Integrating collaboration tools with SDA

Faculty Advisor
Dr. Perez Davilla

MENTOR
Mr. Abbasi Dhilawala
Software Engineer, Tietronix Software Inc

TEAM MEMBERS
Omkar Wadkar
Prachi Phatak
Adebiyi Olukayod
Zhen-Yu Fang

SPRING 2009
April 22,, 2009
**ABSTRACT**

SDA is a web-based portal that provides comprehensive and customizable tools to guide software professionals through various software processes and maintain quality software. SDA is built on advanced J2EE technologies like EJBs, Struts, Servlets and JSP. The user interface framework of SDA is based on Liferay, a leading open source portal. Liferay supports most of the application servers, databases and runs JSR-168 compliant portlets. The project aims to enhance the existing features of SDA by integrating collaborating tools like a rich text editor and staging area. The rich text editor which supports all the rich text including tables, images and bullets etc. And also develop a staging area so that only authorized users can modify the content at run time. These tools should be open source, java based, support multi databases and have a good user interface. After extensive research and investigation the best tool is selected for integration and deployed into the system. The existing system doesn't have an editor and also a staging area so the mechanism of updating an instructions and background for a process is quite time consuming and takes lot of efforts. The team have devised an ingenious way to solve both the above mentioned problems. The team has not only integrated a rich text editor but also developed a staging area with the help of databases. By enhancing these features now it’s possible for an authorized user to modify the content and save it as well. He can publish that content after finished editing. By resolving these issues the usability and functionality of the SDA tool is enhanced.
# TABLE OF CONTENTS

1.0 INTRODUCTION .............................................................................................................................................. 4

2.0 USE CASE

2.0 EXISTING SYSTEM ........................................................................................................................................... 5

2.1 REQUIRED SYSTEM ......................................................................................................................................... 6

3.0 TECHNICAL DETAILS

3.1 USER INTERFACE REQUIREMENT FOR RICH TEXT EDITOR ................................................................. 7

3.1.1 USER INTERFACE DEVELOPMENT FOR RICH TEXT EDITOR .......................................................... 8

3.2 USER INTERFACE REQUIREMENT FOR STAGING AREA ........................................................................... 9

3.2.1 USER INTERFACE DEVELOPMENT FOR STAGING AREA ............................................................... 9

4.0 SERVER SIDE REQUIREMENTS AND DEVELOPMENT

4.1 SERVER SIDE REQUIREMENTS ......................................................................................................................... 12

4.2 SERVER SIDE DEVELOPMENT ....................................................................................................................... 12

4.2.1 DATABASE DEVELOPMENT ...................................................................................................................... 12

4.2.2 CLASS DEVELOPMENT ............................................................................................................................ 14

4.2.2.1 ACTION CLASS ................................................................................................................................ 14

4.2.2.2 MANAGER CLASS ............................................................................................................................... 15

4.2.2.3 CONTENT MODEL CLASS .................................................................................................................. 16

4.2.2.4 CONTENT DAL CLASS ....................................................................................................................... 17

5.0 LESSON LEARNED .......................................................................................................................................... 18

6.0 CONCLUSION ....................................................................................................................................................... 18

7.0 REFERENCES ...................................................................................................................................................... 18
1. Introduction:

SDA is an engineering process automation tool that already has built in support for software development. The tool is web based and is geared towards team development. SDA is built on J2EE technologies like EJBs, Struts, Servlets, JSP and more. The SDA UI framework is built upon a leading open source portal called Liferay. This project will enhance the SDA’s features by adding a content management tool to it.

SDA has the complete information about all the processes instruction and background belonging to a particular package. But in the current system there is no way an authorized user can modify the content of the process. The mechanism used currently works in an indirect manner where user sends the modifications to be made to a company person who converts it into an XML file and then updates the system. This process is very much time consuming and takes lots of efforts to implement.

The team has devised a technique that allows users to modify the content directly. And also the rich text editor is integrated which enables wide variety of features like Text, URL, Table, Images, Bullets. To be inserted into the content, the document further explains all the technical details of how exactly it has been performed. The team was divided and then worked on two different backgrounds. One started with server side development and the other started working on the user interface.
2.0 Use Case: Existing System

The following use case help understand the requirements.

**After publish**
2.1 Use Case: Required System

After publish
3.0 Technical details

As mentioned in introduction the team has started development from two different backgrounds. The team was divided and one team started with user interface development while other started with server side development. Following are the tasks that are completed by both the teams. We will look all the technical details with respect to the given requirements followed my team approach to fulfill the requirements.

