PHP Framework Performance for Web Development

Håkan Nylén

Blekinge Institute of Technology, Karlskrona, Sweden,
`hakan@dun.se`

Abstract. **[Context]** The new PHP Frameworks ideas has become popular for developers. It is important also to know how big impact they have on the performance. Bringing two top frameworks against each others can bring up some light how performance do look like today on the web with PHP as its grounds. **[Problem]** .... **TODO** this need to be written. **[Contribution]** This paper is one of the first trying to perform performance experiments on PHP Frameworks. It can help to push people to the right decisions in future work in PHP Framework Performance. The lack of data in this area is also one of the decision to make this paper as well.

1 Introduction

The World Wide Web, the internet started to be popular in the beginning of the 90s. The face towards the users was different websites, running on HTTP-servers, based on the official HTML language. The websites exploded in amount and it started to be some request to save information and to do something more then show static pages and files for the user. You could use several languages to do so, The Founder of PHP made PHP because of the massive amount of code you needed to code in PERL, a language PHP is based on,[10] to do a specific thing. He created PHP just as it could handle the quest he needed to handle on his page. He released it just for fun and it become famous as a good idea to a simpler language for web development. Many join the project and it is now a big project, a script language that more and more become a more object oriented script language, if not already.

The web is nowadays different. Many are talking about speed, speed and performance. For a website you have many different approaches to make the website faster, or you will lose visitors if the pageload is too slow. A new service, that came famous when cloud came the thing everyone should use, is CDN[1] and also different performance approaches, like the web-server, the server in general or the page. This could mean configuration or tweaks making the page load faster.

---

[1] CDN is Content Delivery Network, a network of servers. where static data for websites and more can be stored, a CDN is spread all over the world, making the request faster.

PHP Framework is mostly focused on fast development, with libraries and helpers, it can make the development of a website faster. This make it very popular among developers wanting to make a fast demo page for a new project. When the page is then up for production on a website, everyone can reach it. This stage make it important to think about performance, because users is not waiting long to a page to load, like mostly did on the 90s because of the slow internet connection. PHP Framework make it fast for the developer but the point of the visitor is missing.

2 The Background

Evaluation is all about the performance. Performance in this paper is about load[2], but it can be hardware, network and more. Evaluation in general is how you can test the performance and in which way. This is very important in the choice of products nowadays.

The history starts when the PHP, PHP: Hypertext Preprocessor (recursive acronym), got born in 1995, Which only was a start. Now is PHP bigger and mainly a own "programming language", more called a script language [10], for websites, inspired by java, C and PERL. [10] Developers using PHP has issues making good websites fast, so some developers started to developed PHP Framework to help to create better website faster. There are two frameworks that has biggest users, one is CodeIgniter [4], that was first released 28 February 2006 [5]. It is focused to great clean, fast framework for developers. The other one is CakePHP [12], which was released in 2005. CakePHP is mainly inspired by Ruby on Rails. [3]

2.1 MVC

TODO need to write more in detail about MVC here. Model View Controller is a structure-think to have 3 different types of classes, types of what they will do. [1] Model will handle datahandling, like data from databases. View is the one that handle how the data should be shown for the client. Controller is the middle, collect data from model and making the data readable and sending it to the view. This paper will be about testing the performance of CodeIgniter [4] and CakePHP [12].

2.2 PHP Framework

PHP is a server-side programming language famous to make it simple to develop in agile methods in web development. The Frameworks are based on MVC with add-ons, like form-helpers and database-controllers, in various languages, like PHP. Frameworks are built to make it even more faster to develop a website. The biggest PHP Frameworks with MVC-thinking are CodeIgniter[4] and CakePHP[12].

---

[2] loads means how long a request take, in milliseconds.

Codeigniter and CakePHP has many libraries to help the developer. Codeigniter has no official inspiration but its goal is to make it faster for the developer to create good websites then creating them from scratch in raw PHP. CakePHP is inspired by Ruby on Rails and you can see that on the structure of how you call the functions in a library in CakePHP. Codeigniter strongest ability is the openness where the developers are publishing libraries and plugins to Codeigniter. CakePHP has this ability too, but doesn't have as many followers making requests for changes in the code as Codeigniter.[3]

The Difference is the structure but in the ground are both the same, they are based on MVC-thinking but over that, they both has difference structures and difference thinking how the libraries for databases and controllers, should behave and been called. The biggest difference was the handling the model. In CakePHP you only type the variables, which the model should handle from the database, normally. In Codeigniter you build the whole model with variables and own functions to get the data from the database. You can do that with CakePHP but by using the already implemented function to get, update, add and remove data from the table in the database.

