Establishing a Remote Lab for Teaching Enterprise Application Development

Sam Lee
sl20@txstate.edu

Mayur R. Mehta
mm07@txstate.edu
Department of Computer Information Systems and Quantitative Methods
Texas State University – San Marcos
San Marcos, TX 78666, USA

Abstract

Many students of Computer Information Systems (CIS) at the McCoy College of Business Administration of the Texas State University – San Marcos live and work in the cities of Austin or San Antonio, which are both more than twenty miles away from campus. It is necessary for these students to use home computers to work on assignments and projects remotely. To study the Java 2 Enterprise Edition (J2EE), students are required to use IBM WebSphere Studio Application Developer (WSAD) as the development tool. This paper presents a method for the establishment of a remote lab in which students can learn how to develop and deploy J2EE applications that connect to database servers.

Keywords: Remote lab, distance learning, J2EE, application development, remote deployment.

1. INTRODUCTION

Colleges and universities have been teaching J2EE and database application development for years. Students of such courses appreciate being provided with real-world practical experience that explores an entire system through the design, implementation, operation, and documentation stages (Wang, 2003). The authors have used IBM WSAD and a database server to teach a course that is entitled "Advanced Business Application Development" that helps students to grasp the skills that are needed to work in the industry.

This course involves many types of computer equipment and many different software packages for the development and deployment of J2EE applications. Moreover, it requires intense interaction with server-based systems and the installation of expensive licensed software. Usually, a specialized lab is built to which only a few people have access. The university provides a CIS lab that is open to teachers and students on weekdays from 8:00 am to 8:00 pm. However, this lab is not convenient for the many CIS students who work and live in the cities that are more than twenty miles away from campus. Therefore, the idea of a "remote lab" has been pursued.

A remote lab that is available around the clock, seven days a week should comprise a lab environment in which students do not have to be present at an in-house lab to perform exercises with real equipment or systems. Several remote labs have been developed for distance learning in various fields. Alhalabi et al. suggested an environment in which physics and electrical engineering labs could be housed on the Internet (Hamza *et al.*, 2002; Hsu *et al.*, 2000), for which the creation of a Graphical User Interface (GUI) that displays responses from real physical materials would be necessary. Five Swiss universities designed a laboratory for remote exercises in the area of telecommunications and computer networking, and found that scheduling and error recovery were issues that particularly needed solving (Zimmerli *et al.*, 2003). In attempting to allow students to remotely develop and deploy J2EE applications, we have encountered a different set of problems. Students need to own copies of the IBM development tools and to remotely manage Web applications that run on real application servers.

This paper presents our experiences in developing a remote lab for students of J2EE application development. The attempt not only benefits traditional classes, but also helps CIS educators who are planning to teach using a distance-learning environment. This paper first provides an overview of J2EE application development, which is followed by a discussion of the requirements of the remote lab and the details of a sample student lab. The paper concludes with some solutions for the setting up of a remote lab for teaching J2EE application development.

2. J2EE APPLICATION

J2EE technologies are the standard for the development of component-based multi-tiered enterprise applications. J2EE applications are often constructed using the Model-View-Controller (MVC) architecture (see Figure 1). The MVC enforces the separation of presentation logic and business logic, and has been suggested as the blueprint that application developers should use to design and implement Web solutions (Takagiwa *et al.*, 2002).

Model-View-Controller

Models are meant to serve as a computational abstraction of real entities. For example, a model of a shopping cart in an e-commerce application contains the identity and quantity of the items that are selected for purchase and offers functions to add and remove items from the cart. Typically, the model component is written as a Java Bean, which uses JDBC technology to access relational database systems.

Views present (render) the information that is contained in a model. For example, a graphical view may render a shopping cart as a list that shows the number and description of items that have been selected for purchase, with a "remove" button next to each item. In a J2EE application, views are rendered by a Web browser using HTML, which is typically produced by Java Server pages (JSPs).

A controller component controls the application flow. It accepts input from the user, interprets the input, and invokes the appropriate operation in the model in response to the input. For example, when the controller receives a request from the "remove" button of an item, it invokes the remove operation for that item. The controller then forwards the JSP view that displays the updated shopping cart without the removed item. The controller is normally implemented as a Java Servlet.

Web Project

A Web project that is created in IBM WSAD organizes J2EE application component files in the following directory structure.

