Success of Grails and Rails in Near Future

Levis Abraham[#1], Krutuja Niphadkar[*2], Neha Chopade[#3]

[#1] *MCA student, SIES College of Management Studies, Navi Mumbai, India*

[#2] *MCA student, SIES College of Management Studies, Navi Mumbai, India*

[#3] *Associate Professor, SIES College of Management Studies, Navi Mumbai, India*

Abstract*: As there is a need for the possibility to deploy new applications in a short period of time there is also a need for a framework, which facilitates those demands. Hence the idea behind was to give a detailed explanation of two powerful web application frameworks which are truly object oriented programming language & dynamic in nature. Both frameworks are derived from different languages i.e. Groovy & Ruby, with different features, similarities & differences. Grails & Rails have been progressing & developed a lot over the years. We chose this topic for our research, so that after studying, analyzing & understanding both the frameworks, we could derive their progress in the market. Both are used for rapid application development which is easy to configure and produce. They work on principles of convention over configuration, scaffolding and don't repeat yourself (DRY). It also has an active user community in the IT market. To further illustrate our work, we gave an overview insight of their growth, by compiling & giving statistics reports & trends which were analyzed by different industry experts & ourselves. Based on the reports & work we are producing, it shows they are well established in the market with huge companies like Twitter, LinkedIn, Vodafone etc, using their frameworks. In conclusion with respect to our work, one should choose framework wisely based on the size of one's application & the scope of the application. We are also highlighting their past success & how it will continue to be successful in the near future.*

Keywords: *Grails, Rails, CoC, DRY, Scaffolding, MVC, Active Record*

I. INTRODUCTION

Ruby was initially designed and developed in the mid-1990s by Yukihiro "Matz" Matsumoto in Japan, under the influence of other similar scripting languages such as Perl, Smalltalk, Eiffel, Ada, and Lisp. A clean, concise, consistent, and structured like languages such as Java, Ruby also offers the speed and ease of use of scripting languages such as PHP. Ruby is also a powerful dynamic, reflective, object-oriented, general-purpose programming language, so instead of writing large amounts of code, developers can declare commands efficiently with subtle inferences via small amounts of code [1]. In the past & also currently, many programmers have wanted a fast, productive approach that produces reliable, clean applications more quickly, using less code. That's were Ruby on Rails started appearing in the market. Rails first released publicly in 2004, which is free and available under an MIT license. David Heinemeier Hansson, a partner and programmer designed Rails by extracting various features from Basecamp, a Ruby-based project-management. Rails turned the general-purpose programming language of Ruby into a specific solution for creating Web applications, giving it direction and thereby putting "Ruby on Rails." The Rails application-development framework is purely based on Ruby. Ruby on Rails attempts to combine PHP's simple propinquity with Java's architecture, purity, and quality. Rails uses integrated programming packages and preset code, known as conventions, designed to be complete and ready to use immediately, without configuration. The language itself has grown into one of the more widely used programming languages with a large active community [2]. Convention over Configuration (CoC) is a corner stone in Ruby on Rails and along with DRY (Don't Repeat Yourself) they are the key concepts used when designing Ruby on Rails.

Groovy was created in 2003, is a powerful, optionally typed and dynamic language that was the first, and currently only, JSR approved language (JSR-241) other than Java for the JVM [3]. Other languages like Ruby fall under the umbrella of JSR-223. It has static-typing and static compilation capabilities, for the Java platform aimed at improving developer productivity thanks to a concise, familiar and easy to learn syntax. Groovy includes features found in Python, Ruby, and Smalltalk, but uses syntax similar to the Java programming language. Scripting for the Java Platform and hundreds of other languages can compile to byte code and run on the JVM. Groovy's similarities to Java and reliance on the JDK set it apart from these other dynamic languages and enable it to achieve a level of integration with Java [4]. Grails is a framework built using the programming language Groovy, which is an agile and dynamic language built to run on the Java Virtual Machine. It is intended to be a high-productivity framework by following the "coding by convention" paradigm, providing a stand-alone development environment and hiding much of the configuration detail from the developer. Groovy uses the strength of Java and new features inspired by other languages such as Ruby, Python and Perl; this

