Design and implementation of a web server application for students to submit assignments

Yuehua Yang
The University of Montana

Follow this and additional works at: http://scholarworks.umt.edu/etd

Recommended Citation
Yang, Yuehua, "Design and implementation of a web server application for students to submit assignments" (2001). Theses, Dissertations, Professional Papers. Paper 5504.
Permission is granted by the author to reproduce this material in its entirety, provided that this material is used for scholarly purposes and is properly cited in published works and reports.

** Please check "Yes" or "No" and provide signature **

Yes, I grant permission __________  \[yes\]
No, I do not grant permission

Author's Signature  \[Yuehua Yang\]  

Date \[April 26, 2001\]

Any copying for commercial purposes or financial gain may be undertaken only with the author's explicit consent.
Design and Implementation of a Web Server

Application for Students to Submit Assignments

By

Yuehua Yang

B.Sc. University of Anhui Agricultural Science, China, 1981

M. Sc. Graduate School, Chinese Academy of Agricultural Science, China, 1986

Presented in partial fulfillment of the requirements

for the degree of

Master of Computer Science

Department of Computer Science, The University of Montana

April, 2001

Approved by

Chairperson

Dean, Graduate School

Date
Design and Implementation of a Web Server Application for Students to Submit Assignments

Chairperson: Alden Wright

This project is a software engineering approach to the design and implementation of a web server application for students to submit assignments. During the design and analysis of the project, a requirements document, high-level use cases and expended use cases documents, class diagrams and sequence diagrams have been finished. The project uses Java servlet programming to implement the application system. There are two parts in the application system, an instructor part and a student part. For an instructor user, the system allows the instructor to set up a course for students to submit their assignments. The system allows the instructor to set up and to modify the assignment size limitation, to edit the course and student passwords, to check student submission information online, and to delete a student account. The system also allows the instructor to delete his/her course account. For a student user, the system allows each student to set up a password-protected account for assignment submission. Students can submit their assignments online after they set up their account. The system also allows students to edit their passwords.

A web demo of the project has been set up. All of the functionalities in the application work. The design and implementation of the project is summarized in the paper. Some questions and experiences with the design and implementation of the project are also discussed in the paper. A recommendation for future work has been suggested. From this project, I learned a lot of software engineering methodology, from requirements analysis, to elicitation of use cases, to architecture and detail design of the project, to extraction of sequence diagrams and class diagrams. Through the project, I increased the abilities of solving technical problems, which were not familiar areas to me. All of these experiences are very helpful for me to continue to work in the J2EE area.
Acknowledgment

I would like to thank my advisor Dr. Alden Wright for his assistance, support and encouragement. I would also like to thank Dr. Joel Henry and Dr. Lee Tangedahl for their helpful suggestions.
## Contents

1. Introduction ....................................................................................................... 1
2. Background ........................................................................................................ 2
3. System Support ................................................................................................. 9
4. Software Development Approach .................................................................... 10
5. Development Processes ................................................................................... 11
   5.1 Requirement Analysis .............................................................................. 11
   5.2 Design .......................................................................................................... 14
      5.2.1 Class Hierarchy ................................................................................. 14
         5.2.1.1 The Architecture of the Web Server Application System .......... 14
      5.2.2 Database Table Design .................................................................... 16
      5.2.3 Sequence Events of a Use Case ....................................................... 17
         5.2.3.1 Use Case of “Set up a course” ................................................... 17
6. Implementation .................................................................................................. 19
   6.1 Coding Languages ...................................................................................... 19
   6.2 Coding and Test ......................................................................................... 19
7. Discuss ............................................................................................................. 21
   7.1 Servlet Programming and Object Oriented Design ..................................... 21
      7.1.1 Require/response Oriented ............................................................... 21
      7.1.2 User-friendly Interface Design ....................................................... 22
   7.2 UML for Web Application ........................................................................ 22
   7.3 Database Connection ................................................................................. 23
   7.4 What I learned in the project .................................................................... 24
8. Summary ........................................................................................................... 26
9. Bibliography ..................................................................................................... 28
10. Appendix A Project Requirements ................................................................. 29
11. Appendix B High-level Use Cases ................................................................ 35
12. Appendix C Expanded Use Cases ................................................................ 37
13. Appendix D Class Diagram ........................................................................... 45
14. Appendix E Sequence Diagrams .................................................................... 49
15. Appendix F System Installation Instruction ................................................... 59
16. Appendix G Project Proposal ......................................................................... 68
17. Appendix H A Real Configuration File of srm.conf ...................................... 73
18. Appendix I A Real Configuration File of jser.conf ........................................ 78
19. Appendix J A Real Configuration File of jser.property .................................... 80
1. Introduction

The Internet and the World Wide Web have experienced explosive growth over the past decade and are now positioned to provide a wide range of services. Computers have become and will continue to be part of our everyday lives. More and more individuals, organization and business use networks to get and send information. Generally, when a student takes a course in a university, he /she has to finish the required assignments and to submit them to the instructor. In computer-related courses, assignments are often submitted electronically in the form of computer files. Traditionally, when a student wants to submit an assignment electronically, he/she has turn in a floppy disk, to send the file to the instructor by email or to copy the file to a course directory, which has been set up for that purpose. However, there are four major problems with these methods of submitting assignments:

(1) If the floppy disk submission method is used, the instructor has to retrieve the assignment from one floppy disk per student, which will take the instructor a lot of time because the machine reads a file from a floppy disk much slower than from a hard disk. Also, both students and the instructor have to use the same format system to write and read a floppy disk.

(2) If the email submission method is used, the instructor has to deal with a large number of email messages, which contain student assignment files. It is not surprising that the instructor may receive more than one hundred emails in a 50-student course for an
assignment. A student may send several modified assignment files for an assignment. The instructor has to save these files to a directory and verify and delete duplicate submissions.

(3) If the copy-file submission method is used, the student must have an account on the machine where the instructor asks students to copy their files to a course directory. The students can go to the course directory and view other students' file information, such as who have submitted their files and when the files have been submitted. Because of this security problem, most instructors only ask students to copy their executable files to this directory. The students have to submit a hardcopy to the instructor.

(4) The students cannot verify whether or not the instructor has received their files when students submit their files by email. Students often submit assignments to the wrong email address. It is a very common situation that when a student is told that he/she has failed to finish an assignment, he/she has to explain that he/she has submitted his/her file and did not know that the file is missing.

Obviously, these disadvantages significantly decrease the efficiency for both instructor and student to send and receive a course assignment. Therefore, there is need to improve the efficiency and reliability of the assignment submission process.
2. Background

In a typical client/server system (two tier architecture), the larger processing loads are given to the PC while the server simple acts as a gateway between the application and database. There are three problems in two-tier architecture as applied to web-based applications:

(1) The web application performance suffers due to the limited resources of the PC

(2) Network traffic tends to increase

(3) The application is hard to be maintained. Even the smallest of changes to the application would involve a complete rollout to the entire user base.

To address these problems, a three-tier architecture has been developed. The application in three-tier architecture is broken up into three separate logical layers, each with a well-defined interface. The first tier is the client-tier. It is responsible for the presentation of data, receiving user events and controlling the user interface. The second tier is the application-server-tier. It is basically the code, which the user calls upon (through first-tier) to retrieve the desired data. Business-objects that implement the business rules "live" here, and are available to the client-tier. This level now forms the central key to solving 2-tier problems. This tier protects the data from direct access by the clients. The third tier is the data-server-tier. This tier is responsible for data storage. Besides the widespread relational database systems, existing legacy systems databases are often reused here. So, a typical web application will collect data from the user (first
tier), send a request to the web server, run the requested server program (second and third tiers), package up the data to be presented in the web browser, and send it back to the Browser for display (Figure 1)

Figure 1 A three-tier architecture

This project will use a three-tier architecture to develop a servlet application system for students to submit assignments. So, a web server, and a database server need to be selected in order to implement the three-tier architecture. Also, an operating system needs to be selected in order to run these servers and the application.

Since the application will be used in a department of a university, which is a non-profit organization, and since there is no financial or software support for this project, it is a good choice to use free software to develop and to run the application.
(1) Web server selection

The web server is a program running on the server computer that listens for incoming requests and services those requests as they come in. Java servlets can run virtually unchanged on different web servers, such as Apache, Microsoft Internet Information Server (IIS), IBM WebSphere, or StarNine WebStar. According to the results of a Web server survey released by http://www.netcraft.com/survey, the leading web servers are Apache web server and Microsoft’s IIS. The Apache web server has been developed as free software in recent years. Its power, flexibility, ease of use, and the availability for multiple platforms have contributed to its rise in popularity over the past few years. Based on the investigation on January 2001, the use of the Apache web server accounted for 58.75% of all web servers while the use of Microsoft’s IIS accounted for only 21.40%. The Apache web server plays an important role in the server-side applications. Since the Apache web server is a free software package and it can be run on a Linux system, a free operating system, the Apache server will be used in this project.

(2) Programming Languages Selection

Servlets are programs that run on a web server, acting as a middle layer between a request coming from a Web browser or other HTTP client and databases or applications on the HTTP server. They provide an easy way for server side code to communicate with web-based clients and a database. Java servlets are more efficient, easier to use, more powerful, more portable and safer than traditional CGI and many alternative CGI-like technologies. There are three advantages to Java servlets:
(I) Platform independence: Java servlets follow the same platform-independent model that the Java language does. You can freely move your servlets to any other platform that supports Java.

(II) Efficiency: Java servlets provide a much more efficient method of handling user requests. When a servlet receives a user request, it spawns another thread within the same process and handles the request, which makes it possible for more users to access the servlet simultaneously.

(III) Reusability: Java is an object-oriented language. It can encapsulate information and provides mechanism for reuse.

The Apache JServ is a module for the popular Apache web server that implements Sun's Java Servlet API for running server-side Java code. Also, JServ is free. So, the project will use the Java Apache web server with the JServ module.