1. The Java Source, which contains the Java source code for the classes, beans, and servlets of the project.
2. The Web Content, which is the mandatory location of all of the Web sources, including the JSP and image files.
3. WEB-INF/classes, which is a directory for the complied classes of the Java Source. The classes in this directory are loaded by an application class loader.
4. WEB-INF/lib, which are the supporting library files that the Web application references, such as the JDBC driver library.

This directory structure maps to the Web application archive (WAR) structure that is defined by Sun. Therefore, a WAR file that is generated from the Web project can be deployed on many different J2EE application servers, including IBM

WebSphere, BEA Weblogic, and open-sourced Tomcat.

3. REMOTE LAB REQUIREMENTS

The remote lab for learning about J2EE application development needs to meet several requirements. First, students must not be forced to purchase software licenses. Second, it must be possible for all of the exercises to be carried out on the private computers of the students. Finally, the exercises must be deployed on a real application server so that they can be seen by anyone using a browser on the Web.

The university has been an active participant in the IBM Scholars Program since 2002, which helps with the first problem of licenses. The offerings and benefits of this program (www.developer.ibm.com/university/scholars/) are described as follows.

"*The IBM Academic Initiative provides accredited and approved faculty with access to the specified range of IBM products available for instructional, learning and non-commercial research. There is an ever-increasing list of products available including software, hardware, documentation, training and educational materials.*"

Our students thus have access to e-business software, educational materials, and other resources. We have also worked closely with an IBM team to develop curricula and to receive training on IBM e-business tools.