gives Grails a strong base to stand on. Grails was previously known as 'Groovy on Rails'; in March 2006 that name was dropped in response to a request by David Heinemeier Hansson, founder of the Ruby on Rails framework [5]. The background of Grails starts with Ruby on Rails gaining popularity and changing the way web development is done and with a lot of companies that have invested money in Java they are losing out on the Ruby on Rails like development, hence Grails was created. Grails is not a Ruby on Rails clone, but tries to offer a Ruby on Rails like environment and incorporate concepts that are familiar to Java developers. There is also a command line interface available for Grails, which can be used for e.g. installing plug-ins or generating an application structure, it can also be used to run self made scripts [6].

Section 2, consists of our research related work which majorly explains what grails and rails have in common. Documentation & Development Approach i.e. their respective MVC Architecture is compared and analyzed for understanding their frameworks in depth. Section 3, consists of our analysis work and their features along with similarities. In order to further understand how successfully they have deployed in the market our 4th section i.e. Section 4, consists of all statistics, trends and our review to showcase how powerful & successful both frameworks are respectively. Section 5, consists of our concluding statements from both frameworks point of view. Section 6, finally contains all the references we went through to make our research paper complete.

II. RELATED WORK

A. *Documentation and Learning*

Grails has a lot of documentations to offer for any developer to facilitate the learning process. It has in-built API and tag library, which are well documented and really helpful when developing with Grails. Grails offers several different tutorials, guides, books and lot more to guide the developer when learning how to use and work around with this framework. Ruby on Rails (RoR) is a much older framework, thus it has also been able build a large foundation of documentation, from the scratch i.e. APIs to guides and books. Rails offer a large community, which includes active users, wiki and blog related channel for asking queries. So it gives the developer a great starting point for getting started with the framework [7].

B. *Development Approach*

Rails

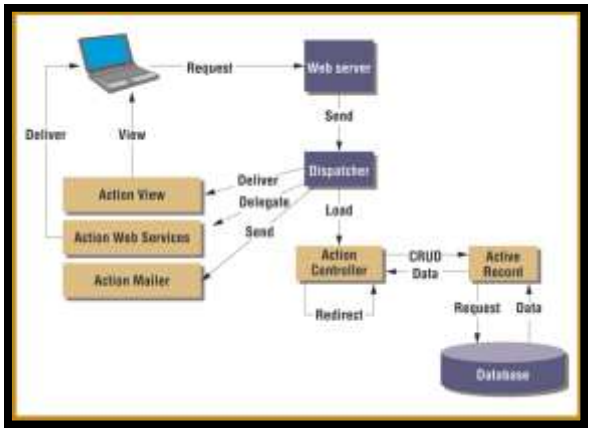


Figure 1 – Rails MVC Architecture, Source – [8].

Rails use a model-view-controller approach to application development. MVC clearly separates code according to its purpose. Three sub-frameworks play a significant part in this separation:
Active Record, Action View, and Action Controller

In the Active Record, Rails' object-relational mapper (ORM) connects programming objects to database tables so that users can access information they want from a database. Rails' ORM is purely based on convention over configuration. This means developers don't have to spend time designing and configuring code that specifies how a table relates to a class of objects.

Rails' Action View template renders the HTML response to the request, which it then sends back to the browser. Action View component generates views, which represent the visual appearance of an application's response to a request. Action View templates work with Ruby and thereby simplifies developers' work by not forcing them to learn a specialized template language.

The Action Controller performs controller development in Rails. In application development, controllers connect the program with its interface and handle communication between them. The Action Controller receives Web-based requests for information in a database, separates and decodes them to determine what the user wants, and then decides which piece of application code should handle the task [9].

