Robust PHP-based App using Laravel Framework

Alfonsus Raditya Arsadjaja / 13514088
Informatics/Computer Science Program
School of Electrical Engineering and Informatics
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13514088@std.stei.itb.ac.id

***Abstract*—Laravel is the modern way to make a php-based web application. It's a php framework with many features inside. Nowadays, application get more complex, so just plain PHP will not work in this way. For making it more robust, Laravel provides many features inside that designed for simplicity, modularity, and readability. It provides some basic features that makes a fundamental parts faster to code. With many projects nowadays, Laravel has become one of the most used framework thanks to its' features and easy-to-maintain app.**

Keywords—Framework; Laravel; PHP;

I. Introduction

PHP (known for recursive acronym PHP: Hypertext Preprocessor) is a programming language designed for web programming. PHP Programming has been known for a long time. It's already famous from the beginning of the internet era, around 1991. Especially for web development, PHP is one of the most famous language in that scope. PHP has been used for many purpose in web technology.

Now, with the increasing customer needs in technology, the language has becoming more and more complex. In the beginning, the language doesn't have much update, also with software. Nowadays, the software can be considered as "out-of-date" only in years count, typically about 2-3 years, even several apps maybe have a optimal lifetime less than a year if it's not updated with the exponentially growth of technology.

There are several things that already solved this problem, like some library or framework out there; Zend Framework is one of them, but most of them doesn't really solve a lot of problem in PHP programming. So with Laravel, we can make it simply faster and lighter code, and more modular code.

In this paper, we will discuss several things, from the details of the PHP itself and framework that we want to use and compare, the example of implementation using Laravel, and then the conclusion itself.

II. Literature Study

A. PHP (PHP: Hypertext Preprocessor)

According to PHP definition, PHP is widely-used open source general-purpose scripting language that is especially suited for web development and can be embedded into HTML. This is a example code of basic php:

Table 1 Print Hello World in Plain PHP

```
<!DOCTYPE HTML>
<html>
    <head>
        <title>PHP Hello World Example</title>
    </head>
    <body>
        <?php
            echo "Hello, World!";
        ?>
    </body>
</html>
```

The PHP code is interpreted, which means that it runs code one line by one line from the head of code and not compiled at first. If some line is interpreted and make a little bugs or similar, it will run to an error and shown to the user. If some line have the same bug, but the flow of the code isn't go there, then the error will not shown to the user. This is contrast with other type of language like C++ or C, which is compiled first before it can be run.

B. Laravel

Laravel is the PHP framework designed by Taylor Otwell on 2011. It runs on PHP 7.0 on the latest version, but still the minimum requirement is PHP 5.6.4. Laravel was designed to make development simpler and follow the conventions so it can be used and developed for a long term application.

It provides some tools that make generating files or configure settings easier. For example, there are a file called artisan, that can be executed for generating many things including model, controller, and migrations (we will discuss about that later).

At first, a newbie programmer will face a lot of challenge while installing Laravel, because it have many dependencies to make it work. It also has a readable and understandable documentation that makes it. Laravel was designed for that: difficult at first, but easy to code after that.

III. The Proposed Method

This section explains my proposed solution for modern PHP application. For making coding simpler, we use some tools for installing libraries that we want to include. The solution uses *composer* and *npm* for installing libraries inside. In fact, this application is mostly provided by third party tools like blade for html template engine, artisan for command-line interface, and Eloquent as simplified ORM (Object-Relational Model).

A. *Composer and NPM*

Composer are tools for requiring some libraries to be included in the project. For Laravel itself, there are so many tools that we have to install to make the framework works. It also includes some php extension to make it want to be installed by composer. For more details, see here.

NPM is acronym for "Node Package Manager", so mostly it is provides template for non-PHP file. It provides libraries for preprocessing CSS and Javascript files instead of processing PHP files that composer did.

