**IFAF Rules Change Proposal and Voting System**

[up746804]

[up746804@myport.ac.uk](mailto:up746804@myport.ac.uk)

*School of Computing, University of Portsmouth, Buckingham Building, Lion Terrace, PO1 3HE*

1. Introduction

The IFAF allows its members to propose change to the rules of the game. This project aims to provide a proposal submission and voting system. The system will have some level of complexity in terms of its members access level and their roles. This report details the development process of this newly proposed IFAF Rules Change Proposal system, from the design and identifying the specification to the technical development process. The document discusses issues encountered during the development process and solution provided to them.

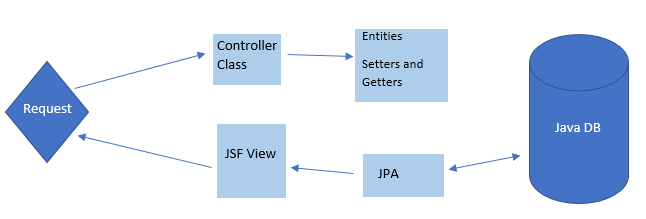
1. Design

The system to be developed has similar functionalities to many social and forum sites, i.e. post, comments and votes (likes). We have kept that in mind when designing the application.

The application is designed to use a template so that it keeps the same appearance throughout. Functionalities based on access level are disabled or abled. For instance, buttons and forms that enables user control are only rendered if the user role matches what is specified.

Information flow is partially through the entity bean and façade CRUD methods and partially through connecting to the server directly. This is because the façade provides easy way of interacting with the database server without writing low level scripts.

Diagram showing flow of information



For more one design, please refer to the design document.

1. Implementation and Testing

**3.1 Development tool**

We are required to use Java EE to complete the development. Therefore, we have used Netbeans 8.1 IDE and Java Web application connected to Glassfish 4.1 server. The web technologies used are JSF and MVC 2. We have found that the tools save development time and tedious code writing as most of the necessary code is created by Java EE. One essential feature in regard to this is the no SQL approach. Once connected to relational database, the application allows for the creation of entities and facades. These files create the necessary SQL script for CRUD operation. Further, the Façade files have access to the relationship between the entities, and therefor can select related records using getCollection method. This is especially useful when dealing with Proposals and their related comments and votes, and makes writing join SQL statements obsolete.

One drawback of using this system is that the learning curve can be steep, especially if coming from other area of development frameworks such as .NET or PHP.

**3.2 Issues encountered**

Implementation of the system started with creating the database using Java DB compatible script. One of the requirements was for the application to create the database automatically when it first run. In the past, using PHP and SQL script, we have managed to create an application that checks if the database exists when the user try’s to log in. If the database doesn’t exists, the application connects to the server and run an SQL script to create database and its tables, and populate the database with initial data before proceeding with the first login. After extensively researching how to do this in the persistence API, we have learned that although the persistence can create tables and populate them, it cannot connect to the server and create the database. The creation of the database for this system therefore must be done manually by creating the schema first, then running the provided SQL script to create and populate the tables.

One area of development we found difficult was understanding the servlet in relation to JSF or JSP. When using JSP, servlet can be configured and used to maintain data access throughout, whilst FaceServlet is used for JSF. Understanding the difference and accessing the required data, for instance using the getCollection method was an issue. This was particularly necessary functionality in order to display proposals with right comments and votes.

Persisting session through the application and accessing user information was also an issue we have encountered. When trying to access logged in user name, we have consistently run into exception ‘java.lang.NullPointException’. This is an issue we have not been able to resolve.

The development of the application required the use of JSF as opposed to JSP. JSF required us to use data tables to retrieve and display data from the database. This caused an issue where by the data displays in tabular format only. We were unable to add or style columns, or add row to the table, hence the application in its current state does not resemble the design. Even with the use of PrimeFaces, we were unable to find a solution. JSF is used by many large organisations like eBay and BMW, which lead us to the conclusion that there is a solution to this issue that we are yet to learn.

