

University of Massachusetts Medical School
eScholarship@UMMS

Population and Quantitative Health Sciences
Publications

Population and Quantitative Health Sciences

2008-4


A Web-based interactive Student Advising system using Java frameworks

V. R. Naini
University of Alabama - Birmingham

Et al.

Let us know how access to this document benefits you.

Follow this and additional works at: https://escholarship.umassmed.edu/qhs_pp

 Part of the [Biostatistics Commons](#), [Computer Engineering Commons](#), [Epidemiology Commons](#), and the [Health Services Research Commons](#)

Repository Citation

Naini VR, Sadasivam RS, Tanik MM. (2008). A Web-based interactive Student Advising system using Java frameworks. Population and Quantitative Health Sciences Publications. <https://doi.org/10.1109/SECON.2008.4494281>. Retrieved from https://escholarship.umassmed.edu/qhs_pp/864

This material is brought to you by eScholarship@UMMS. It has been accepted for inclusion in Population and Quantitative Health Sciences Publications by an authorized administrator of eScholarship@UMMS. For more information, please contact Lisa.Palmer@umassmed.edu.

A Web-based Interactive Student Advising System using Java Frameworks

V. R. Naini, R. S. Sadasivam, and M. M. Tanik
Electrical and Computer Engineering Department
The University of Alabama at Birmingham

Abstract

The use of open source frameworks and tools has become popular in Java development. These frameworks and tools have core strengths and weaknesses and are selected accordingly for development. Consequently, one of the key issues that developers face is to integrate and configure these tools together. This paper demonstrates the use of popular Java frameworks and tools to develop a Web-based interactive Student Registration and Advising system.

1. Introduction

One of the problems in developing large scale applications with Java 2 Enterprise Edition (J2EE) is the complexity associated with it. Open source frameworks, such as Spring, and Java Server Faces (JSF), and object relational mapping tools, such as Hibernate, have been developed to reduce the complexity of developing large applications with J2EE. These open source frameworks and tools have core strengths and weaknesses. Therefore, one of the issues in developing large systems with these frameworks is to integrate and configure these frameworks and tools together to leverage their strengths. This paper demonstrates the use of Spring, JSF, and Hibernate together to develop a Web-based Interactive Student Registration and Advising System (WISRAS).

The current registration process for graduate students in the Electrical and Computer Engineering Department is a paper-based process. The Masters of Science in Electrical Engineering (M.S.E.E) course plan is available as a downloadable Portable Document Format (PDF) and Microsoft Word document at the Electrical and Computer Engineering Web site. The graduate students have to fill this course plan document and send it to their advisor as an email attachment for the advisor's approval. The advisor then approves and signs the course plan. To proceed further with the registration process, the student should submit the approved document to the department. This paper-based

registration process is tedious and has several drawbacks, such as requiring the physical presence of the advisor to approve the course plan document and give it back to the student.

The WISRAS simplifies the registration process of the graduate students by providing an electronic and interactive registration process. In the WISRAS process, the student fills the course plan page on the intranet site of the department and submits it online, which generates an email confirmation of the course plan submission to both the student and the advisor. The advisor then checks the course plan of the student on the intranet site and electronically approves it, generating an email confirmation of approval to the student and the department.

The WISRAS is developed in Java and using open source frameworks. JSF is used for the front-end design to handle the presentation issues of the course plan page. Hibernate is used to handle the data retrieval from the database. Spring is used to handle the flow of execution that is integrating different parts of the application such as front-end and back-end.

2. Frameworks used for Development

2.1 JavaServer Faces

JavaServer Faces (JSF) simplifies development by providing a component-centric approach for developing Java Web user interfaces [1], [2]. JSF ensures that applications are well designed with greater maintainability by integrating the well established Model-View-Controller (MVC) design pattern into its architecture.