B. *Third-party Tools*

a) Artisan Console

Artisan is a tools for command-line interface that provided by Laravel, to making MVC more readable using standard conventions. It can be used with the "**php artisan <<task>>**" command. There are so many different task that can be used and generated from artisan command.

Artisan can be required together with installing with the composer itself, so we don't have to install it seperately.

b) Blade PHP

Blade is the html template engine for processing html page. It can be used to reduce the redundancy of the code, using Object-Oritented-Programming paradigm, such as **extends** or **include** sections from other file, so we don't have to copy-and-paste the code all over again.

c) Eloquent

Eloquent is a Object-Relational Model example for laravel to represent relationship between Models in the project. It has some magic feature that can be used as a validation while inputting the files. Details of this will be explained later.

d) Tinker

Tinker is an artisan command for testing and make an interaction with the project. It can be used to test the model, or even save data to real database in the project. Last, we can see and make a query like in MySQL to perform CRUD in data.

IV. Experiment Results

From the method I proposed before, these are several screenshots from a little project that I designed with command included above.

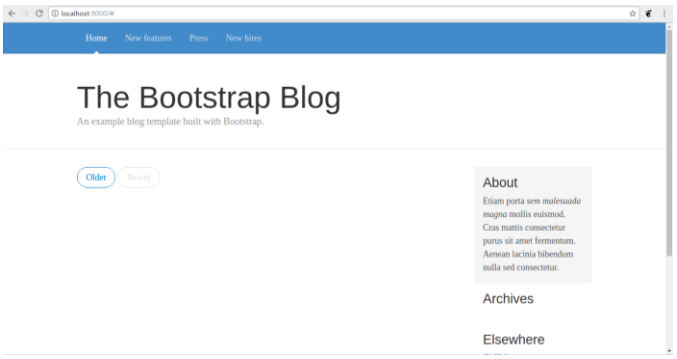


Figure 1 Blog App Home Page

Figure 2 Installation of composer dependencies

As citated in the explanation above, these are several commands that can be invoked using artisan console. The documentation is well enough to be read, can be invoked using "**php artisan**" command.

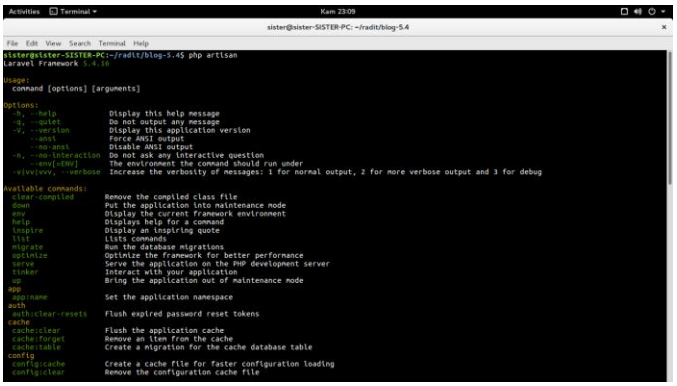


Figure 3 PHP Artisan Command

In above picture can be seen some command that provided by artisan console, which its purpose is shown beside it's command. For example, we can see there are **tinker** command, which is used to interact with application, and **serve** command, which is used to make application can be tested and be run. The usage of these commands are shown below.

Figure 4 PHP Artisan Tinker command

Figure 5 PHP Artisan Serve command

If we run "**php artisan tinker**" command, we can interact directly with application. "interact" here means we can update the database too without opening SQL interface application again! For example, see Figure 6 below.

