RoamPA Reports and Orientation Web App(s)

John Frizzell

Andrew Sweeney

B.Sc.(Hons) in Software Development

April 24, 2016

Final Year Project

Advised by: Dr John Healy
Department of Computer Science and Applied Physics
Galway-Mayo Institute of Technology (GMIT)

GMIT
INSTITIÚID TEICNEOLAÍOCHTA NA GAILLIMHE-MAIGH EO
GALWAY-MAYO INSTITUTE OF TECHNOLOGY

Contents

1 Introduction **4**

2 Context **5**

2.1 Why we chose this project . . . 5

2.2 Challenges we had to face . . . 5

2.3 Objectives . . . 6

2.4 Specifications . . . 6

3 Methodology **7**

3.1 Introduction . . . 8

3.2 Body . . . 9

3.3 Analysis . . . 11

3.4 Conclusion . . . 12

4 Technology Review Breakdown **13**

4.1 Spring MVC and MVC . . . 13

4.2 Hibernate ORM . . . 15

4.3 HTML and CSS, Javascript/jQuery . . . 19

5 System Design **20**

6 System Evaluation **21**

7 Conclusion **22**

About this project

Abstract This project was contracted to us through Enterprise Ireland from a company called RoamPA founded by Una McNeill. RoamPA contacted us to build two modules for a J2EE (Java Enterprise Edition) web application. RoamPA was building a web application aimed at business users and contracted two modules to be developed for them externally. One of the modules consisted of converting an already existing legacy reports module so it would work in a Tomcat environment. This module would allow administrator users to create reports using a streamlined query builder. This system was primarily aimed at administrator users from the human resources department. The other module was a reviews system. The reviews system allows users to generate reviews about existing businesses,faculties, points of interest etc. The reviews system would also let users create,update,view,like and dislike reviews, similar to Trip Advisor. Both of these modules had to be created from scratch and had to integrate into RoamPA's existing server environment. This integration restrains the technologies used and furthermore required the modules to be integrable with their RDMBS (Relational Database Management System). The project involved research, learning many frameworks and understanding the architectural patterns involved with developing a J2EE application.

Authors The authors of this project are Andrew Sweeney and John Frizzell. We are both graduates of the BSc in Computing in Software Development which was the preceding course to the current honors degree. We both have a great interest in programming and technology. We are constantly seeking ways to improve our knowledge and skills.

Chapter 1

Introduction

RoamPA Ltd is a start-up company founded by Una McNeill. They are developing a web application which is used to streamline the process of moving employees globally for work and business purposes. They wish to provide a much improved employee experience, significant cost savings to clients and greater return on investments with assignments and business travel.

We had previously worked with Una McNeill through another Enterprise Ireland contract. The previous contract involved creating a reports module and involved four students altogether. Our last project was our first real collaborative experience where we programmed together whilst incrementally updating the project. This project was delivered successfully to RoamPA and served to build a relationship which acquired us a further contract.

Creating the previous report module was our first real experience with developing in web and server languages such as PHP and JavaScript. The reports module is now a legacy module because the environment for the web application had now shifted from a PHP MVC web application (Model View Controller) environment to now a J2EE web application environment. This shift in environments meant having to redo the reports module as the server language used was now Java and not PHP. Recreating the reports module was one of the two modules we had to produce.

The main module we had to create was a reviews module. This again was restrained by the current environment and involved a lot of researching and agile iterative development. The reviews module consisted of a map element and being able to add reviews to filtered establishments that populated the map. This system would be used to view and create reviews. Based off a reviews popularity or recommendation an employee could judge whether or not to give their business to the reviewed establishment.

Chapter 2

Context

2.1 Why we chose this project

Throughout our college course we have developed in many different languages. Java, C#, C and Python to name a few. We have not had the chance to really to dig deeply into web development. Web languages like PHP and JavaScript were not part of our syllabus. Our sister course ”Computing in Digital Media” does some of the design elements of web development but not much of the back-end stuff that we've both liked. Throughout our course we had a few semesters that had Database modules involving MYSQL. We heard the current project involved full blown MVC, Servlets, ORM and JSP pages. We also learned that the main language required for the project was Java which is one of the core languages for our college course. This project gave us a whole new area to experiment with and learn. The project would give us the opportunity to develop for a J2EE environment. Due to the challenge and interest we accepted the contract for nthis project.

2.2 Challenges we had to face

Straight from the beginning, the challenge was to understand all of the frameworks and how all of the different parts of the environment work together. This meant completely understanding concepts like ORM and how they are applied through the framework Hibernate. Other concepts included learning how MVC is applied through SpringMVC with routes etc. There was a lot to research and a lot of frameworks to read up on. There was also numerous dependencies to figure out. The amount of research, tutorials and understanding to be put into the project was also a big challenge as well as the actual development of the project itself.