Grails

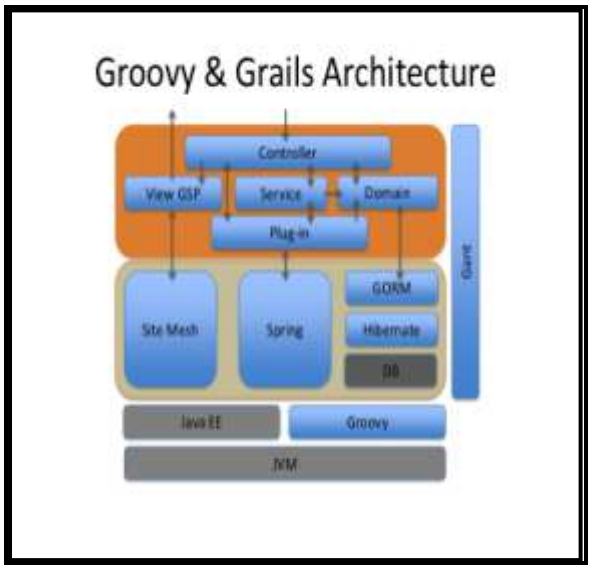


Figure 2 – Grails MVC Architecture, Source – [10].

Grails follows a very popular pattern in web applications development, called Model – View – Controller.

Model -
In Grails, data models are defined using "domain classes". Grails comes with a persistence manager based on Hibernate, called GORM (Grails Object Relational Mapping), which manages entities life cycles and provides default searching methods. GORM is based on Hibernate, a popular Object-Relational Mapping (ORM) tool which provides a framework to map object oriented entities to relational database. Every operation performed over a domain class will be translated by Hibernate into any necessary SQL query to reflect the performed operation in the database. All domain classes of our application are generated in the "domain" folder within our application folder structure [11]

View -
In Grails, views are developed using Groovy Server Pages (GSP), which is a simplified JSP version allowing to place expressions within HTML code and use JSTL like tags. When our application is running, we can decide which view should be processed and send to the client by means of the "render" method in the controllers or we can let grails to use a default view. All the views of our applications are generated in the "views" folder within our application folder structure [11].

Controller -
In Grails, controllers are the components responsible for handling incoming client requests, managing the execution of the business logic and updating the view so the user the can know the final status of the data model upon the execution of the action. The convention in Grails is that any class in the "controller" folder of our application structure and with the name ending in "Controller" will be considered a controller by our application. Grails will identify upon each HTTP request, which controller it should call based on the definition set in –

"conf/UrlMappings.groovy"
"Conf" is the first part of the name of our controller.
"UrlMappings"- "action" is the method to be executed.
Groovy- "id" is the identifier of a domain-class instance [11].

III. ANALYSIS

A. GORM & ActiveRecord

GORM is the Grails Object Relational Model interface. It is primarily backed by Hibernate 3 (A very popular, flexible & powerful open source Java ORM); although more recently GORM has been separated from Hibernate to allow for connectivity to alternative database mappers and NoSQL data stores. Due to the dynamic nature of Groovy with its static and dynamic typing, along with the convention of Grails, there is far less configuration involved in creating Grails domain classes [12].

Grails Example:
i) package spud.cms
class SpudSnippet {
  String name
  String content
  String format = 'html'
  Date dateCreated
  Date lastUpdated
  static mapping = {
  }
  static constraints = {
    name blank:false
    content nullable:true
  }
}

ii) def book = Book.findByTitle("Sherlock Holmes")
book
  .addToAuthors(name:"Sir Author Conan Doyle")
  .addToAuthors(name:"Dr. John Hamish Watson")
  .save()
The ActiveRecord sub framework establishes the connection between the domain objects and the database. It transforms the CRUD (Create/Retrieve/Update/Delete) functions that come from the Action Controller into SQL commands, sends the requests to the database, and returns the received results to the Action Controller. Active Record also validates whether a user is permitted to access or change a specific record. The sub

framework follows the Object/Relational Mapping pattern.

