2003

Web-based e-mail client for computer science

Jichuan Wu

WEB-BASED E-MAIL CLIENT FOR COMPUTER SCIENCE

A Project

Presented to the

Faculty of

California State University,

San Bernardino

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in

Computer Science

---

by

Chi-Chuan Wu

December 2003

WEB-BASED E-MAIL CLIENT FOR COMPUTER SCIENCE

A Project

Presented to the

Faculty of

California State University,

San Bernardino

---

by

Chi-Chuan Wu

December 2003

Approved by:

[redacted]

Dr. Richard Botting, Chair, Computer Science — Nov. 23, 2003 — Date

[redacted]

Dr. Ernesto Gomez

[redacted]

Dr. Kerstin Voigt

ABSTRACT

The project is a web e-mail application to provide a web page interface for all CSCI faculty, staff and students to handle their e-mails. The application is written by JSP, Java servlets, JavaScript and custom JSP tag libraries. It uses JavaMail API to develop a fully functional e-mail client application. Users can login the WebMail page, then list all the messages in the INBOX, SENT, DRAFT, TRASH, PAST, and FUTURE folders, view selected messages with html document in each folder, retrieve and view attachments from selected messages, compose and send messages with attachments, reply or forward messages with original and new attachments, move any messages to the TRASH folder, delete messages from the TRASH folder, store an incomplete message to the DRAFT folder, store sent messages after sending messages. And the most important design in this project is: store and manage your messages by day. You can store your messages to any daily folders, and view messages in any daily folders. You can also make a note for that day. The note is stored and showed immediately in your daily folders.

ACKNOWLEDGMENTS

I thank the faculty of Computer Science department for giving me an opportunity to pursue my M.S. in Computer Science at California State University, San Bernardino. I express my sincere appreciation to my graduate advisor, Dr. Richard Botting who directed me through this entire effort, and give me the whole ideas of my project. In the beginning, we try to use Java Applet to write this project. After several month efforts, unfortunately I didn't make it, but I learn more and more from Java network programming skills. Such experiences help me building a JSP web application using not only JSP and JavaBeans, but also Java servlets and custom my own tag libraries.

Finally, I also want thank my other committee members, Dr. Kerstin Voigt and Dr. Ernesto Gomez for their valuable idea to improve my detail designs.

TABLE OF CONTENTS

ABSTRACT iii

ACKNOWLEDGMENTS iv

LIST OF TABLES viii

LIST OF FIGURES ix

CHAPTER ONE: INTRODUCTION 1

1.1 Purpose of this Project 2

1.2 Project Products 3

CHAPTER TWO: WEB-BASED E-MAIL CLIENT FOR COMPUTER SCIENCE ARCHITECTURE 4

2.1 Software Interfaces 6

CHAPTER THREE: DATABASE DESIGN

3.1 Data Analysis in Mail Server Database and Web Server Database 8

3.2 Tables in Web Server Database 14

CHAPTER FOUR: PROJECT IMPLEMENTATION 16

4.1 The Class Diagram of Web-based E-mail Client for Computer Science 18

4.2 Web-based E-mail Client for Computer Science Graphical User Interface Design 42

4.2.1 Web-based E-mail Client for Computer Science Login 43

4.2.2 Inbox Folder 45

4.2.3 Sent Folder 49

4.2.4 Create a New Message 50

4.2.5 Draft Folder 54

4.2.6 Trash Folder 54

4.2.7 Future Folder 55

4.2.8 Past Folder 59

4.2.9 Error Handle 60

4.2.10 Logout 71

CHAPTER FIVE: SECURITY

5.1 Login Page 74

5.2 Secure Sockets Layer between User and WebMail Client 74

5.3 Secure JavaMail with Java Secure Socket Extension 77

CHAPTER SIX: SYSTEM VALIDATION

6.1 Unit Test 78

6.2 Subsystem Testing 84

6.3 System Testing 85

CHAPTER SEVEN: MAINTENANCE MANUAL

7.1 Software Installation 87

7.1.1 JAVA 2 Platform, Standard Edition (J2SE) 87

7.1.2 JavaMail API 87

7.1.3 JavaBeans Activation Framework (JAF) Extension 88

7.1.4 The com.Oreilly.Servlet Package 88

7.1.5 Tomcat 88

7.1.6 Install Secure Socket Layer Support on Tomcat 89

7.1.7 MySQL Installation 89

7.1.8 JAVA Database Connectivity (JDBC) 90

7.2 Variables Modification 91

7.2.1 System Variables 91

7.2.2 Batch Files Modification 92

7.2.3 Copying Files 92

7.2.4 Secure Sockets Layer Configuration ... 93

7.3 Web-based E-mail Client for Computer Science Installation/Migration 93

7.4 Backup 96

7.4.1 System Backup 96

7.4.2 Database Backup 96

CHAPTER EIGHT: CONCLUSION AND FUTURE DIRECTIONS

8.1 Conclusion 97

8.2 Future Directions 98

APPENDIX: SOURCE CODE OF JAVA CLASSES 100

REFERENCES 137

LIST OF TABLES

Table 1. DB Table Note 15

Table 2. Unit Test Results 78

Table 3. Subsystem Test Results 85

Table 4. System Test Results 86

LIST OF FIGURES

Figure 1. Web-Based E-mail Client Architecture 4

Figure 2. The "Mail" Folder in Computer Science User Account 8

Figure 3. Pine Main Menu 9

Figure 4. Pine Folder List 10

Figure 5. The "Schedule" Folder in Computer Science User Account 11

Figure 6. Pine "Schedule" Folder List 12

Figure 7. Pine Message List in "Dec.1" Folder 13

Figure 8. View Message in Daily Folder by Pine 14

Figure 9. Use Case Diagram 16

Figure 10. Class Diagram 17

Figure 11. Class Diagram of AttachmentServlet.java 18

Figure 12. Class Diagram of ByteArrayDataSource.java 19

Figure 13. Class Diagram of ComposeBean.java 19

Figure 14. Class Diagram of MailUserBean.java 20

Figure 15. Class Diagram of ComposeModel.java 21

Figure 16. Class Diagram of Csmail.java 22

Figure 17. Class Diagram of Deletemail.java 23

Figure 18. Class Diagram of ListAttachmentsTEI.java.... 24

Figure 19. Class Diagram of ListAttachmentsTag.java.... 25

Figure 20. Class Diagram of ListMessagesTag.java 26

Figure 21. Class Diagram of ListMessagesTEI.java 27

Figure 22. Class Diagram of MessageInfo.java 28

Figure 23. Class Diagram of MessageTag.java 29

Figure 24. Class Diagram of MessageTEI.java 29

Figure 25. Class Diagram of MoveToTrash.java 30

Figure 26. Class Diagram of MultipartFormdataRequest.java 31

Figure 27. Class Diagram of MonthMap.java 32

Figure 28. Class Diagram of SendMail.java 33

Figure 29. Class Diagram of SendMailAuthenticator.java ... 34

Figure 30. Class Diagram of SendMailServlet.java 35

Figure 31. Class Diagram of StoreTo.java 36

Figure 32. Class Diagram of StrutMultipartIterator.java ... 37

Figure 33. Class Diagram of StrutMultipartParser.java ... 38

Figure 34. Class Diagram of StrutUploadFile.java 39

Figure 35. Class Diagram of UploadBean.java 40

Figure 36. Class Diagram of UploadException.java 41

Figure 37. Class Diagram of UploadFile.java 41

Figure 38. Class Diagram of UploadServlet.java 42

Figure 39. Security Alert before Entering the Login Page 43

Figure 40. Login Page 44

Figure 41. Messages in the Inbox Folder 45

Figure 42. View the Message Content 46

Figure 43. Handle the Message Attachments 47

Figure 44. Message Reply 48

Figure 45. Messages in the Sent Folder 49

Figure 46. Create A New Message 50

Figure 47. Select A File to Attach 51

Figure 48. Browse your Computer to Select A File to Attach 52

Figure 49. Remove Attached Files 53

Figure 50. Messages in the Draft Folder 54

Figure 51. Messages in the Trash Folder 55

Figure 52. Store the Message by Day 56

Figure 53. Messages in Oct. 26 57

Figure 54. Make A Note on Oct.26 58

Figure 55. The Note "Fall Saving" on Oct.26 59

Figure 56. A Decreasing Date List 60

Figure 57. Error Messages for Blank Username Input 61

Figure 58. Error Messages for Blank Password Input 62

Figure 59. Error Messages for Incorrect Username or Password 63

Figure 60. Error Messages for Blank or Error Address Inputs 64

Figure 61. Error Messages for Incorrect Address Format 65

Figure 62. Error Messages for Exceeding the Maximum Size of one Attachment 66

Figure 63. Error Messages for Exceeding the Total Maximum Size of the Attachments 67

Figure 64. Error Messages for Blank Note Input 68

Figure 65. Virus Alert Message1 69

Figure 66. Virus Alert Message2 70

Figure 67. Some Information About A Virus Possible File 71

Figure 68. A Confirm Message for Logging Out 72

Figure 69. Logout Page 73

Figure 70. The Project File Directory 93

CHAPTER ONE

INTRODUCTION

Before I decide to write my project, there was no web-based e-mail interfaces for CSCI. So I make a decision to write a web-based e-mail client for CSCI as my master project. But now, if you surf on the CSCI website, you will find a WebMail hyperlink exists. The WebMail is provided by the SquirrelMail Project Team, and can be free download from http://www.squirrelmail.org/download.php. Even though, Dr. Botting decide that we can keep doing this project by different way and develop our own web-based application. Since the SquirreMail is written by PHP. We decide use another language to implement it. There is much debate about the relative merits of various scripting languages such as ASP, JSP, PHP etc. and their use for web pages. In the beginning I decide use Java applet to write the project. After several month efforts, I fail to do so. One of the biggest issues with using JavaMail API in applets is the default applet security restrictions. These restrictions only allow applets to connect to the host from which they were loaded. Thus, for such applets to use JavaMail API, the mail server will need to be located on the same machine as the web server

from which the applet is loaded. Since in this project the CSCI mail server and the web-based e-mail client are located on different machines, the JavaMail API failed to use in applet directly. I try using JavaMail API in servlets then make a connection between applets and servlets. This method makes applets successfully send and receive e-mails. But in this case, using pure servlets programming can also do all the jobs. If applet cannot communicate directly to the mail server, the using of applet will lower the efficiency of networking. Finally, the project was written in JSP. With the power of JSP, the web-based e-mail client has a powerful functionality to handle e-mails by interactive and dynamic web sites.

1.1 Purpose of this Project

The purpose of this project is to design, build, and implement a web-based e-mail client for CSCI at CSUSB. All e-mails of CSCI faculty, staff and students are stored in CSCI mail server database. The e-mail client is another web server which helps users to retrieve and sent e-mails by IMAP and SMTP protocol. The e-mail client provides an interface to communicate with the CSCI mail server. With a username and password of your CSCI account, you can retrieve your private e-mails, download or view attached

files, send e-mails with attached files, store and manage your e-mails by daily folders. Since there is one folder for each day, you can put mails in any daily folders to remind you in the future, or you can check past daily folders to see what important messages did you ever store in that day. When you make a note on that daily folder, this daily notes remind you to schedule and manage your daily life. Moreover, the web-based e-mail client supports SSL network connections over two sides. One is between web browsers and the web server client, another one is between the web server client and the CSCI mail server. Only users with their username and password are authorized to access their mail accounts and handle their e-mails.

1.2 Project Products

This project would lead to the following products:

- Implementation of WebMail: a working web site with JSP, Java programs and MySQL database.
- Systems Manual: a project report (this report) will be available with design details and specifications.

CHAPTER TWO

WEB-BASED E-MAIL CLIENT FOR COMPUTER SCIENCE ARCHITECTURE

This project, Web-Based E-mail Client for CSCI (WBECC), implements a web system to provide an interface for all CSCI faculty, staff and students to handle their e-mails. Thus, the components needed to implement WBECC are the existed CSCI mail server and the CSCI e-mail database server, a web client server, a web client database server, and a graphical user interface components. The following figure describes the interactions among the components used in WBECC.

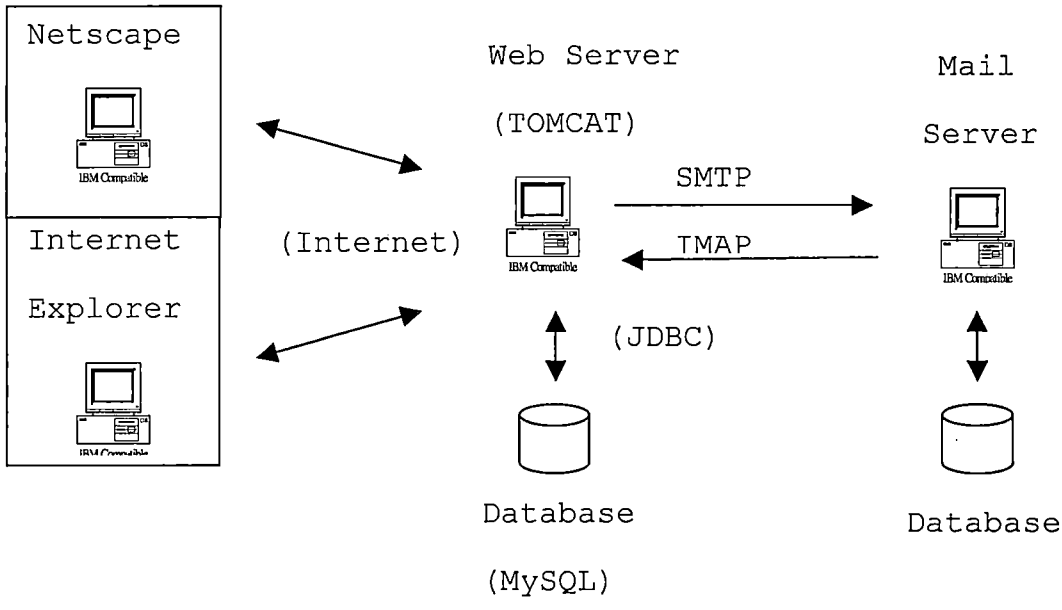

Figure 1. Web-Based E-mail Client Architecture

The components used to build WBECC were chosen with the following criteria: (i) the components should be shareware, i.e., available freely for non-commercial purposes, (ii) be part of a standard, i.e., they do not depend on a specific operating system and hence are easily portable across systems with ease, (iii) database server independent, so that new and different versions of the server can be plugged in easily.

The user interface components are built by using HTML 6.0 forms, frames and Javascript. And the applications are launched using the JavaServer Pages (JSP). JSP was used because it can use javaBeans which provide a common way for all programs and java containers. JSP technology is an extension of servlet technology. The application also uses servet technology directly to handle most parts of composing mails and attaching files. This is because, for some multipart http requests, the servlet technology is the best way to handle it. The application also builds custom tag libraries to read mail. This is because JavaBeans cannot interact with the JSP page: they do not have access to the request and response objects and you cannot use them to perform some decision-making processes. Of course, you can see the use of JavaBeans all of the programs. A lot of classes are needed to be reused in the

application. The web server, Tomcat, can be installed under Windows or Linux. Also, it is easy to process whole user input from the HTML forms. Moreover, Java provides convenient function, Java Database Connector (JDBC), to connect database.

