CHALMERS

Evaluation of web application frameworks

Evaluation of web application frameworks with regards to rapid development.

Master of Science Thesis in Software Engineering and Technology

MARTIN BJÖREMO

PREDRAG TRNINIĆ

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
Göteborg, Sweden, June 2010

Evaluation of web application frameworks
Evaluation of web applications frameworks with regards to rapid development

MARTIN BJÖREMO
PREDRAG TRNINIĆ

Examiner: SVEN-ARNE ANDREASSON

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

[Cover: Logos of frameworks evaluated in this report, from upper left to right; Ruby on Rails, Wicket, CakePHP, Stripes, Grails, Spring Roo.]

Department of Computer Science and Engineering
Göteborg, Sweden July 2010

Abstract

This report focuses on evaluating several web application frameworks for use in rapid development. 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. Creating an application from scratch would most likely be too time consuming and not very rapid. Thus the report will look closer at some of the application frameworks (CakePHP, Grails, Ruby on Rails, Stripes, Spring Roo and Wicket) to see what they have to offer and how they do it. The frameworks are evaluated based on six criteria (documentation and learning, convention over configuration, integrated development environment, internationalization (localization), user data input validation and testing) and one promising is chosen to be used to implement a web application. The conclusions of this evaluation are that there is no superior framework and one should not learn a new programming language just for using a recommended web framework. And also, one should choose framework wisely based on the size of one's application and the scope of the application.

Sammanfattning

Denna rapport fokuserar på utvärdering av ett antal webbramverk för användning i snabb utveckling. Eftersom det finns ett behov av att kunna distribuera nya applikationer på kort tid finns det också ett behov av ramverk som underlättar detta krav. Skapa en webbapplikation från början skulle sannolikt bli alltför tidskrävande och inte särskilt snabb. Således kommer rapporten att se närmare på några av webbramverken som finns (CakePHP, Grails, Ruby on Rails, till Stripes, Spring Roo och Wicket) och se vad de har att erbjuda och hur de löser vissa tidskrävande uppgifter det. Ramverken bedöms utifrån sex kriterier (dokumentation och lärande, konvent över konfigurationen, integrerad utvecklingsmiljö, internationalisering (lokalisering), kontroll och validering av indata från användare) och ett lovande ramverk väljs ut för att användas för att skapa en webbapplikation. Slutsatserna i denna utvärdering är att det inte finns ett överlägset ramverk och man bör inte lära sig ett nytt programmeringsspråk bara för att använda en rekommenderat webbramverk. Dessutom bör man välja webbramverk noga baserat på storleken av ens webbapplikation och omfattningen av webbapplikationen.

Acknowledgements

This master's thesis report has been written as the final part of the Master's program Software Engineering and Technology at Chalmers University of Technology, Sweden 2010. The subject was chosen in cooperation with Logica in Gothenburg, where the thesis also has been performed.

The writers of this report would like to thank their supervisor at Logica, Jesper Forslund, and at Chalmers, Sven-Arne Andreasson, for their support during this project.

CHALMERS

Martin Björemo
Predrag Trninić

Gothenburg, Sweden
June 2010

Table of Contents

Glossary

API – Application Programming Interface
ASP – Active Server Pages
CGI – Common Gateway Interface
CRUD – Create Read Update Delete
I18n - Internationalization
IDE – Integrated Development Environment
Gem – Ruby package manager
GUI – Graphical User Interface
PHP – PHP: Hypertext Preprocessor
MVC – Model-View-Control
WYSIWYG – What You See Is What You Get

1. Introduction

Currently there are a lot of different web frameworks available, ranging over several different programming languages, although they are most commonly built on PHP and Java. They all advertise themselves as the best framework with a lot of possibilities, but when it comes down to practice this might not be the case. This is why there is a need to evaluate these frameworks to find out which ones actually support the criteria one would want from an application and development environment. As the size of the frameworks can vary a lot the focus of the evaluation are medium to large sized frameworks, which does not include fully fledged enterprise frameworks.

1.2. About Logica

Logica is a business and technology service company, employing 39 000 people across 36 countries. With 5 500 employees in Sweden and about 500 Goteborg Logica is one of the major IT-companies in Sweden. Logica deliver business consulting, systems integration and outsourcing across all industries and business functions. Core values for Logica is open, innovative and committed to create effective, sustainable business ecosystems.

2. Background

The World Wide Web (often known as the web) was created in 1990 (1) and in the beginning the web was very static. The user could not interact much with the content, and to update a piece of text (or something else as well) in a published material on the web, the author had to edit the page locally and upload it to the server. To get rid of this manual work the Common Gateway Interface (CGI) standard was created for interfacing external applications with web servers. (2) CGI created a new process for each request to the server resulting in heavy load on server side when dealing with a great amount of requests, thereby the demand for something more efficient grew. In 1995-1996 the growth of web pages increased dramatically (3) and at the same time e-commerce got about. ColdFusion, PHP and ASP (Active Server Pages) was created during the same period. Today ASP is replaced with ASP.NET, but ColdFusion and PHP are still developed and used. The term "web application" was first introduced in 1999 in the Servlet Specification version 2.2 for the Java language. Today most web sites are interactive in some way which has made them more complex to develop and the term web application has become generic.

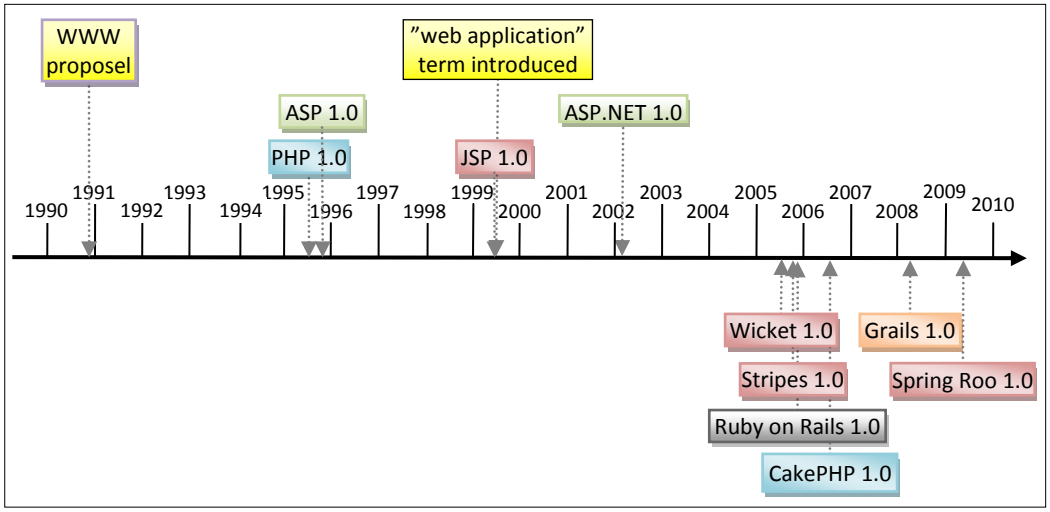

Figure 1Time line of web history, showing first releases of evaluated web frameworks. Notice, web frameworks is a trend of later part of first decennium of third millennium.

2.1. What is a web framework?

A framework in context of software development is a set of prewritten code or libraries which provide functionality common to a whole class of applications. The framework can be seen as a base or a skeleton to build upon. In relation to libraries a framework gives a broader span of functionalities. Libraries are often more focused on solving a narrow scope. Typically a framework provides a predefined structure for the application. The purpose of using a framework is basically to speed up development by not having to rewrite features and structure that are commonly used in most web applications. Instead of inventing the wheel over and over again the developer has many wheels (functions/features etc) that are already tested and working properly. Most (web) frameworks today are free to use (under license from Creative Commons, Apache, etc.) and are open source and anyone can join to contribute in some way. Either, be part of development, propose improvements, code review/inspection or just gain

experience by using the framework and thus being able to help others. Most of the frameworks today have quite many contributors and users (developers) inspecting the source code for security risks and evil pieces of code. (4) (5) (6)

2.2. Framework vs. libraries

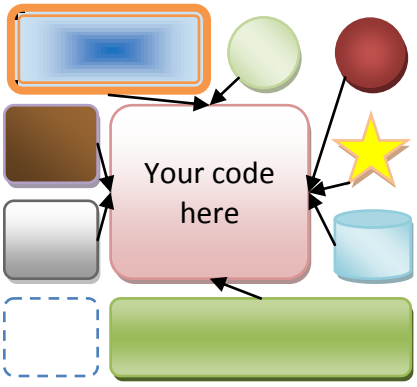

Figure 2 Showing how libraries interact with one's application. Everything is connected to one's code.