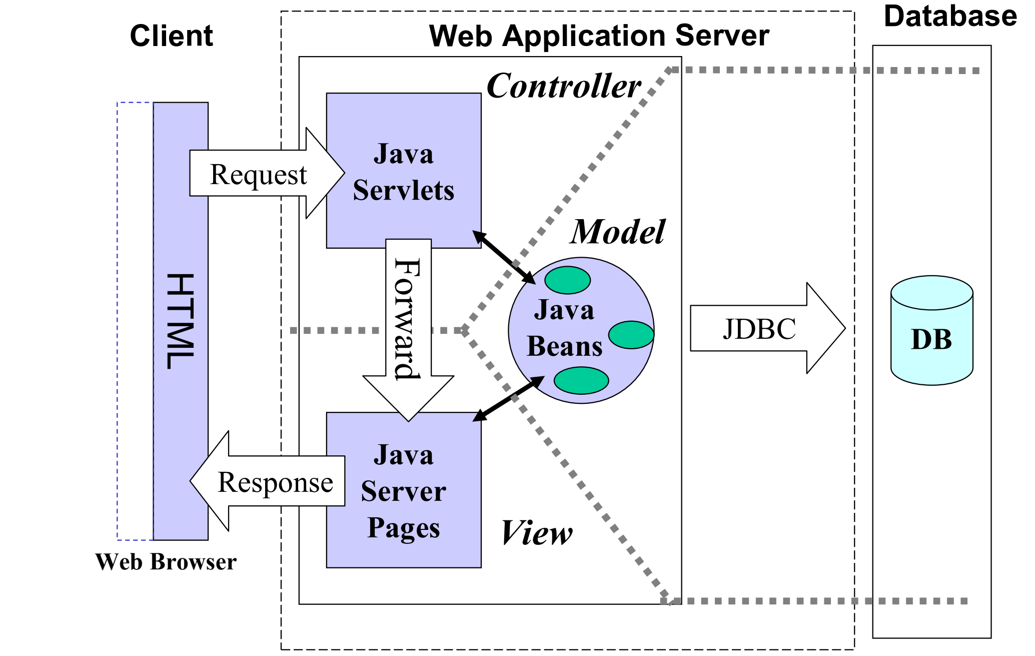
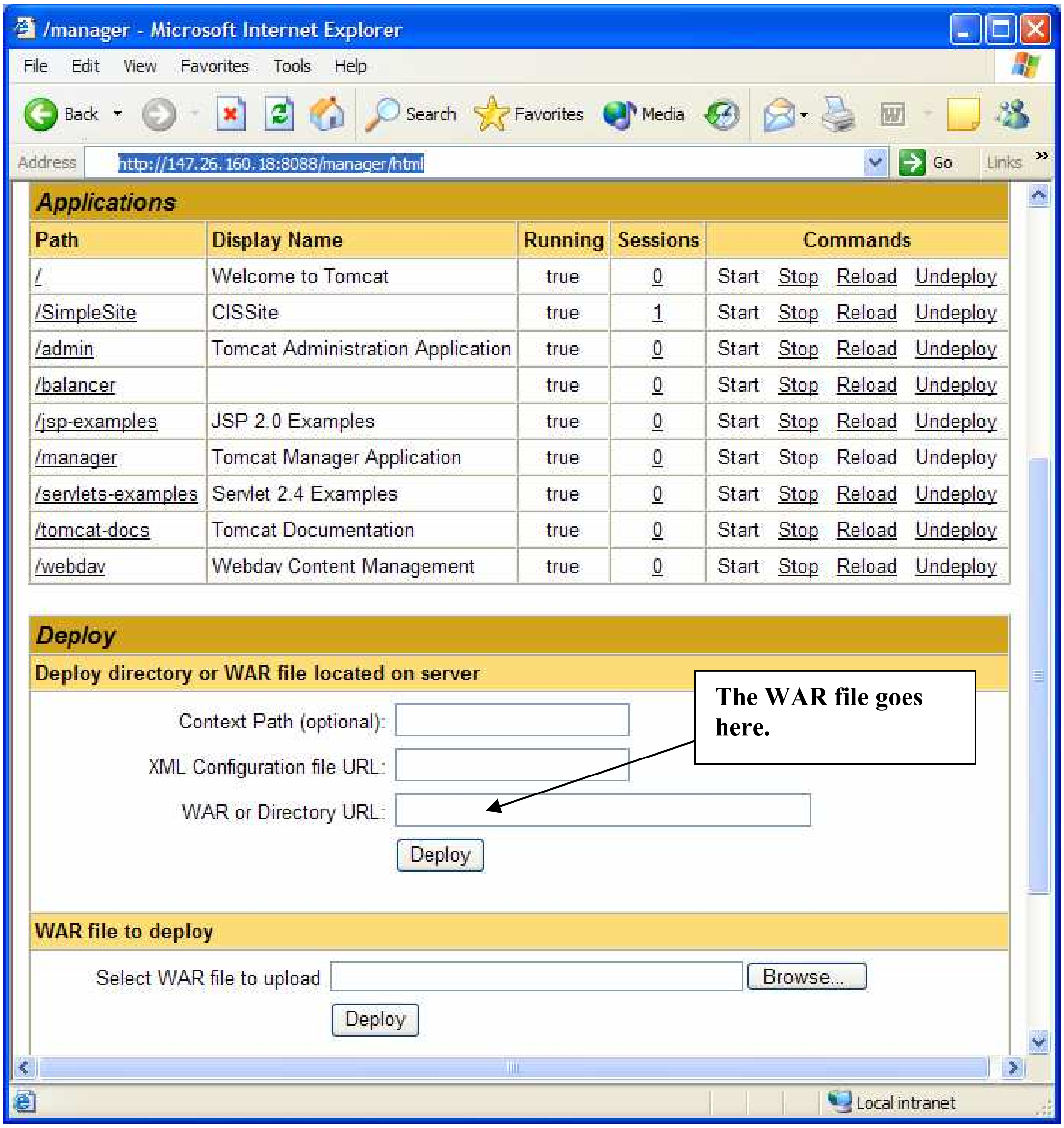


Figure 1. J2EE Application Using the MVC Architecture

The students use IBM tools to develop applications that support the J2EE open standard. When an application or exercise is ready, they can deploy the applications on a Tomcat 5.0 server that is available through the departmental server. The Tomcat server implements Servlet and JSP specifications, and includes many useful features for the development and deployment of Web applications and Web services. In particular, a Web-based manager tool provides the capability to remotely deploy individual Web applications without having to shutdown and restart the entire server (see Figure 2). A Web application is deployed using a WAR file, and each application can be started, stopped, reloaded, or undeployed separately.