Rails Example:

```
class SpudSnippet < ActiveRecord::Base
  validates :name, :presence => true
  validates_uniqueness_of :name, :scope => :site_id
end
```

B. Convention over Configuration

Grails focuses heavily on conventions to remove the need for the developer to spend time on configuration. First of all, it has a well defined folder structure of were each functionality should be placed, such as controllers for views, test classes, translations and so on. This enables the developer to almost not have to do any configuration to get started, unless it is for customization purposes such as URL-mapping. Other than the structure, Grails relies on naming conventions across most part of the framework, such as specific suffix depending on the functionality of the file, which with the folder structure really enhances the purpose of the file. Rails have a similar attitude to conventions as compared with Grails, which is not surprising considering Grails got influenced heavily by Ruby on Rails. Rails also have a specific folder structure, which it uses in order to find the various application functionalities without using configuration. As well as the folder convention, Rails also uses naming conventions for specific files to clarify their purpose within the project [7].

C. Scaffolding

Grails also offers scaffolding, which lets the developer to auto generate views and controller actions for Create-Retrieve-Update-Delete (CRUD) for any given domain class. The way for an application to express a dependency on the scaffolding plug-in is by including the following in build.gradle.

```
dependencies
{
      // ...
      compile "org.grails.plugins:scaffolding"
      // ...

   }
```

Grails provide two types of scaffolding: dynamic and static.

Dynamic Scaffolding -
Dynamic scaffolding is achieved by enabling the scaffold property to “true” in the controller (provided the controller follows the same naming convention as the domain class). In cases where the domain names are different, it is necessary to specify the name of the domain to the scaffold property.

```
class BookController {
 static scaffold = true
}
```

With this configuration, if the application is now started, it will auto-implement all the CRUD-related actions within the controller and also generates the respective views.

Static Scaffolding -
With static scaffolding, Grails allows the developer to generate a controller and the views used to create the user interface from the command line.
This can be achieved by running the:
-> “grails generate-*<domain>” command.
-> grails generate-controller Book

Both uses the default templates, which come bundled with Grails, to generate the view and controllers for a particular domain. Static scaffolding is beneficial when working to build a prototype, test an idea or create an admin interface [13], [14].
The database scheme and the scaffold console command lets Rails create a basic skeleton of controllers and view templates that have create-retrieve-update-delete (CRUD) functionality. In most of the programming situations, Rails also offers generators that save a lot of time for recurring elements. Example: A login form.
The automatic creation of project scaffolding saves time and lets the developer immediately start work on the application’s core functionality.

Eg: - $ rails generate scaffold Post name:string title:string content:text

D. Validation

Grails creates validation when generating views and controller from a domain model. Grails gives a standard error message telling in which field the errors occurred. It is possible to customize these error messages and use localization to get error messages in the user’s native language. Grails also changes color of the fields with errors so the user finds it easier were a mistake is made. The user’s data are still in the fields so the users do not have to rewrite everything [7].

Below is a sample code:

Controller:

```
myDomainInstance.validate()
```

Domain:

```
static constraints = {
 myField(blank:false)
}
```

Input page:

```
<g:hasErrors bean="${myDomainInstance}">
```

```
 <div class="errors">
 <g:renderErrors bean="${myDomainInstance}"
as="list" />
 </div>
</g:hasErrors>

<div class="value ${hasErrors(bean:
myDomainInstance, field: 'myField', 'errors')}">
 <g:textField name="myField"

value="${projectInstance?.myField}" />
</div>
```

Code snippet - Sample code of how input validation is solved in Grails with MVC architecture.

Rails has a scaffold generating command and if one is reusing the code given by this command, one just needs to add one single line of code for adding validation of one or more fields. Then one gets validation of the fields, an error message and fields with error highlighted. Rails are very powerful when it comes to input validation.

