Development with Ruby-on-Rails

A SENG 403 Term Paper

Y. Al-Bender, A. Eidelberg, Z. Huang, C. Mah, B. Triebwasser, K. Tuitoek
Department of Computer Science
University of Calgary
Calgary, Canada

Abstract: The purpose of this paper is to present the Ruby on Rails framework to the reader. No previous knowledge of Ruby (or any other programming language) is assumed for the first few sections; however technical details are discussed in a later section.

Keywords: Ruby on Rails, Ruby, Rails, web application framework, open source project, software engineering patterns

I. INTRODUCTION

Time is money. This has always been the case, but is especially true in today's age where new technologies spawn on a monthly basis, and businesses launch competing applications to the web almost on a daily basis. Due to this rapidly changing environment, time is an important factor when choosing a web application for a development project. Whether they are small or large projects, time should be spent developing features and fixing bugs, instead of configuring details and developing general purpose tools.

Therefore, most developers decide to use software frameworks. But what is a software framework? It is a layer of abstraction introduced to the working parts of the system [1]. A framework is more appropriate for general cases, and not suitable for individual circumstances. In contrast with code libraries, the flow of control for the system, as well as the majority of the code, is usually not meant to be modified [1]. Instead, extensibility is provided in other ways so that the general framework can be narrowed down to support specific programs and applications. Another common attribute to software frameworks, is that they usually provide a default behavior [1]. While this default does not solve all the problems in a project, it is often a good place to start for developers by providing some out-of-the-box functionalities [1].

Thus, frameworks are essential for a businesses' success; while some companies decide to build their own frameworks, many solutions are currently available for them. In the latter case, businesses should research into frameworks as there are several to choose from. They include many default options, and often use specific technologies with different advantages and disadvantages. This means that a certain task on one framework might be more difficult to implement than on another framework. In some cases, a task might be nearly impossible to implement on certain frameworks!

This means that knowledge about existing tools is an essential to project development. This paper helps entry level developers get a general feel for the Ruby programming language on the Rails framework. It will start by discussing the roots and history of Ruby on Rails, how it became successful and the factors of its creation. All of these concepts will demonstrate the need for a framework, as briefly discussed earlier. This is followed by a section that compares key features of Ruby on Rails to other popular frameworks. Lastly, the paper will provide some technical details about the language and framework, and lists some resources for those interested in pursuing Ruby on Rails.

There are many books and articles on the details of the system, all of which aim to provide detailed explanations and help make the reader an expert of the system. It is important to stress that, in contrast to those resources; this paper aims to provide a high

overview of the framework in general. It is intended to give a first impression of the framework, and to provide just enough information so that the question, "why would I choose the Ruby on Rails framework over other available frameworks and solutions?" is answered to readers.

II. History

Before we ride on rails, it is worth mentioning Ruby, which is a programming language that is a base for Rails. Ruby is a dynamic, reflexive, object-oriented and general-purpose programming language developed by a Japanese developer called Yukihiro Matsumoto in 1995 [2]. Ruby embodies the syntax inspired by Smalltalk's conceptual elegance, Python's ease of use and learning, Perl, and Lisp, to name a few [2].

Today, Ruby is the fastest growing language that is rapidly gaining mindshare in the United States and Europe [2]. The reasons for this is because of its rich build-in libraries, ease of learning, ease of extensibility, and overall lower code overhead which results in fewer bugs. In addition to all these benefits, it is also an open source project.

The Rails framework is a productive web-application framework written by David Heinemeier in 2003 [2]. He extracted Ruby on Rails from his work on Basecamp [2], a project management website famous for its ease of use on multiple platforms. One of the biggest advantages of using the Rails framework is that one could develop a web application at least ten times faster than with a typical Java framework, which is widely used in the industry [3]. In Java, a lot of time is spent on configuring XML configuration files, but in Rails the code and database schema are included in the configurations [4, section 2.17]. Due to its inspiration from Smalltalk's minimalistic approach, similar applications have fewer lines of code in Ruby on Rails than in the Java framework [4, section 2.17]. This design paradigm is known as "coding by convention", which decrease the number of decisions developers are required to make, making things simple [5].

Over the years, many features have been introduced into the Rails framework for a developer's convenience. Examples include templates, engines, and the ability to view paths and models [2]. Templates are used to enable developers to generate skeleton applications, which decrease development time; and engines are miniature applications that provide functionality to their host applications [4, section 2.5]. In addition to these, Rails allows developers to optimize pieces of code which help deal with multiple concurrent HTTP requests [4, section 12.3].