All glory to code reuse, but how is the best way of doing it? The debate has been going on for as long as the existence of different approaches. One way to go is using libraries, which will give one what one need when one need just by adding suitable library to the application. The advantage with this approach is that one will only get the functionality one needs when one needs it. They can be applied anywhere in the code and is not hindered by itself due to a design pattern. Any number of libraries can be added to an application this way. The lack of design pattern from libraries can be either an advantage or a disadvantage, depending on who one asks and what the intended usage is. A library is a collection of classes which provide reusable functionalities while a framework is about reusing behaviors by how abstract classes and components interact with each other. With a library one call/inherit from one's application, but a framework provides services for one's code. Figuratively speaking; one's application calls the library and a framework calls one's application.

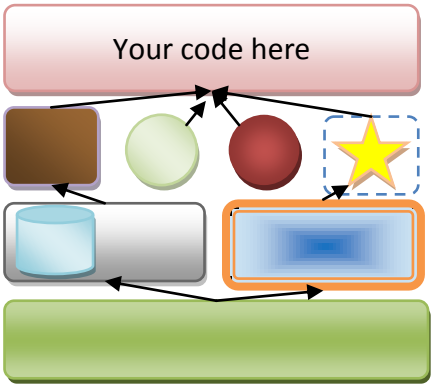

Figure 3 Showing how a framework with components/libraries is connected to one's code. One's code is build upon the web framework.

Most frameworks usually include functionalities that can be seen as libraries, like gems or plug-ins, and can be extended with more of those and libraries. If one already has an application and decides to add a framework or a library to get access to more features, the library can often easily be added by a simple import into the application while adding a framework afterwards is often more complicated. If one want to delete a library it is as easily done as adding it, and one will also lose all functionalities that came with the library. A framework is often easy to uninstall, and some frameworks offers the possibility to keep or convert all necessary classes needed to still be able to run the application. A library is much more rapid to learn; more or less it is just about reading the API and use it. The fact that libraries do not come with a predefined design pattern that the developer is forced to follow is why this report is focusing on frameworks, and not libraries. Following a well known design pattern makes it easier for new members of the developing team to get involved and contribute without making harm of the design of the application. All developers know how the application should be built and no need for tedious discussions and misunderstandings and in the end this more often leads to a longer

life time of an application. A good library can also be added to chosen framework later, and one can get the best of both worlds. (7) (8) (9)

2.3. Degrees of activeness web frameworks

Results from this evaluation showed that the frameworks have different degrees of activeness under and after installation. See more under Results chapter.

- "Passive" framework - are frameworks that are just a bunch of files to start working from. Unpack and one is ready to create a project from it. Some frameworks support to be located apart from the project files, so the framework files can be reused by several other projects. This setup might require some extra configuration.
- "Semi-active" framework - are frameworks that can generate code from one's existing code by a command from the developer. Some of these frameworks can also create whole new projects from the developers command, with or without options.
- "Active" framework - are frameworks which got what semi-active frameworks got and also are working in the background by writing/creating/generating code automatically without any command from the developer.

(4) (10)

2.4. Advantages of using a framework

This list is trying to pinpoint the major benefits of using a framework:

- Reuse of working code that has already been built, tested, and used by other developers increases reliability and reduces developing time. Time is major cost in software development and by decreasing time one can cut costs.
- Get (free) help from others. By letting others (the framework) take care of common issues, like security, internalization, localization, etc., one will get (free) "outsourcing" from skilled developers. If a developer starts from scratch with a project the developer will have to think of everything. And, if the developer knows how to solve everything; why do it once more? There is just risk of bugs and security leaks.
- A framework can help the developer to use a specific architecture or design pattern, often Model-View-Control (MVC), and general best practices. This will make it easier for incoming developer to understand the code and quicker start working with it.
- Framework can support "high level" of programming, by code modularity. Basic tasks, like login and database handling, can be in the framework and separated in another layers is business logic.
- By upgrading one's framework it might give extra features without extra implementation. Take an example if an e-commerce is using a framework and the team behind the framework releases a new version it might give new ways of payment methods.
- It helps new inexperienced developers to easier adopt design patterns and understand the thinking behind following a design pattern. A framework can help beginners to have a steeper learning-curve.

(4) (10)

2.5. Disadvantage of using a framework

Not everything with a framework is on the positive side. Here are some of the most common negative aspects:

- To be able to use the framework at its best, it often requires significant education and experience, in other words long learning curve. But on the other hand if one has great experience from one framework the learning curve for the second one is much steeper. (Assumed a framework about the same size.)
- Performance might be suffering from common code that is built to handle as much as possible, and is not optimized for a specific task.
- If a bug or a security risk in the framework is found it will be in all applications using the framework. If an intruder finds out about this weakness the intruder might be able to use this for evil purposes on all applications using this framework. If the weakness is discovered in someone else's application then one's application might be at risk even though none have ever tried to hack it before.
- Some framework are very stiff and do not give the developer enough flexibility needed for some applications. Good frameworks provide utility and structure while still leaving enough flexibility to not require the developer to do work around. It is not necessarily the framework's fault; the framework might be intended for some other use than one have in mind. Choosing the right framework is crucial if time will be saved or lost.
- Building from scratch often gives a feeling of more productive which can make the developer more peaceful and less feeling of being stuck and thereby more creative and less bored. But remember, this is just a feeling, not fact.
- If using a design pattern, like MVC, it might compartmentalize towards a fault, since it may use six or seven files for showing a simple page.

(11) (4)

2.6. Who should use a framework?

Different frameworks might aim for different goals, but in general they all aim for supporting developers in their work of creating an application with as little effort as possible with as little hinder as possible.

A framework can be used for many purposes, not only serve common functionalities, it can also help learning a design pattern. Therefore new developers, that knows some programming but do not know so much about design patterns and best practice, can benefit from a framework by gaining understanding and programming skills. A beginner will not feel that the framework is getting in the way as much as an experienced developer. An experienced developer already knows the basics and how everything functions and the various parts are related to each other and might therefore sees hinders instead of possibilities. Experienced developers are also more likely work on more advanced projects where a framework might not be suitable, due to the broad use of the application and greater demand for high performance.

The advantages of using a framework should be greater than the disadvantages, otherwise it would, obviously, not be any point of using a framework.

2.7. Rapid development

When rapid development is mention in this report, it does not refer to Rapid Application Development (RAD) which is a type of software development methodology. RAD uses minimal planning in favor of rapid prototyping, which is might be good for rapid development, but the development methodology is not of interest in this report.

In this report rapid development refer to creating an application from an idea to a working application on the web. The development process preferable would be agile, since the first release of this kind of application will not necessarily contain all features.

3. Problem

Today one might come up with a brilliant idea of business concept for the web, and the only problem is the time it takes to create the web application. If one is not too sure it will be a huge success one might not want to spend too much time developing a test application. Building a test application is a very good way of finding out if the idea is on the right track. When one has a ready application the effort, in time and money, needed to launch the application is very low. The significant cost is to develop the application.

To speed up developing one can do just what is needed to make it work, without thinking of further development. This often leads to spaghetti code that will virtually become impossible to maintain. It is here a framework can come in handy. It has predefined structure of how things should be done, the relations between classes are there waiting to be used.

Today the number of web application frameworks is growing faster than ever, and all of them sell themselves as 'The one framework'.

One should not spend more time on building the test application than one feel it was worth a try.

3.1. Purpose

The main argument for this report is to find one, or more, frameworks or combination of frameworks that best supports rapid development of web applications.

3.2. Scope

This evaluation will only consider free frameworks downloadable from Internet. The frameworks should be free to download, use, deploy and distribute.
The evaluation will not include .NET framework, since the .NET framework is not free to use.

4. Method

To be able to perform the evaluation of the frameworks in an effective way a process was set up and used, which was split into three phases with each phase focusing on its own part. The first being the investigation phase, followed by the evaluation phase and concluded with the implementation phase. Each phase plays a specific role in the full process, discussed more in the following paragraphs.

