Comparative analysis between two web application design patterns Ruby on Rails Using Model View Controller over Sinatra

Abubakar Muhammad Dadile[1] Hauwa Abubakar[2,*] Umar Mohammed Abatcha[3]
Department of Computer Science, School of Secondary Education (Sciences),
Umar Suleiman College of Education Gashua Yobe State

Corresponding Author:
E-Mail: bbkrhauwa@gmail.com
Phone No. : 08069018040

Abstract

Web applications on Rails use a programming architecture known as MVC which divides important application functionality into three components: Model, View, and Controller. Contrarily, the Sinatra is considered a lightweight Ruby framework for creating simple web applications with minimal effort. They are all solving a diverse set of issues, even though they indeed overlap. In comparison, the rails framework focuses on writing a model-driven application. Sinatra is a library for managing all activities with HTTP from the server side. An intensive observation was carried out on Rails and Sinatra which are strong options for web application development, each with its pros and cons, and of which Rails with MVC is more reliable among them. All these design patterns are largely helping in developing applications that are loosely combined, and easy to test and maintain. But MVC is faster, more efficient, and preferably most suited to Ruby on Rails nowadays.

Keywords: MVC, XML, JSON

Introduction

Ruby on Rails is an open-source framework for web application development built on the Ruby programming language. Often simply referred to as Rails, it is a server-side web application framework built with MVC philosophy, which enables modularity and extensibility of application that leads to less coupling and high cohesion.

"Model View Controller can be defined as an application design pattern that comprises of three interconnected parts which include a model (program data), the views (user interfaces) and the controller (processes that handle input)". Christensson, P. (2018)

This architecture helps to structure an application by separating Data, business logic, and UI. i.e. three components: Model is where we do all data management, parsing, fetching, basically everything data. The controller handles the business logic, data flow from the model, views, and incoming user requests. The view is responsible for how and what the user views.

According to Haarika R. (2021), one way to speed up the productivity of a developer is by enforcing the structure of the application. That means Rails organizes the folder structure to file and variable naming. Because it standardizes structure and program conventions, the decision-making process is a lot easier and allows powerful abstractions in the code.

By delving into the core web application development with the architecture MVC in creating Rails projects to be considered as an application, not websites, as they deliver more than HTML for browsers. Rails apps also offer XML, JSON, and other rich content. Ruby on Rails is a fully stacked framework. Compared to Sinatra which is a very lean micro-framework. With Sinatra, there is no MVC pattern and does not encourage using that convention.

Why MVC is better than other design pattern such as Sinatra for web applications design.

Web applications on Rails use programming architecture known as MVC which divides important application functionality into three components: Model View and Controller. On the other hand, Sinatra is considered a lightweight Ruby framework for creating a simple web application that requires minimal effort. They are solving a diverse set of issues, even though they overlap. At the same time, the Rails framework focuses on writing a model-driven application. Sinatra is a library for managing all activities with HTTP from the server side. For example, a programmer that needs full integration and standards as conceivable as possible goes with MVC. Cooper P. (2009)

Model View Controller is one of the most commonly used web application development design patterns where Rails generates and promotes code for each element as a separate file which is categorized into three main interconnected layers these are the model, the view, and the controller. Each of these components is built to handle specific development aspects of the application. Ruby on Rails is built upon the MVC paradigm, MVC is an integral part of the Ruby on Rails framework, which signifies that the model houses code that works with the

application data and is responsible for centralizing the business logic of the application and the rules of manipulating those data. The view is responsible for an embedded ruby code that determines how the user interacts with the application and the presentation of data. Patrick et. al., (2016), states that the controller communicates with the model and views by receiving a request from the browser, works with the model to process data, and instructs the view on how to display the result to the user accurately as depicted in figure 1. below.



Fig. 1 Ruby on Rails MVC

Building a new application using only Ruby is a tedious job. Therefore, a dedicated framework was created to streamline the programmer's application development process. It provides the fundamental pieces of programs for desktop applications, as well as web application development. MVC is commonly used to develop modern user interfaces and it works well with an object-oriented programming language since the different models, views, and controllers serve as an object and reuse within an application. Christensson, P. (2018)

Ruby on Rails MVC allows for parallel development and enables programmers to speed up the engineering process as fast as possible. The framework provides ready-to-use baskets for the separation of the app business logic, so a Ruby on Rails developer can save time through its utilization. Olha A., (2019)

The main task and feature of Ruby on Rails is its fastest creation of new projects with high performance by separating the software application into three distinct components, which is a good idea in practice based on the following reasons:

Improved scalability: (is the ability for an application to be upgraded) – For example, when the application is experiencing some performance issues because the result of database access is slow, you can upgrade the hardware running the database without other components being affected.