2.3 Objectives

- Follow the specifications provided to make integration easier
- Research the frameworks and environment
- Create a reports module
- Create a reviews module

2.4 Specifications

We were creating two modules for RoamPA. Both modules had to be integrated onto RoamPA's existing Application server. The Frameworks required of us were to use Hibernate ORM and SpringMVC. The existing database RoamPA uses is MySQL. All of the code handed over to RoamPA must be integrable on their server.

The first module involved creating a reviews system. The reviews system should allow the customer to look up an area using a search box and using filters then find locations for buildings like hospitals,restaurants and numerous other points of interest. The customer could then click on the Point of Interest (POI) and create a review for a business. They can also rate a current review that exists on there and like or dislike it.

The second module

JOHNS BIT ABOUT REPORT GENERATOR

Chapter 3

Methodology

Consisting of a team of two we communicated weekly through Skype calls,reviewing over each others code. The process was incremental as we first primarily spending a lot of time researching and understanding the environment. We went through cycles of researching and going through tutorials on Hibernate and then later on tutorials for SpringMVC. This helped reinforce our understanding of how everything worked. At times we would meet up to do some paired programming, this proved as a great experience as we were able to help each other understanding the project a lot quicker. The was numerous cycles of experimenting with Hibernate, trying pass information through it to the database and gradually building up knowledge from there.

We used GitHub as a repository for our code. This is how the code you later be passed over to the developers at RoamPA. Using GitHub enabled us to download each others work and test locally. GitHub greatly helped our work-flow enabling us to share code between us in minutes remotely from home.

We were restrained to using technologies that would only work on an Apache Tomcat server. This means programming using the following

- Programming all of the server back-end in Java
- Designing all of the web pages using HTML/CSS JavaScript and jQuery, these then were later fitted into JSP pages
- Keeping all of the database queries to conform with MySQL
- Using Hibernate to facilitate the passage of data too and from the database
- Using SpringMVC to manage the structure of the project

This meant the only real choice in technologies to use we had was deciding on what map API to use.

Google Maps was always the first consideration for what mapping API would use. The most well known and popular choice may not always be the best solution to the problem. This lead us to experimenting with several other map API's. Alternatives involved looking at Bing Maps, OpenStreetMap and Leaflet. Even with all of the choice Google Maps still proved to be the best selection. There were varying reasons for not using tha alternatives primarily because of terms of contract, the cost and the map API's being relatively new left them open to change in future. Google Maps has been around 11 years and is constantly updated. Google Maps API allowed querying exactly what we wanted. Google Places is part of the Google Maps API and allows for querying using different type filters. An example of some of the type filters are airports,libraries, department stores, restaurants etc. You can pass in a location, a filter and a radius to the API and have all businesses of that type returned back in JSON format.

3.1 Introduction

In the context of the time span of computing, Word of Mouth (WOM) has long been shown to be efficacious and an important source of information when consumers wish to make purchasing decisions[1]. The usage of Word of Mouth as a marketing tool dates back to, at least, the 1970s, after psychologist George Silverman created his 'Word of Mouth Model' after observing focus groups of physicians discussing pharmaceutical products [2]. Here he found that within a focus group the positive opinion of one physician could sway other members of the group who had previously been sceptical, and even the opinions of those whose patients had had negative experiences with a particular drug.

With the rapid proliferation of digital services that allow consumers to review services online this model is now applied as an Electronic Word of Mouth (eWOM), in 2008 Nielsen Online found that 81% of online shoppers read online reviews from other customers and 71% agreed that these reviews made them more comfortable in purchasing the product, in comparison only 14% sought out reviews from an established source [1]. In the case of tourism B. Brown of the University of Glasgow concluded that tourists will use online reviews as a way to "pre-visit" locations [3].

Many sites and web platforms supply the user with the ability review services or goods, however, the quality and quantity of these reviews vary greatly [4].

3.2 Body

We know that in the field of traditional marketing, by 'Word of Mouth', that the perceived expertise of the reviewer correlates to how likely it is that the receiver will pay attention to the information on offer [5]. Given that most reviews online are anonymous or tied to identities which the receiver of the review knows nothing about this perceived expertise must be attained in other ways:

Lopez et al[6] performed research on the *yelp.com* platform comparing the 'usefulness' rating of reviews to the traditional WOM model, and found that many of the 'cues' provided by the site to indicate a reviewers credibility or expertise strongly correlated to 'high usefulness' ratings for reviews, particularly in the case of 'local expertise' they found:

"Compared to a reviewer with only one review in the neighbourhood, the count of votes increases 14.3% when the reviewer has more than five reviews in the neighbourhoodd."