On December 2008, Merb [6], another web application framework, was launched. Ruby on Rails announced that it would work with the Merb project to bring the best ideas of Merb to Rails [2]. This would avoid unnecessary duplication across both communicates, and Merb was merged with Rails as part of the Rails 3.0 release [2]. The main focus in Merb was on its speed and extensibility, which were factors that developers complained about in the early Rails framework [2]. Since the rails 3.0 release, developers are able to spend more time on coding instead of working inside the framework stack.

Rails continuously released new versions for the developers. The newest release, Rails 4.0, came out on June 25, 2013 [2], and introduced new features. One of them was called the Russian Doll caching system, which help reduce overhead on the framework stack by updating the cached elements when necessary [7]. Another feature that was introduced is Turbolinks, a system which reduces the amount of overhead caused by reloading web pages [8]. This is achieved by using javascript to replace HTML contents dynamically without the need to reload the page [8]. The last big feature introduced was the highly anticipated Live Streaming, a commonly used feature in websites such as Twitch [9]. In addition, Rails created other optional components such as the Active Resource, and the Active Record Observer. The Active Resource allows easy communication between multiple Rails applications [10, section 2], and the Observer reduces clutter in classes by having one class observe other classes for changes [10, section 2].

Currently, Ruby on Rails were used for popular websites such as Twitter [11], a social media platform to create and share ideas[1]; GitHub [12], an online revision control platform; Groupon [13], a platform to get daily deals for local companies; Basecamp [14], a project management tool, and so forth. Major corporations such as Amazon, BBC, CNET, IBM, NASA, Yahoo! and eBay even use Ruby on Rails for some of their projects [15]. This is because Ruby on Rails is a full-stack framework used to make dynamic websites that require information from a web server [4, section 2]. It is used to make complex websites or applications involving a database [4, section 2]. It also emphasizes software engineering patterns and principles such as active record pattern, where data is stored in relational databases [4, section 2]; convention over configuration, also known as coding by convention; don't repeat yourself principle, reducing repetition of information and syncs logically-related elements; and model-view-controller, which will be discussed later in the paper[4, section 2].

As with any open source projects, the community's activity is a large and important factor in its success. Ruby on Rails has a number of large communities where best practices are shared, reinforced, and even questioned. They are also responsive and helpful to novice developers trying to learn the language [2].

III. Comparison to Other Solutions

As mentioned before, there are many languages and frameworks available for developing web applications. Since the middle of 2006, statistics have shown that Ruby has had an increase in usage; while other languages such as Java and PHP have shown a decrease in usage [4, section 1]. By comparing Ruby on Rails to other programming languages and frameworks, it is clear that Ruby on Rails has risen in popularity due to its distinct features and simplicity.

Comparative to PHP, Ruby on Rails is more object-oriented [4, section 1]. This provides the advantages of object-oriented programming, such as modularity, maintainability, and extensibility [4, section 1]. Another advantage of Ruby on Rails comes from the fact that it is an entire web framework, whereas PHP is merely a scripting language [4, section 2]. This provides the convenience of a built-in web server that Ruby on Rails has, while PHP requires other components such as an Apache HTTP to be used [4, section 2]. Ruby on Rails also implements the Model-View-Controller architecture, while PHP provides no native support for this architecture [4, section 2]. This separates responsibilities and allows flexibility in the code as developers can add a subclass to the model for new tasks [16].

Another popular web programming language is Java, which is considered more secure, scalable, and has more widely available development tools than Ruby on Rails [4, section 1]. However, Ruby on Rails provides simplicity and fast development speed [4, section 1], which is attractive to web application developers in today's fast-paced work environment. Ruby on Rails provides faster development time compared to Java, because it provides a seamless framework that allows developers to jump straight into development, whereas Java (JEE[2]) requires more configuration of components in order to start a project [4, section 2].

Developing a web application requires the ability to access and manipulate data stored in a database, such as MySQL [17], Oracle [18], or MSSQL [19]. Ruby on Rails provides a Model-View-Controller architecture to work with databases [16]. Implementing this architecture must be external as well. A database table is represented by a class whose attributes represent columns of the table [16]. An instance of that class represents a single row in the table. Typically, one database table is represented by one model [16]. The model represents information from the database, but it should be independent from the database [16]. It validates the data before it goes into the database, and contains the business logic [16]. The view is a presentation layer of the application, and renders models to format such as XML or

[1] However when Twitter expanded, it switched to Scala for scalability reasons, and built-in guarantees about application state [22].