The Java Apache web server with the JServ module only supports Java servlets, html, and cgi. It does not support JSP. Thus, Java servlets, html and Perl are the only possible languages that can be used in the application. JSP is a new web-development language, its specification was published in June 1999. Since JSP was a newly-released specification and there was not a free, stable module, which could be run on the Apache server available at the time when I designed the project, JSP was not used in the project.

(3) Database server selection
MySQL is a database management system that allows you to store, access, and manage information on a scope ranging from a single small list to a large collection of tables containing millions of rows. It can handle business records, scientific data, images, documents, poll results; the possibilities are endless. In the realm of open source database server software, MySQL is the most popular database server.

There are five advantages of using MySQL:

(I) FREE. MySQL is an Open Source database, available over the Internet at no cost. This contrasts with other proprietary commercial databases that are available only from specific vendors, typically at very high cost.

(II) FAST. It's widely regarded as faster than other databases. MySQL is particularly suited for Web-based applications on high-traffic sites that generate content on the fly based on database lookups.

(III) SQL-BASED. MySQL understands Structured Query Language (SQL), the most common database language in use today.

(IV) EASY TO USE. The MySQL distribution is a relatively small package, not hundreds of megabytes like some other database products. The development philosophy is to focus on features people actually want rather than to toss everything possible into the distribution. This makes MySQL easier to understand.
PORTABLE. MySQL runs under several varieties of UNIX, Linux, and Windows. It is easy to install, set up, use, and administer.

(4) Operating system selection

Linux is a free, UNIX-like operating system, developed originally for Intel-architecture PCs, but which now runs on a variety of platforms including PowerPC, Macintosh, Amiga, Atari, DEC Alpha, Sun Sparc, ARM, and many others. Linux systems excel in many areas, including stability, speed, networking and so on.

(I) Stability: Linux has long been praised for its stability--Linux boxes are known for running months or even years at a time without crashing, freezing, or having to be rebooted. Viruses are seldom a problem on Linux systems.

(II) Fast: Linux machines are also known to be extremely fast, because the operating system is very efficient at managing resources such as memory, CPU power, and disk space. It is natural for Apache web server to run on Linux.

(III) Networking: Networking comes naturally to Linux. After all, Linux is based on UNIX, where computer networking more or less developed. Linux gives very good support for TCP/IP-based networking.

Based on above reasons, this project will use the Apache JServ as its web server, MySQL to manage the database, and Java servlets to develop the application. The Linux operating system will be used.
3. System Support

3.1 System Support

The programs run on the Linux operating system. The following system versions will be used to develop and demonstrate the application:

(I) Apache 1.3.9(including JServ)

(II) JSDK 2.0

(III) JDK 1.2

(IV) MySQL 3.22.32

(V) MySQL.jdbc-2.0pre5

A document to describe how to install and configure related system software is included in Appendix F.
4. Software Development Approach

This project is a software development project. A software engineering approach is used to ensure that the developed product satisfies the requirements of the clients. There are several major software engineering methodologies associated with such areas as software requirements analysis, software design, software implementation and integration, and so on. In this project, the software development approach was as follows:

(1) Project requirements were analyzed and a software requirements document was generated.

(2) In order to further specify the requirements, a high-level use case document and an expanded use case document were generated. These describe the sequences of events that are generated when the actors (student and instructor) use the software system to do specific tasks.

(3) A class diagram describes the types of objects in the system and the static relationship that exists among them. In this project, a class diagram was finished to describe the relationship among the classes.

(4) A system sequence diagram is a picture that shows for a particular scenario of a use case, the events that external actors generate, their order, and inter-system events. The sequence diagram for each use case in the project was generated after finishing the use cases.
(5) In order to improve the understanding of sequence diagram, an English narrative description of a use case was made. It explains the sequences of events generated by actors.

(6) During the software implementation phase, the software was finished, integrated and tested.

The details of the approach followed are provided in subsequent parts of this paper. The software engineering approach helped to ensure the project success.
5. Development Processes

5.1 Requirements Analysis

In this project, a web server application system was designed and implemented. The purpose of this application is to allow students in a university course to submit assignment files and allow an instructor to edit and manage a course for the purpose of submitting assignments.

Two levels of security protect the application system. The first level is a web access control level. A system administrator (or other person with root access to the server computer) sets up the first level of security. Each instructor has a user id and password to access the instructor web page. The second security level is the user personal security level for students and the course security level for instructors. The authorized student users and instructor users will set up the second level of security.

For student users, the system allows each student to set up an account for assignment submission. Students can submit their assignments online after they set up their account. The system also allows students to edit their passwords.

For instructor users, the system allows the instructor to set up a course for students to submit their assignments. The system allows the instructor to set up and modify the assignment size limitation, to edit the course/student password, to check
student submission information online, and to delete a student account. The system also allows the instructor to delete his/her course account.

A more detailed requirements document is included in Appendix A.

In order to improve the understanding of requirements and to describe the sequence of events of an actor (an external agent) using a system, a high-level use case document and an expanded use case document were written. The high-level use case document is included in Appendix B.

It is useful to start with high-level use case to quickly obtain some understanding of the overall major processes. Ten use cases are included in high-level use case document of the project. In these use cases, a use case “Set up a Course” will be described as follows:

In the “Set up a Course” use case, the actor is an instructor user. The instructor user can specify a course via inputting related course information and instructor information, then the course calls “Set up file limitation” use case to specify a file size. The system is now ready to allow students to set up accounts.

Other high-level use cases were described in Appendix B.

An expanded use case shows more detail than a high level one; they are useful in order to obtain a deeper understanding of the processes and requirements. For an example, the typical course of events in “Set up a course” use case is described as following:

(1) This use case begins when instructor user is ready to set up a course.

   Suppose the user is authorized to access the instructor web page.

(2) The user selects to set up a course.
(3) The system shows a form to ask the user to input the course number, section number, maximum assignment number, course id and password.

(4) The user inputs the course number, section number, the instructor first name, last name, the number of assignments, the course id and password.

(5) The system shows a form to ask the user to input the file size limitation.

(6) The user specifies a file size limitation.

(7) The system saves the file size limitation into database table. The system gives information to show that the course has been set up.

The expanded use case document is included in Appendix C.

5.2 Design

5.2.1 Class Hierarchy

5.2.1.1 The Architecture of the web server application System

In this project, 4 java programs named SubmissionServlet.java, Course.java, Student.java and Submission.java are developed. All data and information are stored in 4 data tables named studentAccountTable, submissionTable, instructorTable and assignmentTable.

A class hierarchy diagram is showed in Figure 2
SubmissionServlet.class will extend javax.servlets.HttpServlet.class. Course class will extend SubmissionServlets class and will use all of the methods in SubmissionServlet class. Student class extends Course class and will use all of methods in SubmissionServlet class and the methods enterCourseSelection (out) and maxAssign (String) in Course class. Submission class will extend the HttpServlet class in javax package.

A more complete class diagram is included in Appendix D. In this class diagram, SubmissionServlet class extends javax.servlets.HttpServlet.class. Course class extends SubmissionServlets class and uses all of the methods in SubmissionServlet class. In Course class, class information will be saved in the instructorTable by the saveCourse function, the file size and the maximum assignment number will be saved in the
**assignmentTable** by the **saveFileSize** function, the student information will be accessed from the **studentAccountTable** by the **getAssignInfo** function and the students’ submission information will be accessed from the **submissionTable** by the **getAssignInfo** function. The **Student** class extends the **Course** class. In the **Student** class, the student information is saved in the **studentAccountTable** by the **saveAccount** function; the file size value will be retrieved from the **assignmentTable** by the **getFileSize** function. The **Student** class sends the file size and the student id into **parser** and **parser** submits the file submission information into **Submission** class, which also extends the **HttpServlet**. In **Submission** class, the instructor name will be accessed from the **instructorTable** by the **getInstructor** function and the student name will be accessed from the **studentAccountTable** by the **getStudent** function. The submission information will be saved in the **submissionTable** by the **saveFileInfo** function.

5.2.2 Database table design

In this project, database tables are used to save the course information, the student information, the assignment information and the submission information. In order to manage the database easily, 4 tables were created to save this information. These tables are listed below. The listing includes the table name and the fields. The primary key / concatenated key field is underlined:

1. **instructorTable**: `tablekey (course number + section number)`, instructor's first name, last name, course id, and password.

2. **assignmentTable**: `tablekey`, maximum assignment number, and file limitation size.
(3) studentAccountTable: tablekey, student's first name, last name, 
university-assigned id, user id and password.

(4) submissionTable: tablekey, user id, file name, submission time and assignment number.

5.2.3 Sequence Events of a Use Case

The detailed sequence diagrams of the use cases in the project are included in Appendix E. There are 10 use cases in the project, which include:

(1) Set up a course
(2) Set up file limitation
(3) Change the course password
(4) Check student assignments
(5) Update a student password
(6) Delete a student account
(7) Delete a course
(8) Set up a student account
(9) Submit a assignment
(10) Change student password

As an example, the sequence events of “Set up a course” use case will be described in order to describe how the sequences diagram works.

5.2.3.1 Use Case of “Set up a course”

When an instructor user logs in the instructor home page and selects “Create Course for Submission”, the home page submits the request from the instructor home page into Course class. The doPost function in Course class gets the request and
calls `createNewCoursePage` function. `createNewCoursePage` generates an html page and asks the instructor to input the parameters, which includes the course id, the password, the course number, the section number, the maximum assignment number, the instructor’s last name, and the instructor’s first name. These parameters are submitted into the `Course` class. The `doPost` function in the `Course` class calls `getNewCourse` function and gets these parameters. The `getNewCourse` function also calls `isEmpty` in `SubmissionServlet` class to check if the user inputs all of the parameters, calls `checkPasswd` in `SubmissionServlet` to check if the password satisfies the required format, calls `confirmPasswd` in `SubmissionServlet` to check if the password is confirmed and calls encoding in `SubmissionServlet` in order to encode password using MD5 method. Then, the `getNewCourse` function calls the `saveCourse` function, which saves the parameters into the `instructorTable`. The `getNewCourse` function also call `createFileSizePage` to ask the instructor to input the file size limitation. When instructor submits file size to `Course` class, `doPost` in the class calls the `getFileSize` function. This function gets the file size and calls the `saveFileSize` function which saves the maximum assignment number and the file size into the `assignmentTable`. The function also creates some of the directories on the hard disk of the server machine where the students’ files will be saved.