In the case of Electronic Word of Mouth via online review systems we see that there are some differences in the model;

The critical factor in this context is not necessarily the perceived expertise but the perceived credibility of the source, which can be hard to judge since frequently the source of an eWOM document is anonymous. The receiver of WOM information is more likely to find it credible if the source is known to them[7], however this is rarely the circumstance when we are dealing with online reviews and eWOMs, to offset this many eWOM aggregators apply ratings systems for the reviews themselves[8], however this has it's own challenges and problems, namely that often a simple 'was this review helpful' binary option does not give a good measurement of how credible the source may be. Additionally we can see that new reviews or reviews on niche products are likely to have very few ratings as to their helpfulness[9].

Zhang et al proposed a Probabilistic framework for ranking helpfulness of eWOMs, however there were problems when some eWOMs did not have enough user ratings [4]. There is also some trouble with the self selective nature of the response to this kind of polling as outlined by Sipos et al[10] where they demonstrate that

". . . the observed connection between context and voting behaviour cannot be captured by a cardinal voting model (Section 3), where users make absolute and independent judgements about helpfulness."

i.e. the context in which a review is presented influences the polarity of the ratings that it receives.

In their paper, Kim et al[11] explored automatically rating reviews by helpfulness without manual user input using a Support Vector Machine regression algorithm relying on the wealth of existing eWOMs from other sites.

Ammar et al[12] of MIT propose a ranking system based on partial data by combining ordinal (comparison) data and first-order marginal (top preference) data, however this requires two forms of ranking to be carried out by the user, both making comparisons and ranking in order of preference.

Another phenomena that one encounters in the context of Electronic Word of Mouth is 'spam', Lim et al[13] define this to be:

". . . all forms of malicious manipulation of user generated data so as to influence usage patterns of the data"

This definition would, in some cases, overlap with fraud, and spam need not be malicious in all situations, however, they refer to this as 'review spam' or 'opinion spam' in the context of online reviews, and characterise it as reviewers habitually leaving reviews rating services at a significant deviation from the mean rating for that service. They propose to implement aggregation of the reviewer to detect this behaviour and preserve the integrity of the review system. This approach also seems to suffer from removing the reviews of people who are simply more or less critical than average.

Centeno et al[14] discuss online review system fraud, defining fraud as:

"The fact of promoting bad entities, or damage the reputation of good entities"

They propose a solution to this problem by replacing the standard rating system with a comparison system asking users:

'Which review do you think users would find more helpful, A or B?'.

Dubey et al[15] state that due to the vast quantity of reviews on some services that users will only read a small sample of these which may not be representative, they propose a technique of summarising reviews by first clustering them by user rank (which has been outlined above) and then selecting the sections of those reviews which are similar to the others in their cluster. They do not consider this an opinion mining activity as there isn't semantic analysis taking place, merely similarity analysis. This approach has a drawback in that it *". . . sacrifices the immediacy and narrative structure of reviews"*[16] according to Lappas et al of Boston University.

Jin et al[17] recommend a true opinion mining algorithm which clusters reviews by ordinal rank (rank they give to the service) and then using the service characteristics as keywords to perform the opinion mining. The ordinal clustering allows for both detecting customers fulfilled and unfulfilled needs without need to resort to elaborate semantic analysis.

Lappas et al[16] also state that ranking by 'usefulness' or 'importance' as we saw earlier *"Leads to redundant or non-representative summaries"*, they propose a system which selects a subset of reviews in those cases where there are a number that it is infeasible for the user to scan through manually, and use a similar technique to that of Jin et al to select reviews that deal with key features of the service. Their aim is to maintain the statistical properties of the superset in their subset with regards to both ordinal distribution and feature discussion.

Ghose et al[9] also propose an opinion mining approach but they tune their algorithm to maximise economic results, i.e. promoting the reviews which have the largest effect on sales. Here they propose to assign the review a Subjectivity/Objectivity score by using text mining to determine whether the review is factual, based on features of the service, or sentimental, based on a personal description of the service or experience with the service. They then analysed these subjectivity values against economic impact. The findings here were not conclusive, however they did find a correlation between reviews that combined objectivity and subjectivity with higher 'usefulness' scores from users.

3.3 Analysis

What becomes clear from the predominance of the literature is that this field of research is booming alongside the prevailing business model of service aggregation with many business being based on the disruptive business plan of 'man in the middle, service aggregator, platform' a la *Uber*, *AirBnB*, *TripAdvisor* etc. All of these platforms feature user reviews prominently and the amount of raw user content being produced mandates aggregation of some sort.

As we have seen there are many competing approaches to this problem each with downsides, whether they be functional or in difficulty of implementation, or both. It is apparent that the approaches to solve the problem have continued to evolve as the commonality of the problem increases. The simplest approaches are allowing users to rate services and simply displaying those reviews in some order, from there reviews are reviewed by other users, reviewers build up aggregate scores to determine credible reviewers, and then there are the statistical approaches, and finally the sentiment mining and machine learning approaches.