4.1. Investigation

The investigation phase played a quite important role in the evaluation process as it was here the frameworks that would be evaluated were picked out. That is why information gathering was the main purpose of this phase to be able to select candidates for the following phase. The information was based on the evaluation criteria, mentioned in previous chapters, as a baseline to be able to see how a framework managed to fulfill them and if it was worth to proceed with that framework. The sources for the information were mostly the frameworks website, including the API, tutorials and such, but included other sources as books and other informative websites.

With the information gathered it was noticeable which frameworks did not measure up to the standard that was set and thus failed to qualify to the evaluation phase. At the same time it also gave a hint about how well a framework was documented. Thus it was possible to reach a consensus on which frameworks would qualify for the evaluation phase. When the choice was made, which programming languages a framework was using was also given consideration so that the final frameworks would be in equilibrium, in respect to the programming languages used.

4.2. Evaluation

The second part of the process is the evaluation phase, where most of the work to explore a framework is done. It is here where each framework that was picked would be used to test how well it aids a developer with the evaluation criteria that were set up. This is done by implementing features that test the functional criteria and while continuously evaluating the non-functional ones. It should be noted that the implementation mentioned did not cover implementing a complete application for each framework, but instead focused on implementing the specific features somewhat independently. All of the implementation was also done in a single IDE, which was the one that was found to be best suited for the framework.

One can easily get an overview how well a framework relieves the developer by following convention, but to get a more entire picture there is a need to take a closer look. Does a framework actually move the configuration to convention or does it actually just move it to a more abstract level to make it easier. Even though a higher level of abstraction, see figure [A], would make it easier for a developer by removing noise it is still configuration and thus valued below conventions in the evaluation.

There might not be any difficulties to measuring documentations, although when measuring learning problems arise. That is why learning is based on a few different factors to ensure that it remains unbiased. These factors are the tutorials and guides that are available for

getting started, how intuitive it is with regards to the language it is using and if the same philosophy is used throughout the framework.

When it comes to the frameworks support in an IDE the focus is on how well the framework integrates with the IDE, e.g. if the frameworks offers a console is it possible to access it through the IDE, as well as the classical functionality such as code completion.

This covered the non-functional criteria, as for the functional ones they are basically implemented with each framework to test and evaluate how well it works.

With the first stage of the evaluation phase done, this includes the evaluation of all of the points of interest, the three most promising frameworks were selected to do additional evaluation on. This was done by taking a deeper dive into these specific frameworks by implementing an application, in this case different for every framework. The reason for the variety at this point is because to be able to test all common features of the framework books were followed and have it remain as far as possible the books were chosen from the same publisher [1] as the books had the same approach to the different frameworks.

4.3. Implementation

The last phase was the implementation phase, which focused on a single framework that was chosen with regards to the evaluation in the previous phase. This framework was to be used to implement a web application for the use of the Pomodoro Technique[2] and do so by connecting to a service provider, which puts extra requirements on the application. These requirements were e.g. not having an own internal database to work with, but instead have to rely on an outside source, also having an reliable and secure way of sending information back and forth between the application and the service provider.

[1] The Pragmatic Bookshelf (http://pragprog.com)

[2] The Pomodoro Technique (http://www.pomodorotechnique.com)

5. Evaluation Criteria

When looking at a framework there usually tends to be a long list of features that the framework offers to the developer to facilitate in the development process. Although there is a lack of a unified list of commonly used features, which indicates if a framework has the ability to aid the developer with that task. This can also lead to the inability to make a proper choice if a framework actually fulfills ones needs or not, thus there is a need to level the playing field for the frameworks by introducing a list of features that frameworks should offer. The features, which should be contained in the list, need to cover commonly used functionality in web applications and more specifically functionality that facilitate rapid development and do so by lowering the effort of development. With this list in place the feature within it are used as evaluation criteria of a framework being evaluated.

The first three criteria are non-functional ones and the last three are functional criteria. What criteria got chosen and why will be discussed more in the following sections, as they will go through all of them one by one.

5.1. Documentation and Learning

Having to learn a programming language or framework is something all developers have done at some point, but how trouble-free that journey is depends on several factors. The availability of documentation such as tutorials, books, online resources etc. and the let's not forget the background of the developer. To be able to give a fair and unbiased evaluation the last factor will be considered so that it does not favor a framework because of this, during the evaluation process, more focus will be given to the official documentation and other resources, online as well as offline. The reason for this criterion is to get a reference to see if the benefits of the framework outweigh the learning curve by looking at the documentation available. As if a framework lacks the resources needed to learn it effectively the benefits it brings might not be worth the time, effort and money to actually learn it so another may be better fitted.

5.2. Convention over Configuration

By having a framework that is built for convention over configuration it will affect a considerable part of the development process and do so in a positive manner. As one might notice, as the name suggest, by removing the focus from working with configuration files and instead follow a few conventions it will enable the developer to spend more time on actually developing the application. (12) It might also force a structure on the application although this fact is an double-edged sword as from one aspect it will limit the developer, but in case new developers join the development they will have an easier time understanding how the application works due to a well defined structure. (13) As the evaluation is considering frameworks for rapid development being able to free up time and confusion by less configuration and a common structure the convention over configuration criteria fits quite well into the list of features a framework should have.

5.3. Integrated Development Environment

The definition of an Integrated Development Environment (IDE) is commonly a source code editor, compiler and debugger. (14) This is a perfectly fine definition although a bit outdated when compared to IDEs that are available today, such as NetBeans. It offers several features beyond the basic definition, such as repository support, code completion, refactoring and much more (15). When considering the support for a framework in an IDE, these larger IDEs that offer more features will be considered. The reason for this is that by gathering several tasks at the same place it would increase the clearness of the work area. Preferably a framework is supported in several IDEs so it gives the developer several options when choosing the environment for development.

5.4. Internationalization (Localization)

Internationalization (I18n) refers to the ability of an application to switch between languages in an easy way. Together with localization it allows for an application to find out where a user is located and adapt the application to that region. This is a very strong combination as the applications can be tailored to fit a larger audience than before, which is getting more and more important with more globalization. With a feature like this it will allow the application to reach a whole new level of usefulness and simplify it for the developer to make the application multilingual.

5.5. Validation

For applications that rely on human interaction handling input will play a large role in the application. It will also be something that has to be done repeatedly, which is why support for checking the validity of the input and give the user feedback in case of error is a valuable feature. This can be especially useful when the application uses it own database for storage of data as it aids in keeping the integrity of the data in the database.

5.6. Testing

Testing might be the bane of software development, but nevertheless it plays a really important role in the development process. To be able to ensure that an application is stable testing must be used and that us why it is so important that a framework allows an easy way of doing it. With the support of the framework it will facilitate the testing process and most likely free up additional time for the developer, especially if the framework supports many different types of testing.

5.7. Additional Criteria

With a set of criteria established there is a strong foundation for the evaluation process, although these criteria do not cover all areas that might be of interest when evaluating a framework. Thus other features will be taken into consideration too during the evaluation process, but with a lower importance. These features can include, but not limited to, the possibility to add extensions to the framework or how well it integrates with JavaScript and existing JavaScript libraries. It should also be noted that speed and scaling of a framework is not

part of the evaluation in any way due to the fact that these factors are dependent on how well the code is written for each framework.

6. The Frameworks

As there are quite a few frameworks available all do not fit the scope of this report for various reasons e.g. missing features, still in development or focus on enterprise solutions.