The database choice available to WBECC is MySQL. MySQL is a real multi-user database and free. Also, the availability of the JDBC driver for MySQL is the most important reason to choose it. Moreover, the same code could be used to link with another version of MySQL database by changing proper JDBC driver. Thereby making it database independent. The only data store in the web server database is daily notes. There are 12 tables for 12 months. The other data for the application, including all messages and folders, are all stored in the CSCI mail server database.

2.1 Software Interfaces

- Internet browser: Netscape or Internet Explorer.
- Operating system: Windows 98/Me/2000/XP or Unix/Linux.
- Database: MySQL.
- Compiler: JDK 1.4.
- Language: HTML / JAVA / JavaScript / JSP.

- Database connector: JDBC.
- JSP Container/Web server: Jakarta Tomcat.

CHAPTER THREE

DATABASE DESIGN

3.1 Data Analysis in Mail Server Database and Web Server Database

Since the application is a web interface between CSCI users and the CSCI mail server, most parts of data are stored in the CSCI mail server database. If you look into your CSCI account you will find a "mail" folder is already been there.

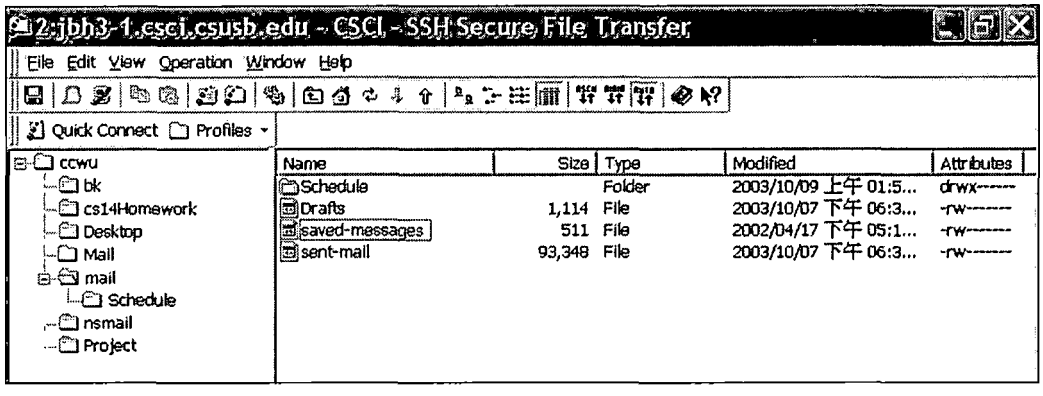

Figure 2. The "Mail" Folder in Computer Science User Account

The "mail" folder is used for Pine to store messages. The project uses the same folders and database as Pine. Therefore, you can view your messages and folders by two different ways: by logining the web page or by the command window of Pine.

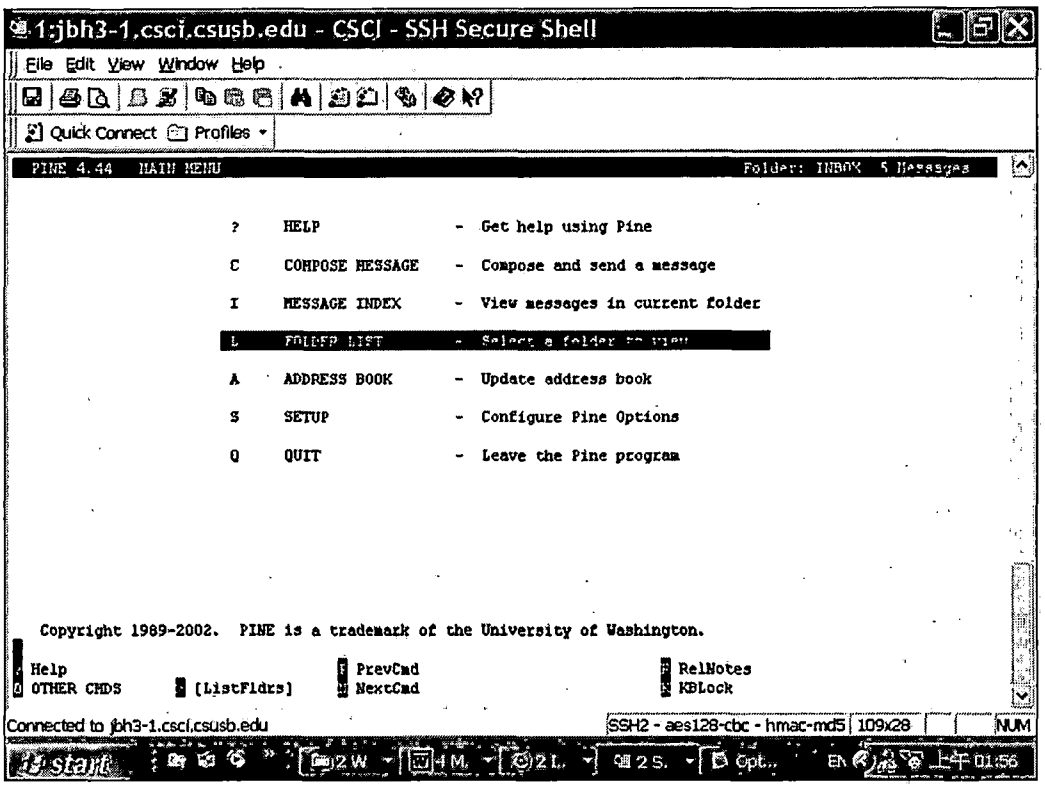

Figure 3. Pine Main Menu

If you view the "folder list" in Pine, you will find all your folders and messages in the "mail" folder.

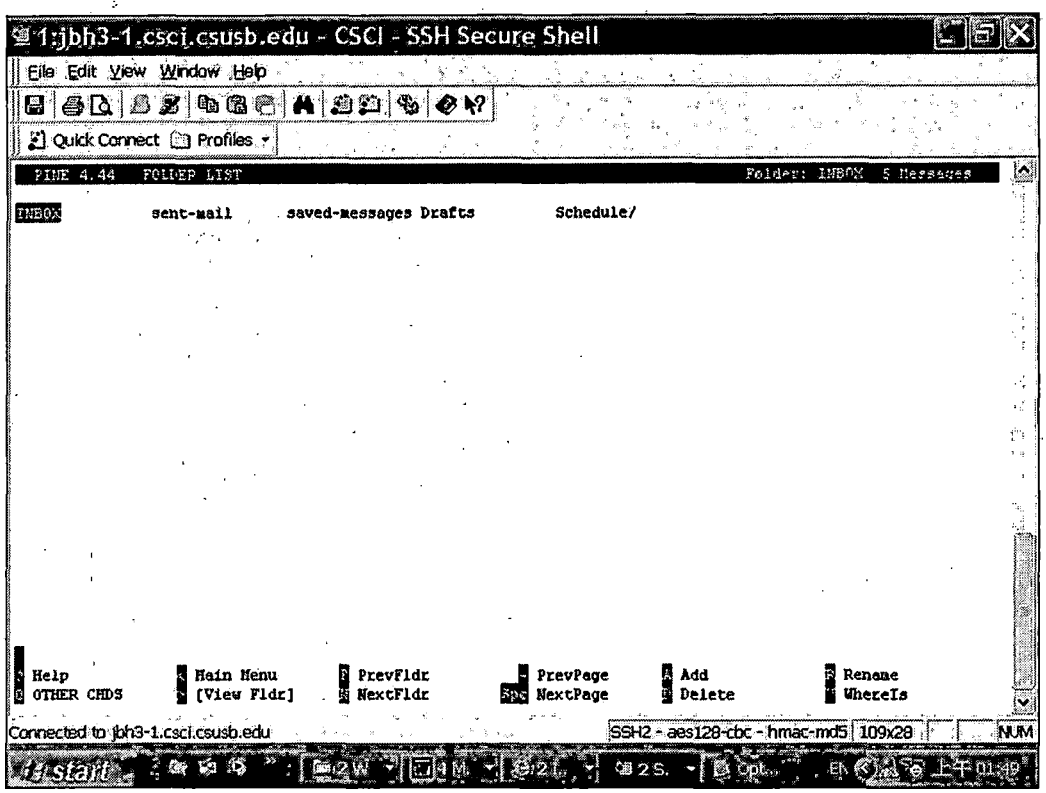

Figure 4. Pine Folder List

The "Trash" folder will be created in your first visit of "Trash" folder from the web-site of Webmail. The "Schedule" folder will also be created in the first time you use daily folders.

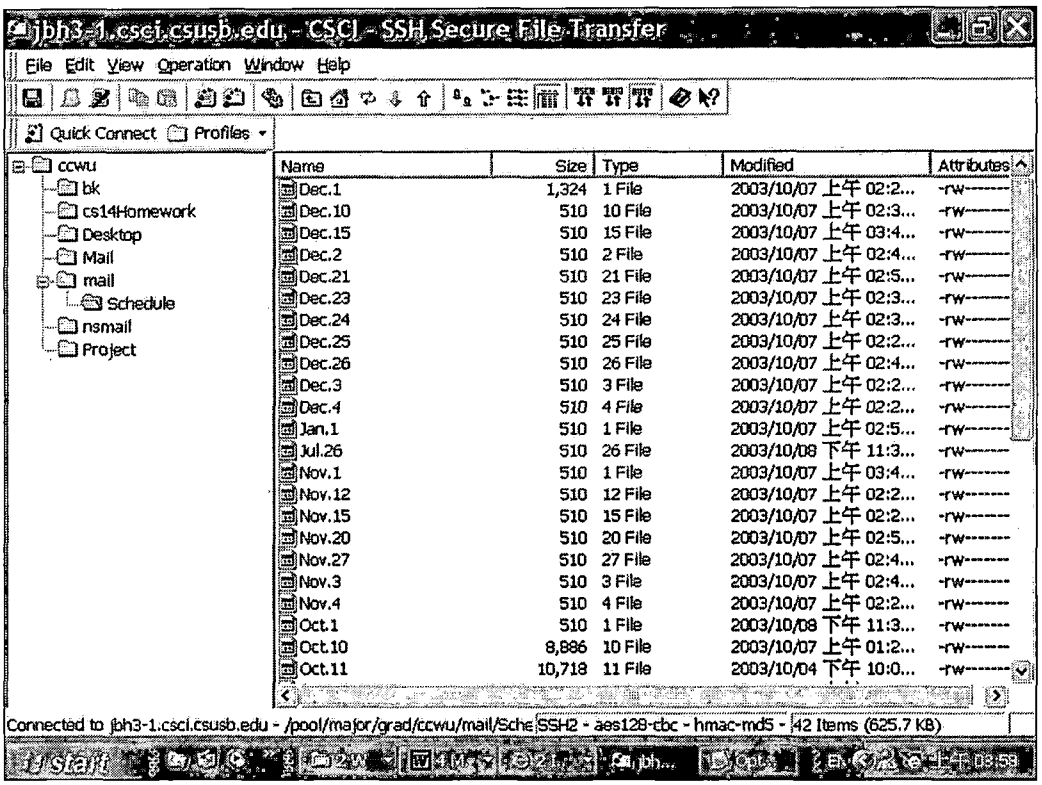

Figure 5. The "Schedule" Folder in Computer Science User Account

The "Schedule" folder is going to stored at most 366 daily folders for one year's usage. All 366 daily folders will not been created in the beginning. Each daily folder is created when you visit that daily folder in the first time.

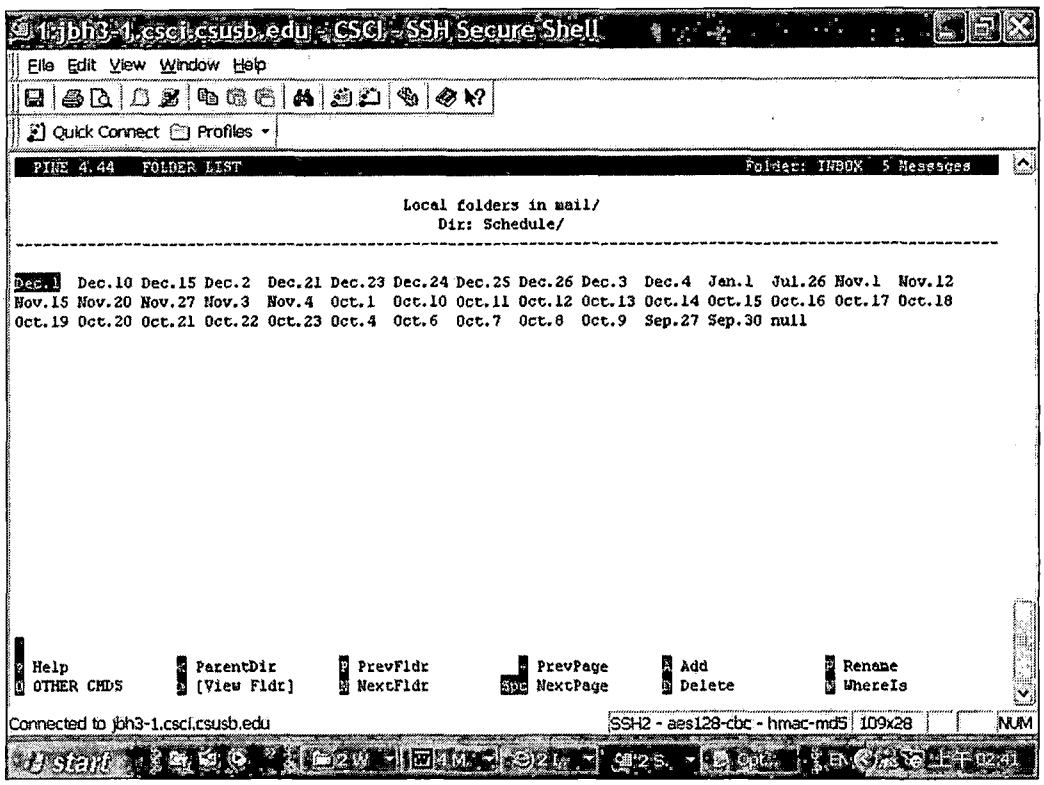

Figure 6. Pine "Schedule" Folder List

Since there is only one daily folder for one day, Dec. 1, 2003, Dec. 1, 2004, and Dec. 1, 2005 ... will use the same daily folder. If you check Dec.1 folder, you will find all Dec. 1 messages from the first year you start to use the system. In this case, even 10 years later, you will not lose any messages in your daily folders. The system will become a really daily messages management system.