We also see from the Lim et al paper[13] and the Centeno et al[14] paper that there are pitfalls that any review system must deal with beyond normal security, the system also has to deal with abnormal user behaviour to ensure integrity and value to other users.

3.4 Conclusion

In conclusion, the literature seems to indicate that a review system must be designed with context in mind, some solutions are better suited to environments where there is an abundance of user data, in the form of reviews and rankings of those reviews, which need to be aggregated, while others are better suited to environments with a scarcity of reviews and rankings. In addition we see that some environments can lend to enhanced 'perceived expertise' (which translates to perceived credibility in an electronic context) from the traditional Word of Mouth model, in that an environment can, at least, 'confirm' that the reviewer is an actual customer (i.e. confirmed purchase on a retailer site). Other environments allow anyone to leave reviews in which makes the case for any given reviewers perceived credibility is harder to make. In the latter situation, the possibility of fraud and undesirable user behaviour (i.e. spamming) is of more concern, so the recommendations of [13]Lim et al and [14]Centeno et al are more important.

It also appears to be of some significance to provide cues as to a users credibility, that is; their level of expertise in a geographical area, how useful their reviews are to other users and so on[6].

Chapter 4

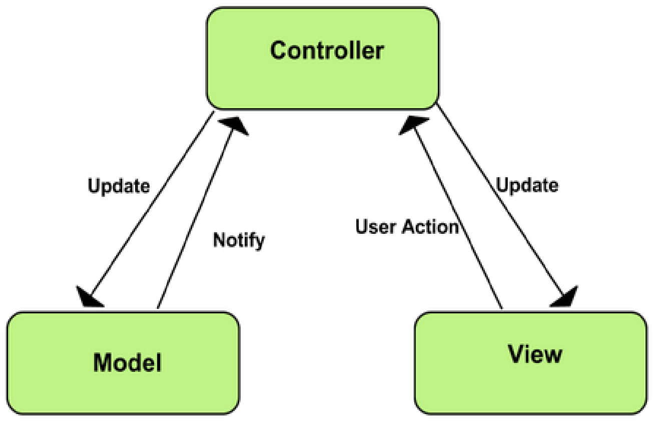
Technology Review Breakdown

- Spring MVC and MVC
- Hibernate ORM and MySQL
- HTML and CSS, Javascript/jQuery
- JSP Pages

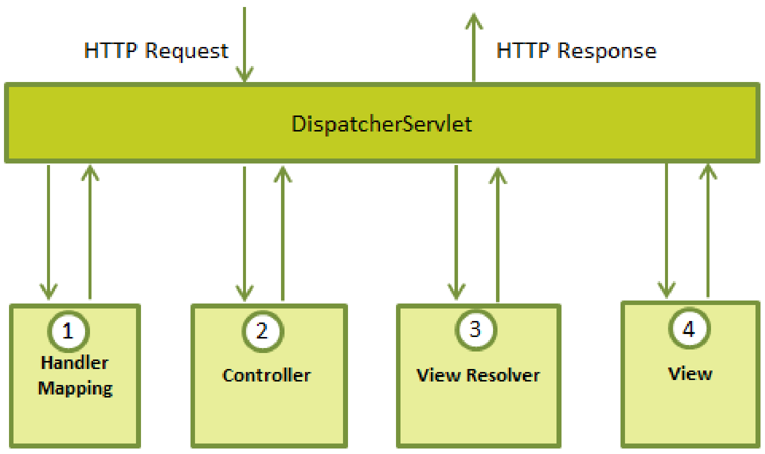
4.1 Spring MVC and MVC

SpringMVC means Spring Model-View-Controller. SpringMVC is the framework we had to conform to. Quite a lot of time was spent learning how to use Spring along with understanding Hibernate ORM.

Model View Controller is a design pattern which consists of a Model, a View and a Controller. MVC is a widely used design pattern by web application frameworks. A user makes a request to the server through a controller and a view is served back. Traditionally the controller will take the request and then manage gathering all of the information from the database and build up a View. Once the information is gathered the controller finally serves the completed view to the user. This view is typically a mix of static and dynamic information based of what's currently in the database. This is a common design pattern for web applications. It scales well and deals with pushing the right information to the user based on the parameters passed to the controller. A request comes in from the user and a response is passed back to the user in the form of a response or a view. Shown in the image below.



SpringMVC has it's own slightly altered version of this. It works off the same principals just with a more indept controller.



After receiving an HTTP request, DispatcherServlet consults the HandlerMapping to call the appropriate Controller.

The Controller takes the request and calls the appropriate service methods based on used GET or POST method. The service method will set model data based on defined business logic and returns view name to the DispatcherServlet.

The DispatcherServlet will take help from ViewResolver to pickup the defined view for the request.