6.1. CakePHP

CakePHP, as can be derived from the name, is built with PHP, with PHP being the most widely used general-purpose scripting language. (16) (17) It comes with a long range of features to facilitate the development process, such as MVC architecture, localization, data sanitization and more. (18) (19)

6.2. Grails

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. (20) Groovy uses the strength of Java and new features inspired by other languages such as Ruby, Python and Perl (20), this gives Grails a strong base to stand on. 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 (21). 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. (21) 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. (22)

6.3. Ruby on Rails

Ruby on Rails, as the name suggests, is built using Ruby, which blends features from several different languages (Perl, Smalltalk, Lisp etc.). (23) The language itself has grown into one of the more widely used programming languages (17) with a large active community. (23) Ruby on Rails focus is on making it easier for the developer to develop, deploy and maintain web applications this has made it grow in popularity and be used when implementing a wide range of Web 2.0 applications. (24) It utilizes the widely used MVC pattern (Model-View-Controller) to enable developers to work in a way they are familiar to and takes it further with the help of convention over configuration (CoC). (24) 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. (25) (26)

6.4. Stripes

As Java web development can be too much work to get anywhere, Stripes focuses on making developing web applications in Java easier and with a lot less configuration [STA1]. It also adds additional features such as easy to use validation and type conversion systems, localization systems and more. (27)

6.5. Spring Roo

Spring Roo is a development tool aimed at Java developers for building full Java web applications fast and easy. (28) It makes use of standard Java APIs, offers a powerful shell to

work with and is not in use during runtime. (28) This means that Roo does its work in the background during development, in that way it also does not interrupt the developer's focus of attention. (29) Another noteworthy feature is that one can remove Roo from a project whenever there is no need or desire to use it anymore, meaning there is no lock in to the framework. (28)

6.6. Wicket

Wicket is a component based framework and separates Java code from the HTML code, making it possible for developers to work on each part independently from each other. (30) As the developer of Wicket sees it there are several flaws with Java web frameworks, from not being easy to use to lacking support for managing server-side state. (31) Thus the development of Wicket has set up goals which it follows and there are easy to use, reusable, non-intrusive, safe, efficient/scalable and complete. (31)

6.7. Other Frameworks

As you might be aware of the list of available frameworks is a lot longer than the frameworks that are part of this evaluation. Since there are far over 30 frameworks that utilize Java (32) and frameworks are not limited to only Java, as seen above there is a mix of several languages.

7. Results

The following section covers the finding of the evaluation made, by looking at how every framework solves each individual criteria that was set up. Beginning with the non-functional ones and finishing with the functional ones.

7.1. Documentation and Learning

When it comes to documentation there is no real standard on how it is done so one has to expect variety on the quality and structure of the documentation offered.

7.1.1. CakePHP

CakePHP really has a well developed documentation and a way of thinking to facilitate the learning process for the developer. It offers an API, a comprehensive manual, a bakery where developers can share code and for getting help there is a Google group and a page which's sole focus is on asking question and getting answers. With all that documentation CakePHP really simplifies it for the developer by having it all in one easily accessible location.

7.1.2. Grails

Grails has quite a lot of documentations to offer the developer to facilitate the learning process. It has an API and tag library, which are well documented and really helpful when developing with Grails. Apart from that it offers several different tutorials, guides, books and more to aid the developer when learning how to use and work with the framework.

Even with the heavy documentation one can find "features" that are not that well documented so one instead needs to utilize other means of documentation to figure out how these features work. The same goes for plug-ins as not all of them are optimally documented.

7.1.3. Ruby on Rails

Ruby on Rails is an older framework, thus it has also been able build a large foundation of documentation, from APIs to guides and books. Along with the large number of guides and a few books Ruby on Rails offers a large community, which includes "How To's" by users, wiki, mailing list and an IRC channel for asking questions. So it gives the developer a great starting point for getting started with the framework.

7.1.4. Spring Roo

Spring Roo documentation is mostly concentrated to a single long documentation page cover the vital areas of Roo, such as getting started, basic functionality and Roo commands. It is accompanied with mostly blog posts discussing various topics and presentations of the framework. This is partly because Roo is relatively new and because Roo uses a lot of other frameworks so the documentation is simply moved to those frameworks and then ones gets overwhelmed with too much documentation. It is especially keen on using other Spring frameworks, which in turn means that learning Spring Roo actually includes learning these other frameworks too. When it comes to the book front there has not been any books released, although there is one that will be released soon putting the number at one.

7.1.5. Stripes

Stripes is probably the framework with the least organized documentation of the final frameworks that were tested. The documentation consist of three different parts, “How To’s”, references (Tag library, JavaDoc and more) and user added documentations, which basically is “How To’s” to certain problems. Unfortunately there is not much more documentations on Stripes except for a book, which is good for getting started with the framework. So in comparison to the other frameworks documentation it is a bit lackluster.

7.1.6. Wicket

Wicket perhaps does not offer the most documentation, but the one that it offers is very useful. With the API and the examples, of how to do many common tasks, one can easily get started with Wicket development. Although it offers more than that with blogs and a reference library that explains how things should be done with Wicket, which is very useful. It is also possible to find other resources on the internet and several books about the framework. With that documentation one can easily get started with Wicket.

7.2. Convention over Configuration

Conventions exist in all the frameworks that were evaluated, although not as heavily used throughout all of them. It was found that the most commonly used convention is naming conventions for files, such as files with certain functionality are named in a specific way or files that work together are named similarly. The frameworks that did offered folder convention also resulted in giving an out-of-the-box structure for the application, where as the ones which mostly focused on naming conventions left those decisions to the developer by moving it to a configuration file. In the end it was the frameworks did not use conventions excessively, instead provided a balance between different convention principles, that had enough conventions to remove configuration and not too many do make it too difficult to learn. This includes Grails and Ruby on Rails with easy to remember conventions and no configuration.

7.2.1. CakePHP

CakePHP is close to being the most convention heavy framework in the line-up, with conventions covering everything from folder structure and file names to model and database mapping. With all the conventions that CakePHP utilizes there is no need for any extra configurations leaving the developer free to focus on various other tasks. Although since the framework uses quite many and specific naming conventions it can make it a bit tricky to learn all of them at ones, but when learned it will enable a smooth development process.

7.2.2. Grails

Grails focuses quite heavily on conventions to remove the need for the developer to spend time on configuration. It first 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. Even the configuration that was available, e.g. database configuration, it was done via Groovy files by assigning values to convention variables

instead of working with an XML file, which would introduces more noise. 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.

7.2.3. Ruby on Rails

Ruby on Rails has a similar attitude to conventions as Grails, which is not surprising considering Grails got influenced heavily by Ruby on Rails. This entails that Ruby on Rails has a specific folder structure, which it uses to be able to find the various application functionality without using configuration. As well as the folder convention Ruby on Rails also uses naming conventions for specific files to clarify their purpose within the project.

7.2.4. Spring Roo

Roo's approach is a bit different from Grails, as it does offer a structure just not as extensive and it shifts more towards using annotations. With the use of annotations and classical naming conventions Roo manages to remove most of the configuration, although with some help from the Roo shell which is used when making a project. The shell takes you through a series of steps that can be considered to be configuration, but these steps are not mandatory. In that way Roo lets the developer decide how much configuration one wants to do. On the downside if one wants to skip using the shell it will basically be a Spring 3 application with its entire configuration.

7.2.5. Stripes

Stripes is quite a lightweight framework which is reflected in its conventions as it does not offer the wide variety the previous two frameworks offer. This in turn means that Stripes main convention types are suffix naming conventions and annotations to get things done, although it still some configuration that is done in a XML file. The reason of the XML file is to give the framework a guideline to where to look for certain things and then the naming conventions and annotations do the rest.

7.2.6. Wicket

Wicket lands somewhere in between convention and configuration, as it has both available to it but for the convention it is mainly the naming conventions of pages and the Java source files that control them. Then it is mostly XML configuration, but that is not to say that there is a need for a lot of XML configuration to get everything running. The configuration is still kept low even with the minimalistic amount of conventions used within the framework.

7.3. Integrated Development Environment

When compared the selected frameworks within this criterion only IDEs recommended by the framework and the two most well known IDEs, NetBeans and Eclipse, was considered. What was found was that all frameworks were able to perform the basics tasks, edit code, compile and debug, using NetBeans or Eclipse. Although debugging in all frameworks was not as easy as "click and debug".

Some frameworks had not only language support built in the IDE, they also had framework support built in, in the IDE.

Many frameworks comes with console commands for creating new projects, creating controllers, domain models, views, deleting and much more with or without options. Those frameworks that do not come with console commands often offers creating new projects via Maven, but after creating the project one is on one's own.