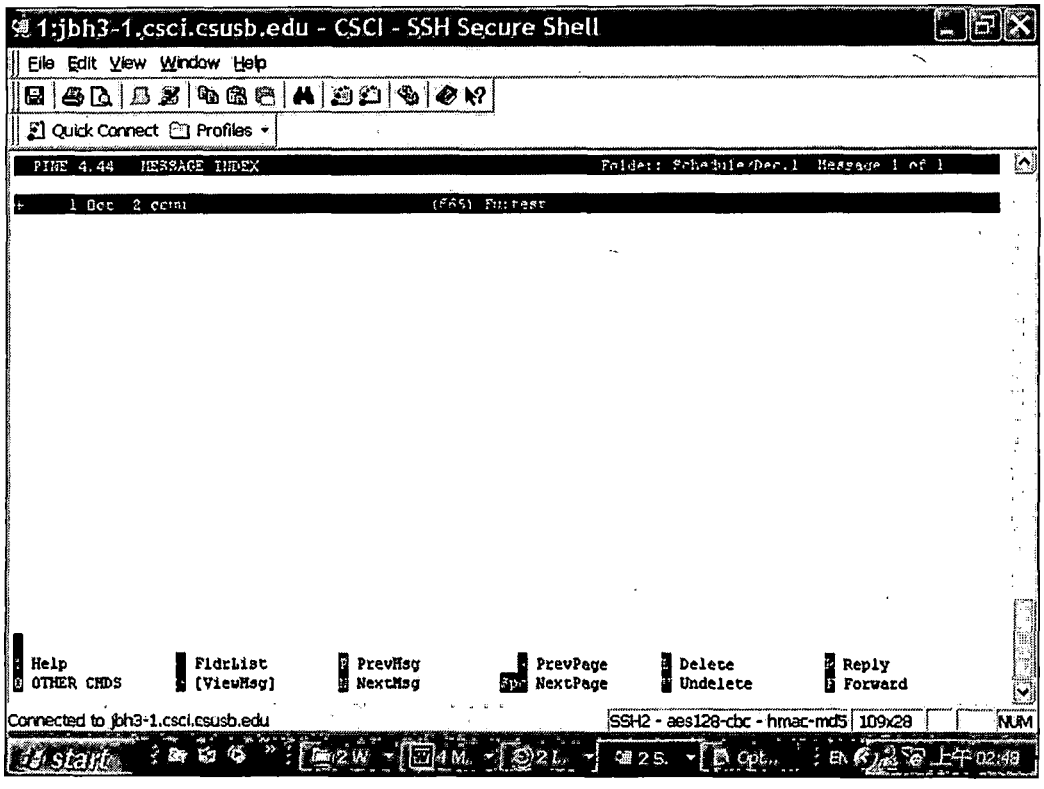

Figure 7. Pine Message List in "Dec.1" Folder

You can store your messages to any daily folder by logining WebMail page, and view the daily folder by Pine or by WebMail page too.

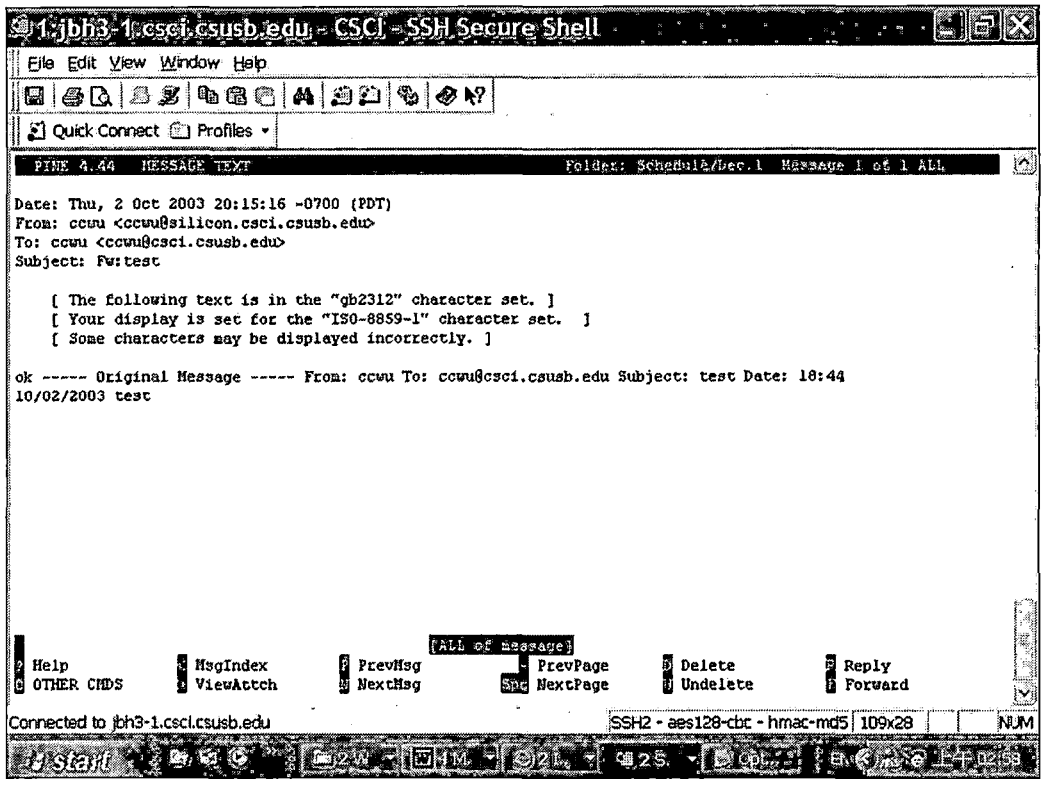

Figure 8. View Message in Daily Folder by Pine

The only database design in this project is in web server. It stores daily notes for each users. Therefore, you cannot see your daily notes from Pine. You can see your daily notes just from the WebMail page.

3.2 Tables in Web Server Database

Since the only data stored in the web Server is the daily note. There is just one table in the web server database. The following are the table describe data type, length, primary key or not, null or non-null key.

Table 1. DB Table Note

field	type	Null	key	default
User	char(10)		Composite PRI	
Folder	Char(10)		Composite PRI	
Detail	char(50)	YES		NULL

The value of the field User is user login name. The value of the field Folder is the name of the daily folder. It will look like "Jan. 1", "Feb. 2", etc. And the value of field detail is the note for that day.

CHAPTER FOUR

PROJECT IMPLEMENTATION

The following Figure 9 is the Use Case Diagram of this project.

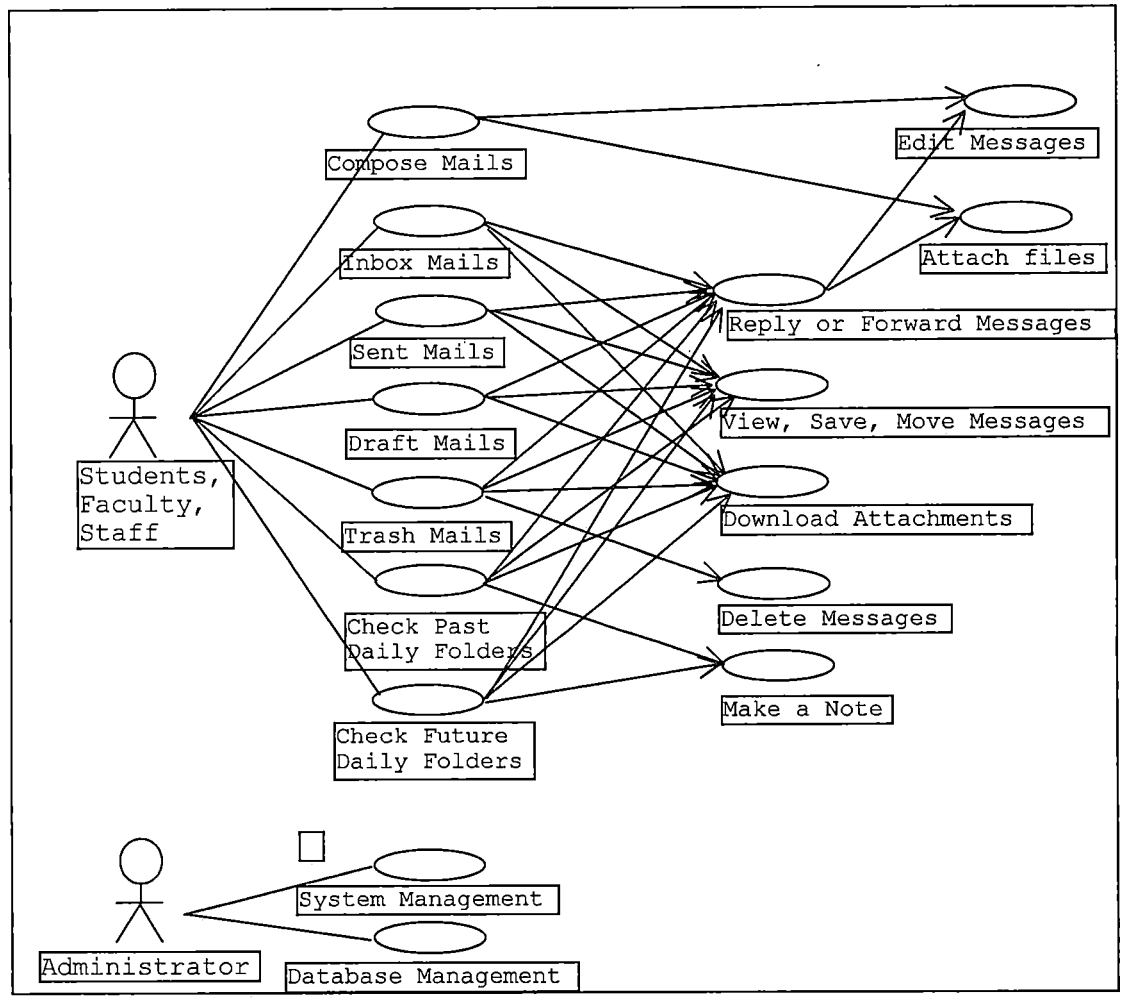

Figure 9. Use Case Diagram

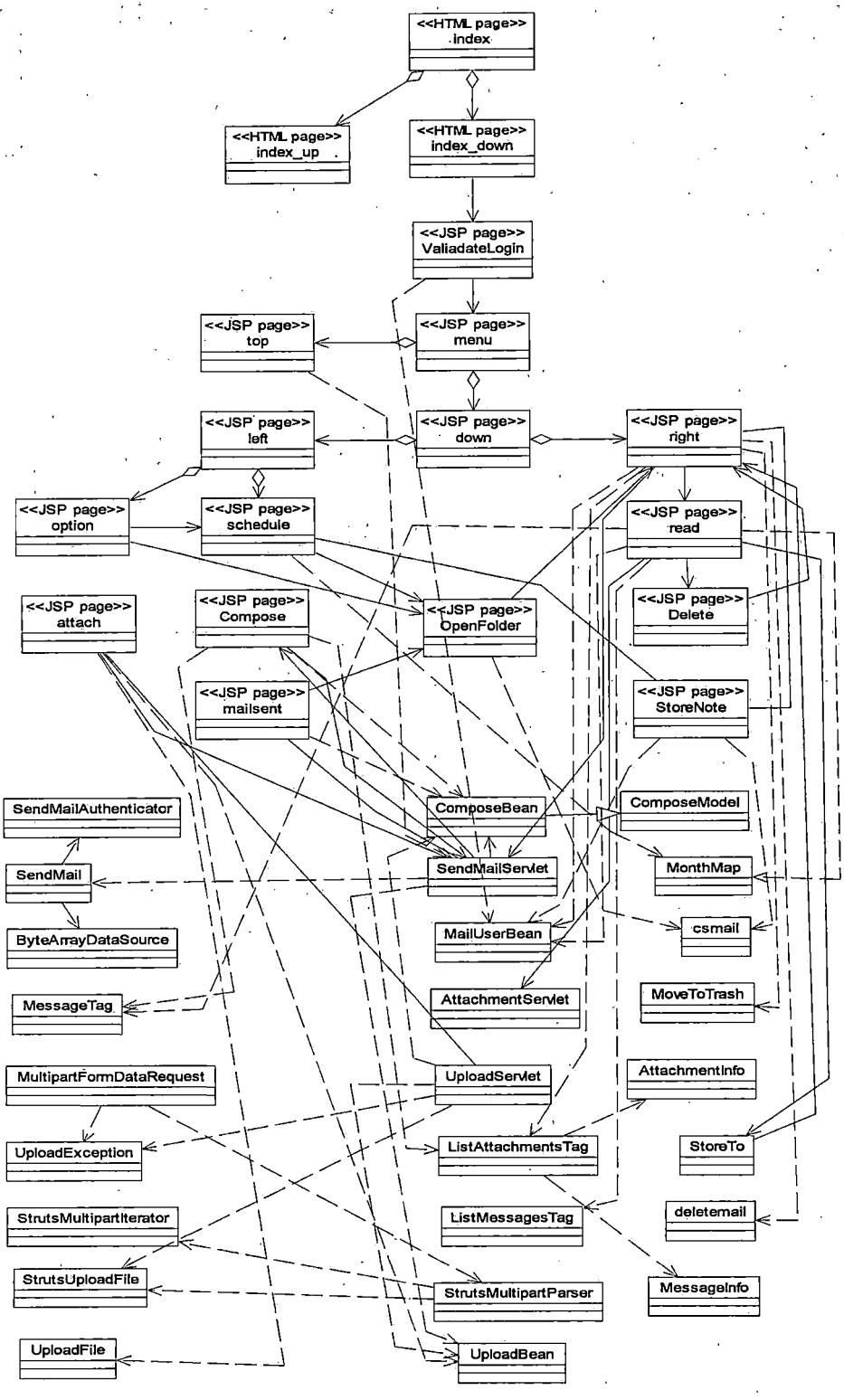

Figure 10. Class Diagram

4.1 The Class Diagram of Web-based E-mail Client for Computer Science

Figure 10 is the relationship of all class diagrams. The Java class includes JavaBeans, servlets, and tag libraries. These Java classes help JSP pages build a more reusable program.

