Advances in Intelligent Systems and Technologies
Proceedings ECIT2004 - Third European Conference on Intelligent Systems and Technologies
Iasi, Romania, July 21-23, 2004

Computer Aided Assessment Project – overview

Eugen Petac, Ph.D.*, Tudor Udrescu*, Marius Atanasiu**

**Department of Informatics and Numerical Methods,*
"Ovidius" University Constantza
epetac@univ-ovidius.ro
tudrescu@univ-ovidius.ro
*** Master in Computational Methods and Modern Information Technologies, "Ovidius" University Constantza*
mariusat@hotmail.com

Abstract. *The problem of assessing the quality of student work is a difficult but important task. This paper describes the details of a project that aims to create a computer aided assessment application for universities. The project will use Struts, which is a Java-based framework for the development of web applications. This application can be best described as a database containing multiple choice questions and used to support various assessment tasks, as well as the learning process by providing feedback based on learning outcomes. While seeking to provide a more efficient process of testing for students, the application will remain highly secure and academically rigorous.*

Keywords. *Online testing, online tutorials, encrypted transmissions, secure environment, Struts framework*

1. Introduction

Student evaluation is of fundamental importance in any university. Tools for the automation of this task can be of real help to the faculty members, but they can also be used to assist individual students to improve the quality of his/her preparation for an exam [6],[8]. There are four interconnected steps that should be taken into consideration when talking about an automated assessment process:

(1) Pre-test to assess the level of student preparation
(2) The online test based on a various number of multiple-choice questions obtained from a database.
(3) Feedback to the student about his performance at the test
(4) Optional presentation of tutorials that aim to remedy the student's poor performance in certain areas of the curriculum

Computer Aided Assessment Project (CAAP) is a project under development at "Ovidius" University that aims to develop software specifically designed to automate various assessment tasks. The software is a web application used primarily for delivering online tests and providing the automated marking and monitoring of students' responses. It will also be used to allow students to access online tutorials specifically tailored to each

student's individualized learning needs as well as other course material. CAAP's design objectives are to support multiple tests, feedback, remote learning and to allow for extensibility and portability. Because it is a web application, CAAP relies on the Model-View-Controller design paradigm. The Model is represented by a database that stores all the information used by the application. Specific functionality includes:

- Provide a secure logon mechanism for each student and for administrators
- Management of multiple tests
- Generate tests with randomly arranged questions for each student
- Display the results at the end of the test
- Provide a mechanism for timing the test
- Automatic or voluntary logoff from the application
- Give the possibility to a student to return to a question and modify the answer
- Generate individual reports for each student and global reports for administrators

The project has established from the beginning the need to use open source technologies in its development. Due to reasons of portability, Java has been chosen as the programming language, MySQL to provide database connectivity and Apache and Tomcat to be the Web server, and Application Server respectively. All these software products can be easily combined into an industrial-strength software product that will impose no additional licensing requirements on the university.

One of the most powerful technologies in web application development is the Apache Software Organization's Struts framework. This is an open source framework based on the Model 2 implementation of Model-View-Controller design patterns, therefore perfectly suited to the needs of our project [4]. The advantage of using the MVC Model 2 paradigm in the development of web applications is that it allows the separation of display code from the flow control logic, in other words there is no Model-related processing within the confines of the View, or presentation component itself [1],[2]. The opposite also holds true; that is, there is no presentation logic in the model layer. This fact allows for an increase in component reusability and a clear delimitation between layers. Consequently, the implementation of one layer can be easily changed without incurring major modifications to the other layers [3].

The core Struts technologies are represented by Java Server Pages and Servlets. The JSP is a server side component that is made up of static HTML and XML elements, tags specific to JSP and optional blocks of Java code called scriptlets. JSP is used as the presentation layer in n-tier web architectures to separate presentation from content. In the Struts framework, JSPs represent the view in the Model-View-Controller. When a JSP is first requested from an incoming request, it is parsed into a Java source file and then compiled into a Servlet class [1],[3].

