MVC ARCHITECTURE: A COMPARITIVE STUDY BETWEEN RUBY ON RAILS AND LARAVEL

Archit Verma

Computer Science and Engineering, Manipal Institute of Technology,
Manipal, Karnataka 576104, India
archit017@gmail.com

Abstract

Model View Controller (MVC) architecture is a standard design pattern used in website design or in web application development. It divides a given software application into three interconnected parts, so as to separate internal representations of information from the ways that information is presented to or accepted from the user. The central component, the model, consists of application data, business rules, logic and functions. A view can be any output representation of information, such as a chart or a diagram. Multiple views of the same information are possible, such as a bar chart for management and a tabular view for accountants. The third part, the controller, accepts input and converts it to commands for the model or view. In this the two popular MVC Frameworks, Laravel which uses the language PHP and Ruby on Rails which uses the language Ruby are compared and analyzed.

Keywords: Rails; Laravel; MVC.

1. Introduction

The concept of Model-View-Controller (or MVC) is a common one and has become increasingly popular as a design framework. With the wide variety of frameworks being used such as ASP.net, CakePHP, Laravel, Ruby on Rails, Spring MVC and Struts, I decided to analyze and compare the popular PHP Framework Laravel and the famous Ruby Framework Ruby on Rails.

Before studying the various aspects of the two frameworks, let us discuss in brief, the research already done in the MVC. The basic components of the MVC architecture are:

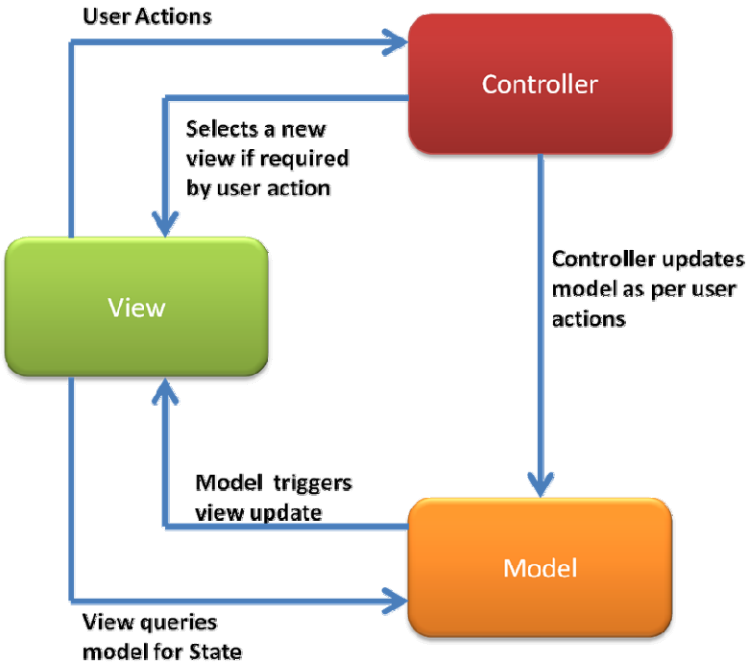

Figure 1: The working of the MVC architecture.

- The Model – The model is the part that interacts with the database to handle the data, logic and the rules
- The View – The view forms the part that interacts with the user by displaying the output and accepting the input in various forms.
- The Controller – The controller sends commands to the model to update the data as well as send commands to the view to modify the data being accepted or displayed.

To compare the two MVC frameworks effectively, we must first analyze the languages that they implement- Ruby and PHP. The major difference between the two are related to the purpose of their creation: PHP was designed to be an embedded scripting language to transform the otherwise static pages into dynamic ones. It was not originally designed to be an object oriented language, rather it was to be embedded into html and solve web specific tasks. Ruby on the other hand was designed to be a fully object oriented language, similar to Perl. However at the current stage, there is not a task which can accomplished in Ruby, that cannot be done so in PHP. Both are now at par when it comes to practical language power, code reuse, development tools and services and application hosting.

2. A Comparison of Features:

2.1. *Scaffolding*

Scaffolding is a technique used in model view controllers to specify the usage of the application database.

