead for this framework is small, thus making CodeIgniter more efficient.

C. CRUD Operations

In this task, three database operations (create a record, update a record and delete a record) were tested for each of the PHP applications. The same data structure and testing data were used for all the applications. Table V shows the results for CRUD Operations.

TABLE V. CRUD OPERATIONS

Operations	*CakePHP Time (milliseconds)*	*Laravel Time (milliseconds)*	*CodeIgniter Time (milliseconds)*
Create	180.85	150.44	98.22
Update	40.62	27.27	12.27
Delete	28.46	14.44	6.74

CodeIgniter application performed the best for the three database operations. CakePHP application performed the worst. This suggested that the database component of CakePHP framework has lower performance than the database component of Laravel framework.

D. Upload Images

A file upload box was used to test the performance of the frameworks when an image is uploaded to the web server. Upon uploading of the image, a report was generated on the server (and stored). This report was also displayed on the browser when the image uploading is finished. A set of 10 image files were selected for testing. The uploaded files were saved to the web server. Table VI shows the results for uploading and saving images.

TABLE VI. UPLOAD IMAGES

File Name	*File Size (KB)*	*CakePHP Time (sec)*	*Laravel Time (sec)*	*CodeIgniter Time (sec)*
1.jpg	89.83	0.634	0.318	0.605
2.jpg	197.202	1.187	1.541	1.32
3.jpg	309.747	1.683	2.22	1.954
4.jpg	416.139	2.186	2.664	2.672
5.jpg	623.068	5.442	4.702	4.679
6.jpg	1119.395	9.306	8.801	8.321
7.jpg	1398.764	10.654	9.991	10.319
8.jpg	1636.173	13.587	12.574	11.676
9.jpg	1810.801	15.3	13.894	13.192
10.jpg	2046.306	14.58	15.462	14.379
Average		7.4559	7.2167	6.9117

On average, CodeIgniter application performed the best for uploading and saving image files. CakePHP application performed the worst. So, the file uploading and saving component of CakePHP framework has the lowest performance.

E. Read/Write Files

Eight text files were used for testing the reading/writing files. Each of the file was copied (read and written), line by line to another file. The copying of the file was timed. A report was produced at the end containing the time taken for copying each of the eight files. The results are as shown in Table VII.

TABLE VII. FILE READ/WRITE

File Name	***File Size (MB)***	***CakePHP Time (millisec)***	***Laravel Time (millisec)***	***CodeIgniter Time (millisec)***
1.txt	2.506204	65.94	52.89	63.93
2.txt	5.087024	110.7	103.04	115.18
3.txt	10.173506	209.5	224.13	207.34
4.txt	20.347014	582.5	434.83	415.54
5.txt	40.69403	1114.37	845.98	865.99
6.txt	81.388062	1696.98	1706.8	1653.79
7.txt	162.77613	3402.53	3338.44	3292.38
8.txt	325.55225	7034.43	6680.45	6850.06
Average		1777.1188	1673.32	1683.02625

On average, Laravel application performed the best for reading and writing text files. CakePHP application performed the worst. So, the file processing component of Laravel framework has the best performance.

V. SUMMARY AND FUTURE WORK

In terms of performance, CodeIgniter performed the best for ***Complex Data***, ***CRUD Operations*** and ***Upload Images*** tasks. Laravel performed the best for ***Read/Write Files*** and performed the worst for ***Complex Data***. This could be due to that CodeIgniter is the simplest PHP framework with the smallest files numbers among the three selected frameworks in this study; therefore, the framework data transferring overhead is small, thus more efficient. In general, CodeIgniter has the best performance and Cake PHP has the worst performance. In terms of reusability, CakePHP has the highest score. This suggested that the performance and the reusability are not in proportional relationship. There could be a number of reasons for this. The regular expressions cannot deal with repeat variable occurrences; this may result in inaccuracy of the reusability score. Further research is required to produce more accurate regular expressions.

In the future, more frameworks should be included to get the general guidelines for PHP framework evaluation. The following factors should be considered in the PHP framework measurements.

- Investigate the benchmark tools for measurements
- Learning curve.
- Neat & clean intuitive API.
- Scalability.
- Security.
- Solid code written on proven design patterns.
- Quality of documentation.
- Security vulnerability

REFERENCES

[1] M. Hills, P. Klint, and J. Vinju, "An empirical study of PHP feature usage: a static analysis perspective," *Proceedings of the 2013 International Symposium on Software Testing and Analysis*. ACM, 2013.

[2] B. Skvorc, "Best PHP Frameworks for 2014,", December 2013, Sitepoint, https://www.sitepoint.com/best-php-frameworks-2014/, last accessed 24/10/2016.

[3] D. Clare, "A comparative evaluation to measure effort between PHP-based frameworks," Project Report, 2005.

[4] A. Fayyaz and M. Munir, "Performance Evaluation of PHP Frameworks (CakePHP and CodeIgniter) in relation to the Object-Relational Mapping, with respect to Load Testing," 2014.

[5] C. Mulder, "*Reducing Dynamic Feature Usage in PHP Code*," Master's thesis, University of Amsterdam, 2013.

[6] Laravel Book, "Laravel Introduction," http://laravelbook.com/laravel-introduction/, last accessed 26/10/2016

[7] C. T. Mon and K. M. Myo, "Framework for Evaluating Reusability of PHP," International Conference on Advances in Engineering and Technology (ICAET'2014), pp25-59. March 29-30, 2014 Singapore.

[8] I. Sarker and K. Apu, "Mvc architecture driven design and implementation of java framework for developing desktop application," International Journal of Hybrid Information Technology 7.5 (2014): 317-322.

[9] A. Paikens and G. Arnicans, "Use of design patterns in PHP-based web application frameworks," *Scientific Papers University of Latvia, Computer Science and Information Technologies* 733 (2008): 53-71.

[10] Penn State University Libraries, "Empirical Research in Education and the Behavioral/Social Sciences," http://guides.libraries.psu.edu/emp , last accessed 30/10/2017.