The instructor must first set up the course/section account. Otherwise, the instructor cannot execute the functionalities for the course/section, and students cannot enroll the course/section.
6. Implementation

At this phase, the detailed design is translated into code.

6.1 Coding Languages

Because the Apache JServ only supports Java servlets, html, and cgi and does not support JSP, Java servlets, html and Perl are the only possible languages, which can be used in the application.

When I started to implement the application, I was going to use only Java servlets and html coding in the application. Thus, when I wrote a Java program to upload the student submission file, I used FileInputStream to upload the file and I used FileOutputStream to save the file into a directory on the server machine. However, when I tested the program through the web, I found that the program was not stable work for uploading the file, sometimes, it worked, sometimes, it didn't. Thus, I decided to use a Perl cgi program to finish the file upload and saving task. The Perl program works well. So, the application includes Java servlets, html and cgi programs.

6.2 Coding and test

This project includes two html files named index.html and index_1.html, four Java programs named SubmissionServlet.java, Course.java, Student.java, Submission.java, and one Perl file named parser.cgi. The java programs are well-commented so that they can be maintained easily.
After the programs were compiled and tested, the application was installed and configured in order to run application using an Apache server. A detailed document was written to describe how to install and to configure the application. The document is included in Appendix F.

The application web demo has been set up. I ran and tested the web application using Netscape Communicator (version 4.0, 4.75, 6.0) and Microsoft Internet Explorer (version 4.0 and 5.0). The application worked fine in all of these versions of Netscape and Microsoft Internet Explorer.
7. Discussion

7.1 Servlet Programming and Object Oriented Design

7.1.1 Require/response oriented

Java servlets extend the javax.servlet.http.HttpServlet class. Java servlets are not user-invocable applications. Servlets interact with a servlet engine through requests and responses. The servlet engine in-turn interacts with the web server by delegating requests to servlets and transmitting responses to the web server. The Java servlet framework provides an object-oriented abstraction of the HTTP request-response paradigm, and is well suited for gluing back-end applications to the web server. However, servlets are not reentrant programs, and therefore cannot maintain state unless they are maintained in HttpSession objects.

One of hard issues in the project is how to design an optimum object oriented servlet application. In this project, I designed two large classes, named Course and Student. In the Course and Student classes, I included several functions to create html web pages, which send requests to web server; I also included several functions to get and process these requests. The combination of "create html page" functions and "get html page" functions ensured implementation of all the functionalities in the project. However, the design also increased the complexity of programming design and decreased the cohesion of the classes.
As a recommendation for future work, this project can be divided up into more classes, which would increase the cohesion of the classes. A combination of composition method to reuse objects/classes and inheritance method is a good choice for the optimizing and implementing the application. Then, we can use interfaces to manage and to call these classes. A good way to finish them is to use both of JSP and servlets to optimize the design and to simplify the coding.

Perhaps the hardest item of the project is how to design an optimum object oriented servlet application. Also, the application must deal with servlets, the servlet API and html. Therefore, the coding, integration and testing for the application are very time-consuming.

7.1.2 User-friendly interface design

A friendly user interface is provided to facilitate the user’s interaction with the system. In the beginning of the project design, my design divided objects based on functionalities such as new account object, new password object and submission object in the student part. Each object had a specific purpose. There was not a direct relationship between the functionalities. The advantage of this design was that it is easy to understand and to maintain. However, it was hard to implement a user-friendly interface with the design. For example, suppose a student selected a course/section in order to submit assignment and after he/she finished the submission and selected other functionalities, the student was forced to re-select course/section for each functionality. Also, the student had to input his/her user id and password for each functionality which needs to check the user’s authorization. This is not a user-friendly interface design. So, I changed my design,
and used the large objects such as Course and Student to encapsulate these functionalities. Each functionality was divided into several methods. When a class was called, these methods could create some of html web pages to implement some of the functionalities. In this design, some of the parameters and methods could be shared with other methods and functionalities. Also, the use of dynamically generated HTML pages could keep the course/section content in a dynamic status and make maintenance of html files easily. So, a much more user-friendly interface design was obtained.

7.2 UML for Web Application

It is obvious that web application designers must work with pages. However, this brings up the question: Should I include the page object in my sequence diagram? At the beginning, I did not include html pages in the sequence diagram. However, I could not illustrate object interactions clearly only by class objects. In fact, web pages, especially in server side, do affect the behaviour of the system application (Jim Conallen 1999). So, these web page objects and script (Perl) must be included in the sequence diagrams for the project. In this project, the users input and submit all the required parameters by means of web pages. The Course class could create nine html pages, and the Student could create four html pages. Each html page passed some special parameters to web server.

7.3 Database connection

Servlets can access a database through vendor-specific Java Database Connectivity (JDBC) driver. To make a connection with the Database, the JDBC driver must be loaded and the driver must be connected in order to create a connection object.
The connection object represents a native database connection and provides methods for executing SQL statements.

The generic code is:

```java
Connection con = DriverManager.getConnection(URL,
"myLogin", "myPassword");
```

For a connection, the hardest thing is what to supply for URL. In this project, the URL = "jdbc:mysql://localhost:3306/research". If you are using some other driver, you must look up the related documentation and figure out what URL should be used. After the connection, Statement and ResultSet objects must be created in order to execute query operation.

During the implementation and test of the project, I found that database connections could not be maintained over multiple executions of queries. Each statement only could create one object, which was only used to execute one query. If you want to execute two or more queries using one connection and statement object, the database will shut down the connection and refuse to execute second query. Also, when you finish database operation, you must close database connection in order to decrease the overhead of acquiring JDBC connection. Sometimes, you may get an unusual result if you do not close the connection for several queries to the database.

### 7.4 What I learned in the project

#### 7.4.1 Software engineering approach

This is first time that I designed and implemented a real world project independently using the software engineering approach. From this project, I learned a lot of software engineering methodology, from requirements analysis, to elicitation of use
cases, to architecture and detail design of the project, to extraction of sequence diagrams and class diagrams. All of these experiences are very useful to my future research and work.

7.4.2 Ability to solve technical problems

This project will use Java servlets to develop a web application. I didn’t have any course studied experiences in this area. While I designed and implemented this project, I have to study web-based programming languages such as Java servlets, Perl and html. Through the project, I increased the abilities to solve technical problems, which are not familiar areas to me.

This project uses Apache JServ, and mySQL server. I have to learn and deal with how to configure and administer these servers independently. These experiences are very helpful for me in my future to work in the J2EE (Java 2 Enterprise Edition) area.
8. Summary

(1) This project is a Java servlet application. The purpose of this application is to allow students in a university course to submit assignment files and to allow an instructor to edit and manage a course for the purpose of submitting assignments. There are two parts to the application system, the instructor part and student part.

(I) For instructor users, the system allows the instructor to set up a course for students to submit their assignments. The system allows the instructor to set up and modify the assignment size limitation, to edit the course/student password, to check the student submission information online, and to delete a student account. The system also allows the instructor to delete his/her course account.

(II) For student users, the system allows each student to set up an account for assignment submission. Students can submit their assignments online after they set up their account. The system also allows students to edit their passwords.

(2) This project used a software engineering approach. During the design and analysis of the project, a requirements document, a high-level use case document and an expanded use case document, class diagrams and sequence diagrams have been finished.
(3) In this project, two html files, four java programs, and one Perl file were written.
   All data and information are stored in four mySQL database tables named
   studentAccountTable, submissionTable, instructorTable and assignmentTable.

(4) The project used Java servlets to generate a lot of html pages. The Course class
    can create nine html pages. These html pages are used to get parameters for
    related functionalities and submit the parameters into the Course class. The
    Student class can create four html pages. These html pages are used to get
    parameters for related functionalities and submit them into Student class.

(5) The application web demo has been set up. It works fine in using Netscape
    Communicator 4.0+ and Microsoft Internet Explorer 4.0+.
9. Bibliography

(1) Jin Conallen. Building Web Applications with UML. ADDISON-WESLEY, 1999
(4) Andrew S. Tanenbaum, Computer Networks, Prentice Hall, 1995
(6) Chris Taylor, Tim Kimmet: Core Java Web, Prentice_Hall, Inc. 1999
(7) Marty Hall: Core Servlets and JavaServer Pages Server, Prentice Hall PTR, Upper Saddle River, NJ 07458, 1999
(10) Randy Jay Yarger et al: MySQL and mSQL, O'REILLY, 1999
(12) JIM Conallen: Building Web Applications with UML, Addison-Wesley, 1999
Appendix A

Design and Implementation of a Web Server

Application for Students to Submit Assignments

(Requirements)

By

Yuehua Yang

Department of Computer Science, University of Montana

1. Overview Statement

The purpose of this project is to create a web server application; which will allow students in a university course to submit assignment files.

2. Clients and Users

The clients are students and instructors in a university course.

3. Set up a course

3.1 A course is associated with the following information:

3.1.1 The course number.

3.1.2 The section number.

3.1.3 The instructor first name
3.1.4 The instructor last name

3.1.5 The course id

3.1.5.1 If the instructor chooses an id, which is identical to the id of an existing account for the other course, the instructor will be required to choose another user id.

3.1.6 The course password

3.1.6.1 The password will be covered in section 3.4

3.1.6.2 The instructor can modify the course password

3.1.7 The number of assignments in the course.

3.1.8 The maximum file size in each assignment

3.1.8.1 The instructor can modify the file size after he/she creates the new course

3.2 For each course, section and assign number, there is a directory in server computer to store student assignments.

3.3 The instructor can delete a course.

3.4 The password shall be at least 7 characters long and must contain at least 3 of the following 4 categories of characters: lower case letters, upper case letters, digits, no alphanumeric characters.