[2] Java Platform, Enterprise Edition [http://docs.oracle.com/javaee/6/firstcup/doc/gkhoy.html]

Javascript [16]. Lastly, the controller is a class that extends the `ApplicationController` class [4, section 2]. It contains the logic to process the data by connecting the model to the view, and should not access the database. The controller knows how to process the data that comes from the model and how to pass it onto the view for rendering.

A Model-View-Controller architecture minimizes external configuration by using naming conventions for mapping database tables to model objects [16]. PHP focuses on a view-only system [4, section 2]. Data is accessed with PHP using SQL queries that are embedded in the logic of the program in order to construct a page to be rendered to the end-user's browser [4, section 2]. The JEE (Java) framework supports the Model-View-Controller architecture, but requires extensive configuration of components in order to support that functionality [4, section 2]. The Java Database Connectivity (JDBC) component can also be used to interact (i.e. execute queries) with a database; this is similar to how PHP achieves database interaction [4, section 2].

Like any programming language, web applications also require proper testing to ensure the software does what it is expected to do and properly handles unexpected cases. In order to test a Java application, a regression testing framework such as JUnit [20] can be used to implement unit tests that test specific portions of the code. Similarly, PHP can use a framework such as PHPUnit [21] (based on JUnit) to implement unit testing on PHP code. In contrast to the external testing frameworks, Ruby on Rails has a built-in testing framework for unit tests. This also tests Ruby code similar to how Java and PHP unit testing solutions work, but the biggest difference is that it dynamically generates folders for testing when a developer creates a new class [10, section 7].

Ruby on Rails has become popular for its ease of use; however it does have limitations that prevent the framework from overtaking other popular web frameworks, such as Java, PHP, or Python. The biggest downfall with Ruby on Rails is its scalability [4, section 1]. Ruby on Rails is not widely used as the main framework in massive software systems. It cannot power large websites, such as Google, Youtube, and Facebook. Twitter switched from Ruby to Scala when it needed to grow in size; although scalability was not the primary reason for this, it is one of them [22].

One reason for the limitation on scalability is that Ruby on Rails focuses on a 'one-size-fits-all' approach. This means that the framework incorporates everything; this makes integration of Ruby on Rails applications with other applications or frameworks complicated and time consuming [4, section 2]. The primary reason is because other applications are not developed in Rails and goes against what it was designed and optimized for. As previously mentioned, Ruby on Rails often has less configuration involved compared to languages like Java, but it can be its downfall as it limits the customizability of a configuration needed for larger web applications.

IV. Differences in Syntax

The main focus in Ruby is simplicity. It tries to follow Python and Smalltalk, where programmers type strictly only what is needed. This also explains the fact that Ruby was designed as a dynamic type language. This means that the same variable can store different types of data [10, section 1].

Therefore the following code does not product an error. Note that `irb>` is the terminal, and `->` is the result.

```
irb> x – “Hello World!”
-> “Hello World!”
irb> x – 3
-> 3
```

While this is not unique to Ruby (as PHP and other languages act similarly), this allows the code to reuse variables, and allows functions to return different types of variables under different circumstances [23, section 3]. This goes well in line with Ruby's general approach towards reducing the time it takes to produce code. However, while dynamic typing has advantages for development time, the compiler cannot predict variable types [23, section

3]. This affects the efficiency of the language, as strict typing allows the compiler to have more information, and thus produce better optimized machine instruction code. This affects a language's ability to catch exceptions and handles them without crashing the entire program [23, section 3]. Finally, one of the biggest disadvantages of dynamic typing languages is that they are not able to recognize nonsensical code [23, section 3]. When a language knows which types an operation supports, during compilation time, the programmer can be alerted of errors. Dynamic type languages are unable to do this [23, section 3]. At times, this can prevent logical errors that result in bugs that are otherwise very difficult to find. For example in Java, when dividing a String type by an Integer type, an `Exception` is thrown during compilation time; in dynamic typing, the program crashes once it reaches to that code [23, section 3].