2.3 Client-side

Client-side means the browser and the scripts and other rendering happening by the visitor on the site. This does have a big impact in the performance on a server. [8] If using many images, the browser first loads the page and then send request to get the image and scripts on the page, adding more load to the server, because it generates more requests making the server handle more load. [11]

2.4 Server-side

Server-side is the server. Here can the network, CPU and RAM be a impact in the performance for the visitor on the site. [11] When Apache, the server software for web is getting many requests on the same time, more CPU and RAM will be used, making it to not handle all the requests when all RAM or CPU is used. Making the feeling of the website as slow. Often is the network that is stopping more load to the server, because the optimized server program such as Apache don not use so much of the CPU and RAM, making the CPU and RAM the less problem. Of course can that be the issue, but not so often.

2.5 Performance

What performance can be different, to make it easy this paper will be about fast request, so network is very important. It can also have big impact how the servers is, Like a big server park in a datacenter. But that cost much money, which is a problem when doing this paper. That is why it will only test this

---

[3] This can been shown on their git repositories on Github, [7] and [6]. Seeing followers of the repository and how many pull requests each project has.

on cloud with one server and see how the framework is changing the ms of a request. This Server is running on Amazon EC2, a isolated server environment on the cloud. A good thing with this setup is that it is more or less this real website is set-up nowadays, making it more trustworthy in the data. The server specification can be seen on table 6.

3 Research

3.1 Research Questions

RQ1: What web performance evaluations exist and how were they performed?
RQ2: What factors impact web performance?[4]
RQ3: To what extent are open source php frameworks evaluated?
RQ4: What performance are different between most commonly used open source php frameworks and how can it be evaluated?

Searched more or less on different types of strings like "web development evaluation" and "Web development performance". found 300 papers and 8 relevant because the others was talking about the evaluation in general, not about performance in web development.

3.2 Research Methodology

Research Question	Methodology
RQ1	Literature
RQ2	Literature
RQ3	Literature
RQ4	Data from RQ1, RQ2 and RQ3 to design experiment

3.3 Literature Survey

The literature was found in different stages. And these sources was used.

- Google Scholar
- IEEE
- Google

These strings were used, as seen in table 1.

The selection was not so hard, Searched for some data that was doing something with evaluation in this area, and it was a few. The important was a good how-to, how they made the research to find out the information in the data, this was often found in the abstract but sometimes in the background. Often it was over 300 results in the search, but only 1-3 relevant pages, in every string, and the same papers was often in many of the strings.

---

[4] Because of the lack of data on php performance, it make it easier to find data about web performance in general.

Strings
Web development Evaluation
Web development Performance
PHP Framework evaluation
PHP Framework performance
PHP Evaluation
PHP Performance
Website Performance
Website evaluation

Table 1. The strings for searching literature

4 Literature review

4.1 The Approach

The search was by the strings presented in Table 1, and byt them the search found some hundreds of papers. This comes to the approach to found some with text such as impact in performance or was something about performance in PHP, because of the lack of papers in this area it become very hard to find any papers about PHP performance at all. This made it easier to find papers in web performance in general. Took everything of value in this area, that have something that could be used for the questions to be answered such as impact in evaluation tests, php frameworks, different evalution types and hows. The rest of the finding approach was depending on the authors knowledge, to find something that has a impact for a website or similar that can be a important thing to bring up in this paper.

4.2 The Papers

Information what the papers contains and what they have in common.

PHP Team Development is about how the MVC idea and techniques make a difference in the development. This is a book about how the development in PHP is and what it does with web development, as MVC is a big part of it. This book is from 2009. [1]

Analysis of model-based mvc framework for php development CodeIgniter is analysing how the mvc based framework Codeigniter is doing in the development in performance and in userability for developers. this is a analysing paper from May 2009, it has kinda much in common with PHP Team Development, the book which have a chapter about MVC. [4]

Ec2 faq: What is a ec2 compute unit shows what EC2 is about and what a compute unit is. This is a website FAQ, which was last visited april 2012. [2]

5

Understanding web performance presents how performance can be used and it is importance in the web business, it also bring up how performance can be tested and what can impact the tests. It is a paper presented in a Business Communication Review October 2001. [8]

A performance comparison of dynamic web technologies describe the different impact in performance when they do a comparison in different performance types and how it can be done. Presented in ACM Sigmetrics December 2003. [11]

Web-based IDE to create model and controller components for mvc-based web applications on cakephp writes how the CakePHP is build and how to code in it and talk a little about how MVC is about. A normal paper from December 2010. Has much in common with the other MVC framework Codeigniter and the book PHP Team Development. [12]

5 Literature Results

5.1 Old Papers

The results below are based on old papers. These papers are based on the general web performance and the web performance has not changed much in 10 years. The papers about web performance that is not 10 years old was very hard to find. They were mostly about java, a other programming language, that is not build up in the same way as php. Making them more or less useless for this paper. The papers bringed up in this paper are taking up impacts and other subject for the answers of the RQ, that has same impact and importance today. So the relevance has not changed.