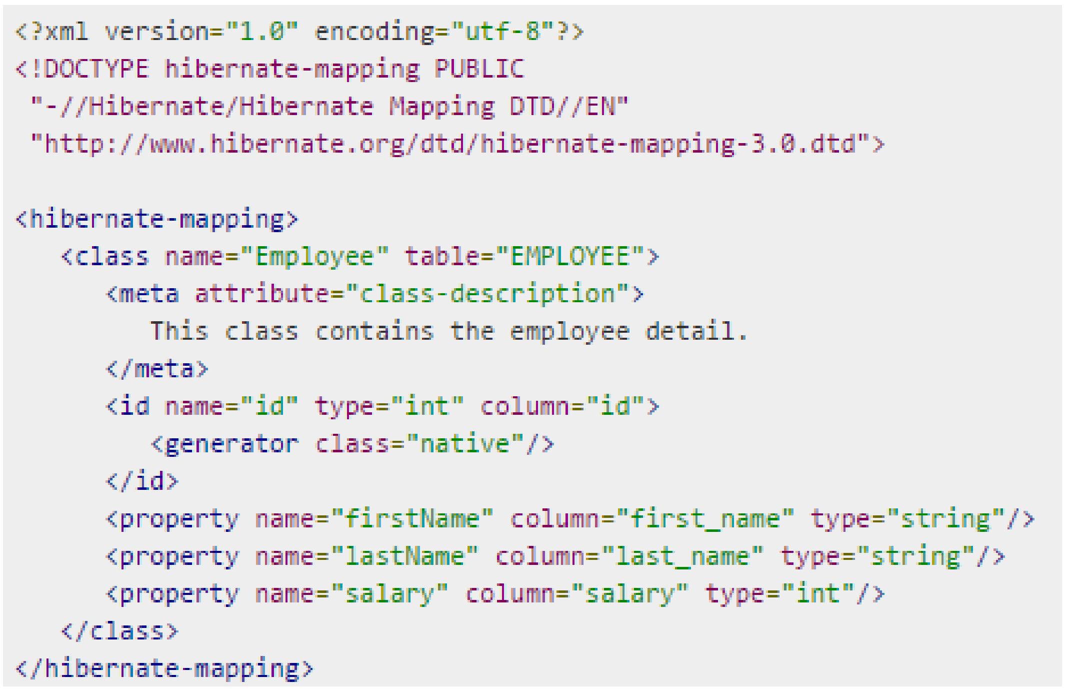
Once the view is finalized, The DispatcherServlet passes the model data to the view which is finally rendered on the browser.

It was very important to the project to understand this MVC concept.

4.2 Hibernate ORM

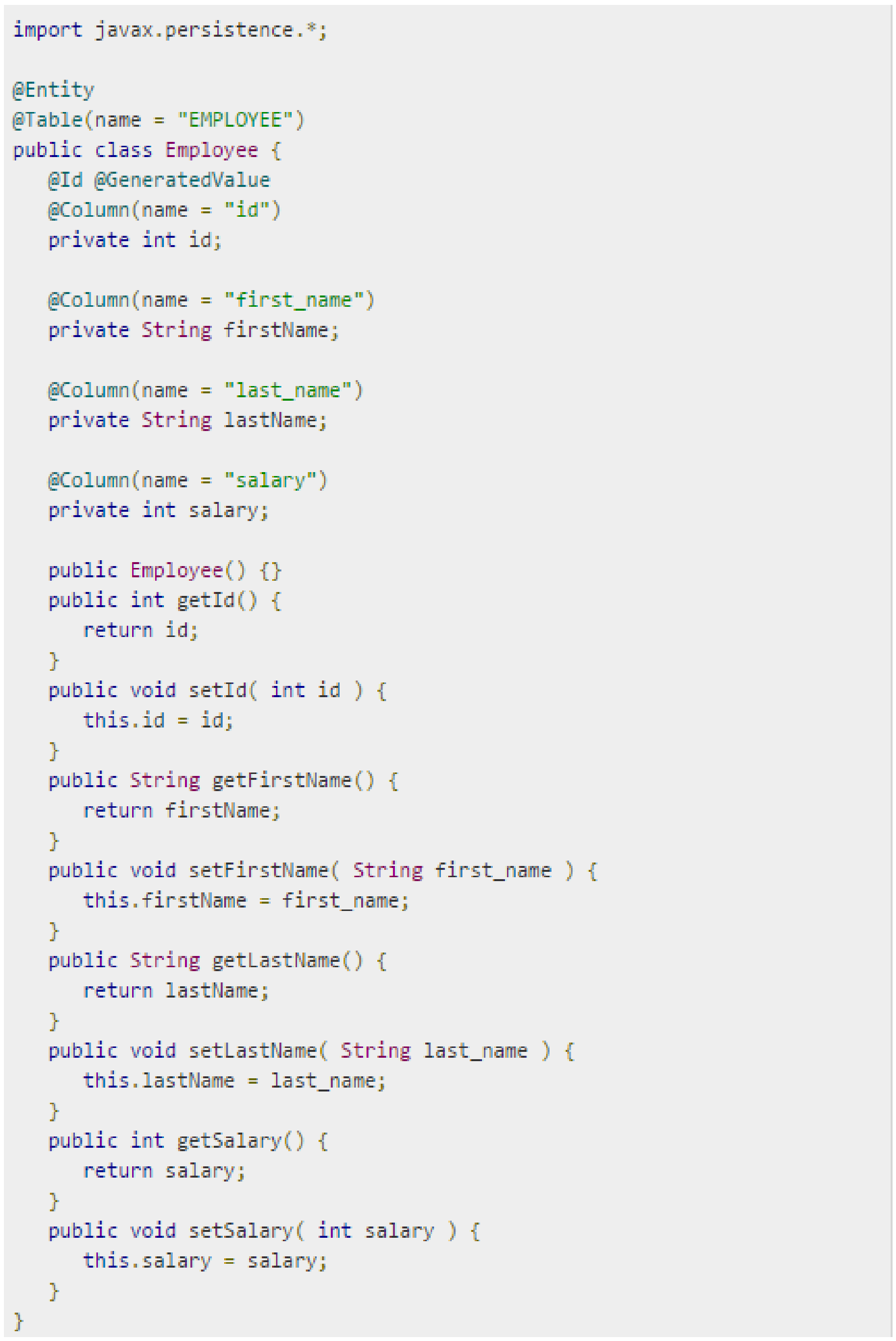
Hibernate ORM is referred usually too as Hibernate for short. Hibernate is an Object Relational Mapping framework which is what ORM stands for. ORM is a technique for mapping objects, in our case, mapping instances of Java classes and relating them to tables and columns in our chosen database. There instances of Java classes are known as POJO's or plain ordinary Java objects. Mapping of the objects to columns and tables in Hibernate is done through XML files or more recently through annotations. Hibernate XML files can be generated through Eclipse but if column names differ from field names you will find yourself spending a lot of time manually editing these files. This leads a lot of margin for error and leads to a lot of overhead. E.g Manually assigning primary keys, fields and making sure relations are properly defined.