Servlets are a Java technology-based solution that is very portable since it runs inside a Java Virtual Machine. They play an important role in the Struts framework since the controller in Model-View-Controller is a Servlet[1].

Struts also uses XML files to store the web application's configuration and to specify all the actions that can be taken by the Struts application. These are *web.xml* and *struts-config.xml*, respectively. A crucial role in any web application is played by web and application servers. The web server handles the HTTP processing while the application server handles container services for JSPs or Servlets[5].

An overview of the MVC pattern as it is implemented in a Struts application is presented in Fig. 1.

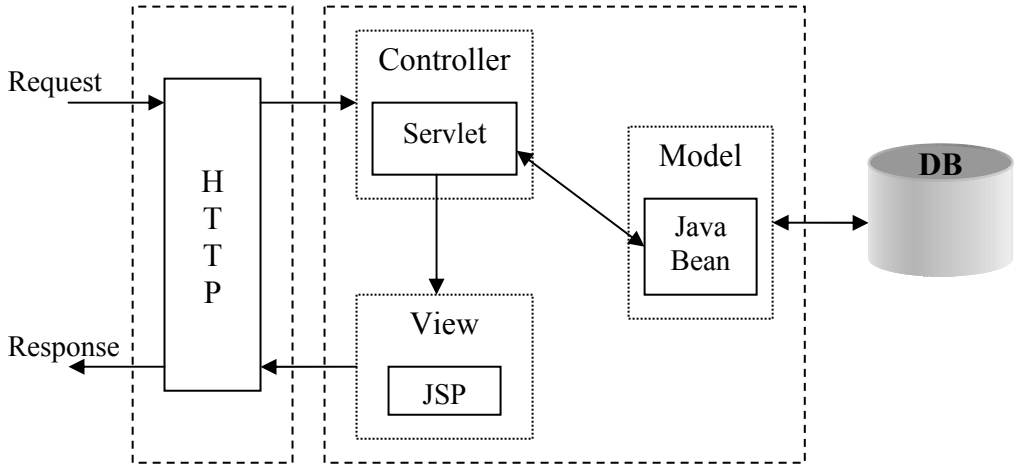

Fig. 1. MVC pattern overview

In the MVC design pattern, the application flow is mediated by the central Controller. The Controller is focused on receiving requests from the client (typically a user running a web browser), deciding what application function is to be performed, and then delegating responsibility for producing the next phase of the user interface to an appropriate handler. In the Struts framework, the handlers are called *Actions*. These actions are part of the Model, and typically use one or more Java Beans to perform the actual application logic. Control is then forwarded back to the Controller that dispatches the response to the appropriate View [3].

The advantages of using Struts over building a MVC-based application from scratch are that the framework provides a set of ready made components and the developer only has to add or modify certain pieces of the puzzle. For the Model, Struts provides the Action classes. These are small and typically rely on other JavaBeans to do the actual work. The View can use the *ActionForm* class provided by the framework to pass information from JSPs to the Model. Finally, the Controller is a Servlet, called the *ActionServlet* and is configured using XML files. The only thing that needs to be done to the Controller when using Struts is to provide the configuration in the form of the *struts-config.xml*, thus simplifying the development of the application [1].

2. Security

Being a web application, the project needs to transmit sensitive data between the web server and the client's browser. This data may include passwords for gaining access to the different level of the application, exam answers and other information that shouldn't be available to someone using a sniffer or other forms of interception. Because CAAP will be used to grade students, we must make sure that the authentication process is secure and only legitimate users have access to certain privileges on the system or are entrusted with certain roles. This level of attention to security is required to preserve the correctness of the academic process.