2.2 Spring

The Spring framework provides a full-featured MVC module for building Web applications [3]. Spring's pluggable MVC architecture provides the option of choosing between using the built-in Spring Web framework or a Web framework such as JSF.

2.3 Hibernate

Hibernate provides a powerful object/relational persistence and query service for Java [4]. Hibernate allows development of persistent classes using object-oriented concepts, such as association, inheritance, and polymorphism. Hibernate handles the mapping from Java classes to database tables and also provides data query and retrieval facilities.

3. Description of Student Advising System

3.1 The WISRAS System

The WISRAS has a common login page for students and advisors as shown in Fig. 1.

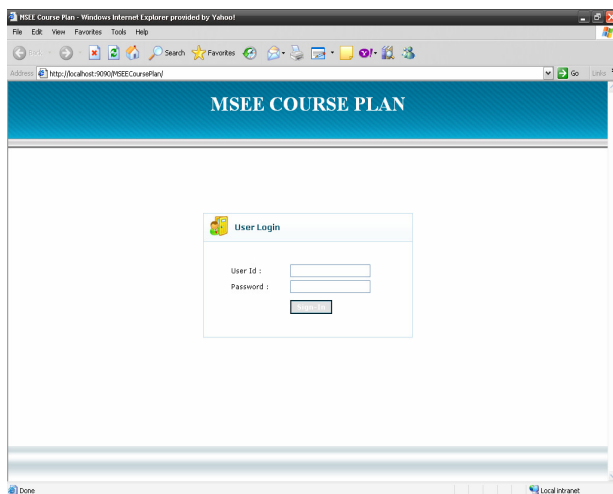


Fig. 1. Login Page

After successful login, the student is directed to the student home page (Fig. 2) where there are three options to choose from: New plan, view plan, and edit plan. New plan option allows the student to create a new plan, view plan allows the student to view the registered plan, and edit plan allows the student to edit the course plan for future semesters. The student can also edit the courses of the current semester if the advisor does not approve a particular course in the course plan.



Fig. 2. Student home page

New plan option directs the student to the M.S.E.E course plan page as shown in Fig. 3(a) and Fig. 3(b). This course plan page consists of four main categories, which represent specialization courses, related courses, math courses, and a thesis/non-thesis selection section. The requirement for the masters program is satisfied with a thesis option selection and eight courses or a project option selection with ten courses. Thesis requirements equal to nine credit hours. A project equals three credit hours.

MSEE Plan

Choose Your Specialization | Select Specialization

EE Specialization Courses		12 Hours		Plan		Completed	
Course	Title	Hours	Grade	Term	Year	Term	Year
Choose Your Specialization first	Choose Your Specialization first	3		Spring	2006	Spring	2006
Choose Your Specialization first	Choose Your Specialization first	3		Spring	2006	Spring	2006
Choose Your Specialization first	Choose Your Specialization first	3		Spring	2006	Spring	2006
Choose Your Specialization first	Choose Your Specialization first	3		Spring	2006	Spring	2006

Related Courses		6 Hours		Plan		Completed	
Course	Title	Hours	Grade	Term	Year	Term	Year
Choose Your Specialization first	Choose Your Specialization first	3		Spring	2006	Spring	2006
Choose Your Specialization first	Choose Your Specialization first	3		Spring	2006	Spring	2006

Maths Courses		6 Hours		Plan		Completed	
Course	Title	Hours	Grade	Term	Year	Term	Year
Choose Your Specialization first	Choose Your Specialization first	3		Spring	2006	Spring	2006
Choose Your Specialization first	Choose Your Specialization first	3		Spring	2006	Spring	2006

Fig. 3(a). M.S.E.E course plan page of the student

Choose Your Specialization | Select Specialization

EE Specialization Courses	
Course	Title
Choose Your Specialization first	Choose Your Specialization first
Choose Your Specialization first	Choose Your Specialization first
Choose Your Specialization first	Choose Your Specialization first
Choose Your Specialization first	Choose Your Specialization first

Fig. 3(b) M.S.E.E course plan page of student. Student can choose the course using the course code or course title

The M.S.E.E course plan page of the WISRAS has a specialization section where the student can select four specialization courses, which address the major requirements for the specialization. The list contains software-computer, power, networks, and telecommunication courses as available specializations. When the student chooses a specialization, data is retrieved from the server using Spring and Hibernate, and the fields are populated with the pertaining courses using JSF. For example, when the student chooses software-computer as the specialization, the courses addressing the software-

computer specialization made available in the specialization course group.