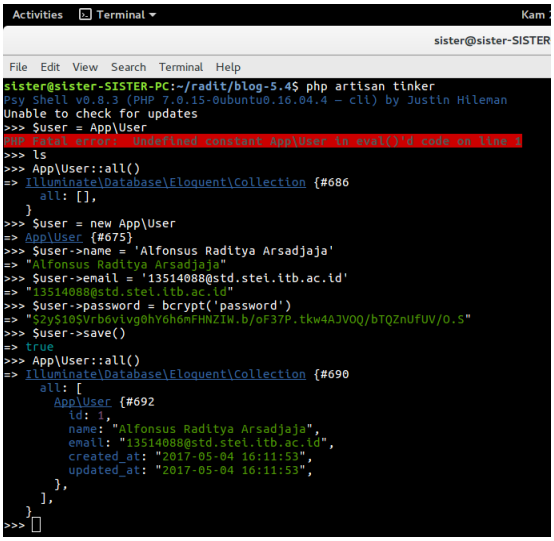


Figure 6 Insert User into Database!

From the example above, there are several things that I want to explain. The overall commands is inserting user into real database. In this case, i'm using MySQL as the DBMS, but it's not important for now. First, I'm querying all user with help from eloquent model and query (**App\User::all()** command). From the results above can be seen there are no users saved in database. Second, I want to make a user with eloquent model, so I'm inputting all attribute that I have to save, and then save it to database (**$user->save()** command). And the last one I'm making a query again to see the list of users, and there it is. There is exist a user that I have made before.

Besides of tinker, there are one last thing that I want to explain, which is **blade template engine**. As described in the previous chapter, It is used to reduce the redundancy of code in html. HTML is known for a long time and a long code. We usually see different pages with only several changes, but recode all HTML in different file. Now, with blade template engine, we don't need to worried about that again.