3.4.1 The password will not be stored. Instead, a message digest (such as MD5) of the password will be stored.

4 Set Up a Student Account

4.1 A student account contains the following information:
4.1.1 The student first name.

4.1.2 The student last name.

4.1.3 The student university-assigned id number.

4.1.4 The student user id.

4.1.5 The student password.

4.1.5.1 The password will be covered in section 3.4.

4.2 A student can create an account by entering the above information.

4.2.1 If the student enters a university-assigned id number, which corresponds, to an existing account, account creation will fail.

4.2.2 If a student chooses a student user id, which is identical to the id of an existing account for that course, the student will be requested to choose another id.

4.3 A student can login into an existing account by supplying the student user id and the password.

4.4 If a student is logged in, the student may change his/her password.

4.5 The instructor can create a list of student accounts by supplying a tab-delimited text file, which contains the information for each account.

4.6 The instructor can delete a student account.

4.7 The instructor can update a student password if the student forgot it.

5 Submit a File

5.1 The students who have set up their account can submit their assignments.

5.2 Provide a user-friendly interface which helps a student to finish submitting his/her assignment step by step.
5.3 Check student authorization

5.3.1 If a student id and password are correct, the system allows the student to submit his/her file.

5.4 Get a file from student's computer.

5.4.1 Check file size

5.4.1.1 If file size is larger than limitation, the system will not save the file. The system will ask student to submit a smaller file.

5.4.2 Save file

5.4.2.1 The system will save the file into the course directory in the server machine.

5.4.2.1.1 The name of directory is listed in 4.2

5.4.2.2 The file name will be saved as user id + file name in order to prevent from one student's file from rewriting another student's file when the file names are the same.

5.4.2.3 The system allows a student to rewrite his/her old file if he/she submits a file twice.

5.4.2.3.1 When a student rewrites his/her submission file, the system will notify the student the file has been rewritten.

5.5 If the system saves the file successfully, it will give the student a receipt on the web and remind the student to save or print the receipt.

5.5.1 The receipt contains the following information

5.5.1.1 The student first name

5.5.1.2 The student last name
5.5.1.3 The course number
5.5.1.4 The section number
5.5.1.5 The assignment number
5.5.1.6 The file name
5.5.1.7 The file size
5.5.1.8 The time that the system received the file
5.5.1.9 The instructor name

5.5.2 The information is listed in the receipt will also be saved in database in order that the instructor can verify it.

6 The instructor check student assignments

6.1 The system will check the course id and the password.

6.1.1 If the course id and the password are correct, the system allows the instructor to login and to check students’ assignments information.

6.1.1.1 The instructor can check students’ assignments using following methods:

6.1.1.1.1 Check all assignments for:

6.1.1.1.1.1 All students

6.1.1.1.2 Specify a student

6.1.1.1.2 Specify a assignment for:

6.1.1.1.2.1 All students

6.1.1.1.2.2 Specify a student

6.2 The instructor can check and verify all the information listed in 5.5.1
7 User Interface

7.1 The system will use user-friendly interface.

8 System Attributes

8.1 The system will be developed and run under Linux/Unix system.

8.2 The following system software will be used to support the application development:

8.2.1 Apache
8.2.2 JSDK2.0
8.2.3 JDK
8.2.4 MySQL
8.2.5 JDBC

8.3 The system will be installed on a Linux PC provided by the CS Department.

The CS department will provide a PC on the network with a standard distribution of Linux installed. The above system software will be installed as part of the project.
## Appendix B

### High-Level Use Cases for a Web server Application for Students to Submit Assignments

<table>
<thead>
<tr>
<th>Use Case:</th>
<th>Set up a course</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actors:</td>
<td>Instructor user</td>
</tr>
<tr>
<td>Type:</td>
<td>Primary</td>
</tr>
<tr>
<td>Include:</td>
<td>Set up file limitation</td>
</tr>
<tr>
<td>Description:</td>
<td>The user specifies a course via inputting related course information and instructor information. The system is now ready to allow students to set up accounts.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use Case:</th>
<th>Set up file limitation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actors:</td>
<td>Instructor user</td>
</tr>
<tr>
<td>Type:</td>
<td>Secondary</td>
</tr>
<tr>
<td>Description:</td>
<td>The user specifies file size for each assignment. The system is ready to save students' file.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use Case:</th>
<th>Change the course password</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actors:</td>
<td>Instructor user</td>
</tr>
<tr>
<td>Type:</td>
<td>Option</td>
</tr>
<tr>
<td>Description:</td>
<td>The user can change the course password after the course is set up. The system is ready to accept new password.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use Case:</th>
<th>Update a student password</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actors:</td>
<td>Instructor user</td>
</tr>
<tr>
<td>Type:</td>
<td>Option</td>
</tr>
<tr>
<td>Description:</td>
<td>The user can change a student password if the student forgets the password. The system is ready to accept new password.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use Case:</th>
<th>Set up a student account</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actors:</td>
<td>Student user</td>
</tr>
<tr>
<td>Type:</td>
<td>Primary</td>
</tr>
</tbody>
</table>
Description: The user can specify his/her account via inputting his/her personal identification information. The system is ready to allow students to submit their files and to allow students to change their passwords.

User Case: Submit a assignment
Actors: Student user
Type: Primary
Include: Give a receipt of file
Description: The user can submit his/her file online. After the system receives and saves the file, it will give a receipt of file.

User Case: Change student password
Actors: Student user
Type: Option
Description: The user can change his/her account password. The system is ready to accept new password

User Case: Check student assignments
Actors: Instructor user
Type: Primary
Description: The user can check student submission information online.

User Case: Delete a student account
Actors: Instructor user
Type: Primary
Description: The user can delete a student account via specifying the student university-assigned id. The system will remove the student account from database.

User Case: Delete a course
Actors: Instructor user
Type: Primary
Description: The user can delete a course via specifying the course name and section. The system will remove the course from database.
Appendix C

A Web Server Application for Students to Submit Assignments
Expanded Use Cases

Use Case: Set up a course
Section: Main

Use Case: Set up a course
Actors: Instructor user
Purpose: Set up a course for students to submit assignments
Overview: The user inputs course identification information. The user sets up the course user id, the password, and the file size limitation. The system is ready to do student account setting up
Type: Primary and Essential
Cross Reference: Functions: R3.1, R3.2, R3.4

Typical Course of Event

<table>
<thead>
<tr>
<th>Actor Action</th>
<th>System Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. This use case begins when user is ready to set up a course. The user is assumed to be authorized to access instructor web pages. See section &quot;Check user authorization&quot;</td>
<td>3. System shows a form to ask the user to input the course number, section number, the instructor first name, last name, maximum assignment number, course id and password</td>
</tr>
<tr>
<td>2. The user selects to set up a course</td>
<td>5. The system shows a form to ask the user to input file size.</td>
</tr>
<tr>
<td>4. User inputs the course number, The section number, the instructor first name, last name, The course id, password, and the number of assignments in the course.</td>
<td></td>
</tr>
</tbody>
</table>

37
6. User specifies a file size limitation: see Section Set up file size limitation.

7. The system saves the file size limitation into database table. The system give a information to show that the course has been set up.

Section: Check user authorization

Typical Course of Events

<table>
<thead>
<tr>
<th>Action</th>
<th>System Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. The user goes to the instructor home page</td>
<td>2. System shows a form to ask the user to input the user id and password</td>
</tr>
<tr>
<td>3. The user inputs the user id and Password</td>
<td>4. System checks the user id and password. If both of user id and password are correct, system shows the instructor home page, otherwise, gives a &quot;authorization fail&quot; information and does not show the home page</td>
</tr>
</tbody>
</table>

Section: Set up file size limitation

Typical Course of Events

<table>
<thead>
<tr>
<th>Actor Action</th>
<th>System Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. The user selects to set up file size limitation.</td>
<td>2. System shows a form to ask the user input File size.</td>
</tr>
<tr>
<td>3. The user inputs the file size value</td>
<td>4. System saves the value into the database.</td>
</tr>
</tbody>
</table>
Use Case: Change the course password

Section: Main

Use Case: Change the course password
Actor: Instructor user
Purpose: Change an existing course password
Overview: The user can change an existing course password. The system is ready to accept new password
Type: Option
Cross Reference: Functions: R3.1.6, R3.4

Typical course of Events

<table>
<thead>
<tr>
<th>Actor Action</th>
<th>System Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. The user selects to change the course password.</td>
<td>2. System shows a form to ask the user to input course id, currently password, and new password.</td>
</tr>
<tr>
<td>3. The user inputs the course id, currently Password and new password.</td>
<td>4. System checks the user authorization, then saves the new password into database.</td>
</tr>
</tbody>
</table>

Use Case: Set up a student account

Section: Main

Use Case: Set up a student account
Actors: Student user
Purpose: Set up a student account for a course to submit assignment files
Overview: The user inputs his/her personal identification information, sets up a user id and password. The system is ready to accept the user file submission.
Type: Primary and Essential
Cross Reference: Functions: R4.1, R4.2, R4.3, R4.4, R3.4

Typical Course of Events
User Case: Change student password  
Section: Main

Use Case: Change student password  
Actors: Student user  
Purpose: Change a existing student password  
Overview: The user can change a existing student password. The system is ready to accept a new password  
Type: Option  
Cross Reference: Functions: R4.4, R3.4

Typical Course of Events

Actor Action  
1. The user selects to change student password  
3. The user input user id, currently password and new password.  