Below is a sample code:

Controller:
No code needed.

Domain model:

```
 validates_presence_of :myField1, myField2
 validates_length_of :myField2, :within => 2..30
```

Input page:
For printing error message this line is used:

```
<%= error_messages_for 'myModel' %>
```

Code snippet - Sample code of how input validation is solved in Rails with MVC Architecture

E. Testing

Testing in Grails is relatively much easier, as it offers a structure for unit testing and integration testing right away. The proper unit test classes are made automatically when a domain model is created. Within the created class, a stub code is created, which leaves one tasks to the developer and that is filling the class with tests. The integration tests have to be created separately depending on what should be tested as they do not belong to a certain part of the application by default. With all tests, they can be run with a single command, which generates statistics about how the tests went. Rails as similar to Grails, offers a structure for adding tests to the application and it also generate certain files automatically. It has the ability to do unit tests on the models, functional tests on the controllers and integration tests between controllers; furthermore it offers additional way of testing other features too. All of these tests can easily be run via a command in the console or the IDE [7].

Features –

Feature	Grails	Rails
Programming Language	Groovy	Ruby
Approach	Domain Oriented	Database Oriented
Object Role Modeling	Hibernate	Active Record
Architecture	MVC	MVC
Support	Static/Dynamic Typing	Dynamic Typing
Servers	Tomcat, Jboss, Weblogic, IntelliJ Ultimate, Jetty & GlassFish.	FastCGI, Jboss Portal, Mongrel, JRuby & GlassFish.
Testing Frameworks	JUnit	Rspec
Infrastructure	Heavy Framework	Light Framework
Community	Active & Popular	Active & Popular

Similarities –

- Both are used for Rapid Application Development.
- Both are easy to configure & use productively.
- Both work on Convention over Configuration (CoC), Scaffolding & Don't Repeat Yourself (DRY).
- Both can be developed as WAR files.
- Both have built-in Web Servers.
- Both support multi-lingual content.
- Both are Full-Stack Frameworks.

IV. STATISTICS AND CURRENT TRENDS

One of the source shows -

Criteria	Grails	Rails
Developer Productivity	1.00	1.00
Developer Perception	1.00	1.00
Learning Curve	1.00	1.00
Project Health	1.00	1.00
Developer Availability	0.50	1.00
Job Trends	0.50	1.00
Templating	1.00	1.00
Components	0.50	0.50
Ajax	0.50	0.50
Plugins or Add-Ons	1.00	1.00
Scalability	0.50	0.50
Testing	1.00	1.00
i18n and l10n	1.00	0.50
Validation	1.00	1.00
Multi-language Support (Groovy / Scala)	1.00	0.00
Quality of Documentation/Tutorials	1.00	1.00
Books Published	1.00	1.00
REST Support (client and server)	1.00	1.00
Mobile / iPhone Support	1.00	1.00
Degree of Risk	1.00	1.00
Totals	17.5	17

Figure 3, Source: Matt Raible's – "Grails & Rails Web Framework Comparison" [15].

Mainly there are:
0 – If it does not have these characteristics.
0.5 – If it has at some level.
1 – If it is fully satisfied.

According to another source from DevRates, they focus on reviews by developers using libraries on their daily work.
DevRates contains projects reviews of most popular tagged categories and programming languages with an active user community who gives latest reviews.

Figure 4, Source - [16].

Based on the top 10 technologies in web framework, Grails stands the highest with **9.3** rating in comparison with Rails which is at **8.0**

Current Job Opportunities & Trends

Due to the ease of development, scalability and learning curve, enterprises are going with Grails & Rails. However, there is a slight demand and job opportunities in Grails with comparison to Rails. According to one of the surveys conducted by Indeed.com, from 2012 to 2016, the job trends are shown below.