Ruby has an interesting set of syntax decisions which are different from other popular languages such as Java or PHP. One key feature that Ruby makes use of, is symbols such as "`!`", "@", and "`?`" to represent specific meanings in the language's syntax [23, section 3]. For example, an "`!`" after a method name indicates that the method changes the content of the variable, and a "?" indicates the method returns a Boolean type [23, section 3]. The following code highlights these concepts.

```
irb> x – "Hello World!"
-> "Hello World!"
irb> x.upcase!
-> "HELLO WORLD!"
irb > x
-> "HELLO WORLD!"
irb > x.empty?
-> false
```

Another fact in the Ruby language is that semicolons are only used to separate multiple statements on the same line, and parentheses and tabs are not required for method calls [23, section 3]. However, those can be optionally provided and used for better coding style. Another difference is that the scope of a method is between the "`def`" and "`end`" tags [23, section 3].

The following code is an example of a `Car` class in Ruby with comments (marked in #), which highlights important syntactical concepts previously discussed:

```
class Car
    # class variables (in Java they are
    static fields)
    @@number_of_cars = 0
    @@wheels = 4

    # constructor for new objects
    def initialize
          @@number_of_cars += 1
    end

    # class method unable to call from
    objects directly
    def self.count
          @@number_of_cars
    end

    # setter method, same name as var
    def mileage=(x)
          # instance variable, does not
          have to be declared
          @mileage = x
    end

    # getter method, same name as var
    def mileage
          @mileage
    end

    def honk
          puts "Honk!" # prints line
    end
end
```

Using the class is as follows:

```
class Limo < Car
    @@wheels = 6
end
```

V. SETTING UP THE ENVIRONMENT

Ruby is very easy to download and install, and it is supported by most common integrated development environments (IDE), such as RubyMine [24], Vim [25], Komodo [26], Eclipse [27], NetBeans[28], and Aptana Studio [29].

The first step is to install Ruby. RubyGems [30], a package manager for Ruby is also automatically installed. This allows other packages to be installed which extend the Ruby language [30]. Rails is a gem that can be installed using the gem interface [10, section A]. Currently Ruby's stable release is 2.1.1 as of February 24, 2014; and Rail's stable release is 4.0.3 as of December 3, 2013 [2].

In Linux or MAC, the following steps should be taken:

1. Download and install Ruby [https://www.ruby-lang.org/en/downloads/]
2. Use the gem installer to install Rails by typing "`gem install rails`" in the terminal

Installation on Windows adds an extra step to ensure the gems will work correctly on the operating system. The SDK adds an adapter to WinGW so the Gems will work properly. The following steps are:

1. Download and install Ruby [http://rubyinstaller.org/]
2. Download and install the corresponding development kit (SDK)
3. Use the gem installer to install Rails by typing "`gem install rails`" in the terminal

On Linux, the Ruby Version Manager (RVM) can be installed. This is an extended command line tool for managing multiple ruby environments [31]. To proceed, enter the terminal and following the steps:

1. Update the currently installed packages by typing "`sudo apt-get update`"
2. Install curl with "`sudo apt-get install curl`"
3. Install the newest stable release of RVM with "`curl –L get rvm.io | bash –s stable`"
4. Execute RVM with "`source ~/.rvm/scripts/rvm`"
5. List the dependencies with "`rvm requirements`"
6. Find the Ruby dependencies and install them with "`rvmsudo`"
7. Install Ruby version 1.9.3 with "`rvm install 1.9.3`"
8. Set Ruby 1.9.3 as the default Ruby with "`rvm use 1.9.3 --default`"
9. Update RubyGems with "`rvm rubygems current`"
10. Install Rails with "`gem install rails`"

VI. RESOURCES

Rails Guide [32] is the best resource to have when coding in the Ruby on Rails environment. It has information on what is possible in Rails and details on how to use it. There is a guide on how to get started on a first application, and has thorough documentation regarding models, views, and controllers. This is an easy-to-read reference and a great starting place for developers.

Railscast [33] is a great website for tutorials on Rails. However, most of the content is not free. It provides tutorials, tips and tricks on using Ruby on Rails, and screencasts. It is well produced and short.

TryRuby [34] is an interactive tutorial website, which is a great starting point when learning the basics of Ruby.

Rails for Zombies [35] is also interactive which offers a series of videos, and a gamefied experience for novices. The tutorial asks to solve challenges to build a website for zombies, while offering badges for achievements.

CodeLearn [36] is another interactive website that allows novices to develop in the Rails environment. It has a terminal, file browser, code editor, and displays the output result.

CodeAcademy [37] also offers interactive tutorials and badges. For a true gamefied experience, novices can sign up for an account, and learn other languages.

The Odin Project [38] is a website that also have a series of tutorials and project specifications. However, this is not interactive and is a comprehensive guide.