Below is an example of this



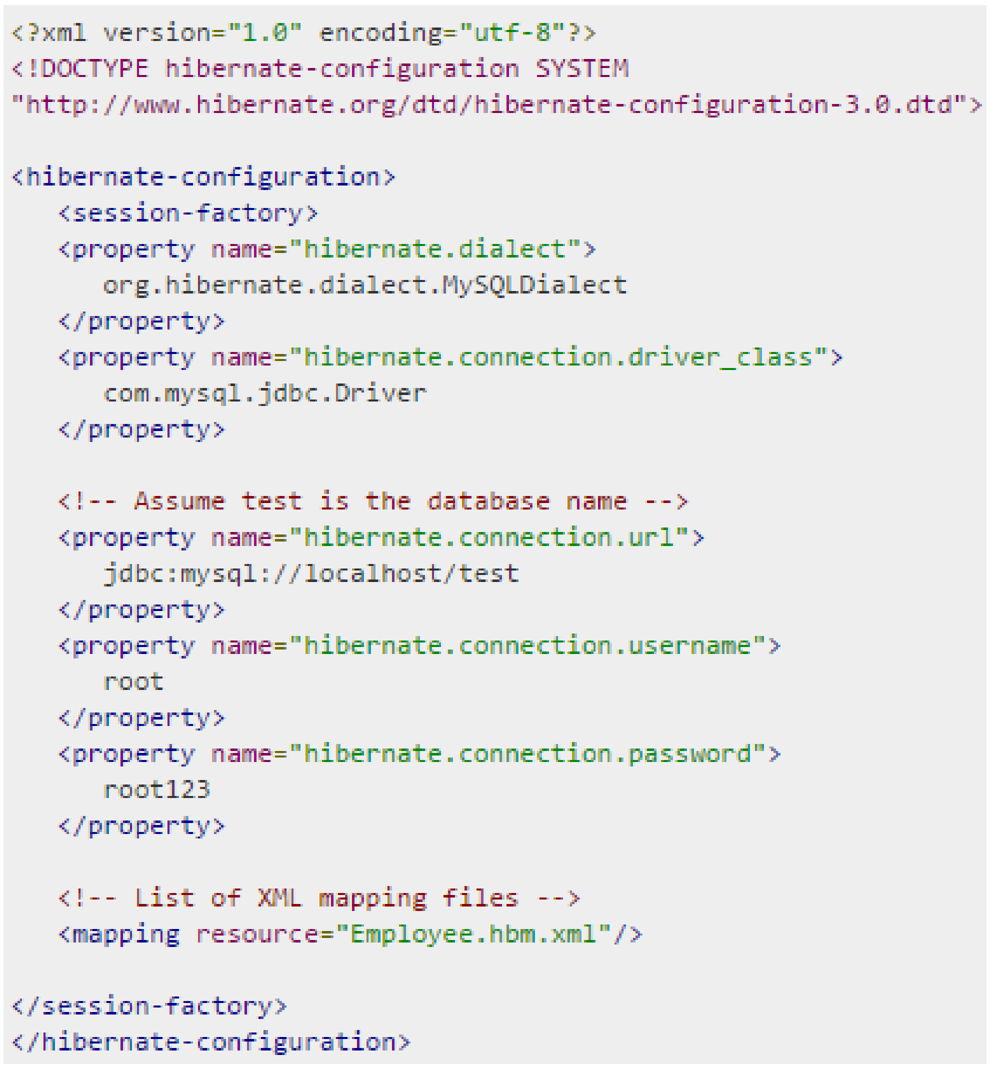
```xml
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
 "-//Hibernate/Hibernate Mapping DTD//EN"
 "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
   <class name="Employee" table="EMPLOYEE">
      <meta attribute="class-description">
         This class contains the employee detail.
      </meta>
      <id name="id" type="int" column="id">
         <generator class="native"/>
      </id>
      <property name="firstName" column="first_name" type="string"/>
      <property name="lastName" column="last_name" type="string"/>
      <property name="salary" column="salary" type="int"/>
   </class>
</hibernate-mapping>
```