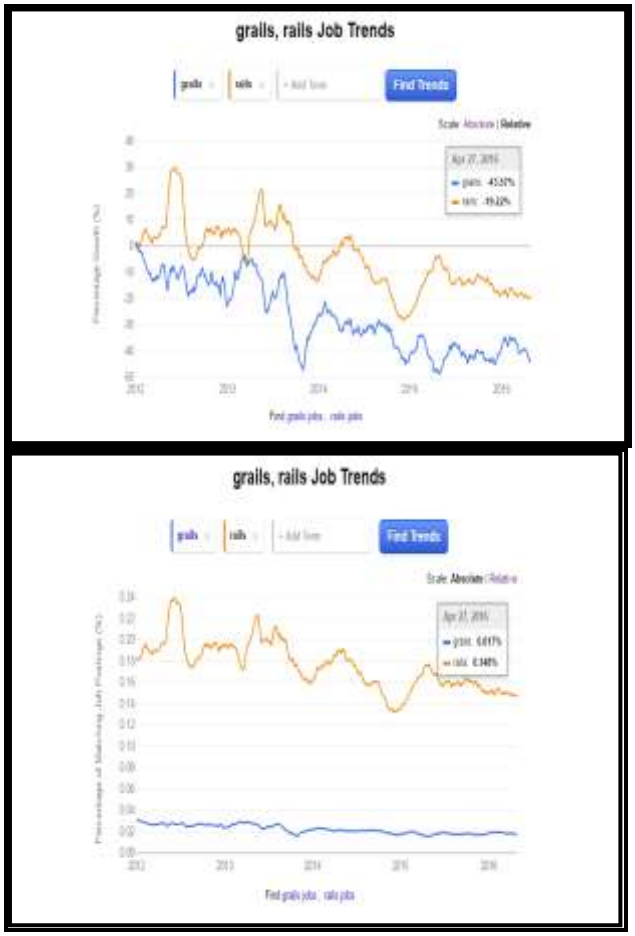


Figure 5, Source - [17].

According to PayScale, based on 31st March 2016, they have given average salary information for Grails & Rails as per the stats given below.

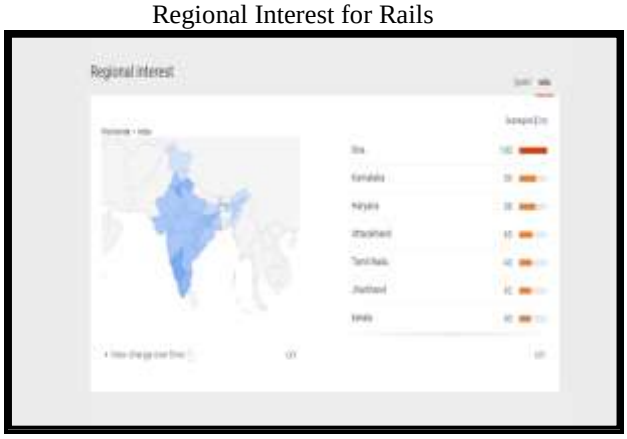
Figure 6, Source - [18].

Figure 7, Source - [19].

Google Trends of Grails & Rails – As of 2016

Regional Interest for Grails

Regional Interest for Rails



Related Searches for Grails in Google

Related Searches for Rails in Google

Figure 8, Sources - [20].

According to all the statistics & reports we have accumulated & on the basis of data gathered in our research, we have created an overview report card.