7.3.1. CakePHP

CakePHP comes with CakePHP Bakery which is CakePHP's shell to the console. It let one "bake" new projects and creating domains, controllers and views etc. It helps one through all options when creating something new and is very useful. The downside with CakePHP is that is has no support in the major IDEs, not built-in and no plug-ins, so far. Although there is no support in NetBeans one can make a work-around to enable some tools like auto-complete, and intelligence, which saves a lot of time for developers. (33)

From version 1.2 CakePHP uses .ctp files by default which means one will probably lose syntax highlighting in most IDEs. To get syntax highlighting working many IDEs support users to add new file-endings to be treated as another file type. CakePHP 1.1 uses .thtml files.
(34)

7.3.2. Grails

Grails comes with the Grails console and is an extended version of the regular Groovy console. The Grails console lets one create projects, domain models, controllers and views based on the domain model, installing plug-in, and much more.

In NetBeans version 6.5 and later (currently stable version is 6.8) has built in support for Grails, no plug-in installation needed. This built in support gives one possibility of code completion, run Grails commands, Grails shell, handle Grails plug-ins, upgrade the framework and more. During the evaluation the support that NetBeans provides integrated very well with the functionality that Grails supplies, which provides a complete development environment for the developer. Only minor problems were discovered. A good example is, if one creates a new class using NetBeans built-in wizard and do not choose a package, by consciously or obliviousness, then Grails shell will ask the developer if one really wants to do this, while NetBeans Wizard has forced focus. This leads to one is not allowed click on the Grails shell to answer yes (y) or no (n), while NetBeans wizard is waiting for the Grails shell to finish its work. This will result in a deadlock, both parties waiting for each other. The only thing one can do if this happens is to kill NetBeans application. If one has a lot of unsaved work it will be lost. Only once during this evaluation this was solved without killing NetBeans application, and was done in a way that was not possible to repeat easily.

Grails also has support in several other IDEs, however was not tested in this evaluation:

- STS Integration – Using Grails projects in the SpringSource Tool Suite, an Eclipse-based IDE (see Spring Roo)
- IDEA Integration – Using IntelliJ IDEA IDE with Grails projects
- TextMate Bundle - TextMate Grails bundle
- Gedit - For Linux, like TextMate
- jEdit – Java based open source with great Groovy plugin.

- Emacs – GSP support for Emacs. Groovy-Mode, Grails-Minor-Mode

(35)

7.3.3. Ruby on Rails

Ruby on Rails comes with a shortcut to startup console with Ruby. I the console one can create projects, controllers, domain models, scaffold, start server, perform tests and much more. One can crate and work with just the console, a text editor and a web browser (for test one's application) and nothing more, if one would like.

Ruby on Rails has great support in NetBeans included. It is possible to create new Ruby on Rails projects directly in NetBeans without downloading any plug-in. Eclipse has support for Ruby on Rails through a plug-in.
(36) (37)

7.3.4. Spring Roo

Spring Roo was, together with CakePHP, the only framework in this evaluation that has its own shell in the console. The advantage of using a shell in the console instead of using commands is one does not have to write the framework prefix before every command. Writing the framework prefix every time do not steal much time, but can be annoying to the developer. Spring Roo is also the only framework has a specialized IDE, SpringSource Tool Suite (STS), which is build from Eclipse. If one used Eclipse before, one will feel like home. The main motivation for using STS instead of Eclipse, or some other IDE, is that STS has Spring Roo shell built in. One can use Spring Roo shell in a console window and use any IDE, but since STS is free and can replace Eclipse, or some other IDE, there is no good reason to not use STS when if one using Spring Roo.

7.3.5. Stripes

Stripes does not have any console commands, but it has support by/from Maven for installation/creating new projects. Stripes has plug-ins to NetBeans and IntelliJ IDEA, which let the developer to create new projects within the IDE. In the evaluation the NetBeans plug-in worked as intended, but one could wish for more possibilities such as creating scaffold.
(38) (39)

7.3.6. Wicket

Wicket has what Stripes has, no console commands and plug-ins to NetBeans and IntelliJ IDEA, and on top of that Wicket also has plug-in for Eclipse. Wicket also has support by/from Maven, as Stripes, but on Wicket's web site one can create the command for creating a new Wicket project with options. This is quite nice for new developers of Wicket, since it is the only command one does with Wicket. One does not have to care of learning the syntax and what to include in the command and so on, and often these command can grow large and it is easy to make a small mistake.
(40)

7.4. Validation

Almost all frameworks utilize MVC architecture and therefore solve validation in like manner. This is demonstrated in Figure 4; the input page sends the user data input to a controller (1.), which in turn calls the domain models validate function (2.). If it was correct, the controller use the input data, most likely store it in a database (3a.). If the input data was not correct it return a list of fields that was incorrect. The controller sends user's input data and which fields that was incorrect back to the input page (3b.). The input page fills in the user's input data, and fields containing corrupt data are marked, in some way.

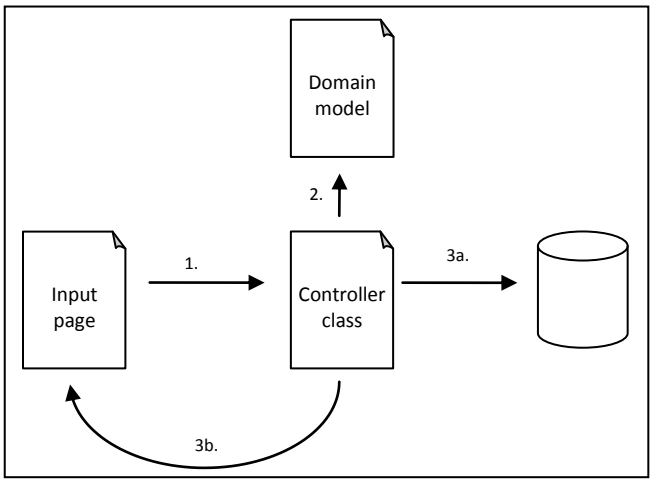

Figure 4 showing how an input page sends user's input data to controller (1.), who calls its domain model's validate function (2.). If there is an error in user's input data then error messages are created and sent back to input page together with all input data. (3b.) Otherwise the data is accepted and most likely stored in a database (3a.)

7.4.1. CakePHP

Uses MVC architecture and everything happens "AutoMagic", which means one does not have to write anything in the controller. The data is saved automatic. (41)

Below is a sample code of how it is done in CakePHP.

```
Controller:
No code needed.

Domain model:
      var $validate = array(
          'myEmailField' => 'email',
     );

Input page:
<?php echo $form->create('MyDomain');?>
            <fieldset>
            <?php
                 echo $form->input('myEmailField');
            ?>
            </fieldset>
```

Code snippet 5 Sample code of how input validation is solved in CakePHP with MVC architecture.

7.4.2. Grails

Grails creates validation when generating views and controller from a domain model. Grails gives a standard error message telling in which field the error(s) occurred. It is possible to customize these error messages and use internationalization to get error messages in the user's

native language. Grails also changes color of the field(s) with error(s) so the user easier finds were a mistake is made. The user's data are still in the fields so the users do not have to rewrite everything.

Below is a code sample how it is done in practice.

```
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 6 Sample code of how input validation is solved in Grails with MVC architecture.

7.4.3. Ruby on Rails

Ruby on 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 field(s), an error message and field(s) with error highlighted. Ruby on Rails is very powerful when it comes to input validation.

```
Controller:
No code needed.

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

Input page:
If one is using the generated scaffold everything already is there.
For printing error message this line is used:

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

Code snippet 7 Sample code of how input validation is solved in Ruby on Rails. It can be done by adding one single line to the scaffold generated code.

7.4.4. Spring Roo

Spring Roo uses Spring Framework's input validation, which is in MCV.

```
Controller:
No code needed.

Domain model:
public class MyDomain {
    @NotNull
    @Size(min = 2, max = 30)
    private String myField;
}

Input page:
<div id="roo_mydomain_myField">
    <label for="_myField_id">MyField:</label>
    <form:textarea cssStyle="width:250px" id="_myField_id" path="myField"/>
    <script type="text/javascript">Spring.addDecoration(new Spring.ElementDecoration
           ({elementId : '_surname_id', widgetType:
                                    'dijit.form.SimpleTextarea', widgetAttrs: {}}));
    </script>
    <br/>
    <form:errors cssClass="errors" id="_myField_error_id" path="myField"/>
</div>
```

Code snippet 8 Sample code of how input validation is solved in Spring Roo, using MVC architecture.