To protect sensitive data we will use the Secure Sockets Layer and the related protocol, HTTP over Secure Sockets Layer (HTTPS). HTTPS employs SSL to protect data by encrypting it at the source and then decrypting it at the destination. This prevents anyone who is monitoring the network from easily obtaining the data that is being transmitted. In order to perform the encryption/decryption process the client and the server exchange public keys. However this procedure is costly in terms of performance, therefore HTTPS will be used only for those pages or processes that transmit sensitive information. Typically, switching back and forth the protocols requires hard-coding the protocol used

and the full URL in every link to each resource in the web application. This leads to development problems each time a server name changes or the requirements for the secure protocol are modified [2].

Struts provides a solution to these problems with an extension named *sslext*. It extends the framework and allows developers to specify the transmission protocol behavior for Struts applications [2]. Within the Struts configuration file, developers specify which action requests require HTTPS transmission and which should use HTTP.

In addition to using SSL, we will integrate JAAS (Java Authentication and Authorization API) with Struts in order to secure the application's resources. There are two points of integration with JAAS [7]. During the logon process and when a resource is requested by the client. At each of these two points the application will dispatch control to JAAS classes that will perform the action. Once the user has been successfully authenticated and his permissions have been stored in his session context, the application can, upon each call to an *Action*, verify if the user has been granted access to the action being called.

3. Development Plan

This section examines the project's development plan in the context of Struts. The development has been broken into several steps [1].

1. Gather and define the application requirements. These have already been outlined in the first section.

2. Define and develop each screen requirement in terms of the data collected and/or displayed. During this stage in the development of the application we have identified the following screens.

Screen	Data Fields	Type	Comments
Main Menu	None		Contains only hyperlinks to the Login screen, Create account screen
Login	Username Password Test	String String List	Editable Editable Not editable
Create account Screen	Name, Surname, Faculty, Username Password	String String PWD	Editable Editable
Start test confirmation screen	None		Displays a hyperlink to the Question screen and starts the timer after activation
Question display Screen	Question (text & images) Radio buttons		Selectable (Allows multiple selections)
Finish Test confirmation screen	None		Stop timer, logs as definitive the answers in the ResponseSheet
Display Results	None		Displays the results of the test.

TABLE 1. Application screens

3. Determine the access paths for each screen, or the application flow. This is illustrated using an activity diagram.

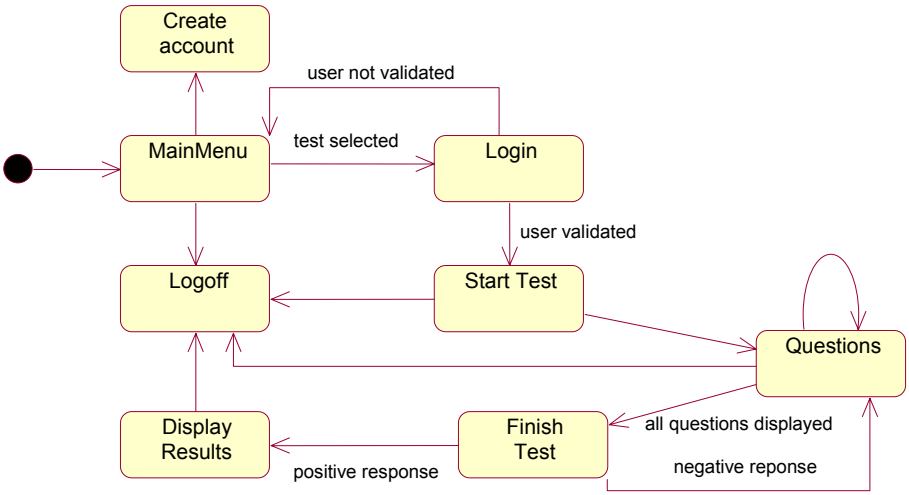

Fig. 2. CAAP activity diagram

4. Develop the *Actions* involved.

- *LogonAction* Once the user logs on , compares the username and password to those stored in a database. If the username and password match, then the activity returns success, otherwise the activity is a failure. The data will be encrypted.