5.2 RQ1: What web performance evaluation exist and how were they performed?

Evaluation in web is mostly done by load testing and counted in ms, but also in how to optimize the page itself to minimize request to server, like less pictures, or move them to another server. This can be seen in table 2.

Evaluation topic	How	year
requests/ms	Experiment	2001[8][11]
server/cpu	Experiment	2003 [11]
on-page optimization	Experiment	2003[11]

Table 2. Different types of evaluation of a web performance

request/ms is mostly used to evaluate performance of web applications. Got two different papers talking about two different things, *Understanding Performance* adds this comment:

> Sure, speed matters, but it is not a one-dimensional problem. And, despite what you have heard, just adding more bandwidth does not always make things go faster. [8]

Large and many requests needs good cpu and network, but to understand *Understanding Performance* speed matters but not always the only problem. *A Performance Comparison of Dynamic Web Technologies* talks about how they do the tests with requests in ms:

> Consideration of overload behaviour may be just as important as the peak request rate when Web site administrators are choosing dynamic Web content generation technologies. [11]

They saw a important to consider how to choose technologies using dynamic web content, because of different peak of request, making a need to handling the overload right. Something that can be changed in how many and how the size is on the requests. They also show how good it can be to evaluate how the performance is by using requests per milliseconds, or seconds.

server/cpu is important when the server need to handle many request at the same time. *A Performance Comparison of Dynamic Web Technologies* tells that:

> Once the servers become overloaded, the CPU utilization of Apache 2.0.45 is lower than that of Apache 1.3.27. Under overload, Apache 2.0.45 is unable to accept TCP connections (and hence requests) as quickly as Apache 1.3.27. [11]

Making it important for cpu to work, but that does not mean that Apache itself make the CPU overload, it can be the network the cpu need to handle.

On-page optimization is about how to create less requests to the server by removing or group images to one, because browsers loads the html first and then sending more request to get the images. This does also have a thing with the size of the images as well, Bigger images makes longer time to download. *A Performance Comparison of Dynamic Web Technologies* is testing with static content [11] that can be images or normal html files. Showing it goes faster then dynamic content, but it is a request.

5.3 RQ2: What factors impact web performance?

There are things that can be a impact of the performance while testing. The impacts are listed in table 3.

The server can have big impact on performance because the cpu and RAM can be to less to handle the big amount of request tested. Small CPU and RAM

Factor	How it impact	comment
Server	network, cpu, ram	This can be fixed with a large server or a data-center, something that can not be tested here.
Network	speed	if the test will do a request/ms load simulation, the network can be maximized without the server being touched to the performance limit which been set.
Wrong config	slowness, bad performance	wrong config on the server, resulting in less performance, but that will not be a problem. The default settings will be used. Change configuration for optimization is not a part of this paper.

Table 3. Different types of evaluation of a web performance

can make it to overload in power and deny new request. Making it impossible to connect when that amount of request is full.

The Network is not very important but the network should be stabile for the amount of request sending to the server. Normal speed would be at least 10mbit down and 10mbit up to send and receive in good speed. The only problem with the network in my home is the stability, making it hard to trust the speed, making the ms higher then it should be. as example, the internet connection where the tests was tested can have 100mbit up and 100mbit down, but only getting 34 mbit of both because of the share of the internet connection, with 60 other people, this can be a big impact if all people is using the network, but that is not a big risk.

Wrong config is the thing that can be spooky and many people do not know that it can be a default config or a wrong setup config that make the server to overload or not accept requests. The tests will use defaults just to make it so clean as possible, because that is how many people actually are using the defaults in the configs.

5.4 RQ3: To what extent are open source php frameworks evaluated?

TODO Need to rewrite this, adding more data why this is not covered. It was nothing about how performance in php framework what could be found, this is one of the things why the author want to experiment and do this. It is a big area in performance in php framework thats not covered.

6 Experiment Design

6.1 Investigated Question

Which of the following php framework provide better performance measured in terms at: page speed in ms.

6.2 Experiment artifacts and variables

Controlled Variables the following variables are controlled

Variable	Description
Request size	constant
Server	see table 6
Framework	Compatible API, look at table 7 for info about the classes.
Response size	constant. See the size in table 8.

Table 4. Controlled variables in Experiment

The tests is as you can see in the list below, all 3 tests will be run for database and without database, for each framework. Making it 6 tests total for a framework, with and without database. the tests are using 5 for the concurrency

	Codeigniter		CakePHP		
	Concurrency	Requests	Concurrency	Requests	
Test 1	5	20,000	5	20,000	with database
Test 2	5	20,000	5	20,000	with database
Test 3	5	20,000	5	20,000	with database
Test 1	5	20,000	5	20,000	withOUT database
Test 2	5	20,000	5	20,000	withOUT database
Test 3	5	20,000	5	20,000	withOUT database

Table 5. How the tests will make for the frameworks.

level, meaning to send 5 request each time, send so fast the previous requests are done. This to help to count the response time more exactly. You can see how the tests will be done in table 5.