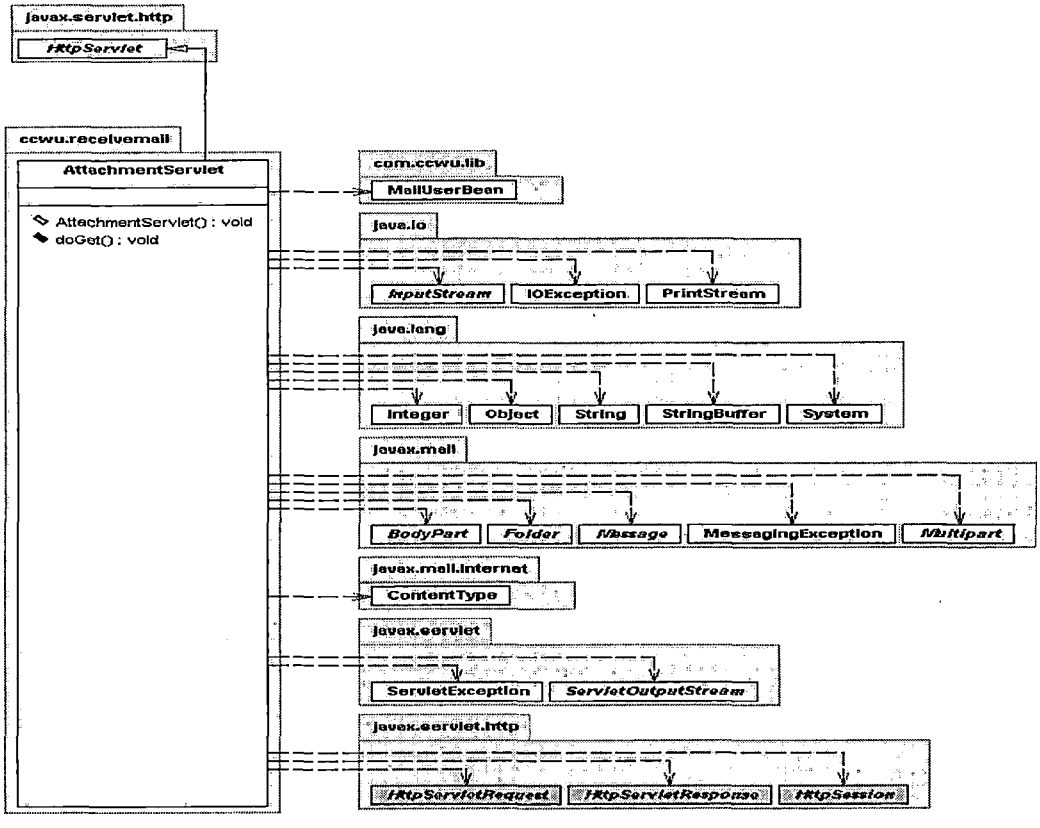

Figure 11. Class Diagram of AttachmentServlet.java

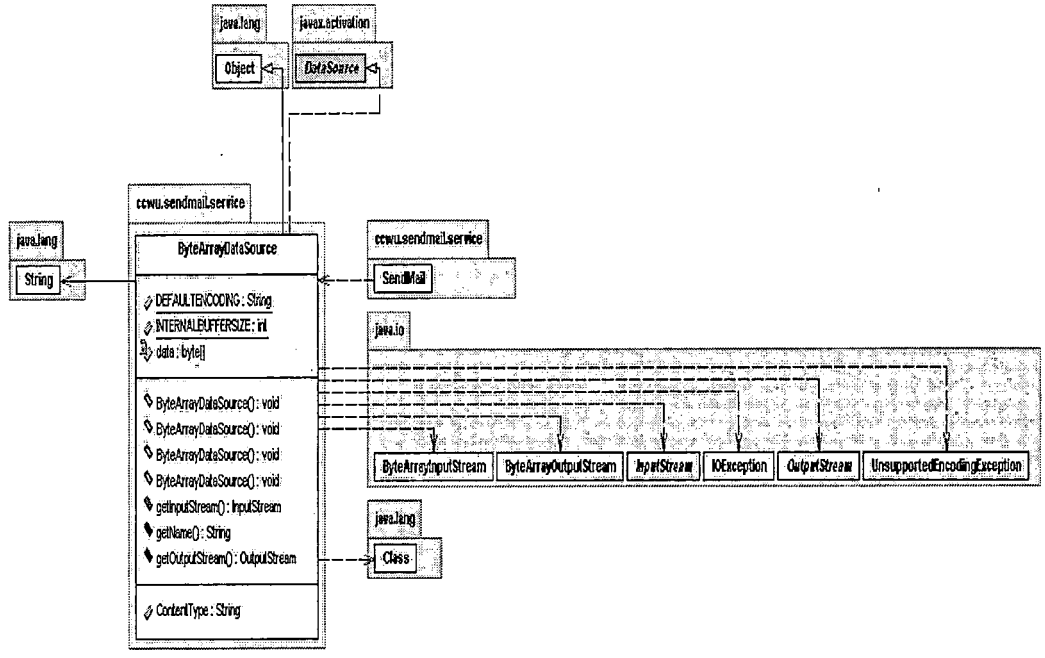

Figure 12. Class Diagram of ByteArrayDataSource.java

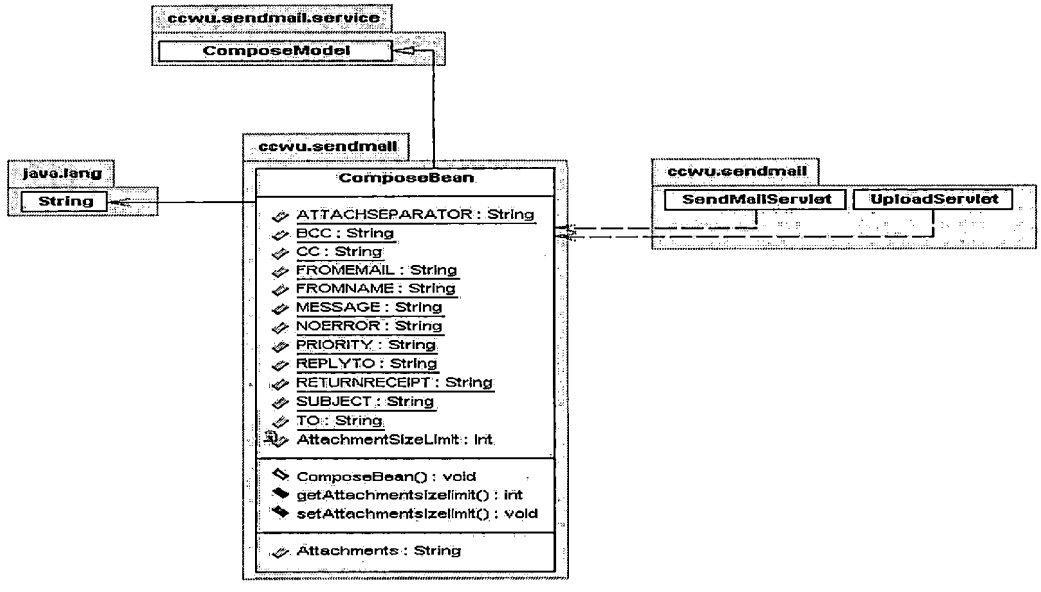

Figure 13. Class Diagram of ComposeBean.java

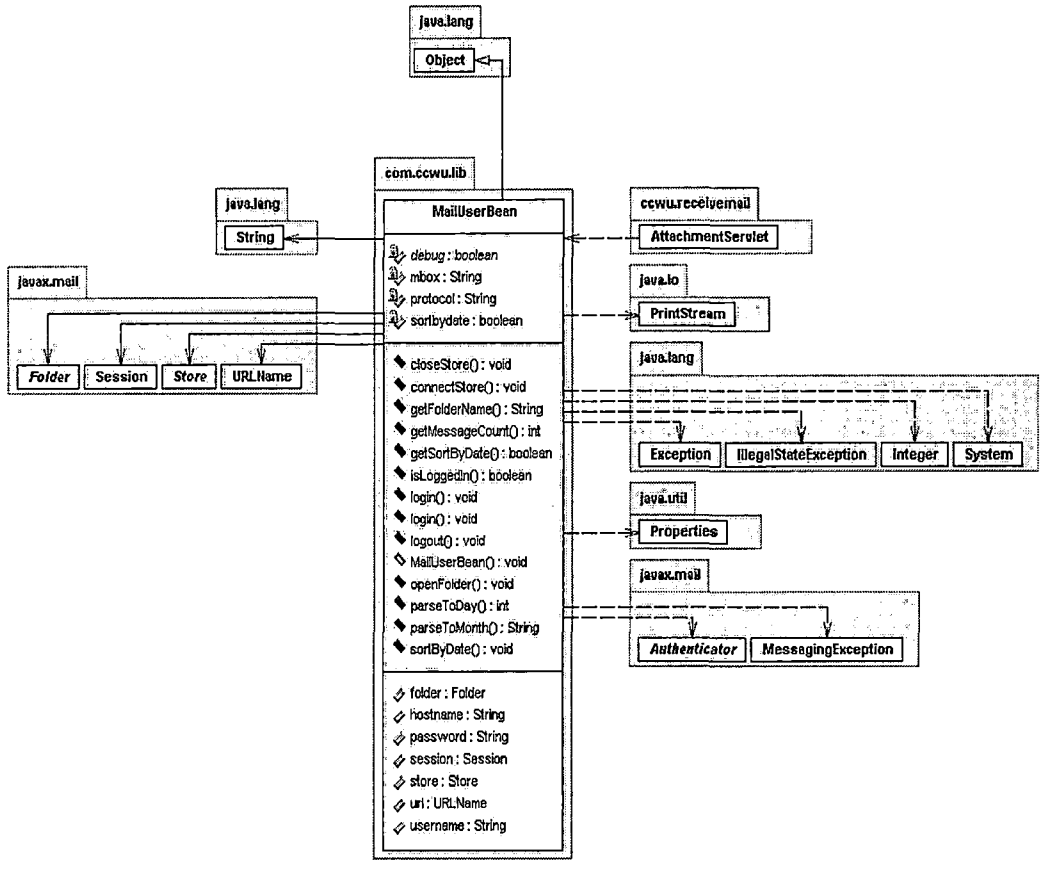

Figure 14. Class Diagram of MailUserBean.java

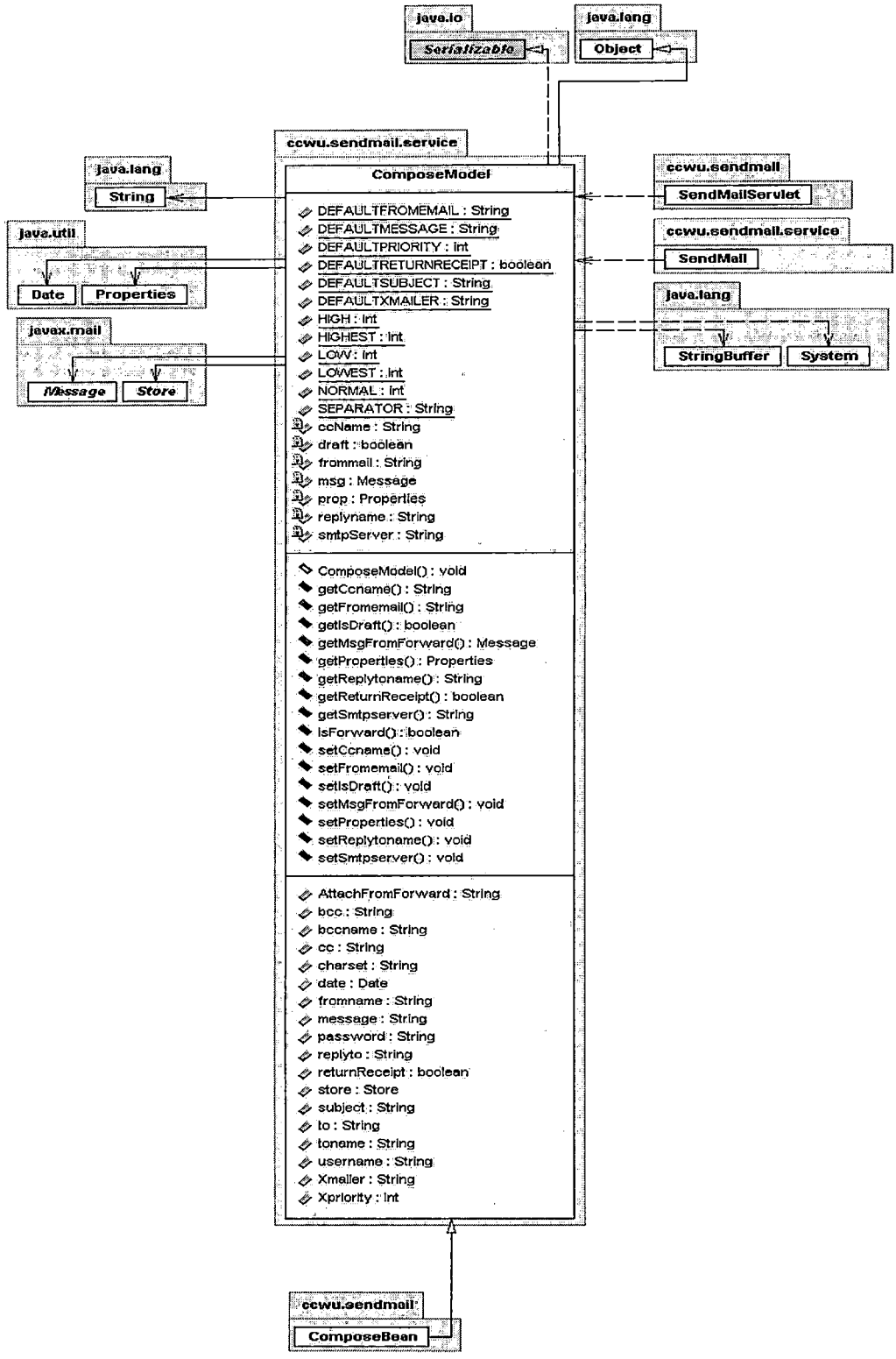

Figure 15. Class Diagram of ComposeModel.java

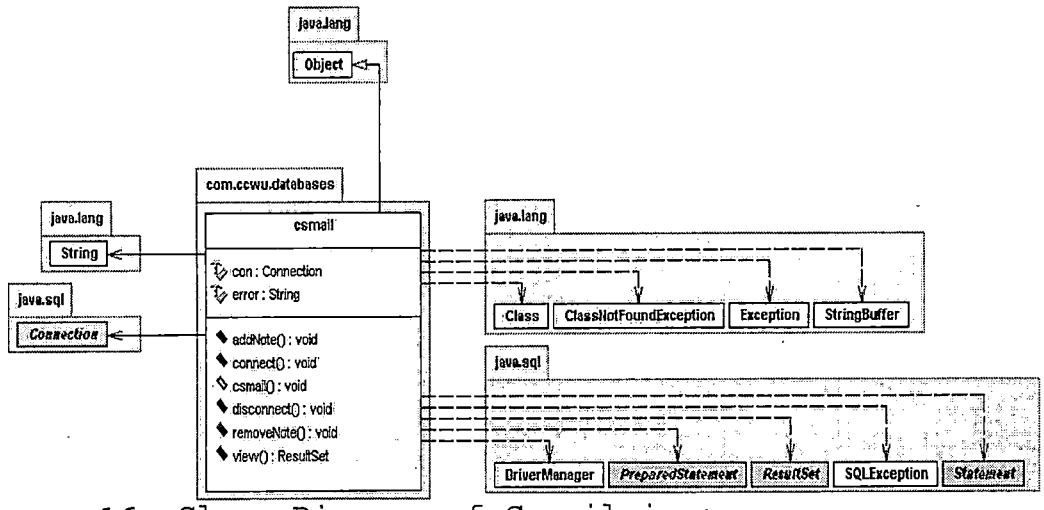

Figure 16. Class Diagram of Csmail.java

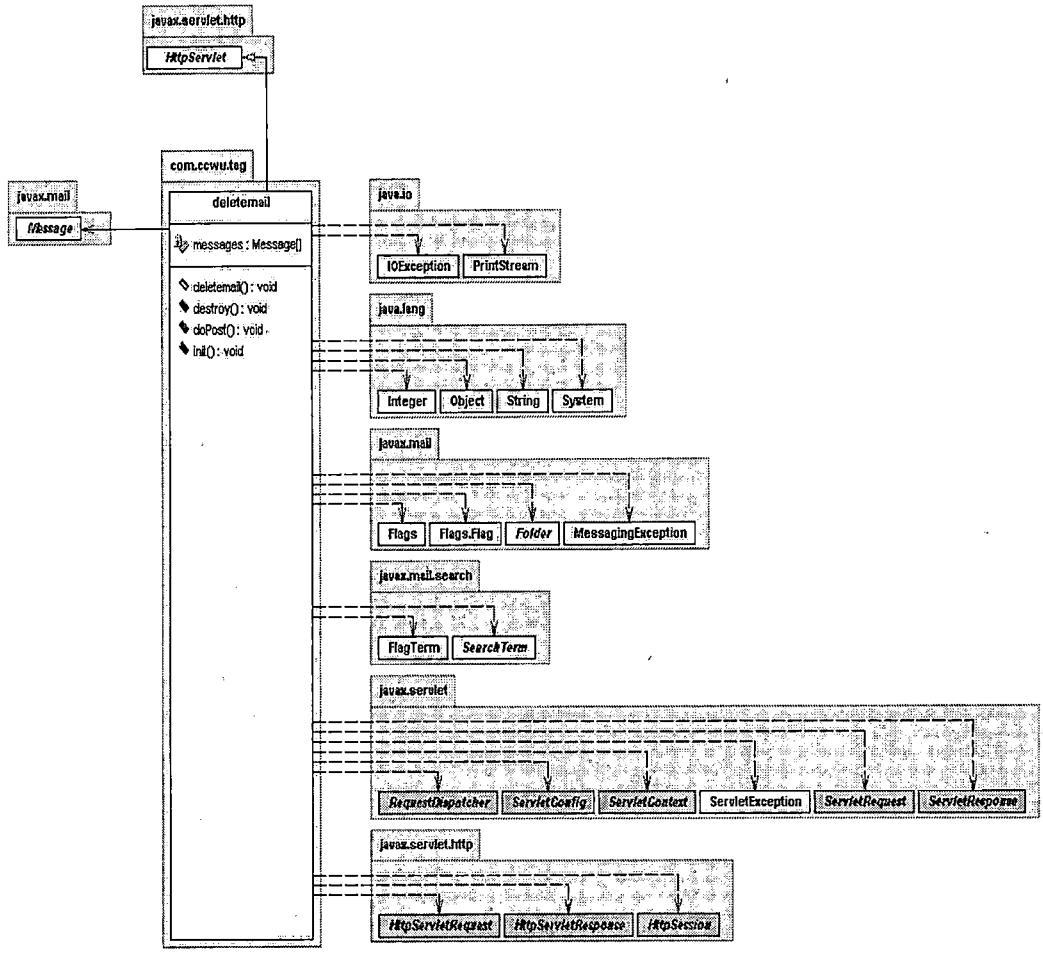

Figure 17. Class Diagram of Deletemail.java

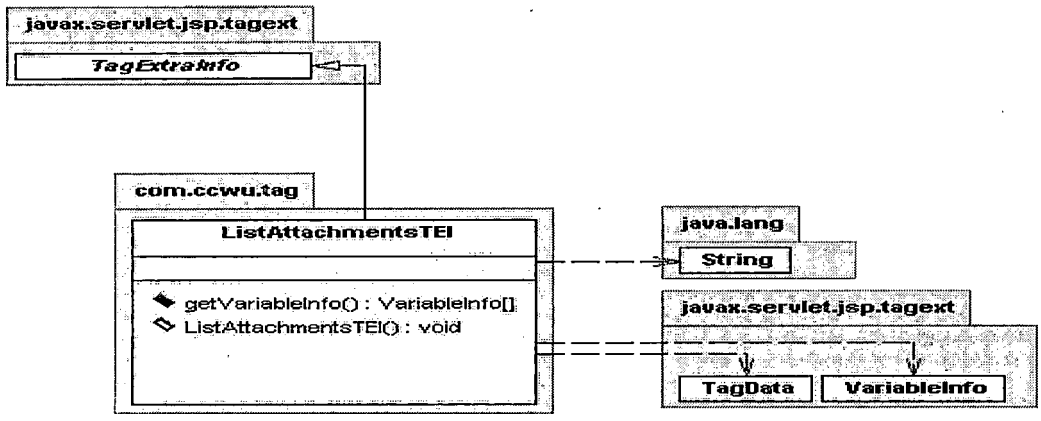

Figure 18. Class Diagram of ListAttachmentsTEI.java