System Response  
2. System shows a form to ask the user to input user id and currently password and new password  
4. System check the user authorization and save the new password into database

Use Case: Submit an assignment  
Section: Main

Use Case: Submit a assignment  
Actors: Student user  
Purpose: Submit a assignment via online  
Overview: The user inputs course number, section number, user id, password, and submit a assignment via online. The file is saved in server machine. The system gives the user a receipt after it has received and saved a file.
Typical Course of Events

<table>
<thead>
<tr>
<th>Actor Action</th>
<th>System Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. The user selects to submit a Assignment</td>
<td>2. System shows a form to ask the user to input user id, course number, section number, password and assignment number</td>
</tr>
<tr>
<td>3. The user inputs user id, password, course number, section number, assignment number</td>
<td>4. System checks the user authorization, then asks the user to input file name</td>
</tr>
<tr>
<td>5. The user input file name with directory using &quot;Browse&quot; button</td>
<td>6. System checks file size, then saves the file into the server machine</td>
</tr>
<tr>
<td></td>
<td>7. System gives the user a receipt, which includes the user name, course number, section number, file name, size, the time that the system received the file and the instructor name. The system saves the same information into the database.</td>
</tr>
</tbody>
</table>

Use Case: Check student assignment

Section: Main

Use Case: Check student assignment
Actors: Instructor user
Purpose: Check student assignment information online
Overview: The user can view all of the student submission information online. The submission includes the student name, assignment number, file name, size, and submission time.
Type: Primary and Essential
Cross Reference: Functions: R6.1, R6.2, R5.5.1

Typical Course of Events

<table>
<thead>
<tr>
<th>Actor Action</th>
<th>System Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. The user selects to check student Assignment submission information.</td>
<td>2. System shows a form to ask the user to input the course id, password, one</td>
</tr>
</tbody>
</table>
3. The user inputs the course number, section number, course id and password, and selects one or all assignments, one or all students.

4. System checks the user authorization, if user selects all students the system will print all students' submission information for one or all assignments. If user selects one student, the system shows all students' name and university-assigned id and asks the user to select one student.

5. The user selects one student.

6. System prints one student's submission for one or all assignments.

Use Case: Delete a student account

Section: Main

Use Case: Delete a student account
Actors: Instructor User
Purpose: Delete a student account online
Overview: The user can delete a student account after he/she inputs the course identification, student's university-assigned id.
Type: Primary and Essential
Cross Reference: Functions: R4.6

Typical Course of Events

Actor Action

1. The user selects to delete a student account

2. System shows a form to ask the user input course identification and all students' university-assigned id and names to ask the user to select one

3. The user inputs the course number, section number, course id, password, and select one student

4. System checks the user authorization, then delete the student account from database.

Use Case: Delete a course

Section: Main

Use Case: Delete a course
Actors: Instructor User
Purpose: Delete a course online
Overview: The user can delete a course from database after he/she inputs the course identification
Type: Primary and Essential
Cross Reference: Functions: R3.3

Typical Course of Event

Actor Action | System Response
--- | ---
1. The user selects to delete a course | 2. System shows a form to ask the user to input the course identification
3. The user inputs the course number, section number, course id, and Password | 4. System checks the user authorization, then deletes the course from the database

Use Case: Update a student password

Section: Main

Use Case: Update a student password
Actor: Instructor user
Purpose: Update a existing student password if the student forgot the password
Overview: The user can change an existing student password. The system is ready to accept new password
Type: Option
Cross Reference: Functions: R4.7, R3.4

Typical course of Events

Actor Action | System Response
--- | ---
1. The user selects to update a student password. | 2. System shows a form to ask the user to input course id, currently password.
3. The user inputs the course id, currently password. | 4. System checks the user authorization, then shows a list of student names and University-assigned ids in database.
5. The user selects a student's university- | 6. System shows a form to ask the user to
assigned id

Input new password.

7. The user inputs a new password.

8. System checks the new password.
Then the system save the password into database.
Appendix D
Class Diagram of The Project(1)
Appendix D

Class Diagram of The Project(2)
Attributes and Methods of Course Class

Course
userid: String
password1: String
password2: String
coursenum1: String
coursesecl: String
lastName : String
firstName : String
assign : String
password : String
Assign : String
studentS : String
tablekey : String
assignment : String
studentFirstName: String
studentLastName : String
newpassword1: String
newpassword2: String
filesize: String
yestodel: String
maxAssignN: String
assignV: String

init(ServletConfig)
doPost(HttpServletRequest, HttpServletResponse)
enterCourseSelection(PrintWriter)
getCourseSection(HttpServletRequest, HttpServletResponse)
enterCourseAuthorization(PrintWriter)
getCourseAuthorization(PrintWriter, HttpServletRequest req, HttpServletResponse)
createNewCoursePage(PrintWriter)
coursePage(PrintWriter, HttpServletRequest, HttpServletResponse)
coursePage(PrintWriter, HttpServletRequest, HttpServletResponse)
coursePage(PrintWriter, HttpServletRequest, HttpServletResponse)
coursePage(PrintWriter, HttpServletRequest, HttpServletResponse)
coursePage(PrintWriter, HttpServletRequest, HttpServletResponse)
coursePage(PrintWriter, HttpServletRequest, HttpServletResponse)
coursePage(PrintWriter, HttpServletRequest, HttpServletResponse)
coursePage(PrintWriter, HttpServletRequest, HttpServletResponse)
coursePage(PrintWriter, HttpServletRequest, HttpServletResponse)
coursePage(PrintWriter, HttpServletRequest, HttpServletResponse)
coursePage(PrintWriter, HttpServletRequest, HttpServletResponse)
coursePage(PrintWriter, HttpServletRequest, HttpServletResponse)
coursePage(PrintWriter, HttpServletRequest, HttpServletResponse)
coursePage(PrintWriter, HttpServletRequest, HttpServletResponse)
coursePage(PrintWriter, HttpServletRequest, HttpServletResponse)
coursePage(PrintWriter, HttpServletRequest, HttpServletResponse)
coursePage(PrintWriter, HttpServletRequest, HttpServletResponse)
coursePage(PrintWriter, HttpServletRequest, HttpServletResponse)
coursePage(PrintWriter, HttpServletRequest, HttpServletResponse)
coursePage(PrintWriter, HttpServletRequest, HttpServletResponse)
coursePage(PrintWriter, HttpServletRequest, HttpServletResponse)
Appendix D

Class Diagram of The Project(3)
Attributes and Methods of Student Class

```
Student

userid: String
password1: String
password2: String
tablekey: String
ssNumber1: String
ssNumber2: String
ssNumber3: String
lastName: String
firstName: String
password: String
newpassword1: String
newpassword2: String
tablekey: String
filesize: String
ssN: String

init(java.servlet.ServletConfig config)
doPost(HttpServletRequest, HttpServletResponse)
checkSSN(String, String): boolean
enterStudAuthorization(PrintWriter)
createNewAccountPage(PrintWriter)
getNewAccountPage(PrintWriter, HttpServletRequest, HttpServletResponse)
createParserPage(java.io.PrintWriter)
getFileSize(tablekey)
createStudentNewPasswordPage(PrintWriter)
getStudentNewPasswordPage(PrintWriter, HttpServletRequest, HttpServletResponse)
saveAccount(PrintWriter)
saveStudPassword(PrintWriter)
selectNewCourse(PrintWriter)
checkUserid(String, String): boolean
```

47
Appendix D

Class Diagram of The Project(4)
Attributes and Methods of SubmissionServlet Class

```
SubmissionServlet

checkPasswd(String): boolean
confirmPasswd(String, String): boolean
doPost(HttpServletRequest, HttpServletResponse)
enCoding(String): String
init(ServletConfig)
isAuthorized(String, String, String): Boolean
isEmpty(String): Boolean
```

Attributes and Methods of Submission Class

```
Submission

lastName: String
firstName: String
insfirstname: String
inslastname: String

init(ServletConfig)
getStudent(String, String)
getInstructor(String)
saveFileInformation(PrintWriter)
doPost(HttpServletRequest, HttpServletResponse)
```
Appendix E
Sequence Diagrams for the Project

Sequence Diagram for Set up a Course
1: instructorHome(index.html) 2: Course 3: newCoursePage 4: SubmissionServlet 5: FileSizePage 6: instructorTable 6: assignmentTable
Sequence diagram for use case Check student assignment

(1: instructorHome(index.html) 2: Course 3: AssignmentPage 4: SubmissionServlet 5: StudentIDPage 6: submissionTable)
Sequence diagram for use case Change the course password
Sequence diagram for use case Change file size
Sequence diagram for use case Update a student password

Sequence diagram for use case Delete a student account

(1: instructorHome(index.html) 2: Course 3: DeleteAccountPage 4: SubmissionServlet 5: StudentAccountPage 6: submissionTable 7: studentAccountTable)
Sequence diagram for use case Delete a course

1: instructorHome(index.html) 2: Course 3: DeleteCoursePage 4: SubmissionServlet 5: studentAccountTable 7: submissionTable 8: assignmentTable 9: instructorTable
Sequence diagram for use case Set up a student account
Sequence diagram for use case Submit a assignment
(1: studentHome(index_1.html) 2: Student 3: SubmissionAuthorizedPage 4: SubmissionServlet 5: ParserPage
6: Parser/perl) 7: Submission 8: submissionTable 9: instructorTable 10: assignmentTable 11: studentAccountTable)
Sequence diagram for use case Change student password

(1: instructorHome(index.html) 2: Course 3: newCoursePage 4: SubmissionServlet 5: instructorTable 6: assignmentTable)
Appendix F

Some of Systems Installation and Configuration

This document describes how to install and configure the software which are used in the project. These software are:

1. Set up the Java servlet application of the project
2. Set up Apache Authentication
3. Set up JDBC Driver for MySQL
4. MySQL Installation and Use
5. Configuration of Apache server

After you follow the instruction of the document to install and configure these software, you should execute the application easily in your machine.

1. Set up the Java servlet application system of the project

(1) Make a directory, which installs project source codes. Suppose it is /home/research.

(2) Make another directory, which will store student assignment source files. Suppose it is /home/research/studentfiles/.