The WISRAS has a math courses section with options for selecting two math courses. Two math courses are compulsory for the master’s degree program. The related courses section requires the student to select two additional courses. The related courses include any of the available courses except those meeting the requirements for the math and the specialization courses.

The thesis/non-thesis section has a radio button group showing thesis and non-thesis options. The thesis option is selected by default. Normally, the thesis is considered as nine credit hours and carried over three semesters. If the student selects the non-thesis option, the section changes, reflecting a table with a project and two related courses. The project option is a text box where the project name can be entered. The two related courses can be selected through the drop-down menus.

After completing the course plan, the student can save the plan by clicking the save button at the bottom of the course plan page. Clicking the save button saves the plan while the student can view, edit, and submit it in the future. If the students choose the save and submit option, the plan is submitted and an email confirmation of the course plan submission is sent to both the student and the advisor.

The student can now view the submitted course plan by clicking on the view plan link in the student homepage. Clicking on the view plan link directs the user to a page that shows the summary of submitted course plan as shown in Fig. 4. Finally, edit plan directs the student to the M.S.E.E course plan and allows the student to edit the courses for the future semesters. The student can also edit the courses of the current semester if the advisor does not approve a particular course in the course plan.

MSEE Plan Details	
Specialization Course 1	SW1
Specialization Course 2	SW2
Specialization Course 3	SW4
Specialization Course 4	SW3
Related Course 1	PW1
Related Course 2	NW2
Related Course 3	TL4
Related Course 4	TL3
Math Course 1	MT2
Math Course 2	MT4
Non Thesis	
Project	MSEE Course Plan

Comments adviser Comments

Submit Plan

Fig. 4. View plan page

Advisors also have an interface to the WISRAS application. As mentioned earlier, the login page is the

same for the students and advisors. The advisor can login into the application with the credentials provided by the administrator.

After the successful login into the application, the advisor is directed to the advisor’s homepage, where a list of students under the advisor is displayed (Fig. 5).

The advisor can click on a particular student link to get the course plan of that particular student (Fig. 5). The advisor can approve or reject the courses selected by the student by checking the approve check box of the courses. There is a comments section to provide comments about the reason for the rejection of a course as shown in Fig. 6. After the advisor’s approval of the course plan, the application automatically generates an email confirmation to the student and the department about the successful approval of the student’s course plan. This confirmation email triggers the department’s administration office to provide the registration code to the student through an email.

Welcome Adv Last

Students Information

- Student 1
- Student 2
- Student 3

Fig. 5. Advisor’s home page

MSEE Plan Details		
Specialization	Software	
Specialization Course 1	SW1	<input checked="" type="checkbox"/>
Specialization Course 2	SW2	<input checked="" type="checkbox"/>
Specialization Course 3	SW4	<input type="checkbox"/>
Specialization Course 4	SW3	<input type="checkbox"/>
Related Course 1	PW1	<input type="checkbox"/>
Related Course 2	NW2	<input type="checkbox"/>
Math Course 1	MT2	<input type="checkbox"/>
Math Course 2	MT4	<input type="checkbox"/>
Non Thesis		
Project	MSEE Course Plan	<input type="checkbox"/>

adviser Comments

Comments::

Update

Fig. 6. Advisor’s view of student’s course plan

The sequence diagrams of the advisor and student interface are shown in Fig. 7 and Fig. 8, which represent the complete flow of the application.