The total amount of request of 20,000 is to get amount of request to get average time from, not too much and not too less. It does not need to be much more because the average will be the same $\pm$ 2 ms. That make it not necessary to have more. But it took as much as 20,000 because it make it more secure to be right numbers in the results.

System	Version/size
OS	Ubuntu 10.04
Apache	2.2.14
MySQL	5.4
PHP	5.3.2
CPU	1 EC2 Compute Unit[4]
RAM	1.7 GB
HDD	160 GB
bits	64 bit

Table 6. The server specification where the test will be done on.

A good thing to know is how the server will be setup, here is the table of the specification of the server that will be used for the experiment, on the Amazon cloud. This is how the Framework classes will be used in the experiment.

Type	Codeigniter class	CakePHP class
Controller	CI_Controller	AppController
Model	CI_Model	AppModel
Database	database	database
Form helper	form	Form

Table 7. The different classes to use in the frameworks while coding a blog.

Manipulated variables Server load measured in terms of amount of request. Manipulated variables

Response Type HTML without database, HTML with database

Response Type is the only thing that will be changed. It is the support of the database that will be removed or added, making it possible to see the difference in performance for the core of the framework and for just the database controller.

Experiment Artifact

blog the blog will have normal features, or more likely very simple. It will have a normal list with pagination on the startpage, every post will have its own page. You can see the code in Appendix A1 and A3.

apache benchmark will use Apaches own benchmark to simulate server load.

Environment the server is mostly already set, see table 6.

---

[4] One EC2 Compute Unit provides the equivalent CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor. [2]

Blog is first a simple blog, with posts list on the startpage and with a own page for single post. The most important with the blog is the index page in the experiment, because it is there the requests will be send. Focusing to make a decent list of posts with title, body and created date at the blog. Both frameworks used the same database making it simple and less chance for the database to make a bad impact in the performance. You can see the code in Appendix A1 and A2.

Apache benchmark is doing as All tools for performance for a web application is doing. It is same idea, to send a amount of request to the server and count how long it took to get a response back. Apache benchmark is the one that is been in while and has a reputation to do fair tests.[9]

7 Experiment

7.1 How it was made

TODO Fixed so the night/day is picked right here and in The Results section Used Apache benchmark to 20,000 requests on a currency of 5 request per time frame to a server. the test was made with a database on the night randomly picked which framework to test. The tests without database was made on the day, making it more sense why the ms is higher on the tests there, it is more traffic on the internet on the day.

I used the commandline below to send the load request, randomly to specific ip-address[5] on the day and night from my own computer to the test server, on the Amazon cloud on Ireland.

```
ab -wn 20000 -c 5 http://11.11.11.11/framework
```

The variables controlled was response size, amount of request and currency of requests. see the data for the variables in table 8.

Variable	Codeigniter db[6]	Codeigniter	cakephp db[6]	cakephp
ResponseSize[7]	6440000 bytes	6440000 bytes	6360000 bytes	6360000 bytes
Requests	20,000	20,000	20,000	20,000
Currency	5	5	5	5

Table 8. The tests with database average request/connection times in ms.

---

[5] the ip is typed as 11.11.11.11 to make it clear that the server is not running no more

[6] with db it means database. In other meanings the framework with the database, the database was used with models for posts, without db it was just plain html in view with controller used.

[7] ResponseSize is total bytes of html sent from the server in one test, on every interval.

7.2   The results

The experiment was made randomly on a night in different times, picked randomly. You can see how the tests was done in Table 5. The results of the tests can you see in Appendix A3.

The result can you see in figure 1. It was almost same results on both, but a small gap between the two framework on just a few Millie seconds, to be exact, it was a gap on 5,34 ms in a average.
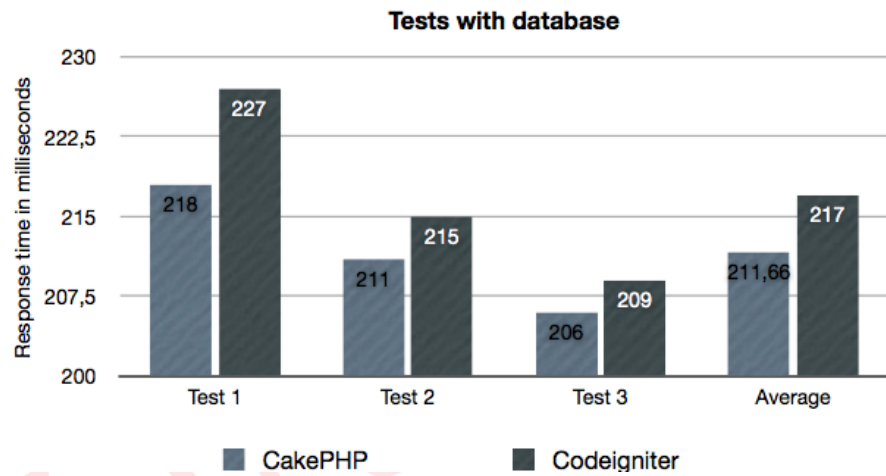

Figure 1. The tests with database average request/connection times in ms.