Two areas of developments that were technically challenging were implementing specification 2e, storing and accessing user last logged in time, and specification 8, set vote closing time. Whilst setting the last logged in time is not difficult (change time stamp when the user logs in) determining the user online status was in terms of having the system check if the user is still using the application. We could not find solution for the later issue, which requires the system to check the vote closing date and time and close the proposal for voting.

Using the edit method of the Façade to populate the edit form was also an issue that took much of our development time to correct. To resolve this issue, we have changed to using PreparedStatement method to directly select and populate edit form. This has caused another issue whereby the select statement returns null as the parameter passed is null.

Accessing the data from the managed beans was the main issue we encountered. As a result, the application as it stands does not have access to a session. This means the application does not remember users once they have logged in to track and associate activities with the account. Other functionalities that depends on the session are rendering buttons and forms that makes functionality available based on the user role.

Some of the development issues we have found to be particularly tricky are;

* persistence issues in creating database automatically,
* Specification 8. - setting timer for voting and automatically disable vote button after voting time is expired.
* Specification 2e – Creating and accessing user log status (last logged in time)
* Choosing between JSF and JSP as JSF is harder to style, therefore makes it hard to display data in a required place and format.
* Understanding servlet and how they work, JSF with FaceServlet or JSP with servlet?
* Getting comments and votes using the getCollection method
* Rendering buttons based on user role

**3.3 Testing**

So far, we component of the application is tested as ii is being created and developed. In the initial data, we have a range of users with different roles. That has enabled us to test the functionality of each role. The application in its current form is not fully complete, therefore final testing is not complete.

User testing is partially completed using new user to the system. However, due to its incompleteness, the users have found it hard to use and follow. One important outcome from this is the tabular form that the data is being displayed in.

Other important test such as load testing using load generator is to be completed once the application is ready for deployment. In its current form, only development test is complete.

**Advice**

In hindsight, we have found that understanding how the technology works during the lecture period is the best of insuring the deliverable of a complete application. Understanding the relationship between entities, facades, servlets and managed beans is essential, even more than how to write code for functionalities. In an MVC structure, information gets passed between these files, therefore it is important to understand the relationship between the file.

Making the right design choice and tools choice in the first around will save a great deal of development time. Wrong file naming and wrong web.xml settings has forced me to abandon an application and start new one. To avoid this, the design of the application as well as the specified structure must be applied first time.

1. Summary

When first creating the application, the specified file naming and structure were not followed. This has resulted in the need to create a new application with the right structure applied. Although we have successfully recreated the application, this has led to us losing a great deal of development time that could’ve been used in completing the application.

Although there is no bad design decision we have mad, we have found it hard to implement what is designed mainly as a result of using data tables as opposed to tables. In hindsight, using JSP would’ve helped us understand the process easily and helped us produce a more complete application.

An alternative to using Netbeans and Java EE would be to use ASP.NET and C#. As we have some experience using this technology, it would’ve be a fitting alternative. However, we have found Netbeans and Java EE to be much more developer friendly as most of the tedious code are automatically created by the system. This is not the case with ASP.NET.

1. References

* Jendrock, E., Cervera-Navarro, R., Evans, I., Haase, K., & Markito, W. (2014). The java ee 7 tutorial (Vol. 1). Addison-Wesley Professional.
* Heffelfinger, D. R. (2014). Java EE 7 with GlassFish 4 Application Server. Packt Publishing Ltd.
* Apte, A. A., & Acker, L. E. (2001). U.S. Patent No. 6,269,373. Washington, DC: U.S. Patent and Trademark Office.
* Hunter, J., & Crawford, W. (2001). Java Servlet Programming: Help for Server Side Java Developers. " O'Reilly Media, Inc.".
* Jamroendararasame, K., Susuki, T., & Tokuda, T. (2002, January). A diagram approach to automatic generation of JSP/servlet Web applications. In Proc. of the 6th IASTED International Conference on Software Engineering and Applications (pp. 292-297).
* Englander, R. (1997). Developing JAVA beans. " O'Reilly Media, Inc.".
* Roman, E., Sriganesh, R. P., & Brose, G. (2004). Mastering enterprise javabeans. John Wiley & Sons.