3.1 User Interface Requirements for rich text editor

The requirements that were given to the team are as following

- The team will either select an open source content management tool or develop one for the integration.
- The tool shall support the following contents
  - Text
  - URL
  - Images
  - Tables
  - Bullets
- The tool shall display the managed content to other users in a read only format.
3.1.1 User Interface Development for rich text editor

The first team initiated their development by searching for a rich text editor online. And tested it for all the requirements given. and the team successfully tested the rich text editor as well.

The following screen shot shows the rich text editor found.

As we can see that all the given requirements are being satisfied. The next task is to integrate this editor with an SDA.
3.2 User Interface requirements for Staging area

The requirements for the staging area are as following

- The content management tool shall have the following features
  - The tool shall store the content in a database (preferably works with multiple leading database management tools like SQL Server, MySQL, Oracle etc)
  - The tool shall allow certain users with the appropriate right to create/modify the content with a rich text editor

3.2.1 User Interface Development for staging area

The development of staging area was critical task. The team is suppose to create and entirely new page. This new page will display all the packages and upon selecting a package the processes and sub processes of the selected package will be shown. The team went ahead with the step by step process which is explained further with the help of screen shots.

Step 1: Add like in process admin tab called “Edit Instruction & Background”
After clicking the “Edit instruction & Background” link

**Step 2:** Open a new tab in process Admin which has list of packages.

![Image of SDA Administrator interface with Process Admin and Edit Instruction and Background]

**Step 3:** upon selecting a package all the processes and activities should be displayed.

![Image showing same interface with selected package and list of activities]
Step 4: After clicking the edit button the new window pops up editor with the content to be modified inside it. The editor has three buttons “save “, “save and publish”, & “save and close”. The user can edit either instruction or background.

4.0 Server side requirements and developments

The server side development was quite complicated and challenging task to perform because the team is suppose to understand the struts architecture that SDA follows. And then build their modules according to the struts hierarchy as following.
4.1 Server Side Requirements

The server side requirements are as following

- The tool shall store the content in a database (preferably works with multiple leading database management tools like SQL Server, MySQL, Oracle etc)
- The tool shall allow certain users (with the appropriate right) to create/modify the content with a rich text editor

4.2 Server side development

The given requirements are tackled using two different approaches of development. The first one takes care of the database which is mentioned here (our team is going to use MySQL Server 5.0). and the second approach is to develop classes to invoke methods. We will discuss both the methods in much detail.

4.2.1 Database development

In order to create a staging area the team came up with a simple design of database. where team has create two different data base tables to store the saved and published data separately. Following are two database schemas that team has developed
Database schema:

- **PublishedContentInfo** –

<table>
<thead>
<tr>
<th>ColumnName</th>
<th>DataType</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Varchar[24]</td>
<td>Not Null</td>
<td>Record ID</td>
</tr>
<tr>
<td>contentId</td>
<td>Varchar[255]</td>
<td>Not Null</td>
<td>BusinessKey</td>
</tr>
<tr>
<td>contentVersion</td>
<td>Integer</td>
<td>Not Null</td>
<td>Version of Content</td>
</tr>
<tr>
<td>contentType</td>
<td>Varchar[50]</td>
<td>Not Null</td>
<td>Background or Instruction</td>
</tr>
<tr>
<td>content</td>
<td>Text</td>
<td>Not Null</td>
<td>Content</td>
</tr>
<tr>
<td>creationDate</td>
<td>Date/Time</td>
<td>Not Null</td>
<td>Creation Date of Content</td>
</tr>
<tr>
<td>author</td>
<td>Varchar[50]</td>
<td>Not Null</td>
<td>Author of process</td>
</tr>
<tr>
<td>lock</td>
<td>Boolean</td>
<td>Not Null</td>
<td>Content Lock</td>
</tr>
</tbody>
</table>

- **WorkingContentInfo** –

<table>
<thead>
<tr>
<th>ColumnName</th>
<th>DataType</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Varchar[24]</td>
<td>Not Null</td>
<td>FK_WorkingContentInfo_PublishedContentInfo/id</td>
</tr>
<tr>
<td>creationDate</td>
<td>Date/Time</td>
<td>Not Null</td>
<td>Creation Date</td>
</tr>
<tr>
<td>author</td>
<td>Varchar[50]</td>
<td>Not Null</td>
<td>Current author of Process</td>
</tr>
<tr>
<td>content</td>
<td>Text</td>
<td>Not Null</td>
<td>Working Content</td>
</tr>
</tbody>
</table>

The working content info stores the data which is being saved but not published and the published content info stores all the content that has been published.
4.2.2 Class development

The classes are part of SDA architecture and they have to follow the struts architecture. As we can see in section 4.0 the layering system in SDA. If we observe carefully we notice that there is a class driving a particular layer and that’s why team has to develop and modify following classes in order to run the mechanism successfully.