12

This was with database, with show that if we will focus to the ms total average, is CakePHP faster. This need to be tested without a database too, You can see the result in figure 2.
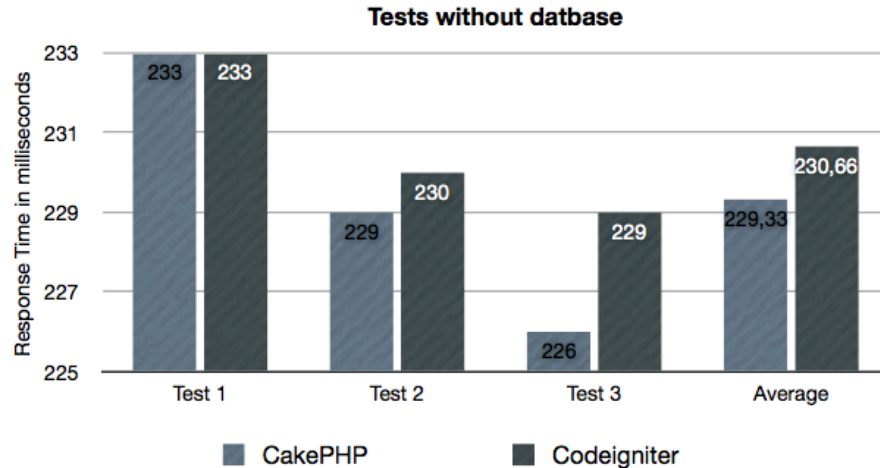

Figure 2. The tests without database average request/connection times in ms.

The test without database shows that they are almost equally fast. only 1,3 ms gap between them, CakePHP was faster here too, showing CakePHP has faster structure in its core.

7.3   Data analysis and discuss

This was good to see how the different the performance is between the framework. This is also a very good way to evaluate the performance on any web service or application, the load of request can show how fast the request can be handled by the framework and show the fastness of it.

The biggest difference of the performance was when using a database, The difference was **5,34 ms**. it sounds little but in big sites just few ms does matter. The difference when using no database, only controller and view was no different, there was the difference **1,3 ms**. very little. and if we bring the size in the thought, the gap is even lesser. Maybe is Codeigniter faster, if they was in same size? That can be true. The default template in CakePHP and Codeigniter was used, making the page in different sizes and could make a difference in the performance of the tests and request.

The difference in milliseconds is so small without database because the frameworks seems to have good structure in their core of controllers and support for views, which we used in the tests without database for both framework. The difference in milliseconds for the tests with a database is bigger, 5.34 ms.

Smaller then a thought but it is a difference. making it noticeable when having a database and the PHP framework on a big website with many visitors. Creating a big queue of requests, making the small difference bigger, 5.34 ms can feel like the double, 10,68 ms or more. That is why a small difference can be important.

A thing to think about is the size difference, which was around **80,000 bytes**, for both the test with and without the database. This can make gap between the framework understandable with db, but without the db was the tests very close to each others. If the size was the same, Codeigniter could be faster.

The difference in the size is because the use of the default in CakePHP and Codeigniter, CakePHP has more style and a usable template as default for the application, Codeigniter do not have it making it confused of the bigger size of the response. But the size is not a problem because of the choice to use the default of each framework, making the difference in the size.

This seems as no problem, used the default tools in both framework to make a very simple blog webpage. The size should in other word be different, it would be weird if it was not. But the thought that the size can make a difference is of course in the mind. It is request time in ms, and it does have a very big impact by the size.

Would love to see any **future work** on different sizes for the PHP framework, which tests how the size does matter of the performance, with same size of both.

7.4   Validity Threats

Tests are not isolated making the tests to be disturb by something from outside, can be the traffic of the line on Amazon cloud or something else that can make the response time longer or not trusted. This was tried to be reduced by been testing in the night. But you can see in the tests without, that was made more on the morning that the response time is already higher. This should not change the gap between the frameworks because all the tests was tested in the same time, making it having all around same response time in general.

Concurrency level <mark>**TODO** the threat of small/big concurrency level need to be written</mark>

Client's impact could be a problem. The tests was running Apache benchmark, but Apache benchmark is ignoring stylesheet and pictures (The blog in the test did not have pictures). It is just getting the generated html page, after php have generated it with the use of the code of PHP Framework. making the impact smaller then a normal browser, that is getting the stylesheet and pictures too.

8   Conclusion

The frameworks performance has big difference, but mostly just if in used of database, the classes fro using database can in other words be more effective.

CakePHP was fastest in both tests, with and without a database. But the size was different on 80,000 bytes. This could be something that could make CakePHP a winner. But my conclusion is that used default template, very simple html page. So it does not have a big impact.