Annotations, which were brought in by newer versions of Hibernate scraps the ideas of these XML files and makes things alot easier. Instead of mapping out every member variable for a class you can set up mapping now with a few simple annotations



```
import javax.persistence.*;

@Entity
@Table(name = "EMPLOYEE")
public class Employee {
   @Id @GeneratedValue
   @Column(name = "id")
   private int id;

   @Column(name = "first_name")
   private String firstName;

   @Column(name = "last_name")
   private String lastName;

   @Column(name = "salary")
   private int salary;

   public Employee() {}
   public int getId() {
      return id;
   }
   public void setId( int id ) {
      this.id = id;
   }
   public String getFirstName() {
      return firstName;
   }
   public void setFirstName( String first_name ) {
      this.firstName = first_name;
   }
   public String getLastName() {
      return lastName;
   }
   public void setLastName( String last_name ) {
      this.lastName = last_name;
   }
   public int getSalary() {
      return salary;
   }
   public void setSalary( int salary ) {
      this.salary = salary;
   }
}
```

Hibernate also has a configuration file which must be set up. This holds connection information for the database and other settings.



```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-configuration SYSTEM
"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
   <session-factory>
   <property name="hibernate.dialect">
      org.hibernate.dialect.MySQLDialect
   </property>
   <property name="hibernate.connection.driver_class">
      com.mysql.jdbc.Driver
   </property>

   <!-- Assume test is the database name -->
   <property name="hibernate.connection.url">
      jdbc:mysql://localhost/test
   </property>
   <property name="hibernate.connection.username">
      root
   </property>
   <property name="hibernate.connection.password">
      root123
   </property>

   <!-- List of XML mapping files -->
   <mapping resource="Employee.hbm.xml"/>

</session-factory>
</hibernate-configuration>
```

Originally experimenting and learning was done using version 3 of Hibernate and used Hibernates Object Relational Mapping using XML files. These files are in typical XML format. To use this method of mapping must define the mapped files in the hibernate configuration file, if not when you try to commit the object to the database an exception is thrown. Furthermore any errors in the XML or mistypes in the mapping will stop any operations using this object.

We were successful with implementing some basic CRUD methods (Create,Read,Update and Delete). This required alot of manual configuration with the XML files. The Eclipse IDE which we used allows you to generate these hbm.xml files but they were not always correct. You must take into account that when generating these xml mapping files Eclipse has no knowledge of the database. If for whatever reason you have configured your column names in the database differently you will have configure them manually for every POJO with mapping in your project. Due to this it was best to keep field names in the POJO classes the same in both the project and the database. Hibernate will also try to generate keys and guess which field is key in the POJO or not even find the key at all. Usually the autogeneration just looked for a field called ”id”. Due to this you'll have to manually configure the id and it's relationship. Another major overhead of autogenerating these files is having to generate a mapping file for each class dependant on the current class you are generating the mapping for. The more relations an object has, the more files you have to configure and this was very tedious. The margin for error was high, autogeneration lead to mistakes and overall it was very rigid. This lead to very slow workflow because the IDE wouldn't return the location of errors in these files and would leave you guessing alot of the time. This experience stood to us as we now knew how to commit the objects to the database correctly. Once we had information going into the database and coming back we went back into another research phase.