- Action class
- Manager class
- Content DAL Class
- Content model class

4.2.2.1 Action Class

Action class contains all the functions that are invoked for every action. For example actions like View, Edit etc. The action class drives the controller layer in SDA.

<table>
<thead>
<tr>
<th>Action class</th>
</tr>
</thead>
<tbody>
<tr>
<td>String UserId</td>
</tr>
<tr>
<td>String ProcessId</td>
</tr>
<tr>
<td>ViewContentAction()</td>
</tr>
<tr>
<td>EditContentAction()</td>
</tr>
<tr>
<td>SaveContentAction()</td>
</tr>
<tr>
<td>PublishContentAction()</td>
</tr>
</tbody>
</table>
4.2.2.2 Manager Class

The manager class contains all the function which drives the business logic for an SDA. The manager class is placed between the DAL class and actions class. So it controls the flow of information from DAL class by checking privileges and checking authorizations. The team has designed the Manager class as following:

<table>
<thead>
<tr>
<th>MANAGER</th>
</tr>
</thead>
<tbody>
<tr>
<td>String UserId</td>
</tr>
<tr>
<td>String packageName</td>
</tr>
<tr>
<td>String processName</td>
</tr>
<tr>
<td>String version</td>
</tr>
<tr>
<td>String id</td>
</tr>
<tr>
<td>String contentId</td>
</tr>
<tr>
<td>String contentType</td>
</tr>
<tr>
<td>Arraylist getEditableProcessesForUser()</td>
</tr>
<tr>
<td>Arraylist getProcessHierarchy()</td>
</tr>
<tr>
<td>ContentModel getContentById()</td>
</tr>
<tr>
<td>ContentModel getLatestVersionContent()</td>
</tr>
<tr>
<td>Boolean publishContent()</td>
</tr>
</tbody>
</table>
4.2.2.3 Content Model Class

As we can observe from the layering structure the content model class is the one which has all the setter and getter functions. The content model class work at all level. Team has developed it in following manner.

<table>
<thead>
<tr>
<th>Content model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Private String id;</td>
</tr>
<tr>
<td>Private Int contentVersion;</td>
</tr>
<tr>
<td>Private String contentType;</td>
</tr>
<tr>
<td>Private String content;</td>
</tr>
<tr>
<td>Private Date creationDate;</td>
</tr>
<tr>
<td>Private String author;</td>
</tr>
<tr>
<td>Private Boolean locked;</td>
</tr>
</tbody>
</table>

```
ContentModel(String contentId, Int contentVersion, String contentType) throws IllegalArgumentException {}

ContentModel(String id, String contentId, Int contentVersion, String contentType) throws IllegalArgumentException {}

ContentModel(String id, String contentId, Int contentVersion, String contentType, String content, Date creationDate, String author, Boolean locked) throws IllegalArgumentException {}

// setting & getting function
```
4.2.2.4 Content DAL Class

The content DAL class contains all the queries to fetch data from a database like INSERT, UPDATE, and DELETE. The queries are fired depending on the kind of operation being performed. Sometimes there is a combination of two or three different queries that are fired. The content DAL which the team came up with is as following.
5.0 lessons learned

While developing this project we have gathered a huge experience. We learned how to work as a team in a better way. We also learned to schedule our work so that any task should get finished within given time limit. While working with the team, we learned how to distribute the task among the team mates so that the all team member will get the equal share. We also experienced the professional approach for development and felt really glad that we got this opportunity during academic period. We arranged lots of meetings and that a good way of
conveying thoughts and which helped everyone to know what he/she exactly doing. It was a good learning curve for all of us.

6.0 Conclusion

We have completed all our requirements and increased the usability by providing enhancements like rich text editor and staging area. The objectives and requirements of the project are met.

7.0 References