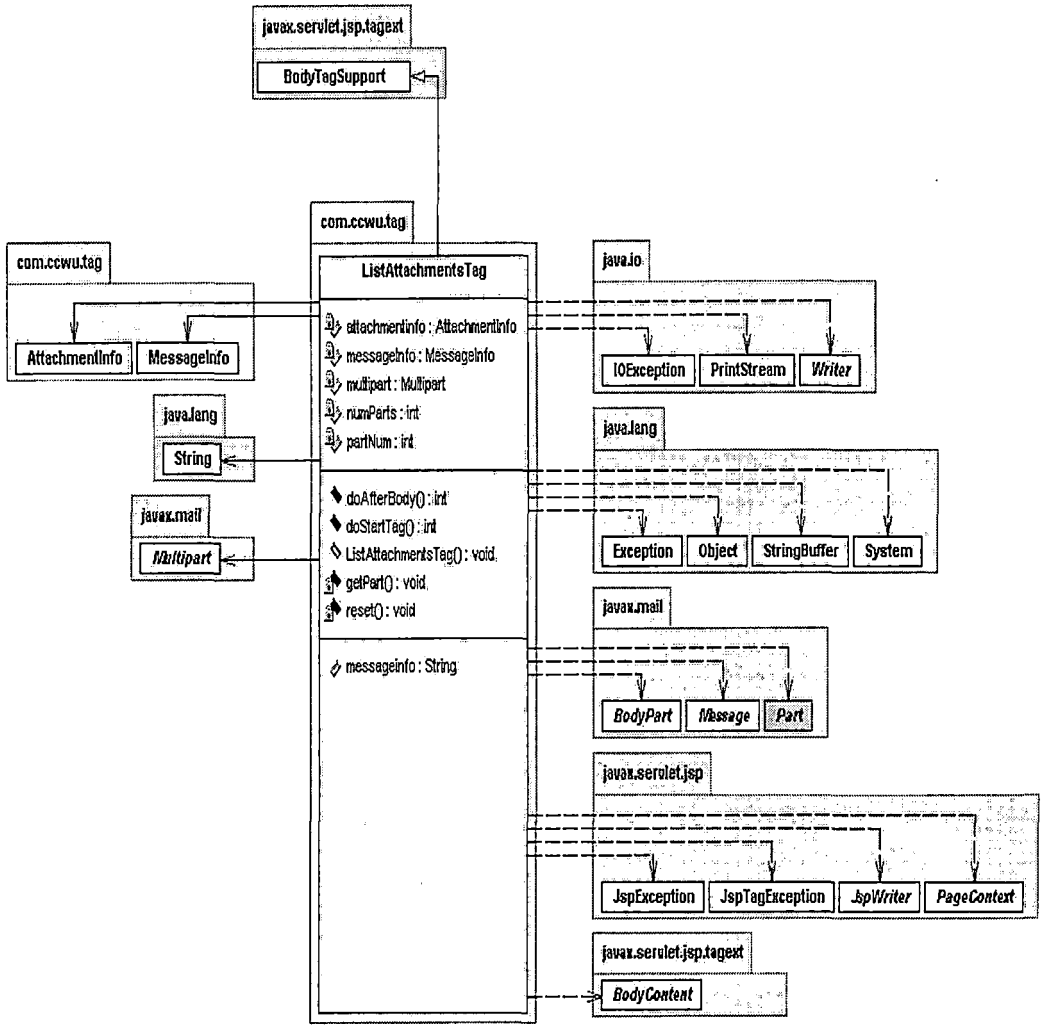

Figure 19. Class Diagram of ListAttachmentsTag.java

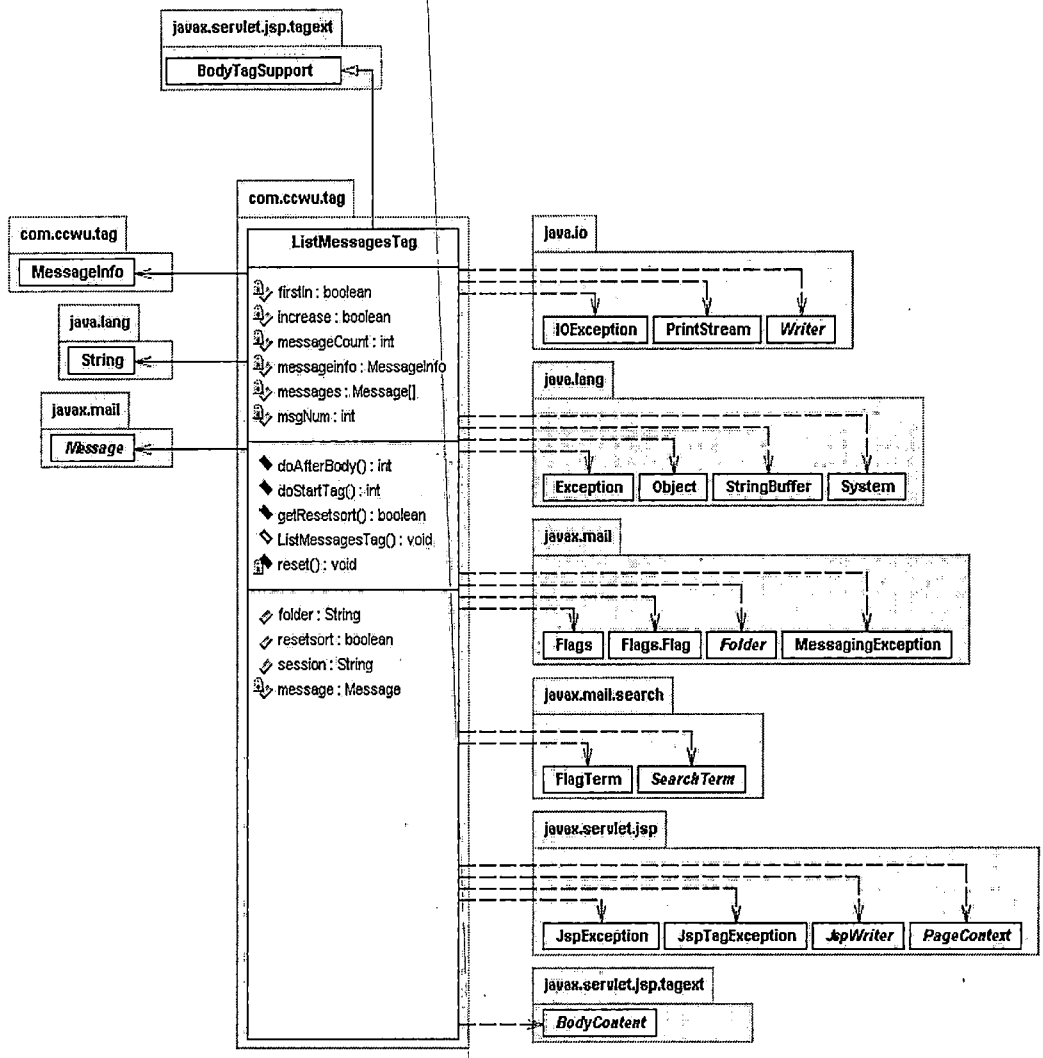

Figure 20. Class Diagram of ListMessagesTag.java

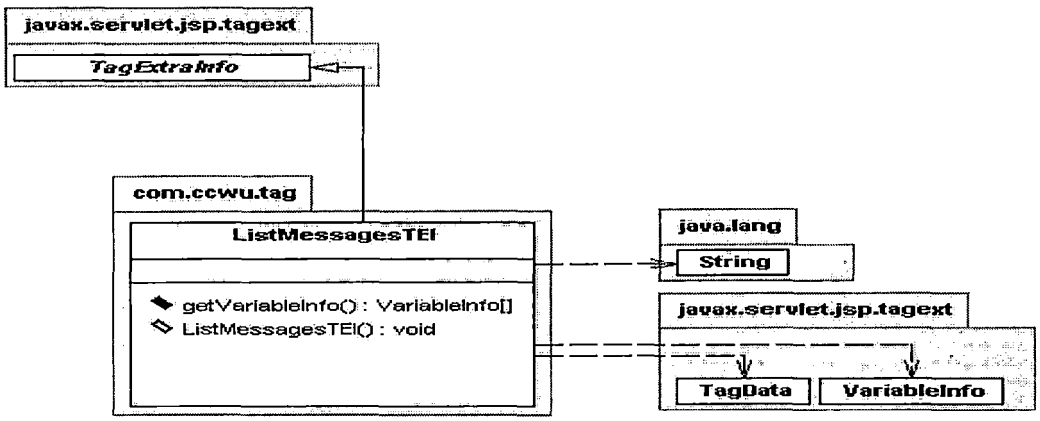

Figure 21. Class Diagram of ListMessagesTEI.java

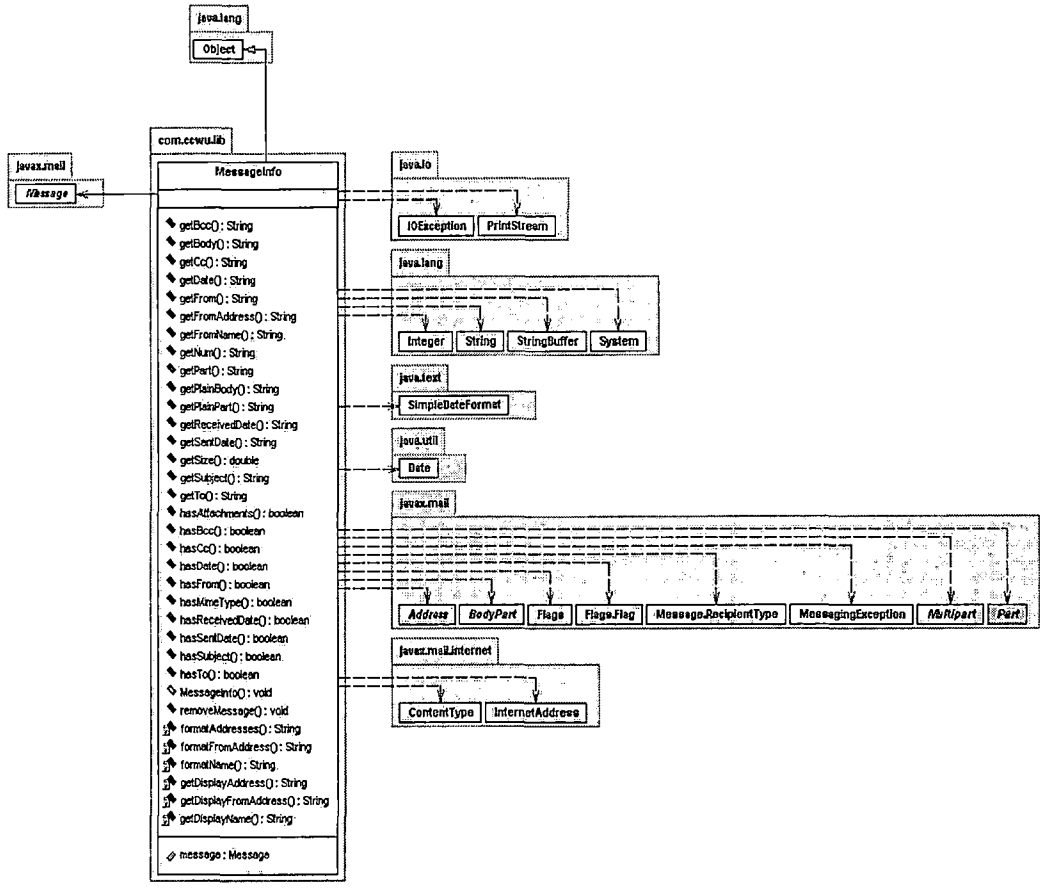

Figure 22. Class Diagram of MessageInfo.java

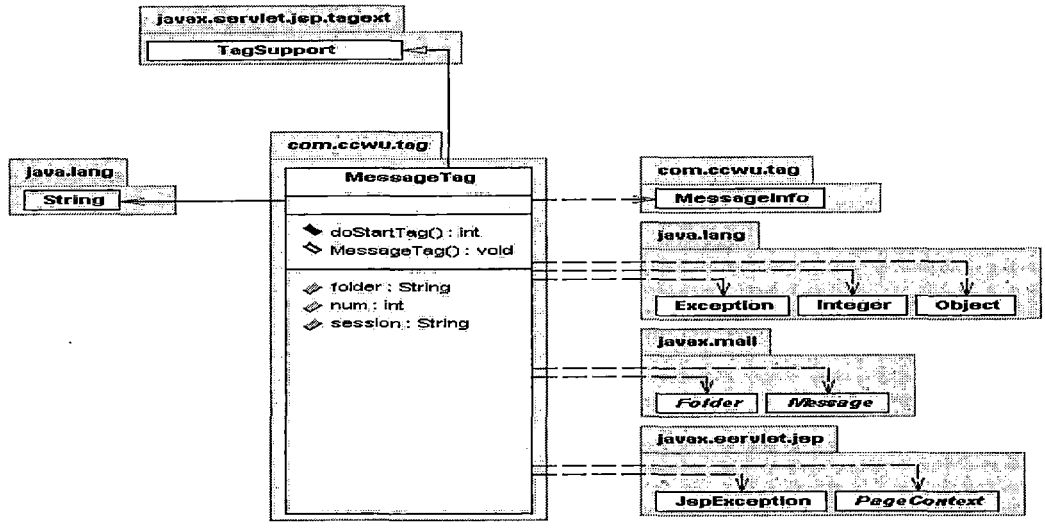

Figure 23. Class Diagram of MessageTag.java

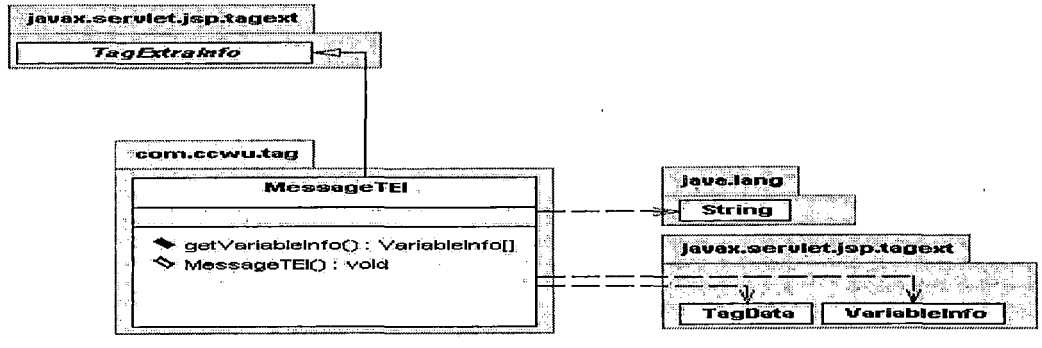

Figure 24. Class Diagram of MessageTEI.java

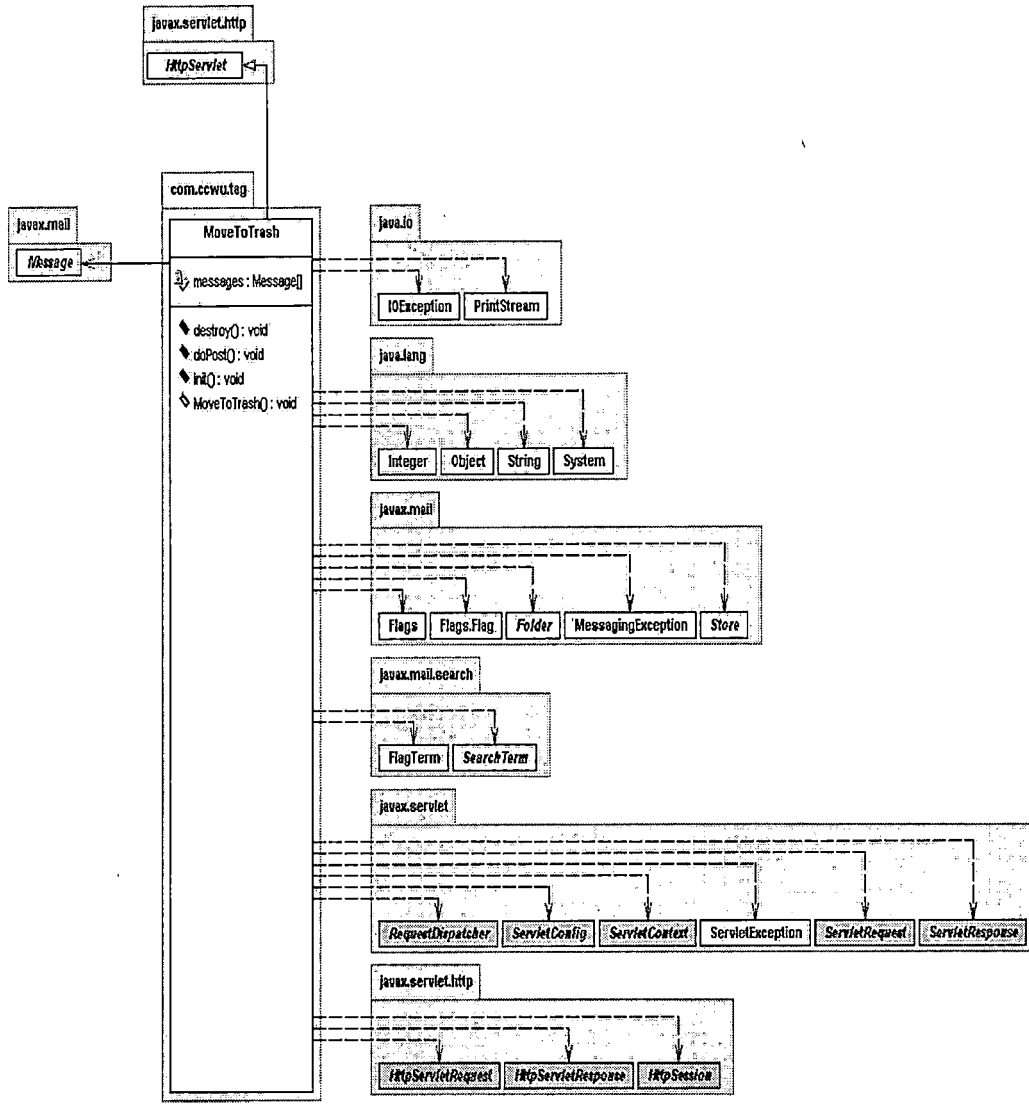

Figure 25. Class Diagram of MoveToTrash.java

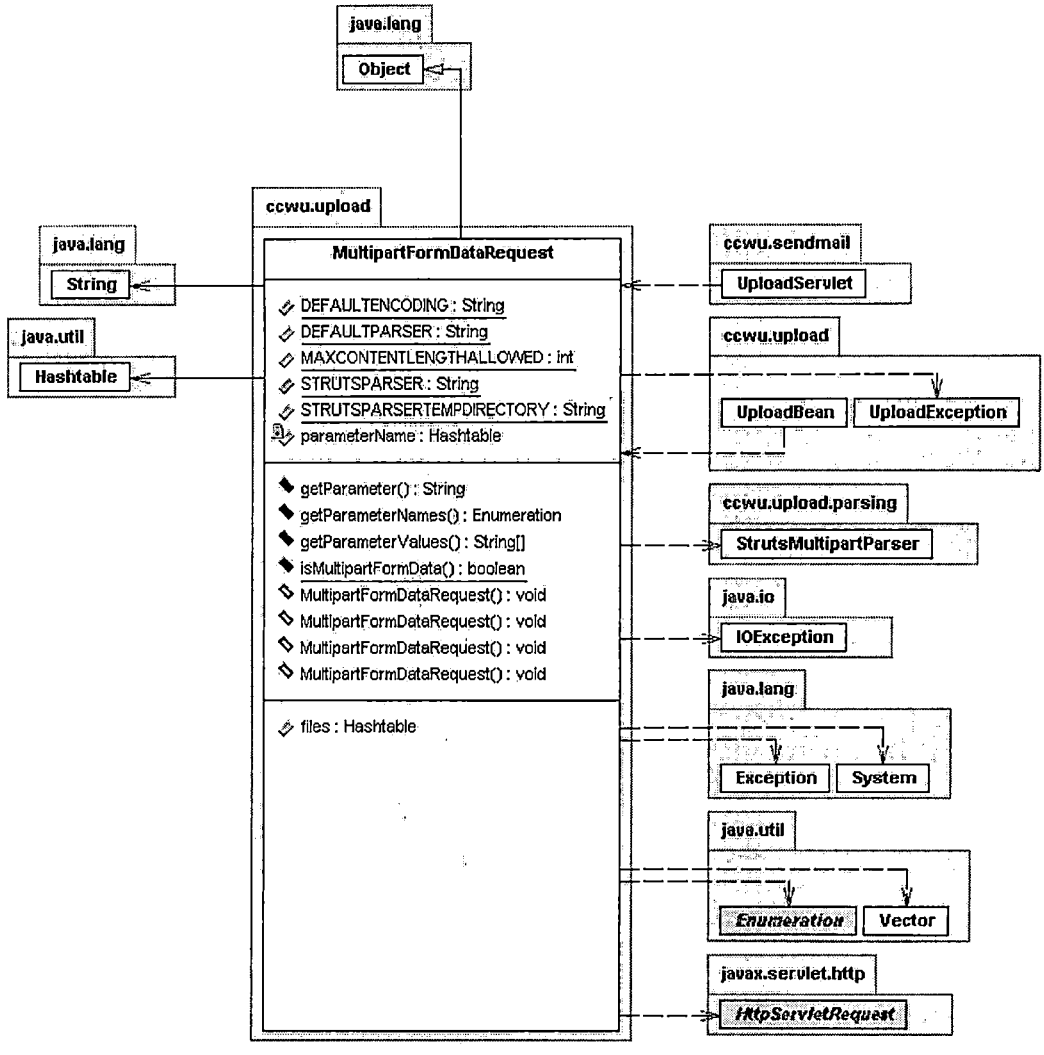

Figure 26. Class Diagram of MultipartFormdataRequest.java