The fastest framework, according to the data is CakePHP, with just 1,3 ms without database and with 5,34 ms with database. This is something that is a big impact in the PHP performance and it can be fixed in newer versions.

The experiment shows that the evaluation can be done it milliseconds for websites and web applications. With this data is the difference of evaluation between PHP framework not big at all, but it is a difference, even if it is a small one.

The relevance and contribution of this paper is big because of the lack of papers and experiment made on PHP Frameworks. This is one of a new area of technology to test in performance.

Let the war begin between the php frameworks in performance.

List of notes

References

1. S. Abeysinghe. *PHP Team Development*. Packt Publishing, 2009.
2. Amazon Web Services LLC. Ec2 faq: What is a ec2 compute unit. `http://aws.amazon.com/ec2/faqs/`, April 2012.
3. Cake Software Foundation. Intro to cakephp, what is cakephp. `http://book.cakephp.org/1.1/view/307/Introduction-to-CakePHP`, June 2012.
4. G. Da-gang. Analysis of model-based mvc framework for php development codeigniter. May 2009.
5. EllisLab LLC. Codeigniter changelog and releasedates. `http://codeigniter.com/user_guide/changelog.html`, June 2012.
6. GitHub Inc. Git repository for cakephp. `https://github.com/cakephp/cakephp`, July 2012.
7. GitHub Inc. Git repository for codeigniter. `https://github.com/EllisLab/CodeIgniter`, July 2012.
8. P. Sevcik and J. Bartlett. Understanding web performance. *Business Communications Review*, 31(10):28, October 2001.
9. The Apache Software Foundation. ab: Apache benchmark specification and information. `http://httpd.apache.org/docs/2.2/programs/ab.html`, July 2012.
10. The PHP Group. Php faq what is php and what does it stand for. `http://php.net/manual/en/faq.general.php`, June 2012.
11. L. Titchkosky, M. Arlitt, and C. Williamson. A performance comparison of dynamic web technologies. *ACM SIGMETRICS*, 31(3):2–11, December 2003.
12. S. Widjaja. Web-based ide to create model and controller components for mvc-based web applications on cakephp. December 2010.

Figures

Tables

Appendix

A1. CodeIgniter Code

controller/blog.php

```php
<?php
class Blog extends CI_Controller {
        public function __construct()
    {
        parent::__construct();
        $this->load->model('Posts_model', 'Posts', TRUE);
    }

    function index() {
            $this->data->query = $this->Posts->getPosts(10, 0);

            $this->load->view('blog', $this->data);
    }

    function view($id) {
            $this->data->query = $this->Posts->getPost($id);
            $this->load->view('view', $this->data);
    }
}
?>
```

view/blog.php

```html
<html>
<head>
<title>Le Blog</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link href="/codeigniter/css/cake.generic.css" rel="stylesheet">
<link href="/codeigniter/favicon.ico" type="image/x-icon" rel="icon" />
<link href="/codeigniter/favicon.ico" type="image/x-icon" rel="shortcut icon" />
</head>
<body>
<div style="hight:100px;font-size:18pt;font-family:helvetica,arial;width:100%">
<h1>My simple blog.</h1>
</div>

<div class="container">
<?
if($query->num_rows() > 0) {
        foreach ($query->result() as $row)
        {
```

18

```php
                ?>
                <div style="width:300px;font-size:8pt">
                        <a href="blog/view/<?=$row->id?>">
                                <h2><?=$row->title?></h2>
                        </a>
                <p>Skrevs <?=$row->created?></p>
                <p><?=$row->body?></p>
                </div>
                <?
        }
}
else {
        ?><p>No posts yet.</p><?
}
?>
</div>
</body>
</html>
```

view/view.php

```php
<html>
<head>
<title>Le Blog</title>
<link href="/codeigniter/bootstrap/css/bootstrap.css" rel="stylesheet">
</head>
<body>
<div style="hight:100px;font-size:18pt;font-family:helvetica,arial;width:100%">
<h1>My simple blog.</h1>
</div>

<div class="container">
<?php
if($query->num_rows() > 0) {
$row = $query->row();
?>
<h2><?=$row->title?></h2>
<p>Created <?=$row->created?></p>
<p><?=$row->body?></p>
<?
}
?>
</div>
</body>
</html>
```

model/posts_model.php

```php
<?php
class Posts_model extends CI_Model {
        public $id = "";
        public $title = "";
        public $body = "";
        public $created = "";
        public $modified = "";

        function install()
        {
                    /* First, create our posts table: */
                $this->db->query("CREATE TABLE posts (
    id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(50),
    body TEXT,
    created DATETIME DEFAULT NULL,
    modified DATETIME DEFAULT NULL
);");
        }

        function addPost($data) {
            if($data) {
                    $this->db->insert('posts', $data);
            }
        }

        function getPosts($amount, $lastId = 0) {
                if($lastId == 0) {
                        $data = $this->db->query("SELECT * FROM posts ORDER
                         BY ID DESC LIMIT 10");
                }
                else {
                        $data = $this->db->query("SELECT * FROM posts WHERE id
                         <$lastId ORDER BY ID DESC LIMIT 10");
                }
                return $data;
        }

        function getPost($id = 0)
        {
                $data = "";
                if($id != 0) {
                        $data = $this->db->query("SELECT * FROM posts
                         WHERE id ='{$id}' LIMIT 1");
                }
```

```php
            return $data;
        }

}
?>
```