There are several other resources on the web, but these are one of the most popular and useful.

VII. Conclusion

Ruby on Rails is a framework that builds from the Ruby language and provides many features for fast development. There are a number of other frameworks available, and it is important for a developer to gain a general overview of a given framework when planning for a project.

This paper has provided the reader with a brief history of the framework; discussing the benefits of using Ruby over other solutions, in addition to which frameworks work in different cases; briefly describing the language design choices and some of the syntax; provided a guide to install Ruby and Rails; and touched on a few resources for interested readers.

Acknowledgments

We would like to thank Rob Kremer and Julia Paredes who provided additional comments and suggestions for the paper.

References

[1] Riehle, Dirk. “Framework Design: A Role Modeling Approach”. Ph.D Thesis, No. 13509. Zurich, Switzerland, ETH Zurich, 2000.
[2] “Ruby on Rails.” rubyonrails.org. RubyonRails, n.p., n.d., 20 Feb 2014.
[3] Farnworth, Demian. “20 Reasons Why Every Web Designer Needs to Learn Rails.” *Treehouse Blog*. TeamTreeHouse., 18 Sep 2012, 19 Feb 2014.
[4] Vohra, Deepak. “Ruby on Rails for PHP and Java Developers”. Berlin: Springer, 2007. Print.
[5] Miller, J. “Design For Convention Over Configuration.” Microsoft, 2009.
[6] “Merb.” Merbivore.com. 19 Feb 2014.
[7] “Caching with Rails: An overview.” guides.rubyonrails.org/caching_with_rails.html, n.p., n.d., 20 Feb 2014.
[8] “rails/turbolinks.” github.com/rails/turbolinks#turbolinks. Ruby on Rails, n.p., n.d., 20 Feb 2014.
[9] “Twitch.” twitch.tv. 19 Feb 2014.
[10] Tate, Bruce; Hibbs, Curt. “Ruby on Rails: Up and Running.” O’Reilly Media, 2006. Print.
[11] “Twitter.” twitter.com. 20 Feb 2014.
[12] “GitHub.” github.com. 20 Feb 2014.
[13] “Groupon.” groupon.com. 20 Feb 2014.
[14] “Basecamp.” basecamp.com. 20 Feb 2014.
[15] Mornini, Tom; Yard, Engine. "Here's Why Ruby on Rails Is Hot." Business Insider. *Business Insider*, 11 May 2011, 20 Feb 2014.
[16] Howart, Chris. “Getting Started with MVC.” *Sitepoint*. Sitepoint, 25 Oct 2012, 18 Feb 2014.
[17] “MySQL.” mysql.com. 18 Feb 2014.
[18] “Oracle Database.” oracle.com/us/products/database/overview/index.html. 18 Feb 2014.
[19] “Microsoft SQL Server.” php.net/manual/en 18 Feb 2014.
[20] “JUnit” junit.org. 21 Feb 2014.
[21] “PHPUnit” phpunit.de. 21 Feb 2014.
[22] “Scaling Twitter: Making Twitter 10000 Percent Faster.” High Scalability. *High Scalability*, 27 Jun 2009.
[23] Williams, Justin. "Rails Solutions: Ruby on Rails Made Easy." Apress, 2007. Print.
[24] “RubyMine.” jetbrains.com/ruby. 21 Feb 2014.
[25] “Vim.” vim.org. 21 Feb 2014.
[26] “Komodo.” komodoide.com. 21 Feb 2014.
[27] “Eclipse.” eclipse.org. 21 Feb 2014.
[28] “NetBeans.” netbeans.org. 21 Feb 2014.
[29] “Aptana Studio.” aptana.com. 21 Feb 2014.
[30] “RubyGems.” rubygems.org. 21 Feb 2014.
[31] “Ruby Version Manager.” rvm.io. 21 Feb 2014.
[32] “Rails Guide.” guides.rubyonrails.org. 21 Feb 2014.
[33] “Railscast.” railscast.com. 21 Feb 2014.
[34] “TryRuby.” tryruby.org. 21 Feb 2014.
[35] “Rails for Zombies.” railsforzombies.org. 21 Feb 2014.
[36] “CodeLearn.” codelearn.org/ruby-on-rails-tutorial. 21 Feb 2014.
[37] “CodeAcademy.” codeacademy.com/tracks/ruby. 21 Feb 2014.
[38] “The Odin Project.” theodinproject.com. 31 Feb 2014.