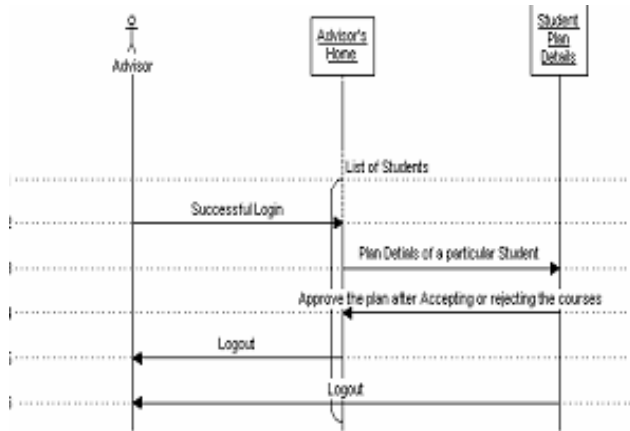


Fig. 7. Sequence diagram of advisor interface

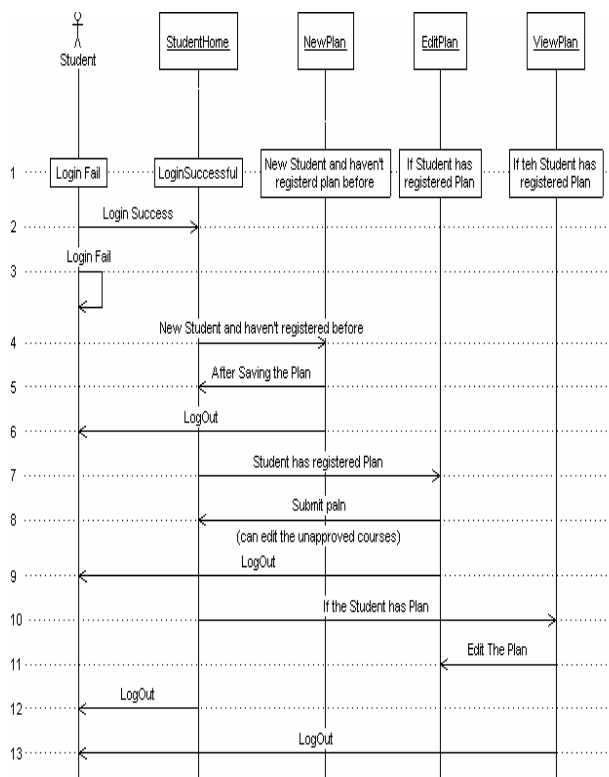


Fig. 8. Sequence diagram of the student interface

3.2 Technical Aspects of the Application

Login.jsp that is written in JSP and JSF is the user login page of the application. Once the user enters their credentials, the *checkValidUser* action verifies the user's credentials and also whether the user is an advisor or a student. If the user is an advisor, then the *checkValidUser* action returns *successAdvisor*. If the user is a student it returns *successStudent*. If the login fails, the user is

redirected back to the *Login.jsp* page with an appropriate message. *Faces-navigation.xml* contains the navigation cases and navigation rules. The advisor is directed to the advisor's home page and the student to the student's home page.

```

<navigation-rule>
  <from-view-id>/view/Login.jsp</from-view-id>
  <navigation-case>
    <from-outcome>successStudent</from-outcome>
    <to-view-id>/view/Home.jsp</to-view-id>
  </navigation-case>
  <navigation-case>
    <from-outcome>successAdvisor</from-outcome>
    <to-view-id>/view/AdvisorHome.jsp</to-view-id>
  </navigation-case>
  <navigation-case>
    <from-outcome>fail</from-outcome>
    <to-view-id>/view/Login.jsp</to-view-id>
  </navigation-case>
</navigation-rule>