Weightage – A (2 point), B (1 point)

	Grails	Rails
Validation & Testing	A	A
Code Conventions	A	A
Developers Tools	A	B
Learning Curve	B	A
Scalability & Growth	A	B
Job Trends	B	B
Total Score	10	9

V. CONCLUSION

To choose a framework based on a suitable application is very important. If the framework is not suitable for the application, the framework might be obstruction instead of a helping hand. It's crucial to find out if the framework is built for the size of application. Choosing between the two i.e. Grails & Rails depends on the requirements, the intensity of the development project and the features you are looking in a particular framework. As a dynamic web framework embracing DRY principles, Grails was designed to be a companion to, rather than a replacement for, Java. Grails can dramatically reduce the complexity of building web applications, helping Java developers create applications with greater speed, agility and flexibility in terms of integrating with existing systems. During the implementation apart from the well documented part of Grails, a really neat feature with Grails is that it works well with existing Java libraries, because of Groovy. This means that when developing one can easily use a Java library without any complications. More over Grails is a very domain oriented framework and offers easy ways of creating a database to work with and relations between tables in that database. Grails is not alone in the realm of dynamic Java web frameworks. Much inspired by peer framework i.e. Ruby on Rails. Some of these frameworks share similar principles of philosophy.

Ruby on Rails has become quite popular over the years. It's a very easy platform to work with & is very mature in the industry. Most people will become very productive with it very quickly. It abstains from heavy tools and focuses instead on "individuals and their interactions." The process of programming is much faster than with other frameworks and languages, partly because of the object-oriented nature of Ruby and the vast collection of open source code available within the Rails community. The principles of CoC, DRY & scaffolding provide a first operable version early in the development cycle & saves lot of time and effort while coding. It is purely database-oriented framework and has developed a strong focus on testing, and has good testing frameworks. Like Rails, Grails is also gaining in popularity, illustrating a growing trend in rapid application development. Like to conclude by saying that both have been successfully developed over the years, adopted by various user communities and will continue to progress in near future based on the current trends in the market.

VI. REFERENCES

[1] "Ruby Programming Language Wikipedia,", Available: https://en.wikipedia.org/wiki/Ruby_(programming_language)

[2] "IEEE Journals and Magazine", Title – Ruby on Rails, Author - Michael Bächle and Paul Kirchberg

[3] "Groovy Programming Language,", Available: https://jcp.org/en/jsr/results?id=2490, https://jcp.org/en/jsr/detail?id=241

[4] "Groovy - An agile dynamic language for the Java Platform," , Available: http://www.groovy-lang.org/

[5] "Grails Framework Wikipedia,", Available: https://en.wikipedia.org/wiki/Grails_(framework)

[6] "Grails – Introduction , Available:
- https://grails.org/wiki/Introduction

[7] "Evaluation of Web Application Frameworks", Available: http://publications.lib.chalmers.se/

[8] Image Source, Available: https://viblo.asia/leminhtuan2015/posts/KE7bGoAOR5e2

[9] "IEEE Journals and Magazine", Title – Will Software Developer Ride Ruby on Rails to Success?, Author – David Geer, http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1597080

[10] "Rapid Development with Neo4J under Grails,", Available: http://www.slideshare.net/stasimus/neo4-grails

[11] "Grails Model View Controller Pattern," , Available: http://www.wideskills.com/grails/grails-model-view-controller-pattern

[12] "Moving from Grails to Rails," , Available: http://www.redwindsw.com/blog/2014-01-15-moving-from-rails-to-grails-differences-and-similarities

[13] "Grails Scaffolding," Available: http://docs.grails.org/latest/guide/scaffolding.html

[14] "Grails Web Framework," Cited, Available: http://www.sapient.com/content/dam/sapient/sapientglobalmarkets/pdf/thought-leadership/MarProg_Grails_WP_Web.pdf

[15] Image Source: Matt Raible's – "Grails & Rails Web Framework Comparison"

[16] Image Source, Available: http://devrates.com/stats/top?tagName=web+framework

[17] Image Source, Available: http://www.indeed.com/jobtrends/q-grails-q-rails.html

[18] Image Source, Available: http://www.payscale.com/research/IN/Skill=Ruby_on_Rails/Salary#by_Years_Experience

[19] Image Source, Available: http://www.payscale.com/research/IN/Employer=Grail_Research/Salary#by_Years_Experience

[20] Image Source, https://www.google.co.in/trends/explore/