- *AdduserAction* The user is prompted for personal data (name, surname, faculty, password, etc). If the data in all the required fields is successfully validated than a new account is added to the database and a message is displayed, otherwise the user is prompted to make the necessary changes inside the application form.

- *StartTestAction* Once the user has indicated that he is ready to start the test, this activity will generate a random sequence of test questions from the database that will be associated with the username and other information into an *ExamSession*. This activity will also start the exam timer. If the user doesn't press the button within a specified time interval, the test will start automatically. Every selection that the student makes from this moment till the end of the examination should be recorded in detail. This measure is required from an educational as well as moral standpoint.

- *DisplayQuestionsActions* This activity involves displaying the various parts of the question (text + images), as well as the answers. These should be displayed in a random order. Any computations regarding the results must be carried out exclusively on the server side.

- *FinishTestAction* Once the user has selected this path , the system will log the answers in the *ResponseSheet* as final and stop the timer. All other relevant information concerning the test will be saved in the applications logs and databases. The results will be computed on the server side by comparing the *ResponseSheet* with the *AnswerSheet*.

- *DisplayResultsAction* Displays the results of the test to the user.

- *LogoffAction* Performes the necessary steps to log off the current user from the system including invalidating any session information, as well as closing the session itself.

5. Define the JSPs to match the application's workflows.

JSP	Purpose
index.jsp	Main Menu
login.jsp	Username/password form
register.jsp	Form for gathering information about new users
startConf.jsp	Start test confirmation screen
stopConf.jsp	Finnish test confirmation screen
display.jsp	Displays the results of the test
questions.jsp	Displays the questions.

TABLE 2. JSP definitions

6. Define the database structure.

Column Name	Data Type	Key
UserID	Autonumber	Primary Key
Username	String	
Password	String	
Name	String	
Surname	String	
RoleID	Number	Foreign Key
FacultyID	Number	Foreign Key
DepartmentID	Number	Foreign Key

TABLE 3. Users Table

Column Name	Data Type	Key
SupervisorID	Autonumber	Primary Key
UserID	String	Foreign Key
Title	String	

TABLE 4. Supervisors Table

Column Name	Data Type	Key
TestID	Autonumber	Primary Key
TestTitle	String	
QuestNr	Number	
SupervisorID	Number	Foreign Key

TABLE 5. Tests Table

Column Name	Data Type	Key
QuestionID	Autonumber	Primary Key
TestID	Number	Foreign Key
Question	String	
MCSA	Boolean	
AnswerID	Number	Foreign Key

TABLE 6. Questions Table

Column Name	Data Type	Key
ChoiceID	Autonumber	Primary Key
TestID	Number	Foreign Key

QuestionID	Number	Foreign Key
Choice	String	

TABLE 7. Choices Table

Column Name	Data Type	Key
ScoreID	Autonumber	Primary Key
SessionID	Number	Foreign Key
Score	Number	

TABLE 8. Scores History Table

Column Name	Data Type	Key
SessionID	Autonumber	Primary Key
UserID	Number	Foreign Key
TestDate	DateTime	
TestID	Number	Foreign Key

TABLE 9. Sessions Table

Column Name	Data Type	Key
FacultyID	Autonumber	Primary Key
Faculty Name	String	

TABLE 10. Faculties Table

Column Name	Data Type	Key
RoleID	Autonumber	Primary Key
Role	String	

TABLE 11. Roles Table

Column Name	Data Type	Key
DeparmentID	Autonumber	Primary Key
Deparment	String	

TABLE 12. Departments Table

Column Name	Data Type	Key
ResultsID	Autonumber	Primary Key
QuestionID	Number	Foreign Key
SessionID	Number	Foreign Key
SelectedAnswerID	Number	Foreign Key

TABLE 13. Results Table

7. Build the configuration files.

8. Build, test, deploy. The last step of development, in which extensive testing will be carried out to uncover possible bugs and security flaws.