```

The student's home page has three links: new plan, view plan, and edit plan. When the student clicks on new plan, the action *plan* is called, which returns the *MSEEPan.jsp* page. If student clicks on view plan link, it will call the method *getPlanDetails()*, which returns the results of the action *viewPlan*.

```

<navigation-rule>
  <from-view-id>/view/Home.jsp</from-view-id>
  <navigation-case>
    <from-outcome>plan</from-outcome>
    <to-view-id>/view/MSEEPan.jsp</to-view-id>
  </navigation-case>
  <navigation-case>
    <from-outcome>viewplan</from-outcome>
    <to-view-id>/view/ViewPlan.jsp</to-view-id>
  </navigation-case>
</navigation-rule>

```

MSEEPan.jsp page has the course plan details. The specialization details are populated while loading the page. The drop-down menu items for specialization selection are Software-computer, Power, Networks, and Telecommunications. On selecting a specialization item in the drop-down menu, the *valueChangeListener* is called and an *onchange* event is triggered and calls the function *pullDownChanged()*.

```

<h:selectOneMenu id="specializationId"
  value="#{MSEEPPlanBean.specializationId}"
  valueChangeListener="#{MSEEPPlanBean.getMenuC
hange}"
  onchange="pulldownChanged()">
<f:selectItem itemLabel="Select
Specialization" itemValue="0" />
<f:selectItems
value="#{MSEEPPlanBean.specializations}" />
</h:selectOneMenu>
<h:commandButton id="myUpdateButton"
value="Submit" action="specialization"
style="visibility:hidden;" />

```

In the javascript function *pulldownChanged()*, a method used to get the button clicked is as follows

```

function pulldownChanged(){
  document.getElementById("plan:myUpdateButt
on").click();
}

```

The related courses and math course drop-downs are populated by retrieving the respective information by calling the methods *getRelatedCourses()*, *getMathsCourses()* through services using the Spring flow and Hibernate. Lastly, the student has to select the thesis or non thesis plan option. By default, the thesis option is selected (counts 9 credits) and the student has to enter the plan details. If the student has selected non- thesis option (project work), it will count as 3 credits and the student has to select two more related courses. *Onchange* event is triggered once the student has selected the non thesis or thesis option. The *show()* and *hide()* functions are used to display the course fields.

```

function show(section) {

(document.getElementById(section)).style.dis
play='block';

}

function hide(section) {

(document.getElementById(section)).style.dis
play='none';

}

```

After selecting the courses, the plan can be saved by clicking the save button, which triggers the action method *savePlan()*.

```

<h:commandButton value="Save Plan"
action="#{MSEEPPlanBean.savePlan}"
immediate="true">

```

The *savePlan()* method retrieves the selected courses details and saves them in the database by using the bean *savePlan()* method in the *CoursesDao* bean.

```

MSEEPPlanBean.savePlan():

public String savePlan() {
  try{
    this.getServiceLocator().getCoursesService
().savePlan(getInfo());
  }catch (Exception e) {
    System.out.println("Exception::::"+e);
  }
  return "success";
}

```

CourseDao.savePlan():

```

public int savePlan(BaseDAOBean bean) {
  CoursePlan plan = (CoursePlan) bean;
  try {
    add(plan);
  } catch (Exception e) {
    e.printStackTrace();
  }
  return plan.getId();
}

```

After the successful saving of the course plan, emails will be generated to both student and corresponding advisor, with the course details and the student name by calling the method *sendMail()* method in *AutoGeneratedMailFactory* class.