2.1.1. *Scaffolding in Rails*

When the scaffold keyword along with the model name is used in a controller, Rails will automatically generate the appropriate data interfaces. However, the interfaces created by this method are difficult to modify. So another method to create easily modifiable data interfaces is using the *generate scaffold <model name>* command.

2.1.2. *Scaffolding in Laravel*

Several functionalities in java are implemented using code generators, scaffolding is one of them. First the generator required for scaffolding is installed using the dependency manager called composer and then the *generate:scaffold* command can directly be used with the required arguments

2.2. *Full Text Search*

The process of searching records to quickly access required data is known as Full Text Search. Both Laravel and Rails implement Full text search using the database only and in a similar manner. The engine MyISAM must be used for Laravel.

3. Architecture

3.1. *Development Principles*

3.1.1 *Convention over Configuration*

Convention over Configuration is the software design principle of decreasing the number of decision to be made by the developer, thus increasing simplicity, although it can result in losing its flexibility. For both Rails and Laravel, Models are named as singular of their corresponding tables which are in plural, while the view pages are conventionally named the same as the functions which call them. These can be changed of course, but such a change is discouraged

3.1.2 *Test Driven Development*

Test driven development involves the developer first writing an automated test case and then writing the minimum code to pass that test.

Testing support was woven into the Rails fabric from the beginning. Every Rails application has three environments: development, test, and production. The database for each one of them is configured in config/database.yml. The command *rails generate scaffold*, can be used to create a test stub in the test/models folder and the *rake test* command can be used to run it.

Laravel uses an in-memory database, which allows for fast and clean tests. The tests are first migrated to the database and then the tests are run through the model.

3.1.2 *Don't Repeat Yourself*

Both Laravel and Rails follow the Don't repeat Yourself concept which implies maximum code reuse. Parts of code can be used like APIs with specific parameters passed for specific uses.

4. System Requirements

Both Rails and Laravel are Cross platform and thus giving them an advantage over .NET. Laravel can use SQLite, MySQL, PostgresSQL, Redis and Microsft BI databases, while Rails can use MongoDB, Drizzle, Oracle, IBM DB2, CouchDB, Cassandra, MariaDB, MemcacheDB, NoSQL, elasticsearch and Microsoft SQL Server Express in addition to these.

5. Others

There are several common and different features of both Laravel and Rails that make them premier MVC architectures. The important ones are listed below

Table 1. Important features of Laravel and Rails.

Feature	Laravel	Rails
Multi-user system	Yes	Yes
Autofocus	No	Yes
Pingback	Yes	Yes
Image processing engine	No	Yes, but via third party
Interpreter	Yes	Yes
Trackback	Yes	Yes
Database model	Object-oriented	Object-relational,Dynamic Schema, NoSQL,XML Database, Schema-less ,Key-value,Pub/Sub,Multidimensional
Template language	Blade, PHP, Smarty, Twig	ERB, HAML, Erubis, Radius, Slim
Target audience	Web Development	Web Development, App developer, Media and publishing sector, System Adminstration

6. Conclusion and Future work

Due to being relatively new, several new features have been added to Laravel within its 3 year lifespan, the major update being from Laravel 3 to Laravel 4 where bundles were replaced by generators and composer. This resulted in a major issue with backward compatibility. However with Laravel 4 being more than at par with other major MVCs it is not expected to have such a major update in the near future.

Rails was one of the first MVC frameworks, and is still one of the most widely used. With relatively frequent, backward compatible updates the most recent of whom being the use of the asset pipeline which greatly improved the way javascript and css is rendered. These updates and a vast open source community ensure that Rails has a secure future in the programming world.

References

[1] Micael Gallego-Carrillo, I. G.-A.-H. (2005). Applying Hierarchical MVC Architecture To High Interactive Web Application . Third International Conference I.TECH - 2005, (pp. 1-6).
[2] Kingsley Sage. (n.d.). Model View Controller (MVC) architecture. 18-27
[3] Vedavyas J , Venkatesulu.S IJCET(2013) Volume 4 Issue 3-A Study of MVC – A Software Design Pattern for Web Application Development on J2EE Architecture
[4] Official Documentation of Laravel at http://www.laravel.com
[5] Officail Documentaion of Ruby on Rails at http://guides.rubyonrails.org/