A2. CakePHP Code

controller/postsController.php

```php
<?php
class PostsController extends AppController {
    public $helpers = array('Html', 'Form');

    public function index() {
        $this->set('posts', $this->Post->find('all'));
    }

    public function view($id = null) {
            $this->Post->id = $id;
        $this->set('post', $this->Post->read());
    }
}
?>
```

view/posts/index.ctp

```php
<!-- File: /app/View/Posts/index.ctp -->

<h1>Blog posts</h1>
<table>
    <tr>
        <th>Id</th>
        <th>Title</th>
        <th>Created</th>
    </tr>

    <!-- Here is where we loop through our $posts array, printing out post info -->

<?php foreach ($posts as $post): ?>
    <tr>
        <td><?php echo $post['Post']['id']; ?></td>
        <td>
            <?php echo $this->Html->link($post['Post']['title'],
array('controller' => 'posts', 'action' => 'view', $post['Post']['id'])); ?>
        </td>
        <td><?php echo $post['Post']['created']; ?></td>
```

21

```
        </tr>
        <?php endforeach; ?>

</table>
```

view/posts/view.ctp

```
<!-- File: /app/View/Posts/view.ctp -->

<h1><?php echo h($post['Post']['title'])?></h1>

<p><small>Created: <?php echo $post['Post']['created']?></small></p>

<p><?php echo h($post['Post']['body'])?></p>
```

model/Post.php

```php
<?php
class Post extends AppModel {
        public $validate = array(
        'title' => array(
            'rule' => 'notEmpty'
        ),
        'body' => array(
            'rule' => 'notEmpty'
        )
    );
}
?>
```

A3. The Tests Results

22

Tests WITH database

This is ApacheBench, Version 2.3 <*$Revision: 655654 $*>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

CODEIGNITER

first test

Server Software:	Apache/2.2.14
Server Hostname:	54.247.130.232
Server Port:	80
Document Path:	/codeigniter
Document Length:	322 bytes
Concurrency Level:	5
Time taken for tests:	911.508 seconds
Complete requests:	20000
Failed requests:	0
Non-2xx responses:	20000
Total transferred:	12100000 bytes
HTML transferred:	6440000 bytes
Requests per second:	21941.67
Transfer rate:	13274.71 kb/s received

Connection Times (ms)

	min	avg	max
Connect:	68	226	38113
Processing:	36	1	1
Total:	104	227	38114

second test

Server Software:	Apache/2.2.14
Server Hostname:	54.247.130.232
Server Port:	80
Document Path:	/codeigniter
Document Length:	322 bytes
Concurrency Level:	5
Time taken for tests:	861.753 seconds
Complete requests:	20000
Failed requests:	0

Non-2xx responses:	20000
Total transferred:	12100000 bytes
HTML transferred:	6440000 bytes
Requests per second:	23208.52
Transfer rate:	14041.15 kb/s received

Connnection Times (ms)

	min	avg	max
Connect:	56	214	38048
Processing:	48	1	0
Total:	104	215	38048

third test

Server Software:	Apache/2.2.14
Server Hostname:	54.247.130.232
Server Port:	80
Document Path:	/codeigniter
Document Length:	322 bytes
Concurrency Level:	5
Time taken for tests:	837.755 seconds
Complete requests:	20000
Failed requests:	0
Non-2xx responses:	20000
Total transferred:	12100000 bytes
HTML transferred:	6440000 bytes
Requests per second:	23873.32
Transfer rate:	14443.36 kb/s received

Connnection Times (ms)

	min	avg	max
Connect:	59	207	20491
Processing:	45	2	0
Total:	104	209	20491

CAKEPHP

first test

Server Software:	Apache/2.2.14
Server Hostname:	54.247.130.232
Server Port:	80
Document Path:	/cakephp

Document Length:	318 bytes
Concurrency Level:	5
Time taken for tests:	873.316 seconds
Complete requests:	20000
Failed requests:	0
Non-2xx responses:	20000
Total transferred:	11420000 bytes
HTML transferred:	6360000 bytes
Requests per second:	22901.23
Transfer rate:	13076.60 kb/s received