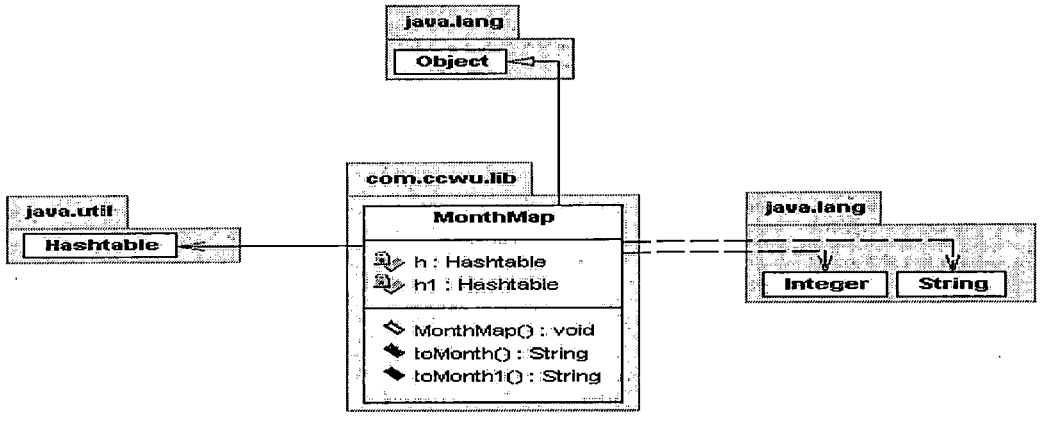

Figure 27. Class Diagram of MonthMap.java

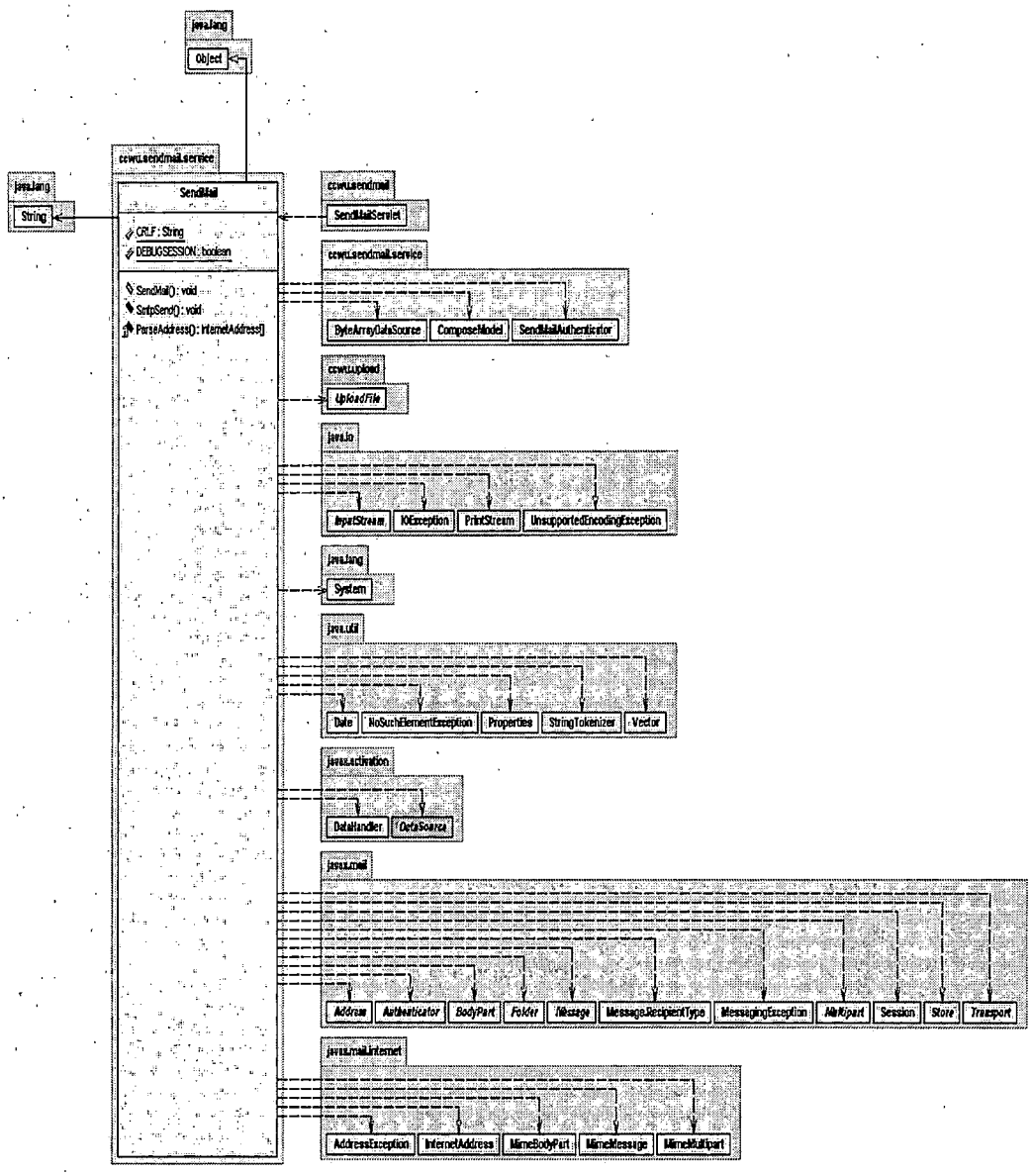

Figure 28. Class Diagram of SendMail.java

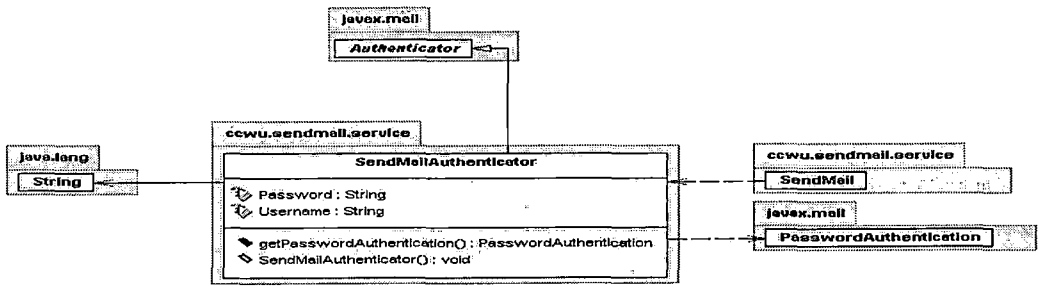

Figure 29. Class Diagram of SendMailAuthenticator.java

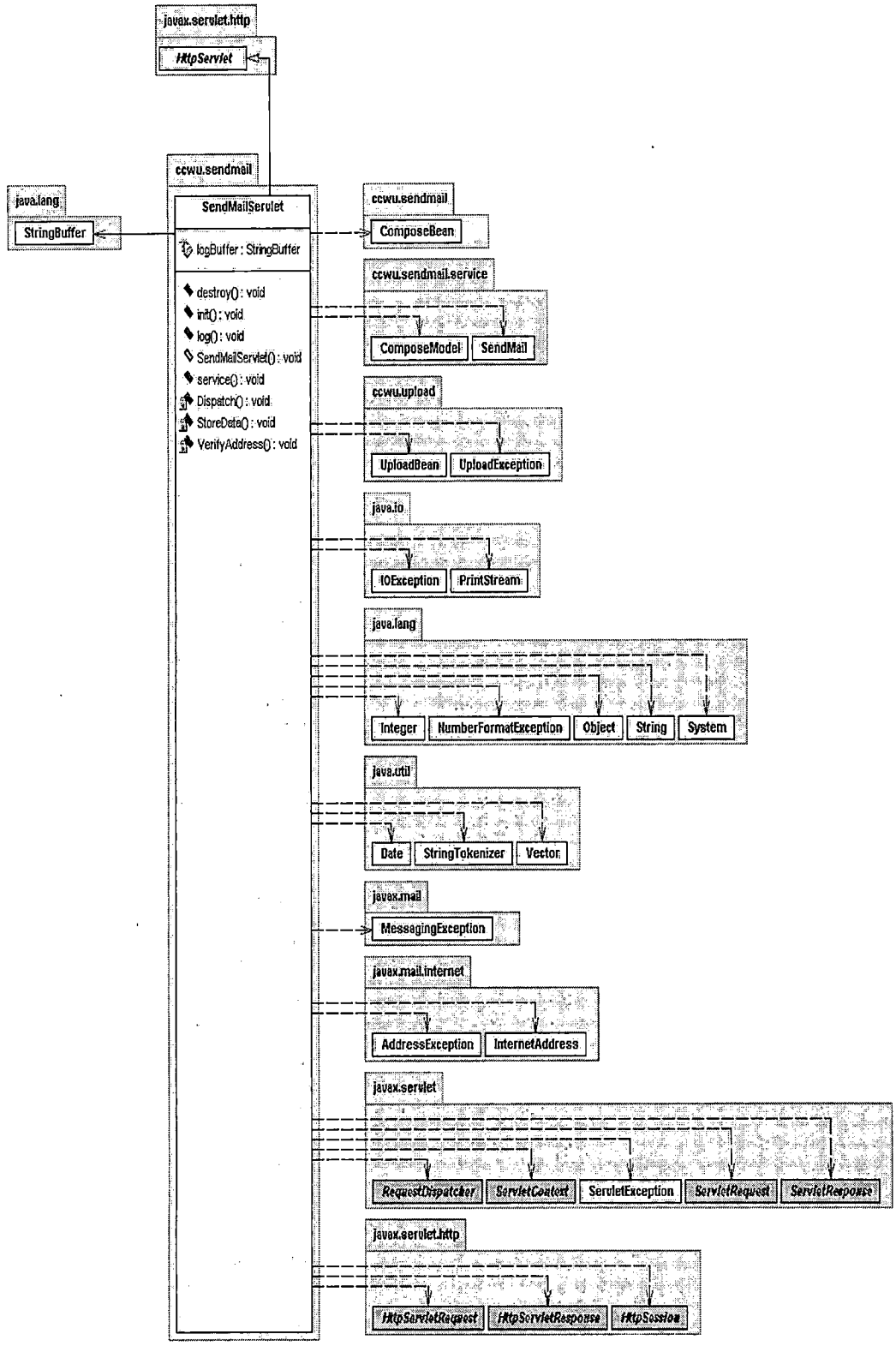

Figure 30. Class Diagram of SendMailServlet.java

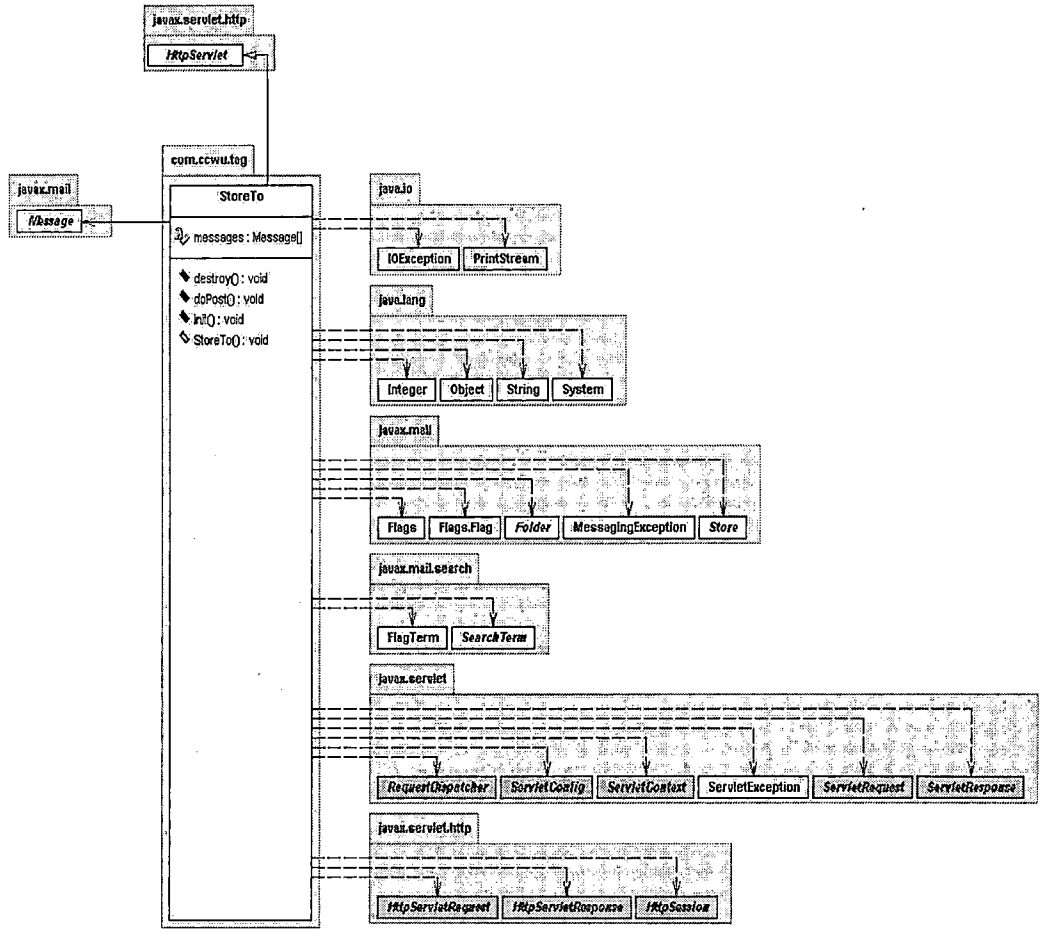

Figure 31. Class Diagram of StoreTo.java

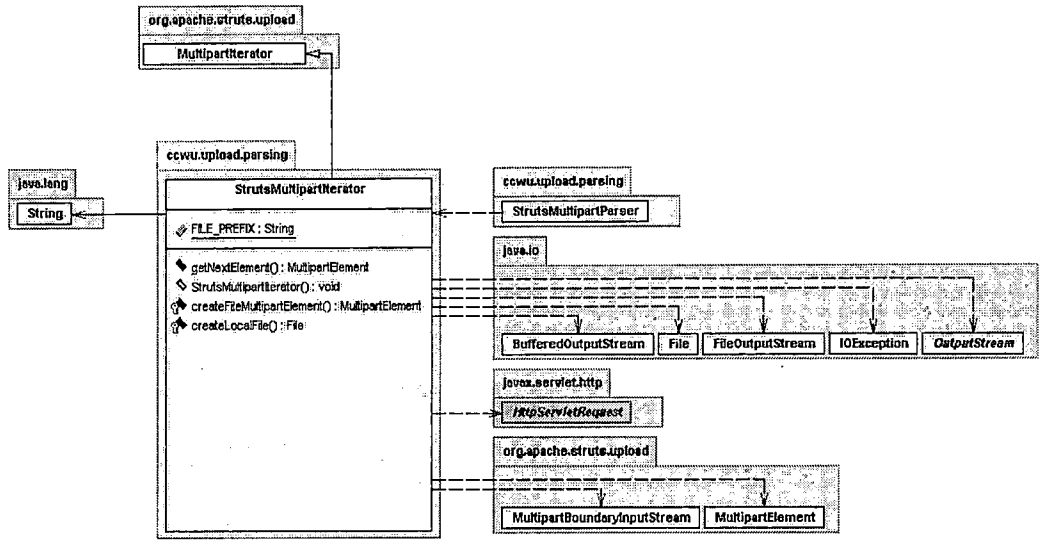

Figure 32. Class Diagram of StrutMultipartIterator.java

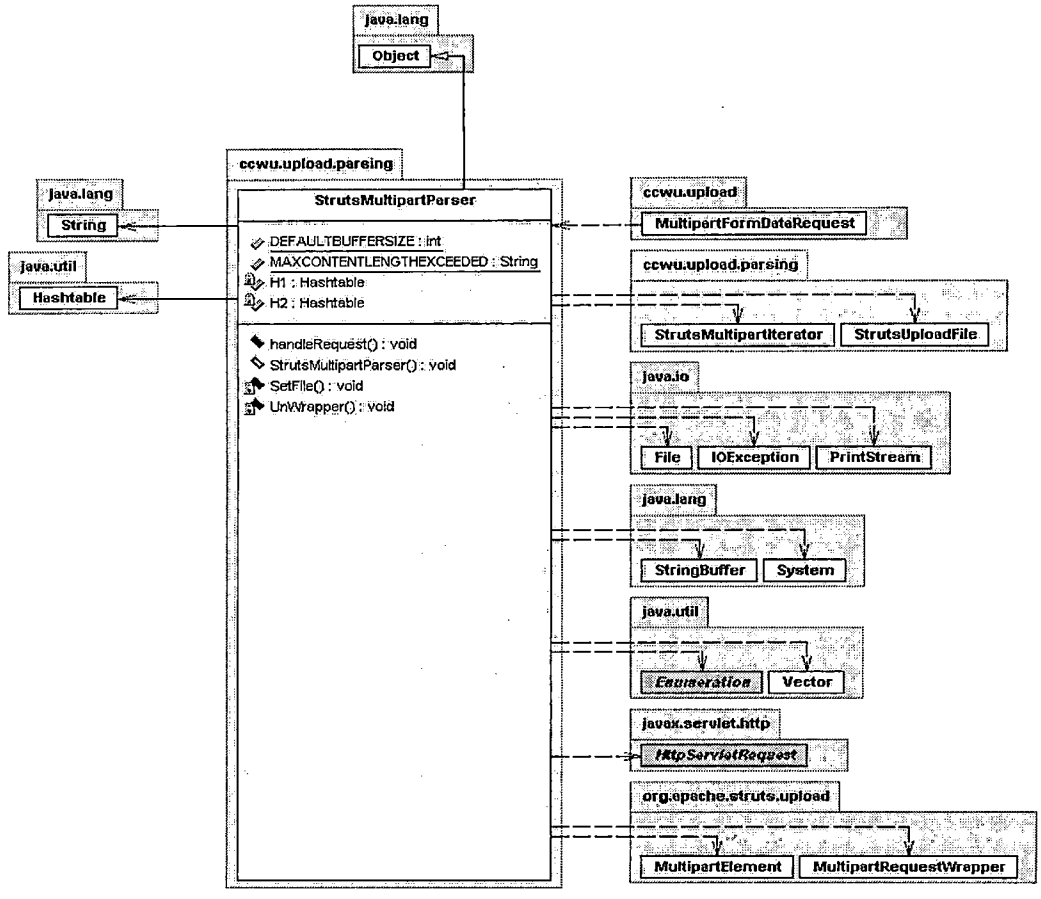

Figure 33. Class Diagram of StrutMultipartParser.java

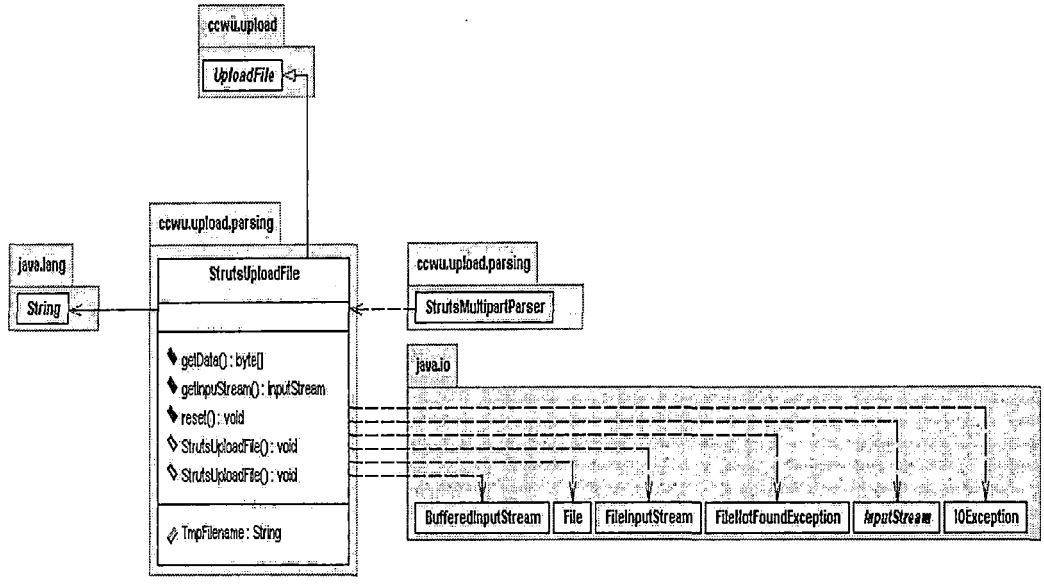

Figure 34. Class Diagram of StrutUploadFile.java