After login, the advisor will be directed to the advisor's home page. The advisor can see the list of students under his or her guidance. The students' list gets displayed by using *dataTable* in JSF. Selecting a particular student name redirects the advisor to that student's course plan. When the advisor clicks on a student's name, the action *studentPlan* is called.

```

<h:dataTable
value="#{AdvisorHomeBean.studentInfo}"
var="data">
  <h:column>
    <h:commandLink action="studentPlan">
      <h:outputText
value="#{data.studentFname}" />
      <f:param value="#{data.studentId}"
name="studentId"></f:param>
    </h:commandLink>
  </h:column>
</h:dataTable>

```

The *studentPlan* action redirects to the *StudentPlanDetails.jsp* pages.

```

<navigation-rule>
  <from-view-
id>/view/AdvisorHome.jsp</from-view-id>
  <navigation-case>
    <from-outcome>studentPlan</from-
outcome>
    <to-view-
id>/view/StudentPlanDetails.jsp</to-view-id>
  </navigation-case>
</navigation-rule>

```

In the *StudentPlanDetails* page, the advisor can accept or reject the student's selected course by checking the appropriate check box. The advisor can also enter comments in the student's course plan if there is any problem with the student's selected courses. Once the advisor approves a course, the student cannot edit that particular course any further. When the advisor clicks on the save button, it will call the *saveStudentPlan()* action and the *isAccepted* flag is set for the accepted courses in the database by using the update method in Hibernate. Emails will be sent to both the advisor and student about the changes of the advisor on the course plan by calling *sendMail()* method in *AutoGeneratedMailFactory* class. Based on that email, the student can login into the application and view the accepted courses. The student can also edit the rejected courses and re-submit the course plan which repeats the process for acceptance of the course plan by the advisor.

4. Discussion and Conclusion

The WISRAS was successfully developed using Spring, Hibernate, and JSF with all the main features.

Presently, the student, advisor, and email interfaces of the WISRAS are completed. The next development phase of the WISRAS would result in an application where the students and advisors have secure access to the application. The extension of the project includes adding an interface to the course schedule for all semesters. An interface to the department's administration office for sending the registration code to the student on successful approval of the student's course plan is also planned.

There were some issues in configuring the application to work in different browsers especially Internet Explorer (IE) and Mozilla. Each browser behaves differently for the same methods in JavaScript. Therefore, the debugging was very difficult using JavaScript. The JavaScript console in Mozilla was more helpful for debugging than debugging options in IE. We also plan to benefit from an early development we have implemented in expanding the functionality of the current system [8].

5. Acknowledgement

We thank Dr. Gary J Grimes and Prof. David Green, Electrical and Computer Engineering Department at the University of Alabama at Birmingham, for their support and help during the course of the development of the project.

6. References

- [1] B. Kurniawan, "JavaServer Faces Programming," J. Peters, Ed. New York: McGraw-Hill, 2003, ch. 14.
- [2] C. Schalk, "Introduction to JavaServer Faces," 2005. [Online]. Available:

- <http://www.oracle.com/technology/tech/java/newsletter/articles/introjsf/index.html>. [Accessed Nov. 20 2007].
- [3] K. Donald, E. Vervaeke, "Spring Framework and web Flow," 2007. [Online]. Available: <http://www.springframework.org/webflow> [Accessed Nov. 20 2007].
- [4] P. Peak, N. Heudecker "Hibernate Basics," 2005. [Online]. Available: http://www.developer.com/open/article.php/10930_3559931_5. [Accessed Nov. 20 2007].
- [5] R. Hightower, "Configuring Hibernate, Spring and JSF," 2005. [Online]. Available: <http://www.thearcmind.com/confluence/display/SpribernateSF/Configuring+Hibernate%2C+Spring%2C+OpenInSessionViewFilter+and+MyFaces+JSF>. [Accessed Nov. 20 2007].
- [6] C. Sagi, "An interactive Web application using AJAX," 2005.
- [7] Exadel Training, "JavaServer Faces Tutorial," 2007. [Online]. Available: <http://www.exadel.com/tutorial/jsf/jsftutorial-kickstart.html> [Accessed Nov. 20 2007].
- [8] S. Lanka, M. M. Tanik, and D. Green, "Design of a Distance Education System," IEEE Southeastcon, Lexington, Kentucky, March 25-28, 1999, 129-133.