Upon researching more we found alot of information about using annotations with Hibernate and that in recent version they had removed away from the traditional method of mapping using the hbm.xml files. In the case where future changes might be made to the work we have worked on it was best to research annotations more as it seemed it could speed up development and remove the need for those mapping files. We moved up a version of Hibernate and discovered Hibernate JPA Annotations. While we succeeded in getting this XML mapping working Hibernate Annotations drastically changed the workflow. Annotations removed the need for hbm.xml files and you could simply add simple annotations to the POJO class.

Creating the actual database from these classes had to be done manually. Hibernate does have the capacity to do this automatically based off the current classes but this lead to issues. The particular component was ”hibernate.hbm2ddl.auto” this is called in the spring.xml file which serves as configuration file for server connection strings and to name the annotated or mapped classes. The reason this wasn't used was because it would force the generation of tables for classes we didn't use and forced the generation of tables for interfaces strangely. Some of these tables weren't needed so the tables that Hibernate generated for us were used as a guide more than anything. Using Hibernate we also found that Hibernate would look for those interface tables, this really complicated things, we only required a minimal amount of tables in the database. Due to this the decision was taken to ignore some of the composition with the POJO's and manage them manually. So in our case the class DefaultOrientationService held a set of Reviews in a ReviewSet. It was much cleaner to assign a review set id to a review and store the reviews individually that way. This solution removed the need for all these interval tables in the database. The idea was to keep our footprint on the database as light as possible. In the end we reduced the amount of tables needed in the database down too four relatively simple tables.

4.3 HTML and CSS, Javascript/jQuery

We didn't use much HTML and CSS because the sidebars and navigation bars around the content would later be updated to the style created by RoamPA's team. A simple Bootstrap was used as a template and was later removed. All the functionality of the map was achieved using Google Maps Places API. This was all done through JavaScript. Anything that needed to be done by the back-end was sent through to the proper MVC routes on the server controller by using JavaScriptt.

Chapter 5

System Design

Chapter 6

System Evaluation

As many pages as needed.

- Prove that your software is robust. How? Testing etc.
- Use performance benchmarks (space and time) if algorithmic.
- Measure the outcomes / outputs of your system / software against the objectives from the Introduction.
- Highlight any limitations or opportunities in your approach or technologies used.

Chapter 7

Conclusion

About three pages.

- Briefly summarise your context and objectives (a few lines).
- Highlight your findings from the evaluation section / chapter and any opportunities identified.

Bibliography

[1] "81 Percent of online holiday shoppers read online customer reviews, Nielsen Online." `http://www.nielsen.com/content/dam/corporate/us/en/\newswire/uploads/2008/12/pr_081218.pdf`. December 18, 2008.

[2] "The history of word of mouth marketing.." `http://www.thefreelibrary.com/Thehistoryofwordofmouthmarketing.-a0134908667`. Date: Jul 1, 2005.

[3] "Tourism and mobile technology." `http://www.ecscw.org/2003/018Brown_ecscw03.pdf`, 2003.

[4] R. Zhang and T. Tran, "A helpfulness modeling framework for electronic word-of-mouth on consumer opinion platforms," May 2011.

[5] "Word-of-Mouth Processes within a Services Purchase Decision Context." `http://jsr.sagepub.com/content/3/2/166.abstract`.

[6] C. López and R. Farzan, "Analysis of local online review systems as digital word-of-mouth," 2014.

[7] F. S. . G. L. U. Yakov Bart, Venkatesh Shankar. `http://ebusiness.mit.edu/urban/papers/are\%20the%20drivers%20and%20role%20%28jppm\%202005%29.pdf`. Date: October 2005, Journal of Marketing.

[8] R. Krestel and N. Dokoohaki, "Diversifying product review rankings: Getting the full picture," 2011.

[9] A. Ghose and P. G. Ipeirotis, "Designing novel review ranking systems: Predicting the usefulness and impact of reviews," 2007.

[10] R. Sipos, A. Ghosh, and T. Joachims, "Was this review helpful to you?: It depends! context and voting patterns in online content," 2014.

[11] "Automatically assessing review helpfulness. In Proceedings of the Conference on Empirical Methods in Natural Language Processing.." `http://www.aclweb.org/anthology/W06-1650`. Date: July 2006.

[12] A. Ammar and D. Shah, "Efficient rank aggregation using partial data," June 2012.

[13] E.-P. Lim, V.-A. Nguyen, N. Jindal, B. Liu, and H. W. Lauw, "Detecting product review spammers using rating behaviors," 2010.

[14] R. Centeno and R. Hermoso, "When opinion request meets majority search: Avoiding fraud in on-line review systems," 2015.

[15] A. Dubey and A. Kumar, "A technique for summarizing web reviews," 2008.

[16] T. Lappas, M. Crovella, and E. Terzi, "Selecting a characteristic set of reviews," 2012.

[17] J. Jin, P. Ji, and Y. Liu, "Product characteristic weighting for designer from online reviews: An ordinal classification approach," 2012.