(3) Modify saveFileSize(out) and removeCourse(out) functions in Course.java and let fileDir="/home/research/studentfiles". Modify parser.cgi and let $MAIN_DIRECTORY="home/research/studentfiles".
(4) When an instructor creates a course and section, the sub directories of storing students' files will be created automatically by the application.

For example, if an instructor creates course number cs 446 and section 1, and maximum assignment 3, the application will create following sub-directory:

/home/research/studentfiles/cs446-1/Assign1
/home/research/studentfiles/cs446-1/Assign2
/home/research/studentfiles/cs446-1/Assign3

(5) Copy project.tar into /home/research/

(6) Untar project.tar. It will create following directory:

/home/research/com/submission/servlets/
/home/research/com/submission/cgi-bin/
/home/research/com/submission/students/
/home/research/com/submission/instructor-bin/
/home/research/com/submission/html/

(7) Add the directory of /home/research to your classpath:

For an example:

You can add the lines

    CLASSPATH=$CLASSPATH:/home/research

    export CLASSPATH

to your .bashrc file

2. Set up Apache Authentication

(1) First, you must have Apache and the htpasswd utility installed on your Linux machine. If you do not, you can download the latest copy of Apache (I use version 1.3.12
installed in /home/apache/apache_1.3.12/) from the Apache Organization web site and install it on your machine. htpasswd comes with Apache.

(2) Go to /home/apache/apache_1.3.12/src/support/

In order to set user authentication using a user name and a password, we need to follow two steps. First, we create a file containing the user names and passwords. Then, we inform the server, which resources are to be protected, and which users are allowed to access them. The first step towards configuring authentication is to set up a list of users and their corresponding passwords using htpasswd commands. This list is saved in a file called, for example, .htpasswd. The .htpasswd file for the instructors will be saved into /home/research/com/submission/instructor-bin/. The .htpasswd file for students will be saved into /home/research/com/submission/students/.

The general htpasswd commands are:

htpasswd [-cmdps] passwordfile username

htpasswd -b[cmdps] passwordfile username password

In the commands:

-c Create a new file.

-m Force MD5 encryption of the password.

-d Force CRYPT encryption of the password (default).

For an example, if you want to create a .htpassword file in /home/research/com/submission/instructor-bin and add user id for John, password abc-123, you can type the following command:

htpasswd -c -b /home/research/com/submission/instructor-bin/.htpasswd john abc-123
The -c flag is only used the first time we use htpasswd to create a new users file. Other users can be added to the existing file in the same way, except that the -c argument is not needed since the file already exists.

If this option is used when adding other users, the file is over written and the old users are lost.

A detail descrip can be found in http://search.apache.org/

3. Set up JDBC Driver for MySQL

(1) Untar mm.mysql.jdbc-2.0pre5.tar.gz, which is included in project tar file. This will create the mm.mysql.jdbc-2.0pre5 directory. In the mm.mysql.jdbc-2.0pre5 directory, there are mysql_1_comp.jar, mysql_1_uncomp.jar, mysql_2_comp.jar, mysql_2_uncomp.jar, mysql_both_comp.jar and mysql_both_uncomp.jar.

(2) Add one of jar to your classpath.

For instance, you can add the lines

CLASSPATH=$CLASSPATH:/home/research/mm.mysql.jdbc-2.0pre5/mys

mysql_1_comp.jar

export CLASSPATH

to your .bashrc file.

4. MySQL Installation and Use

(1) Get mysql source codes. you can get mysql source codes in http://www.mysql.com.

In this project, I get a mysql version named mysql-3.22.32.tar.gz. I will describe the installation based on the version:

shell> gunzip < mysql-3.22.32.tar.gz | tar xvf -
shell> ln -s mysql-3.22.32 mysql

shell> cd mysql

shell> scripts/mysql_install_db

shell> bin/safe_mysqld &

A detail installation document can be found in http://www.mysql.com/doc/

(2) create a database and 4 tables for the project

All operation is under root:

(I) create research database

If mysql server does not run:

shell> mysql/bin/safe_mysqld&

then:

shell> mysql

mysql> CREATE DATABASE research;

(II) Create 4 tables under research database

mysql>use research;

mysql> create table instructorTable (INSFIRSTNAME CHAR (20), INSLASTNAME CHAR (20), COURSEID CHAR (20), PASSWD CHAR (50), TABLEKEY CHAR (20));

mysql> create table assignmentTable (TABLEKEY CHAR (20), MAXASSIGN INT (6), FILESIZE CHAR (20));

mysql> create table studentAccountTable (STUDFIRSTNAME CHAR (20), STUDLASTNAME CHAR (20), USERID CHAR (20), PASSWD CHAR (50), STUDSSN CHAR (40), TABLEKEY CHAR (20));
mysq1> create table submissionTable (USERID CHAR (20), FILENAME CHAR (40),
ASS CHAR (20), DATE CHAR (30), TABLEKEY CHAR (20));

Right now, the tables have created and ready to save information for the project

5. Configuration of Apache server

(1) Installation

Go to http://www.apache.org/ web page and download apache server
source codes.

In this project, I get and use apache-1.3.9-4.i386.rpm version.

I will descript how to install and configure apache server using this version.

All of operations are executed under root:

(l) rpm -i apache-1.3.9-4.i386.rpm

It will create /etc/httpd/ for operation and /home/httpd/for default server directory.

Now, you can run apache server by

shell>httpd start

if you want to stop apache server,

shell>httpd stop

(2) Configuration

(l) Modify srm.conf:

A more complete srm.conf file is included in Appendix H.

(a) Go to /etc/httpd/conf/.

suppose the project source codes are put in /home/research/com/submission/

(b) Add and modify following codes in srm.conf:

<Directory /home/research/com/submission/instructor-bin>
order allow, deny
allow from all

AuthName "Assignment Submission Site"

AuthType Basic

AuthUserFile /home/research/com/submission/instructor-bin/.htpasswd

<Limit GET POST>
require valid-user
</Limit>
</Directory>

<Directory /home/research/com/submission/students>
order allow, deny
allow from all

AuthName "Assignment Submission Site"

AuthType Basic

AuthUserFile /home/research/com/submission/students/.htpasswd

<Limit GET POST>
require valid-user
</Limit>
</Directory>

AddHandler cgi-script .cgi

Alias /students/ /home/research/com/submission/students/

Alias /instructor-bin/ /home/research/com/submission/instructor-bin/

ScriptAlias /cgi-bin/ /home/research/com/submission/cgi-bin/
(II) modify jser.conf

A more complete jser.conf file is included in Appendix I.

(a) go to /etc/httpd/conf/jserv/

(b) open jser.conf file

© add following codes:

ApJServMount /servlets /research
ApJServMount /servlet /research

save the file

(III) Modify jser.properties

A more complete jser.properties file is included in Appendix J.

go to /etc/httpd/conf/jserv/

open jser.properties file

add following codes:

wrapper.bin=/home/myJDK/jdk1.2/bin/java

wrapper.classpath=/home/myJDK/jdk1.2/lib/dt.jar

wrapper.classpath=/home/myJDK/jdk1.2/lib/tools.jar

wrapper.classpath=/home/myJDK/jdk1.2/jre/rt.jar

wrapper.classpath=/home/myJDK/JSDK2.0/lib/jsdk.jar

wrapper.classpath=/home/myJSDK/mm.mysql.jdbc-2.0pre5

wrapper.classpath=/home/research

zones=research
research.properties=/etc/httpd/conf/jserv/zoon.properties

save the file

(IV) modify zoon.properties

(a) Remove zoon.properties in /etc/httpd/conf/jserv/

(b) Copy zoon.properties file in project.tar into /etc/httpd/conf/jserv/
Appendix G

Master's Project Proposal

Design and Implementation of a Web Server

Application for Students to Submit Assignments

By

Yuehua Yang

Department of Computer Science, University of Montana

1 Introduction

The Internet and the World Wide Web have experienced explosive growth over the past decade and are now positioned to provide a wide range of service. Computers have become and will continue to be part of our everyday lives. More and more individuals, organization and business use network to get and send information. Generally, when a student takes a course in a university, he/she has to finish the required assignments and submit them to the instructor. In computer-related courses, assignments are often submitted electronically in the form of computer files. Traditionally, when a student wants to submit an assignment electronically, he/she has to turn in a floppy disk, to send
the file to the instructor by email or to copy the file to a course directory; which has been set up for that purpose. However, there are four major problems with these methods of submitting assignments:

(5) If the floppy disk submission method is used, the instructor has to retrieve the assignment from one floppy disk per student, which will take the instructor a lot of time to do that because the machine reads a file from a floppy disk much slower than from hard disk. Also, both students and the instructor have to use the same format system to write and read floppy disk.

(1) If the email submission method is used, the instructor has to deal with a large number of emails, which contain student assignment files. It is not surprising to an instructor that he/she may receive more than one hundred emails in a 50-student course for an assignment. A student may send several modified assignment files for an assignment. The instructor has to save these files to a directory and verify and delete duplicate submissions.

(2) If the copy-file submission method is used, the student must have an account on the machine where the instructor asks students copy their files to a course directory. The students can go to the course directory and view other students file information, such as who have submitted their files and when the files have be submitted. Because of this security problem, most instructors only ask students to copy their executable files to this directory. The students have to submit a hardcopy to instructor.

(3) The students cannot verify whether or not the instructor has received their files when students submit their files by email. Students often submit assignments to the wrong
email address. It is very often situation that when a student is told that he/she has failed to finish an assignment, he/she has to explain that he/she has submitted his/her file and did not know that the file is missing.

Obviously, these disadvantages significantly decrease the efficiency for both instructor and student to send and receive a course assignment. Therefore, there is need to improve the efficiency and reliability of the assignment submission process.

In this project, a web server application system will be designed and implemented. The purpose of this application is to allow students in a university course to submit assignment files and allow an instructor to edit and manage a course for the purpose of submitting assignments.

The application system will be protected by two levels of security. First level is web access control level. A system administrator (or other person with root access to the server computer) sets up first level security. Each instructor has a user id and password to access the instructor web page. The second security level is the user personal security level for students and the course security level for instructors. The authorized student users and instructor users will set up second level security.