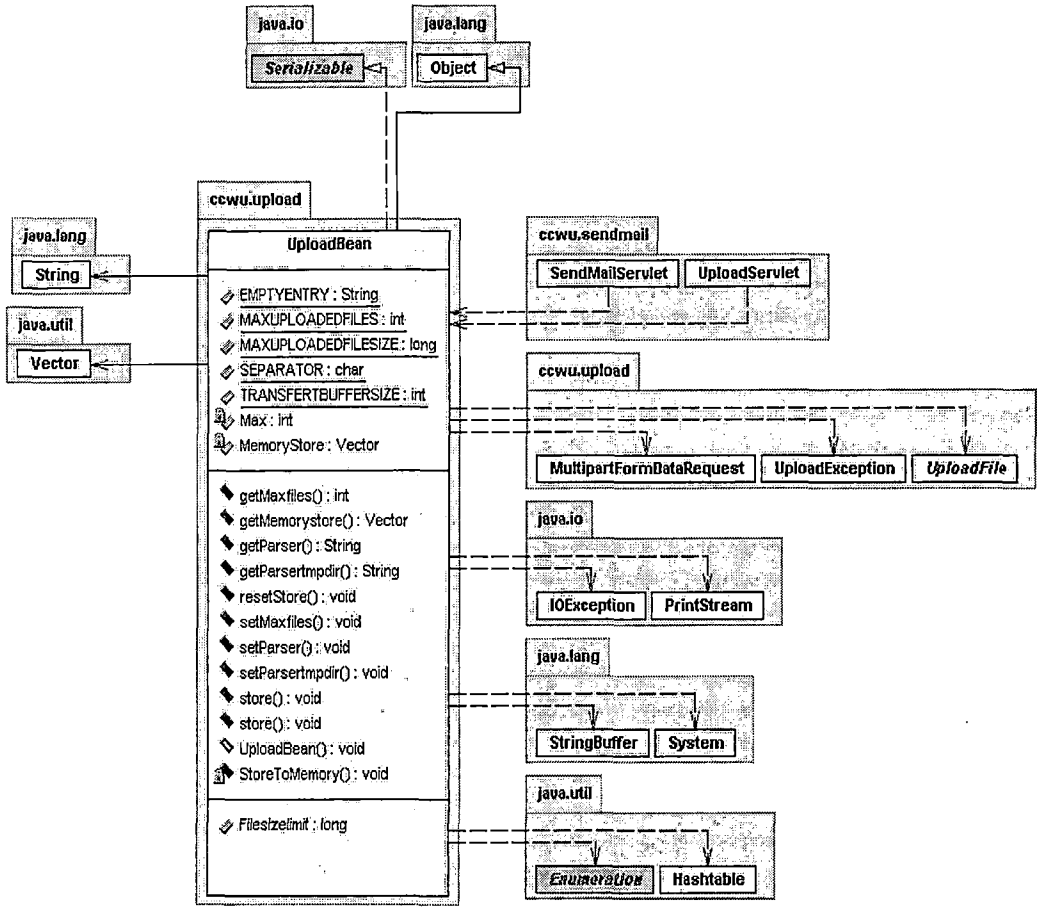

Figure 35. Class Diagram of UploadBean.java

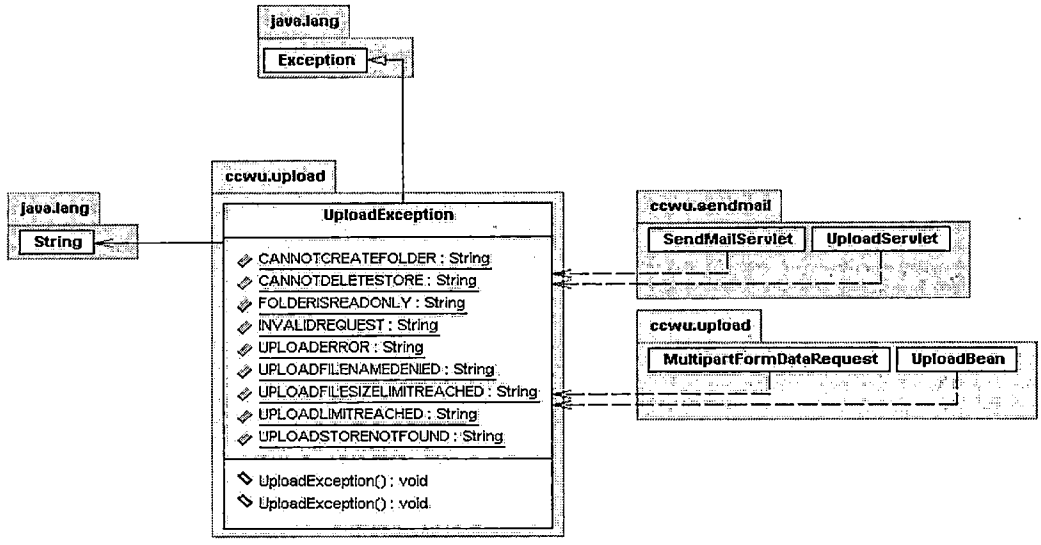

Figure 36. Class Diagram of UploadException.java

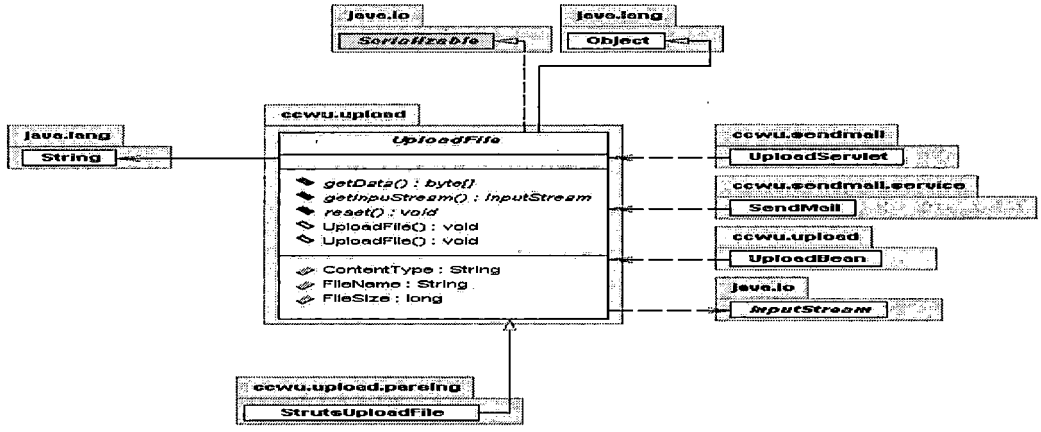

Figure 37. Class Diagram of UploadFile.java

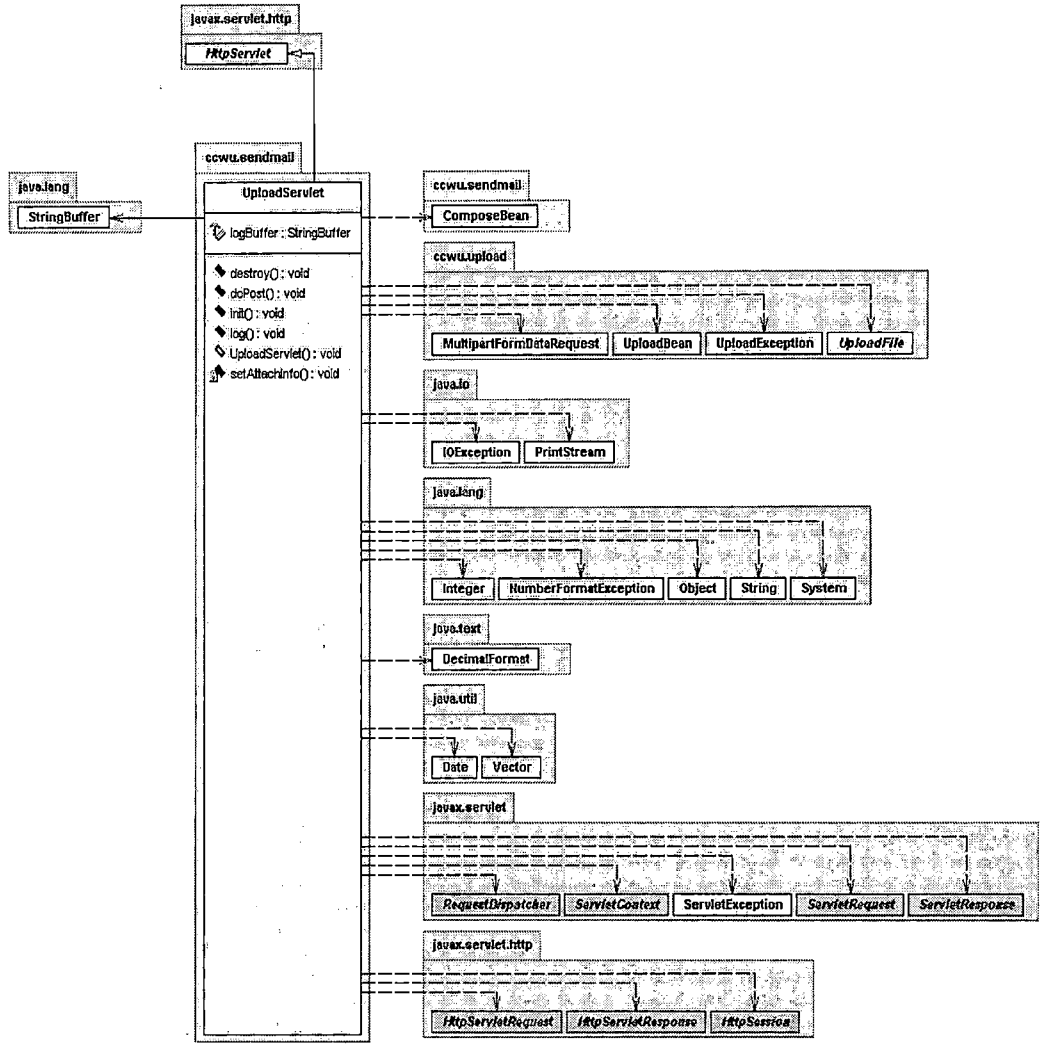

Figure 38. Class Diagram of UploadServlet.java

4.2 Web-based E-mail Client for Computer Science Graphical User Interface Design

WBECC GUI is easy to use. The GUI is written using Hyper Text Markup Language (HTML) Version 6.0 forms and frames. Also, it also uses JavaScript to check the user input's accuracy. Hence, the WBECC GUI is executable under Internet Explorer 5.0 or greater. The following sub sections explain the GUI work and details.