4. Conclusions

On-line quizzes can be used as an instrument for providing *feedback* to students on the degree of their understanding of course material. Such quizzes can be used at the beginning of a course for *diagnostic* purposes to indicate any areas where prerequisite knowledge may be inadequate, during the course to measure *progress* in understanding, or at the end of a course to assist in *revision* [11].

In the initial phase of our research we have evaluated a number of online testing tools to determine if one of these would be suitable for our needs. A number of other projects of this type have been going on around the world, including the CASTLE Project at the University of Leicester and TRIADS (Tripartite Assessment Delivery System), a joint project involving the University of Derby, the University of Liverpool and the Open University. The CASTLE (Computer ASsisted Teaching & LEarning) toolkit is one of the most popular tools for creating on-line multimedia assessments. Using CASTLE managers can create on-line interactive assessment tools quickly and easily *without* any prior knowledge of HTML, cgi, or similar scripting languages. This tool has been developed for educational purposes , and it is available free of charge [12].

An alternative approach, which can be used to create on-line quizzes, is to purchase one of the growing numbers of commercial software tools. The world leader in the field of commercial testers is *Question Mark Perception*. It can be used to develop and deliver both self-assessment tests and formal computer-mediated exams. A variety of standard question types are supported including numeric response and diagram/hotspot response as well as multiple choices, multiple response and text based questions. However, performance comes at a cost, and a license is very expensive [12].

Web@ssessor from ComputerPREP is another recent product designed with Java to run on the Web. In addition to the usual range of question types, links to graphics, video clips, audio files, and animations, Web@ssessor allows tests to be authored or taken in any language[12].

After the software review we have reached the conclusion that because of various problems like licensing requirements or security issues, we should develop our own project. We have settled on Struts to implement the project for a number of reasons—some technical and some not [4]. The Struts technical features that were important to us were:

- Struts performs well.
- Struts has a sound architectural model with a modest learning curve.
- Struts is very solid and stable.
- Struts has a strong set of custom tag libraries that simplify JSP development.

In addition, Struts simplifies the development process by supplying prebuilt components to assemble applications from and last but not least Struts was built using industry best practices including the MVC design pattern and it can be deployed in a wide range of environments. After finishing the development of the software at the end of the summer, we expect to deploy CAAP in a realistic environment at the beginning of Winter Semester 2004-2005.

References

[1]. Sue Spielman , The Struts Framework: Practical Guide for Java Programmers, Morgan Kaufmann Publishers (2003)
[2]. Apache Struts website, http://jakarta.apache.org/struts
[3]. Tim Holloway, Struts: a Solid Web-App Framework, www.fawcette.com/javapro (2002)
[4]. Harry Rusli and John Yu , Issues In Struts Adoption , www.scioworks.com (2003)
[5]. Don Denoncourt, Struts: A Standard Architecture for Web Applications, www.e-promag.com (2004)
[6]. Dinah Ceplis and Jack Moes, Industry Certification: A Technology Enhanced Testing Tool , Assiniboine Community College (2001)
[7]. Dan Moore, Using JAAS for Authorization and Authentication, www.mooreds.com /jaas.html
[8]. Eric Foxley, Colin Higgins and Cleveland Gibbon, The Ceilidh System. A General Overview, Learning Technology Research, Computer Science Department University of Nottingham UK (1996)

[9]. Eric Foxley, Colin Higgins, Pavlos Symeonidis and Athanasios Tsintsifas, The CourseMaster Automated Assessment System – a next generation Ceilidh, Computer Assisted Assessment Workshop (2001) , University of Warwick
[10]. Review of CourseMarker, Learning Technology Research , Computer Science Dept. Nottingham University UK (2003)
[11]. Guy Judge, The production and use of on-line Web quizzes for economics, The Virtual Edition, Volume 13, Issue 1(1999) University of Portsmouth
[12]. University of Leicester website , http://www.ulst.ac.uk/cticomp/CAA.html