For student users, the system allows each student to set up an account for assignment submission; which includes the user id and the password. Students can submit their assignments after they set up their account. The system also allows students to edit their password.

For instructor users, the system allows the instructor set up a course for students to submit their assignments. The system will give students receipts after they submit their
assignments successfully. The system allows the instructor to set up and modify the assignment size limitation, to edit course/student password, to check student submission information online, and to delete a student account. The system also allows the instructor to delete his/her course account.

2 Method of Implementation

(1) Programming and System Support

All programs for the web-based client-server application will be developed under the Linux operating system. Java servlet programs will be written in order to implement the application. The programs will be written in Java, html or perl.

(2) Demo Environment

The programs can be run on any machine; which has, the required tools installed. The required tools are listed in the requirements document. The client side of the application can be demonstrated in Netscape 4.0+ or Internet Explorer 4.0 +.

3 The plan for the project

The project will be finished based on following plan:

(1) Requirements phase:

A requirement document and use cases will be elicited

(2) Design phase:

A class diagram and sequence diagrams will be written in the design phase

(3) Implementation phase:
Programming will be finished

(4) Summary phase:

The paper will be finished
Appendix H

A Real Configuration File of srm.conf

##
## srm.conf -- Apache HTTP server configuration file
## This file is modified by Yuehua Yang
##
## With this document, you define the name space that users see of your http
## server. This file also defines server settings which affect how requests are
## serviced, and how results should be formatted.

# See the tutorials at http://www.apache.org/ for
# more information.

# Originally by Rob McCool; Adapted for Apache

<Directory /home/yuehua/research/com/submission/instructor-bin>
# Options Includes Indexes FollowSymLinks
order allow,deny
allow from all
AuthName "Yuehua Yang Site"
AuthType Basic
AuthUserFile /home/yuehua/research/com/submission/instructor-bin/.htpasswd
<Limit GET POST>
require valid-user
</Limit>
</Directory>

<Directory /home/yuehua/research/com/submission/students>
# Options Includes Indexes FollowSymLinks
order allow,deny
allow from all
AuthName "Yuehua Yang Site"
AuthType Basic
AuthUserFile /home/yuehua/research/com/submission/students/.htpasswd
<Limit GET POST>
require valid-user
</Limit>
</Directory>
# DocumentRoot: The directory out of which you will serve your
# documents. By default, all requests are taken from this directory, but
# symbolic links and aliases may be used to point to other locations.

DocumentRoot /home/yuehua/research/com/submission/html

# UserDir: The name of the directory which is appended onto a user's home
# directory if a ~user request is received.

UserDir public_html

# DirectoryIndex: Name of the file or files to use as a pre-written HTML
# directory index. Separate multiple entries with spaces.

DirectoryIndex index.html index.shtml index.cgi

# FancyIndexing is whether you want fancy directory indexing or standard

FancyIndexing on

# AddIcon tells the server which icon to show for different files or filename
# extensions

AddIconByEncoding (CMP,/icons/compressed.gif) x-compress x-gzip

AddIconByType (TXT,/icons/text.gif) text/*
AddIconByType (IMG,/icons/image2.gif) image/*
AddIconByType (SND,/icons/sound2.gif) audio/*
AddIconByType (VID,/icons/movie.gif) video/*

AddIcon /icons/binary.gif .bin .exe
AddIcon /icons/binhex.gif .hqx
AddIcon /icons/tar.gif .tar
AddIcon /icons/world2.gif .wrl .wrl.gz .vrm .vrm .iv
AddIcon /icons/compressed.gif .Z .z .tgz .gz .zip
AddIcon /icons/a.gif .ps .ai .eps
AddIcon /icons/layout.gif .html .shtml .htm .pdf
AddIcon /icons/text.gif .txt
AddIcon /icons/c.gif .c
AddIcon /icons/p.gif .pl .py
AddIcon /icons/f.gif .for
AddIcon /icons/dvi.gif .dvi
AddIcon /icons/uuencoded.gif .uu
AddIcon /icons/script.gif .conf .sh .shar .csh .ksh .tcl
AddIcon /icons/tex.gif .tex
AddIcon /icons/bomb.gif core
AddIcon /icons/back.gif 
AddIcon /icons/hand.right.gif README
AddIcon /icons/folder.gif ^DIRECTORY^ 
AddIcon /icons/blank.gif ^BLANKICON^ 

# DefaultIcon is which icon to show for files which do not have an icon 
# explicitly set.

DefaultIcon /icons/unknown.gif 

# AddDescription allows you to place a short description after a file in 
# server-generated indexes. 
# Format: AddDescription "description" filename 

# ReadmeName is the name of the README file the server will look for by 
# default. Format: ReadmeName name 
#
# The server will first look for name.html, include it if found, and it will 
# then look for name and include it as plaintext if found. 
#
# HeaderName is the name of a file which should be prepended to 
# directory indexes.

ReadmeName README
HeaderName HEADER 

# IndexIgnore is a set of filenames which directory indexing should ignore 
# Format: IndexIgnore name1 name2...

IndexIgnore .??* *~ *# HEADER* README* RCS 

# AccessFileName: The name of the file to look for in each directory 
# for access control information.

AccessFileName .htaccess 

# TypesConfig describes where the mime.types file (or equivalent) is 
# to be found.

TypesConfig /etc/mime.types 

# DefaultType is the default MIME type for documents which the server 
# cannot find the type of from filename extensions.

DefaultType text/plain 

# AddEncoding allows you to have certain browsers (Mosaic/X 2.1+) uncompress 
# information on the fly. Note: Not all browsers support this.
AddEncoding x-compress Z
AddEncoding x-gzip gz

# AddLanguage allows you to specify the language of a document. You can
# then use content negotiation to give a browser a file in a language
# it can understand. Note that the suffix does not have to be the same
# as the language keyword --- those with documents in Polish (whose
# net-standard language code is pl) may wish to use "AddLanguage pl .po"
# to avoid the ambiguity with the common suffix for perl scripts.
AddLanguage en .en
AddLanguage fr .fr
AddLanguage de .de
AddLanguage da .da
AddLanguage el .el
AddLanguage it .it

# LanguagePriority allows you to give precedence to some languages
# in case of a tie during content negotiation.
# Just list the languages in decreasing order of preference.
LanguagePriority en fr de

# Redirect allows you to tell clients about documents which used to exist in
# your server's namespace, but do not anymore. This allows you to tell the
# clients where to look for the relocated document.
# Format: Redirect fakename url

# Aliases: Add here as many aliases as you need (with no limit). The format is
# Alias fakename realname

# Note that if you include a trailing / on fakename then the server will
# require it to be present in the URL. So "/icons" isn't aliased in this
# example.
Alias /icons/ /home/httpd/icons/
Alias /students/ /home/yuehua/research/com/submission/students/
Alias /html/ /home/yuehua/research/com/submission/html/
Alias /instructor-bin/ /home/yuehua/research/com/submission/instructor-bin/

BrowserMatch "Mozilla/2" nokeepalive
BrowserMatch "MSIE 4\..0b2;" nokeepalive downgrade-1.0 force-response-1.0

# The following directive disables HTTP/1.1 responses to browsers which
# are in violation of the HTTP/1.0 spec by not being able to grok a
# basic 1.1 response.
BrowserMatch "RealPlayer 4\0" force-response-1.0
BrowserMatch "Java/1\0" force-response-1.0
BrowserMatch "JDK/1\0" force-response-1.0
Appendix I

A Real Configuration File of jserv.conf

#******************************************************************************
# Apache JServ Configuration File
# This file is modified by Yuehua Yang
#******************************************************************************

# Note: this file should be appended or included into your httpd.conf

# Tell Apache on Win32 to load the Apache JServ communication module
# LoadModule jserv_module modules/ApacheModuleJServ.dll

# Tell Apache on Unix to load the Apache JServ communication module
# For shared object builds only!!!
# LoadModule jserv_module /usr/lib/apache/mod_jserv.so

<IfModule mod_jserv.c>

# Whether Apache must start Apache JServ or not (On=Manual Off=Autostart)
# Syntax: ApJServManual [on/off]
# Default: "Off"
ApJServManual off

# Properties filename for Apache JServ in Automatic Mode.
# In manual mode this directive is ignored
# Syntax: ApJServProperties [filename]
# Default: "/conf/jserv.properties"
ApJServProperties /etc/httpd/conf/jserv/jserv.properties

# Log file for this module operation relative to Apache root directory.
# Set the name of the trace/log file. To avoid possible confusion about
# the location of this file, an absolute pathname is recommended.
#
# This log file is different than the log file that is in the
# jserv.properties file. This is the log file for the C portion of Apache
# JServ.
#
# On Unix, this file must have write permissions by the owner of the JVM
# process. In other words, if you are running Apache JServ in manual mode
# and Apache is running as user nobody, then the file must have its
# permissions set so that that user can write to it.
# Syntax: ApJServLogFile [filename]
# Default: "/logs/mod_jserv.log"
# Note: when set to "DISABLED", the log will be redirected to Apache error log
ApJServLogFile /var/log/httpd/mod_jserv.log

# Log Level for this module
# Syntax: ApJServLogLevel [debug|info|notice|warn|error|crit>alert|emerg]
# Default: info (unless compiled w/ JSERV_DEBUG, in which case it's debug)
ApJServLogLevel notice

# Protocol used by this host to connect to Apache JServ
# (see documentation for more details on available protocols)
# Syntax: ApJServDefaultProtocol [name]
# Default: "ajpv12"
ApJServDefaultProtocol ajpv12

# Default host on which Apache JServ is running
# Syntax: ApJServDefaultHost [hostname]
# Default: "localhost"
#ApJServDefaultHost java.apache.org

# Default port that Apache JServ is listening to
# Syntax: ApJServDefaultPort [number]
# Default: protocol-dependant (for ajpv12 protocol this is "8007")
ApJServDefaultPort 8007

# The amount of time to give to the JVM to start up as well
# as the amount of time to wait to ping the JVM to see if it
# is alive. Slow or heavily loaded machines might want to
# increase this value.
# Default: 10 seconds
# ApJServVMTimeout 10

# Passes parameter and value to specified protocol.
# Syntax: ApJServProtocolParameter [name] [parameter] [value]
# Default: NONE
# Note: Currently no protocols handle this. Introduced for future protocols.