Connection Times (ms)

	min	avg	max
Connect:	57	216	38197
Processing:	47	2	0
Total:	104	218	38197

second test

Server Software:	Apache/2.2.14
Server Hostname:	54.247.130.232
Server Port:	80
Document Path:	/cakephp
Document Length:	318 bytes
Concurrency Level:	5
Time taken for tests:	847.232 seconds
Complete requests:	20000
Failed requests:	0
Non-2xx responses:	20000
Total transferred:	11420000 bytes
HTML transferred:	6360000 bytes
Requests per second:	23606.27
Transfer rate:	13479.18 kb/s received

Connection Times (ms)

	min	avg	max
Connect:	58	210	20516
Processing:	46	1	0
Total:	104	211	20516

third test

Server Software:	Apache/2.2.14

Server Hostname:	54.247.130.232		
Server Port:	80		
Document Path:	/cakephp		
Document Length:	318 bytes		
Concurrency Level:	5		
Time taken for tests:	826.001 seconds		
Complete requests:	20000		
Failed requests:	0		
Non-2xx responses:	20000		
Total transferred:	11420000 bytes		
HTML transferred:	6360000 bytes		
Requests per second:	24213.03		
Transfer rate:	13825.64 kb/s received		

Connnection Times (ms)

	min	avg	max
Connect:	53	204	11978
Processing:	51	2	0
Total:	104	206	11978

Tests without database

CODEIGNITER

first test

Server Software:	Apache/2.2.14
Server Hostname:	54.247.130.232
Server Port:	80
Document Path:	/codeigniter
Document Length:	322 bytes
Concurrency Level:	5
Time taken for tests:	933.235 seconds
Complete requests:	20000
Failed requests:	0
Non-2xx responses:	20000
Total transferred:	12100000 bytes
HTML transferred:	6440000 bytes
Requests per second:	21430.84
Transfer rate:	12965.66 kb/s received

Connection Times (ms)

	min	avg	max
Connect:	57	232	38060
Processing:	47	1	1
Total:	104	233	38061

second test

Server Software:	Apache/2.2.14
Server Hostname:	54.247.130.232
Server Port:	80
Document Path:	/codeigniter
Document Length:	322 bytes
Concurrency Level:	5
Time taken for tests:	922.767 seconds
Complete requests:	20000
Failed requests:	0
Non-2xx responses:	20000
Total transferred:	12100000 bytes
HTML transferred:	6440000 bytes

Requests per second: 21673.96
Transfer rate:　　13112.74 kb/s received
Connection Times (ms)

	min	avg	max
Connect:	59	229	71415
Processing:	45	1	0
Total:	104	230	71415

third test

Server Software:	Apache/2.2.14
Server Hostname:	54.247.130.232
Server Port:	80
Document Path:	/codeigniter
Document Length:	322 bytes
Concurrency Level:	5
Time taken for tests:	918.796 seconds
Complete requests:	20000
Failed requests:	0
Non-2xx responses:	20000
Total transferred:	12100000 bytes
HTML transferred:	6440000 bytes
Requests per second:	21767.63
Transfer rate:	13169.41 kb/s received

Connection Times (ms)

	min	avg	max
Connect:	60	228	37012
Processing:	44	1	0
Total:	104	229	37012

CAKEPHP

first test

Server Software:	Apache/2.2.14
Server Hostname:	54.247.130.232
Server Port:	80
Document Path:	/cakephp
Document Length:	318 bytes
Concurrency Level:	5
Time taken for tests:	935.934 seconds

Complete requests:	20000	
Failed requests:	0	
Non-2xx responses:	20000	
Total transferred:	11420000 bytes	
HTML transferred:	6360000 bytes	
Requests per second:	21369.03	
Transfer rate:	12201.72 kb/s received	

Connection Times (ms)

	min	avg	max
Connect:	57	232	20396
Processing:	47	1	1
Total:	104	233	20397

second test

Server Software:	Apache/2.2.14
Server Hostname:	54.247.130.232
Server Port:	80
Document Path:	/cakephp
Document Length:	318 bytes
Concurrency Level:	5
Time taken for tests:	917.710 seconds
Complete requests:	20000
Failed requests:	0
Non-2xx responses:	20000
Total transferred:	11420000 bytes
HTML transferred:	6360000 bytes
Requests per second:	21793.38
Transfer rate:	12444.02 kb/s received

Connection Times (ms)

	min	avg	max
Connect:	54	228	74417
Processing:	50	1	0
Total:	104	229	74417

third test

Server Software:	Apache/2.2.14
Server Hostname:	54.247.130.232
Server Port:	80
Document Path:	/cakephp

Document Length:	318 bytes
Concurrency Level:	5
Time taken for tests:	907.820 seconds
Complete requests:	20000
Failed requests:	0
Non-2xx responses:	20000
Total transferred:	11420000 bytes
HTML transferred:	6360000 bytes
Requests per second:	22030.79
Transfer rate:	12579.58 kb/s received

Connection Times (ms)

	min	avg	max
Connect:	58	225	37924
Processing:	46	1	1
Total:	104	226	37925