7.4.5. Stripes

To validate an input in Stripes is very easy. Just add a single line in the actionbean and the rest is already taken care of.

```
Controller:
No code needed.

Domain model/ActionBean:
@Validate(required=true)  // the new line
private String myField;

Input page:
<stripes:form beanclass="action.RegisterActionBean">
     Title of myfield:
    <stripes:text name="myField"/>
    <stripes:errors field="myField"/>  // needed to show error message
...
</stripes:form>
```

Code snippet 9 Sample code of how input validation is solved in Stripes, using MVC architecture.

7.4.6. Wicket

Wicket is using separation of concerns, but it is not traditional MVC. Wicket is instead closely patterned after stateful GUI framework, using threes of components. Wicket clearly distinguishes between views and logics by using both a Java and an XHTML file for each web page. By doing so one can edit the XHTML files in one's favorite WYSIWYG-editor.
From a developers perspective this is beautiful way of solving the separation of concern issue, but on the other hand Wicket does not provide any scaffolding and creating forms (and other CRUD operations) is therefore very time-consuming.

```
Domain model:
public class MyDomain implements Serializable {
    public String myField;
}

Domain model-application:
public class MyDomainApplication extends WebApplication {

    public MyDomainApplication(){}

    @Override
    public Class getHomePage() {
        return InputPage.class;
    }
}



Input page-Java:
public CommentForm(final String componentName) {
      super(componentName);

      // Construct form and feedback panel and hook them up
      final FeedbackPanel feedback = new FeedbackPanel("feedback");
      add(feedback);
      add(new TextField("myField", new PropertyModel(myDomain, "myField"))
                                                                  .setRequired(true));
Input page-HTML:
<span wicket:id=”feedback”/> //prints error messages

<wicket:message key="myField">My Field: </wicket:message>
<input type="text" wicket:id = "myField" />
```

Code snippet 10 Sample code of how input validation is solved Wicket. Wicket uses a variation of MVC architecture.

7.5. Internationalization (Localization)

The support and the way a framework implements i18n does vary quite a bit, but they usually have a similar foundation. The basic building blocks can be divided into three parts, the first being a place where the translated information is located, the second a way of retrieving the correct translation and the third being a way of switching the language. It is noteworthy that

some frameworks really enable the developer to internationalize an application with minimal effort, enabling for using more time on developing the functionality of an application. The frameworks that do this are Grails and Spring Roo with Ruby on Rails right behind them.

7.5.1. CakePHP

CakePHP is the only framework that recommends using a third party tool for working with i18n, to be more precise to manage the language files. This recommendation is there to ensure that the files are properly encoded and avoid corruption, which partly forces a developer to learn that tool. It also adds extra work to the management of the .pot and .po files (language files).

When it comes to control the language the sites shows it is easily done by a single command, although this cannot be done through the URL (as some previous frameworks) unless that functionality is implemented as it is not out-of-the-box functionality. Then to actually use the translation on the pages one only needs to call a translate function on the string to be translated.

7.5.2. Grails

Grails i18n support is built with the Spring and the Java i18n mechanics at its foundation, which makes adding languages into the application an easy task. By following the convention of Grails all languages files are put into a specific folder with the correct language code in the name. These language files are ordinary Java property files encoded in UTF-8 that contain codes followed by the text these codes translate into. The language codes are than easily fetched by using a Grails tag or a function call depending on the context, such as views or controllers, with the possibility to send parameters into the translations. Deciding which language will be used is decided automatically by reading the "Accept-Language" header that falls back on default it a match is not found. Although one can still change language by simple passing a "lang" parameter in the URL and then everything is handled in the background to switch the language and keep it set to the new language.
(42)

7.5.3. Ruby on Rails

Ruby on Rails offers an approach to i18n that is similar to that of Grails and Roo, although as Ruby does not use Spring for i18n there are a few differences. The language files are done in the same way apart from a few conventions as naming of the file and how a code and the corresponding translation are written in the files. Information from these files is easily fetched with the *t* (translate) helper Ruby on Rails provides, which uses the code to get the correct message. The simplicity really facilitates the work that has to be done with i18n.

When it comes to changing the language it is done by changing a variable that out-of-the-box cannot be changed with the URL, but implementing the functionality only needs 4 lines of code.

7.5.4. Spring Roo

Spring Roo also uses Spring for enabling the i18n support, which makes the management and functionality of languages very similar to Grails. Because of the same underlying framework Roo basically handles i18n in the exact same way as Grails. It makes use of property files for keeping track of the various translations and uses a tag to fetch the information so it can be output to the user. The way of changing language is also done in the same way, by passing a parameter and the rest is handled automatically.

7.5.5. Stripes

Stripes really differs from the previous frameworks when it come to handling i18n, as first of all it does not offer a way of making your application multilingual instead it leaves the choice of how to do it to the developer and second the help if offers needs some configuration before it can be used as desired. Although Stripes tries to facilitate the way i18n is implemented by the means of configuration and allow the use of tags to make it easy to get hold of the messages. The difference from previous frameworks is still quite large, although as mentioned because of the lack of full support there are more options for the developer, but it makes the implementation of i18n take up additional time. This is a direct result of the fact that Stripes is a more lightweight framework.

7.5.6. Wicket

Wicket, just as Stripes, has its own way of dealing with i18n, although for the developer it will be similar to the Spring way that Grails and Roo use. One major difference is that fact that Wicket does not put focus on having the files in a special i18n folder put instead it focuses more on the naming of the files. Where it is a good practice to have a property file for each page, but one is not limited to doing it that way put can instead put all the information in one file. The messages are retrieved with Wicket tags that take a value as parameter, which is the code of the value gotten from the property file. These look-ups also work for images, if named property the images change depending on the locale being used. The actually changing of locals differs from previous framework, with the difference being the fact in Wicket the session locale value has to be set and it is not done automatically be sending a parameter. In the end there is a bit of an extra work, but it still simplifies working with i18n.

7.6. Testing

The approaches to testing and ways of testing differ a lot between the frameworks, although they all offer some support to testing. What type of testing depends on the framework, so there is no one type of testing that all the frameworks offer.

7.6.1. CakePHP

CakePHP offers testing the help of SimpleTest, which does not ship with the core CakePHP functionality to additional configurations are needed to get everything up and running. Although when configuration is completed one can utilize all of the functionality offered for

performing unit testing. In addition to the unit testing CakePHP also offers testing of views in the application. Everything is then gathered in one command in the CakePHP console, which enables running of any of the tests created.

7.6.2. Grails

Testing in Grails is relatively easy, 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, assuming the create domain class command is used. Within the created class 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, one wants to run, are created they can be run with a single command, which generates statistics about how the tests went.

7.6.3. Ruby on Rails

Ruby on Rails, as 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.

7.6.4. Spring Roo

Roo, like Grails, aids the developer with testing as it offers a command to auto generate test classes. These generated test classes will automatically verify database operations like find and delete, with a total of eight tests per entity and with the ability for the developer to add additional tests.

7.6.5. Stripes

Stripes has a quite different approach to testing, which is basically the “do it yourself” approach. One can do regular unit testing on the actionbeans or alternatively use Mock objects to get a more accurate testing environment for the actionbeans.

7.6.6. Wicket

Wickets focus in testing is being able to render pages and then assert that the components rendered contain the correct values. As it does not have any classic unit testing because of how the framework works. Also the testing is done manually as Wicket does not offer any generation of classes and such.

8. Lessons Learned

During the evaluation process there were a few things noticeable with frameworks in general and some things with Grails as it was chosen to be the framework used in the implementation part.

8.1. General

When looking at all the frameworks out there one notices that a lot of programming languages have some kind of web application framework even if some languages, like Java and PHP, have a lot more frameworks. This enables these two languages to have frameworks specializing in certain areas, but with the most programming languages having some sort of larger MVC framework and one can see that these do differ, but they have common structures and features. Meaning that whatever language ones pick there is always something one can use and switch between languages and frameworks and still use the same thinking when developing.

8.2. Grails

During the implementation with Grails one notices that apart from the well documented part of Grails there are a few things missing, such as some plug-ins are not always properly documented as one would hope.

There are also some “features” that are not documented as the fact that one cannot change language in the index page by sending a parameter as with all other pages. This is basically see as a bug by developers using the framework, but as a “Won’t fix” by the developers of Grails. So this “feature” exists, but is not properly documented so when using the framework this will come as a surprise.