Reusability – a model can be reused by multiple views. For example, this helps programmers build applications that are lightweight and give full control to build and reuse multiple views (DRY). Cooper P. (2009)

Ease of maintenance – as the components have a low dependency on each other, For example, making changes to one such as fixing bugs and changing functionality does not affect another. Patrick et., al., (2016)

Consistency and clean code – provide ready-to-use code that simplifies the implementation of many features. For example, you can update all features as fast and seamless code to read and sort through. Patrick et., al., (2016)

Brief descriptions of another design pattern such as Sinatra and its weakness:

Sinatra is a small, lightweight web framework and domain-specific language written in Ruby designed for the speedy creation of websites, web services, and web applications. Sinatra is built upon Rack and It emphasizes a minimal approach to development, offering only what is essential to handle HTTP requests and deliver a response to clients. Even though Sinatra does the most amazing things with routing. For example, it is flexible when routing which is essentially a regular expression that matches URL requests using some commands like Get, Post, Puts, and Delete as shown in the figure below.

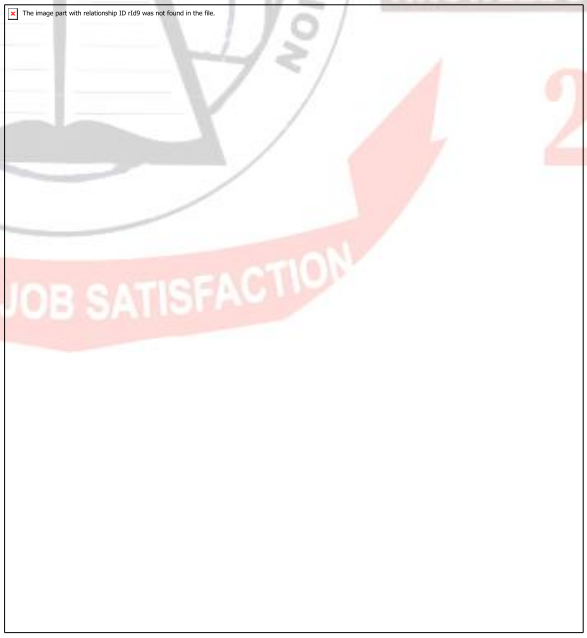

Fig.2 Sinatra internal structure

Sinatra is well-fitted for small micro-apps or APIs, by implementing simple APIs for easy access to data, Yates, J., (2013). It does not support scaffolding and asynchronous workload. For example, when you tend to have some asynchronous I/O file uploads, web sockets or other high-performance concurrent workloads are not suitable.

Sinatra Drawback

Sinatra has no built-in ORM (Object Relational Mapper) tools, no pre-fab configurations file, you can't even get a project folder until you create one.

Sinatra requires running Unicorn, Passenger, Thin, or some other threaded Rack-compatible server.

Rails are better suited for big projects with MVC. Therefore, MVC is preferable to the other sorts of design patterns based on the above statements and also offers many advantages as follows.

It can handle multiple views using the same business model, it is easier to maintain, test, and upgrade the whole system.

The model is completely decoupled from view to allow lots of flexibilities and implement the model considering the reusability and modularity. This model can be extended for further distributed applications to design. Easier to add new clients just by adding their views and controller, Neeraj, S. (2013)

Conclusion

All these design patterns generally help in developing applications that are loosely coupled, and they are easy to test and maintain. But with Rails MVC architecture, web applications appear to be faster, more efficient, and reliable and as well can handle asynchronous overloads. Web applications built with ruby on rails are considered appropriate and have basic divisions between layers. The model is completely decoupled from view to allow a lot of flexibilities and implement the model considering the reusability and modularity. This model can be extended for extensive distributed applications to design. Whereas, Sinatra is a small, lightweight web framework, and domain-specific language written in Ruby established for the speedy creation of websites, web services, and web applications. Sinatra is built upon Rack and it emphasizes a minimal approach to development, offering only what is essential

to handle HTTP requests and deliver a response to clients but does not support asynchronous workload.

References

Christensson, P. (2018). *MVC Definition*. Retrieved 2023, Sep 21, from https://techterms.com

Cooper P. (2009) Web Application Frameworks: Rails, Sinatra, and Ramaze. In: Beginning Ruby. Apress, Berkeley, CA.[22 September 2023]. pp 349-411.

Haarika R., (2021) Introduction to Ruby on Rails and How MVC works. Nerd for Tech Publisher

Neeraj, S. (2013). Comparative Introduction to MVC and MVP. Retrieved 2023, Sep 23, from https://www.codeproject.com/Tips/31292/MVC-v-s-MVP-How-Common-and-How-Differentw

OlhaAnurina, (2019). *What is Ruby on Rails Used for: Tips from Back-end Developers.* Retrieved 2023, Sep 21, fromhttps://mlsdev.com/blog/what-is-ruby-on-rails-used-for

Patrick L., Glenn G. (2016). Rails: Novice to Ninja, 3rd Edition.SitePoint

https://www.oreilly.com/library/view/rails-novice-to/9781492017554/

Yates, J (2013), Instant Sinatra Starter, Packet Publishing, Limited, Olton. Available from: ProQuest Ebook Central. [25 September 2023].pp 35-42.

Krzysztof W. (2023). Ruby on Rails vs Sinatra: Use the right framework for your next project. https://www.netguru.com/blog/ruby-on-rails-versus-sinatra