# Apache JServ secret key file relative to Apache root directory.
# Syntax: ApJServSecretKey [filename]
# Default: "/conf/jserv.secret.key"
# Warning: if authentication is DISABLED, everyone on this machine (not just
# this module) may connect to your servlet engine and execute servlet
# bypassing web server restrictions. See the documentation for more information
#ApJServSecretKey DISABLED
ApJServSecretKey /etc/httpd/conf/jserv/jserv.secret.key
# Whether <VirtualHost> inherits base host mount points or not
# Syntax: ApJServMountCopy [on/off]
# Default: "On"
# Note: This directive is meaningful only when virtual hosts are being used
ApJServMountCopy on

# Executes a servlet passing filename with proper extension in PATH_TRANSLATED # property of servlet request.
# Syntax: ApJServAction [extension] [servlet-uri]
# Defaults: NONE
# Notes: This is used for external tools.
ApJServAction .jsp /servlets/org.gjt.jsp.JSPServlet
ApJServAction .jhtml /servlets/org.apache.servlet.ssi.SSI

# Enable the Apache JServ status handler with the URL of
# "http://servername/jserv/" (note the trailing slash!)
# Change the "deny" directive to restrict access to this status page.
<Location /jserv/>
  SetHandler jserv-status
  order deny,allow
  deny from all
  allow from localhost
</Location>

# A R N I N G # Remember to disable or otherwise protect the execution of the Apache JServ # # Status Handler (see right above) on a production environment since this may # # give untrusted users the ability to obtain restricted information on your # # servlets and their initialization arguments such as JDBC passwords and # # other important information. The Apache JServ Status Handler should be # # accessible only by system administrators.

</IfModule>
Appendix J
A Real Configuration File of jser.properties

# The Java Virtual Machine interpreter.
# Syntax: wrapper.bin=[filename] (String)
# Note: specify a full path if the interpreter is not visible in your path.
wrapper.bin=/home/yuehua/myJDK/jdk1.2/bin/java

# Arguments passed to Java interpreter (optional)
# Syntax: wrapper.bin.parameters=[parameters] (String)
# Default: NONE
wrapper.bin.parameters=-verbose

# CLASSPATH environment value passed to the JVM
# Syntax: wrapper.classpath=[path] (String)
# Default: NONE (Sun's JDK/JRE already have a default classpath)
# Note: if more than one line is supplied these will be concatenated using
# ";" or ";" (depending wether Unix or Win32) characters. JVM must be
# able to find JSDK and JServ classes and any utility classes used by
# your servlets.
# Note: the classes you want to be automatically reloaded upon modification
# MUST NOT be in this classpath or the classpath of the shell
# you start the Apache from.

wrapper.classpath=/usr/lib/apache/ApacheJServ.jar
wrapper.classpath=/home/httpd/classes/servlet-2.0.jar
wrapper.classpath=/home/yuehua/myJDK/jdk1.2/lib/rt.jar
wrapper.classpath=/home/yuehua/myJDK/jdk1.2/lib/tools.jar
wrapper.classpath=/home/yuehua/myJDK/jdk1.2/jre/rt.jar
wrapper.classpath=/home/yuehua/myJDK/jaf-1.0.1/activation.jar
wrapper.classpath=/home/yuehua/myJDK/JSDK2.0/lib/jsdk.jar
wrapper.classpath=/home/yuehua/myJDK/servlet.jar
wrapper.classpath=/home/yuehua/myJSDK/mm.mysql.jdbc-2.0pre5
wrapper.classpath=/usr/local/lib/perl5/site_perl/5.005
wrapper.classpath=/home/yuehua/research
# Syntax: bindaddress=[ipaddress] or [localhost] or [*]  
# Default: localhost  
bindaddress=localhost

# Set the port Apache JServ listens to.  
# Syntax: port=[1024,65535] (int)  
# Default: 8007  
port=8007

#
# Servlet Zones parameters  
###################################

# List of servlet zones Apache JServ manages  
# Syntax: zones=[servlet zone],[servlet zone]... (Comma separated list of String)  
# Default: NONE  
zones=research

research.properties=/etc/httpd/conf/zone.properties  
#
# Thread Pool parameters  
###################################

# Enables or disables the use of the thread pool.  
# Syntax: pool=[true|false] (boolean)  
# Default: false  
# WARNING: the pool has not been extensively tested and may generate deadlocks.  
# For this reason, we advise against using this code in production environments.  
pool=false

# Indicates the number of idle threads that the pool may contain.  
# Syntax: pool.capacity=(int)>0  
# Default: 10  
# NOTE: depending on your system load, this number should be low for contantly  
# loaded servers and should be increased depending on load bursts.  
pool.capacity=10

# Indicates the pool controller that should be used to control the  
# level of the recycled threads.  
# Syntax: pool.controller=[full class of controller] (String)  
# Default: org.apache.java.recycle.DefaultController  
# NOTE: it is safe to leave this unchanged unless special recycle behavior  
# is needed. Look at the "org.apache.java.recycle" package javadocs for more  
# info on other pool controllers and their behavior.  
pool.controller=org.apache.java.recycle.DefaultController
# Security parameters

## Enable/disable the execution of org.apache.jserv.JServ as a servlet.
# This is disabled by default because it may give informations that should be restricted.
# Note that the execution of Apache JServ as a servlet is filtered by the web server modules by default so that both sides should be enabled to let this service work.
# This service is useful for installation and configuration since it gives feedback about the exact configurations Apache JServ is using, but it should be disabled when both installation and configuration processes are done.
# Syntax: security.selfservlet=[true|false] (boolean)
# Default: false
# WARNING: disable this in a production environment since may give reserved information to untrusted users.
security.selfservlet=true

## Set the maximum number of socket connections Apache JServ may handle simultaneously. Make sure your operating environment has enough file descriptors to allow this number.
# Syntax: security.maxConnections=(int)>1
# Default: 50
security.maxConnections=50

## Backlog setting for very fine performance tunning of JServ.
# Unless you are familiar to sockets leave this value commented out.
# security.backlog=5

## Enable/disable connection authentication.
# NOTE: unauthenticated connections are a little faster since authentication handshake is not performed at connection creation.
# WARNING: authentication is disabled by default because we believe that connection restriction from all IP addresses but localhost reduces your time to get Apache JServ to run. If you allow other addresses to connect and you don't trust it, you should enable authentication to prevent untrusted execution of your servlets. Beware: if authentication is disabled and the IP address is allowed, everyone on that machine can execute your servlets!
# Syntax: security.authentication=[true,false] (boolean)
# Default: true
security.authentication=true

## Authentication secret key.
# The secret key is passed as a file that must be kept secure and must be exactly the same of those used by clients to authenticate themselves.
# Syntax: security.secretKey=[secret key path and filename] (String)
# Default: NONE
# Note: if the file could not be opened, try using absolute paths.
security.secretKey=/etc/httpd/conf/jserv/jserv.secret.key

# Length of the randomly generated challenge string (in bytes) used to
# authenticate connections. 5 is the lowest possible choice to force a safe
# level of security and reduce connection creation overhead.
# Syntax: security.challengeSize=(int)>5
# Default: 5
#security.challengeSize=5

#
# Logging parameters

 Enable/disable Apache JServ logging.
# WARNING: logging is a very expensive operation in terms of performance. You
# should reduced the generated log to a minimum or even disable it if fast
# execution is an issue. Note that if all log channels (see below) are
# enabled, the log may become really big since each servlet request may
# generate many Kb of log. Some log channels are mainly for debugging
# purposes and should be disabled in a production environment.
# Syntax: log=[true,false] (boolean)
# Default: true
log=true

# Set the name of the trace/log file. To avoid possible confusion about
# the location of this file, an absolute pathname is recommended.
#
# This log file is different than the log file that is in the
# jserv.conf file. This is the log file for the Java portion of Apache
# JServ.
#
# On Unix, this file must have write permissions by the owner of the JVM
# process. In other words, if you are running Apache JServ in manual mode
# and Apache is running as user nobody, then the file must have its
# permissions set so that that user can write to it.
# Syntax: log.file=[log path and filename] (String)
# Default: NONE
# Note: if the file could not be opened, try using absolute paths.
log.file=/var/log/httpd/jserv.log

# Enable the timestamp before the log message
# Syntax: log.timestamp=[true,false] (boolean)
# Default: true
log.timestamp=true

# Use the given string as a data format
# (see java.text.SimpleDateFormat for the list of options)
# Syntax: log.dateFormat=(String)
# Default: [dd/MM/yyyy HH:mm:ss:SSS zz]
log.dateFormat=[dd/MM/yyyy HH:mm:ss:SSS zz]

# Syntax: log.queue.maxage = [milliseconds]
# Default: 5000
log.queue.maxage = 5000

# Syntax: log.queue.maxsize = [integer]
# Default: 1000
log.queue.maxsize = 1000

# Servlets exception, i.e. exception caught during
# servlet.service() processing are monitored here
# you probably want to have this one switched on
log.channel.servletException=true

# JServ exception, caught internally in jserv
# we suggest to leave it on
log.channel.jservException=true

# Warning channel, it catches all the important
# messages that don't cause JServ to stop, leave it on
log.channel.warning=true

# Servlet log
# All messages logged by servlets. Probably you want
# this one to be switched on.
log.channel.servletLog=true

# Critical errors
# Messages produced by critical events causing jserv to stop
log.channel.critical=true

# Debug channel
# Only for internal debugging purposes
# log.channel.debug=true