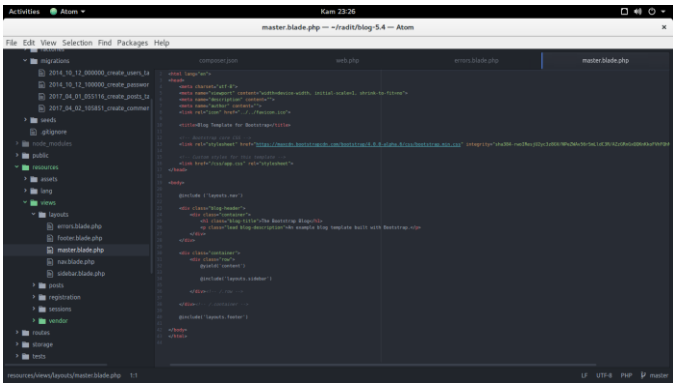


Figure 7 Master Layout using Blade Engine

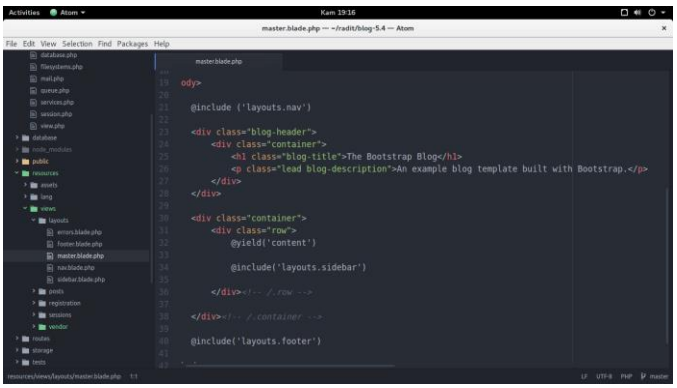


Figure 8 Master Layout Magnified

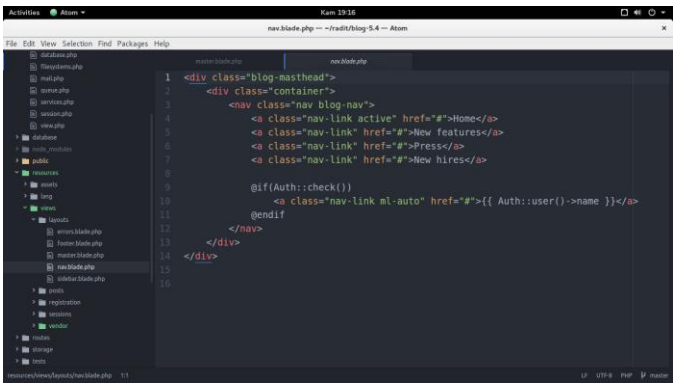


Figure 9 Files included with Blade Engine

From Figure 7, 8, and Figure 9 above, those are the basics of layouting using blade. We can see there are includes in some places in master page (Figure 7 and 8), which is used if the same section is used in other files rather than master page. The file included in master can be shown in Figure 9 (navbar file). Now because the include feature, navbar file can be reused in other file and developer only have to type one line of code rather than all the code over again.

Still from master pages, there are some sections that we want to specify with the pages specification later. It can use **yield** command for it. In the above example, there are a line contains "**@yield('content')**" code. The details will be explained later in this chapter.

Now, what if I want to extends the master code above to specific pages as well? Of course Laravel thinks about it, and the example are shown in Figure 10.

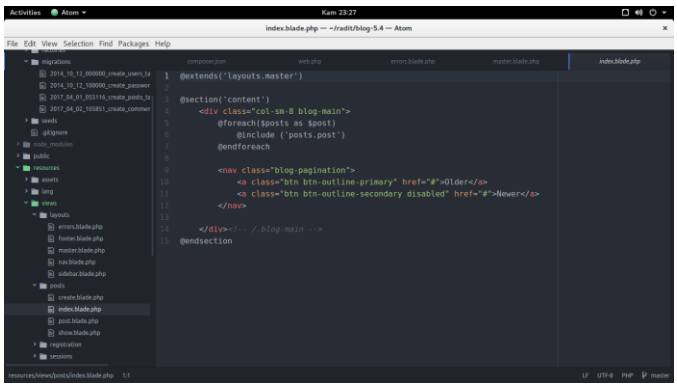


Figure 10 HTML Page extended using Blade Engine

In the code above, the index page are extending the master layout file (**@extends('layouts.master')** code). It is used for make the developer doesn't have to write it all over again. Now, the "**@section('content')**" above is what we want to specify in 'yield' command in Figure 7 and 8. The name of section and yield must be same to make it includes in the right place in the html. For example, the "name" is '**content**', so there are @yield('content') in the master file and @section('content') in the specific page, like in OOP paradigm.

V. Conclusion

From the above example, we can see that we don't have to configure much things after we have installed it. It's a lot, but there are several of them that makes a core of laravel, such as composer, npm, php artisan, tinker, and eloquent. Actually there are much more features that I can't describe here, which is also useful for development. Its already provided by Laravel itself and you only have to learn it for some time, but after that it will be fast while developing your application.

Acknowledgment

First of all, the author want to express a special thanks to Dr. Rinaldi Munir, ST. MT., Dr. Eng. Ayu Purwarianti, ST., MT., and Mrs. Dessi Puji Lestari ST, M.Eng., Ph.D. for an opportunity for making this paper. I hope that we can make a new and better solutions for making robust web application. Lastly, I want to say thanks to God for making this all happened well and this paper can be used and make an inspiration for a long time.

References

[1] http://www.hongkiat.com/blog/best-php-frameworks/, accessed May 4th 2017, 02:31

[2] http://php.net/manual/en/intro-whatis.php, accessed May 4th 2017, 03:03

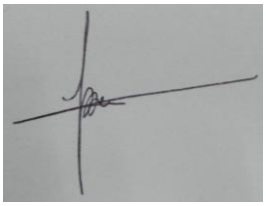
[3] https://laravel.com, accessed May 4th 2017, 19:31

[4] https://laracasts.com/series/laravel-from-scratch-2017, accessed May 4th 2017, 19:48

Statement

I hereby state that this paper is original, not a translation or an adaptation of ones paper, and not plagiarism.

Bandung, May 5th 2017



Alfonsus Raditya Arsadjaja
13514088