/manager - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Media
Address
Go
Links
Local intranet

Address: http://147.26.160.18:8088/manager/html

Applications

Path	Display Name	Running	Sessions	Commands
/	Welcome to Tomcat	true	0	Start Stop Reload Undeploy
/SimpleSite	CISSite	true	1	Start Stop Reload Undeploy
/admin	Tomcat Administration Application	true	0	Start Stop Reload Undeploy
/balancer		true	0	Start Stop Reload Undeploy
/jsp-examples	JSP 2.0 Examples	true	0	Start Stop Reload Undeploy
/manager	Tomcat Manager Application	true	0	Start Stop Reload Undeploy
/servlets-examples	Servlet 2.4 Examples	true	0	Start Stop Reload Undeploy
/tomcat-docs	Tomcat Documentation	true	0	Start Stop Reload Undeploy
/webdav	Webdav Content Management	true	0	Start Stop Reload Undeploy

Deploy

Deploy directory or WAR file located on server

Context Path (optional):

XML Configuration file URL:

WAR or Directory URL:

Deploy

The WAR file goes here.

WAR file to deploy

Select WAR file to upload Browse...

Deploy

Figure 2. Tomcat Web Application Manager

4. A SAMPLE LAB

To demonstrate the process of building an enterprise application, we discuss the development of a simple Web-based registration system. The system is to be designed and implemented in accordance with the J2EE architecture. The system permits users to enter a user ID, password, and personal information to become members of the Web site; displays a welcome page if the users register successfully; and displays the response to the user ID step, including a message to enter another ID if the ID that the user has entered is already taken.

First, a Web project called "SimpleSite" is created to start a new lab, and then a JDBC driver is imported into a WEB-INF/lib directory for database connectivity. Second, a model of a "member" Java Bean is built that contains the properties of the member's ID, password, name, and address, and provides methods for setting the property values, searching a member by ID, and creating a new member in the database. Third, views are developed, including the registration and welcome JSP pages. Finally, a controller that is implemented by a servlet is designed to receive the submitted registration information and then to call on the Java Bean search method by passing the entered ID. The controller calls on methods to set the member properties and create a member if no existing member is found to own the entered ID. It either forwards the view that displays welcome messages if registration is successful, or forwards and redisplays the registration page if registration is unsuccessful.

After all of the application components have been developed and tested on a private computer, they are remotely deployed on the real Tomcat 5.0 server through the following steps. First, the Web project is exported to a WAR file using a WSAD wizard. The WAR file must be named using the project name (e.g., SimpleSite.war). The Tomcat Web Application Manager is then opened remotely, as shown in Figure 2, using a username and password, and the WAR file is uploaded. The Web project (or application) is then listed with the path information (e.g., /SimpleSite), the application is started, and the WAR file is automatically expanded and deployed on the Tomcat server. The application is then executed using an appropriate URL (e.g., http://lab01.cis.univ.edu:8088/SimpleSite/Registration.jsp).

5. CONCLUSION

Many working professionals that have busy schedules seek advanced CIS education at colleges and universities. Learning server-based application development does not only involve reading online materials and writing reports, but also practical exercises, and remote labs are a convenient way to foster practical skills in students who have to commute to campus.

This paper describes our experiences in setting up a remote lab for teaching J2EE application development in traditional classes. We have provided solutions to some issues that include access to IBM WSAD software and the remote deployment of the student exercises on a Tomcat server. In the spring of 2005, many of the 20 students in the Advanced Business Application Development class installed WSAD in their home computers to work on assignments. However, only a few students succeeded in remotely deploying Web applications. Different project names were assigned to individual students. They tended to fail the deployment by overlooking the requirement that Web project names must be identical to WAR file names.

The authors believe that this remote lab solution is potentially effective for the delivery of distance courses that involve Web programming exercises. Additional work is being pursued, including improvements to the remote deployment process and the development of a remote environment for teaching Web services.

6. REFERENCES

Hamza, K., B. Alhalabi, S. Hsu and M. Petrie (2002). "Remote Labs: the Next High Tech Step beyond Simulation for Distance Education."

Journal of Computers in the Schools, 19 (3/4), pp. 171-190.

Hsu, S., B. Alhalabi & M. Ilyas (2000). "A Java-based Remote Laboratory for Distance Education." International Conference on Engineering Education, Taipei, Taiwan.

Takagiwa, O., J. Korchmar, A. Lindquist and M. Vojtko (2002). "WebSphere Studio Application Developer Programming Guide (1st ed.)." San Jose, California: IBM Corporation, International Technical Support Organization.

Wang, M. (2003). "Teaching Web-driven Database Application with Templates." Information Systems Education Journal, 1 (11), pp. 1-8.

Zimmerli, S, M.-A. Steinemann and T. Braun (2003). "Resource Management Portal for Laboratories Using Real Devices on the Internet." ACM Computer Communication Review, 33 (3), pp. 145-151.