One 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, which is great if Grails does not have a plug-in that supports a certain feature one can use a Java library.

It should also be mentioned that 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. Although in the implementation of the web application a service provider was used, which is not the standard way of working but nevertheless worked without any difficulties. One just loses some of the functionality Grails has that would otherwise aid in the development.

9. Conclusion

In this evaluation no superior framework was found. Instead, there are many frameworks working pretty well, and since all of them might not have the same application areas it is not fair to really say one is better than another.

As expected there are many frameworks lacking in some way, mostly in documentation. It was also found that to be an excellent (or 'expert') developer in a framework it takes time and experience. There is an advantage to be skilled in same language as the framework is written in, in order to get the most out of the framework. Therefore, a skilled developer of a programming language should use framework in same language instead of learning a new language just for using a recommended framework.

In the evaluation it was found at least one good framework in each programming language, at least one of those programming language frameworks were tested in this evaluation. In other words, do not learn a new programming language just for using a framework, especially if one will only use it a few times.

In some cases there are other external factors that will impact the situation. With frequent use of many applications e.g it is a request from a customer, or you just want to get more experience. Then you should go ahead and learn the new language and the new web framework.

To choose a framework, effort in finding a suitable application is very important. If the framework is not suitable for the application, the framework might be a hinder instead of a helping hand. It's crucial to find out if the framework is built for my size of application. For a very small application it is more likely that the framework is just too complex, with regards to the size of the application. If one is building a small weblog with just reading and saving to a database, then a simple application written in PHP is enough to do CRUD operations. Likewise if one will build a gigantic application where scaling is important, one might build it from scratch to be able to optimize clock cycles per instruction.

10. References

1. **Time Magazine.** Tim Berners Lee - Time 100 People of the Century. *Time.* [Online] [Cited: 05 17, 2010.] http://205.188.238.181/time/time100/scientist/profile/bernerslee.html.
2. **NCSA.** CGI: Common Gateway Interface. [Online] [Cited: 05 31, 2010.] http://hoohoo.ncsa.uiuc.edu/cgi/intro.html.
3. **William F. Slater, III.** Internet History and Growth Presentation by William Slater III - Chicago Chapter of the Internet Society. [Online] 09 18, 2002. [Cited: 05 29, 2010.] http://www.isoc.org/internet/history/2002_0918_Internet_History_and_Growth.ppt.
4. **Framework Docforge.** Framework. *An Open Wiki For Software Developers.* [Online] DocForge. [Cited: 06 01, 2010.] http://docforge.com/wiki/Framework.
5. **Alex Chaffee.** What is a web application (or "webapp")? *jGuru.* [Online] 08 17, 2000. [Cited: 06 02, 2010.] http://www.jguru.com/faq/view.jsp?EID=129328.
6. Web application framework. *An Open Wiki For Software Developers.* [Online] DocForge. [Cited: 05 28, 2010.] http://docforge.com/wiki/Web_application_framework.
7. **Dot Net Guy.** Agileprogrammer. [Online] 05 30, 2007. [Cited: 06 02, 2010.] http://www.agileprogrammer.com/dotnetguy/archive/2007/05/30/22819.aspx.
8. **Matt Raible.** The History of JSP. *Raible Desgins.* [Online] 01 29, 2003. [Cited: 06 01, 2010.] http://raibledesigns.com/rd/entry/the_history_of_jsp.
9. Designing Reusable Classes. [book auth.] Ralph E. Johnson & Brian Foote. *Journal of Object-Oriented Programming.* 2. University of Illinois at Urbana-Champaign : s.n., 1988, Vol. 1, pp. 22-35.
10. **Martijn Faassen.** Web frameworks considered useful. [Online] [Cited: 06 01, 2010.] http://faassen.n--tree.net/blog/view/weblog/2010/04/15/0.
11. **Kate Cook.** Pros and Cons of an MVC Framework. *J House Media.* [Online] 02 11, 2009. [Cited: 06 01, 2010.] http://www.jhousemedia.com/blog-profile.php?ID=22&Category_ID=20.
12. **Miller, Jeremy.** "Convention over Configuration". *MSDN Magazine.* Feb 2009.
13. **Elegant Code.** Convention over Configuration. *Elegant Code.* [Online] 11 28, 2009. [Cited: 06 01, 2010.] http://elegantcode.com/2009/11/28/convention-over-configuration.
14. **Farlex, Inc.** Integrated development environment. *The Free Dictionary.* [Online] [Cited: 05 28, 2010.] http://encyclopedia.thefreedictionary.com/integrated+development+environment.
15. **Oracle Corporation.** NetBeans IDE 6.8 Release Information. *NetBeans.* [Online] 05 19, 2010. http://netbeans.org/community/releases/68/.
16. **The PHP Group.** *PHP: Hypertext Preprocessor.* [Online] [Cited: 05 19, 2010.] http://php.net/index.php.
17. **Tiobe Software.** TIOBE Programming Community Index for June 2010. [Online] [Cited: 06 06, 2010.] http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html.
18. **Cake Software Foundation, Inc.** Basic Features. *CakePHP: the rapid development php framework .* [Online] [Citat: den 28 05 2010.] http://cakephp.org/pages/features.
19. —. 1.1 What is CakePHP? Why Use it? *The Cookbook.* [Online] [Cited: 06 01, 2010.] http://book.cakephp.org/view/8/What-is-CakePHP-Why-Use-it.
20. **Codehaus Foundation.** *Groovy - An agile dynamic language for the Java Platform.* [Online] [Cited: 05 25, 2010.] http://groovy.codehaus.org.

21. **SpringSource.** Grails - Introduction. *Grails.* [Online] [Cited: 05 28, 2010.] http://www.grails.org/Introduction.
22. **Klein, Dave.** Grails, A Quick-Start Guide. s.l. : The Pragmatic Bookshelf, p. 22.
23. About Ruby. *Ruby Programming Language.* [Online] [Cited: 05 28, 2010.] http://www.ruby-lang.org/en/about/.
24. **Sam Ruby, Dave Thomas, David Heinemeier Hansson.** Agile Web Development with Rails. Third Edition. u.o. : The Pragmatic Bookshelf, s. 14.
25. —. Agile Web Development with Rails. Third Edition. p. 15.
26. The core team. *Ruby on Rails.* [Online] [Cited: 05 28, 2010.] http://rubyonrails.org/core.
27. *Stripes Framework.* [Online] [Cited: 05 19, 2010.] http://www.stripesframework.org/display/stripes/Home.
28. **SpringSource.** *Spring Roo.* [Online] [Cited: 05 20, 2010.] http://www.springsource.org/roo.
29. —. Chapter 1. Introduction. *Spring Framework.* [Online] [Cited: 05 28, 2010.] http://static.springsource.org/spring-roo/reference/html/intro.html.
30. **Apache Software Foundation.** Wicket - Features. *Apache Wicket.* [Online] [Cited: 06 01, 2010.] http://wicket.apache.org/features.html.
31. —. Introduction. *Apache Wicket.* [Online] [Citat: den 01 06 2010.] http://wicket.apache.org/introduction.html.
32. Open Source Web Frameworks in Java. *Java-Source.net.* [Online] [Cited: 05 28, 2010.] http://java-source.net/open-source/web-frameworks.
33. **TipLite.** CakePHP support in NetBeans. *TipLite.* [Online] [Cited: 05 19, 2010.] http://www.tiplite.com/cakephp-support-in-netbeans/.
34. Using CakePHP in your IDE. *Cakephp tutorials.* [Online] [Cited: 05 19, 2010.] http://cake-php-tutorial.blogspot.com/2009/06/using-cakephp-with-your-ide.html.
35. **SpringSource.** Grails - IDE Integration. [Online] [Cited: 05 19, 2010.] http://www.grails.org/IDE+Integration.
36. **Sébastien.** Ruby / Rails IDE Comparison. *The Nameless one.* [Online] 02 28, 2007. [Cited: 05 19, 2010.] http://tnlessone.wordpress.com/2007/02/28/ruby-rails-ide-comparison-idea-netbeans-radrails/.
37. **Deitel.** Eclipse for Ruby on Rails Development. *Eclipse Resource Center.* [Online] [Cited: 05 19, 2010.] http://www.deitel.com/ResourceCenters/Software/Eclipse/EclipseforRubyonRailsDevelopment/tabid/1467/Default.aspx.
38. **Oracle Corporation.** NetBeans Plugin Portal. [Online] 01 28, 2008. [Cited: 05 19, 2010.] http://plugins.netbeans.org/PluginPortal/faces/PluginDetailPage.jsp?pluginid=5115.
39. **Atlassian Confluence.** Stripes IDE integration. *Stripes Framework.* [Online] [Cited: 05 19, 2010.] http://www.stripesframework.org/display/stripes/IDE+integration.
40. **Apache Software Foundation.** Apache Wicket IDEs. *Apcache Wicket.* [Online] [Cited: 05 19, 2010.] http://wicket.apache.org/ides.html.
41. **Cake Software Foundation, Inc.** 7.3.3 Automagic Form Elements. *The Cookbook.* [Online] [Cited: 05 23, 2010.] http://book.cakephp.org/view/189/Automagic-Form-Elements.
42. [Online] [Citat: den 25 05 2010.] http://grails.org/doc/latest/guide/10.%20Internationalization.html.

APPENDIX I

This appendix will cover most of the information about the application, from purpose to design and implementation.

Pomodoro Technique

The Pomodoro Technique is used to make use of time more efficiently, by setting up intervals to work in, using a timer. For each interval one or more activities are chosen and worked on nonstop for the duration of the interval. One working interval is called a *Pomodoro* and recommended length of a Pomodoro is about 25 minutes. During a Pomodoro one takes notes of interruptions that occurred while working. An interruption might be internal where one lose focus do to self reasons or external when some outside source distracts one from the work. Between every Pomodoro there is a shorter break, and after about four Pomodoros there is a longer break. By working in these intervals one utilizes time more efficiently and can thus be more productive. Activities that are not completed in one interval can be continued in another Pomodoro.

Design

The Pomodoro web application is designed to be able to work with a service provider from where it gets all the data a user needs and stores while using the application. By using a service provider it enables the user to switch platform if wanted. The web application is one out of three being developed, and the other two are an Android and an iPhone application, see figure 1. Android and iPhone will offer the same functionality and are used in a similar manner as the web application. The service provider connects these three platforms by using a REST API to enable retrieving data and OAuth for user authentication, which puts certain limitations on the application.

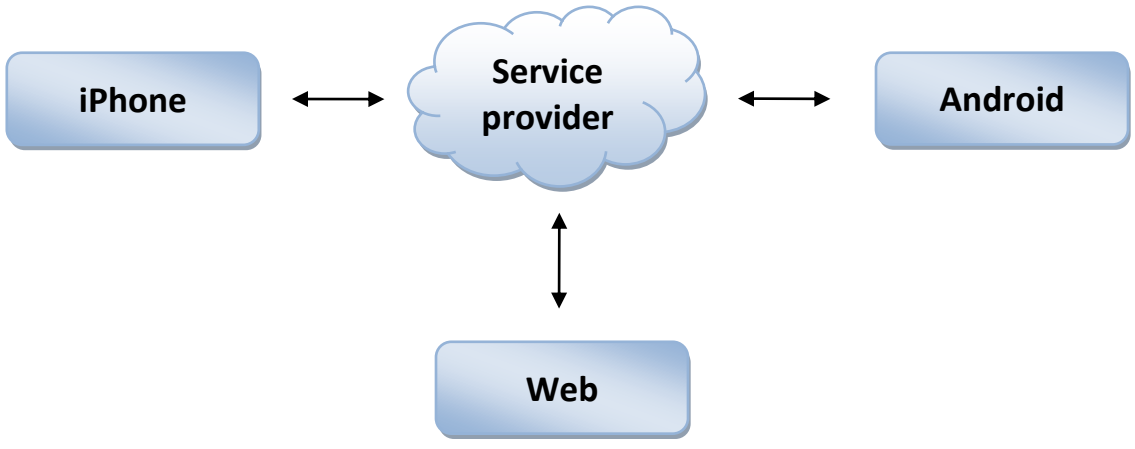

Figure 1 visually showing how the systems will interact with the service provider.

Domain Model

The domain model covers all the basic entities found in the Pomodoro Technique and also has some additional ones that add to the functionality of the application, as seen in figure 2.

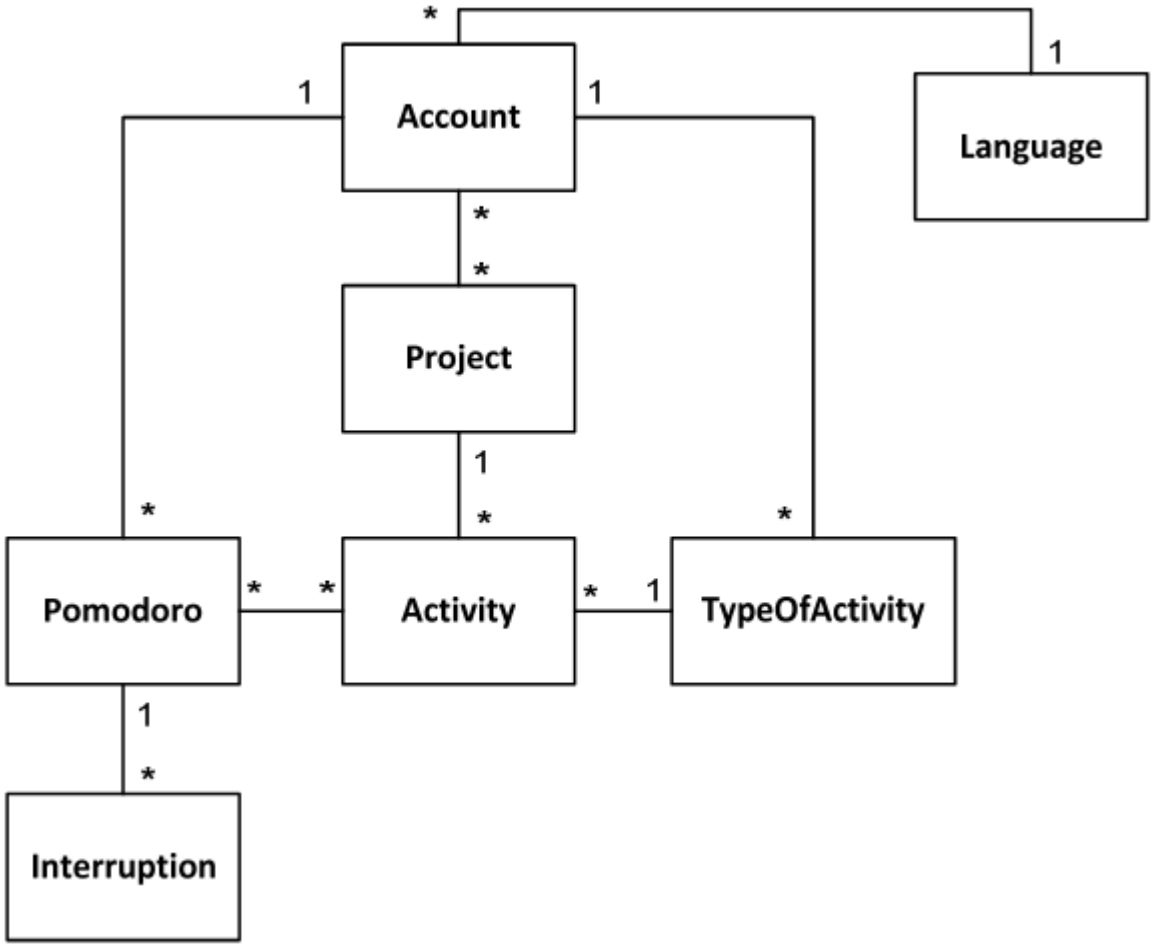

Figure 2 showing domain model over the Pomodoro application.

The biggest difference from the basic Pomodoro Technique is the introductions of projects to be able to easier organize activities and enable several users to work together.

Features

The application first of all supports users, which means each user has its own workspace. This workspace is then used to create and manage projects enabling users to then organize their activities, which also can be managed here. Projects can be connected to several users by allowing the project creator (owner) to invite other users and set roles to these users. With this functionality all users in a project can share the activities within the project. In addition to this the application can run a Pomodoro (timer), add activities to it and report interruptions while running. This allows the users to use the full power of the Pomodoro Technique and if needed switch between platforms.