4.2.1 Web-based E-mail Client for Computer Science Login

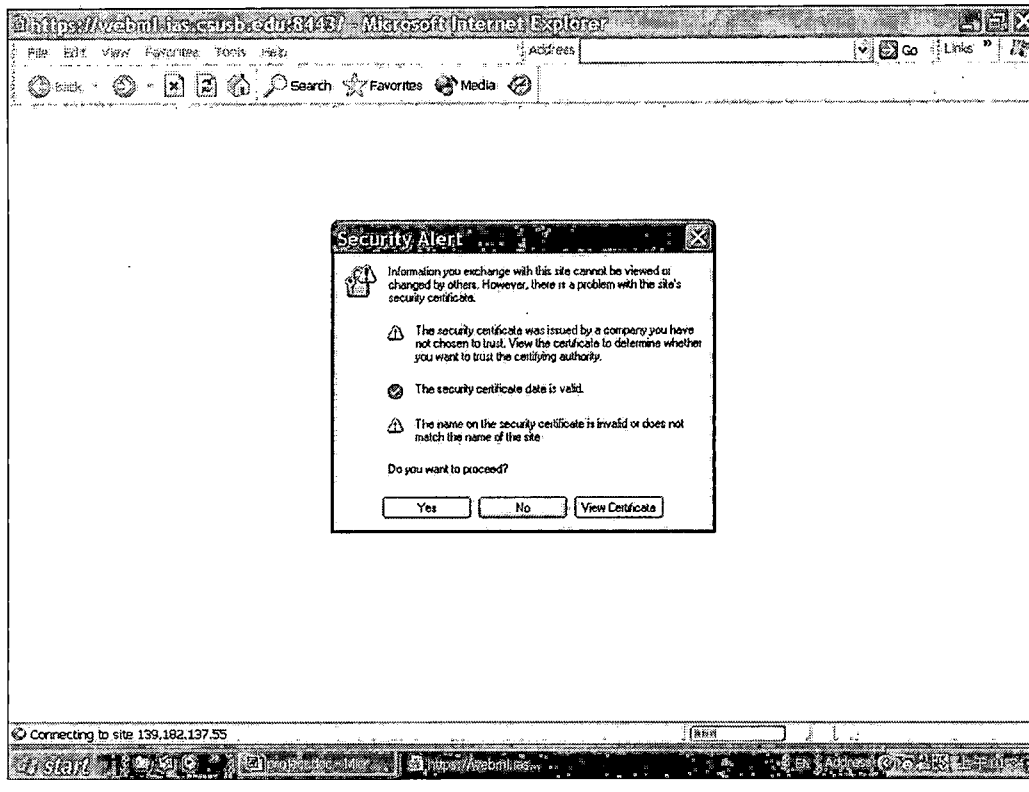

Figure 39. Security Alert before Entering the Login Page

When the user connects to WebMail page in CSCI, A security alert will pop up. This is because the entire WebMail applications were run over Secure Socket Layer (SSL). This technology allows Web browsers and Web servers to communicate over a secured connection. In this secure connection, the data that is being sent is encrypted before being sent, then decrypted upon receipt and prior to processing. Both the browser and the server encrypt all traffic before sending any data. After the user click

"yes" button in the secure alert page, then s/he can enter the Login page.

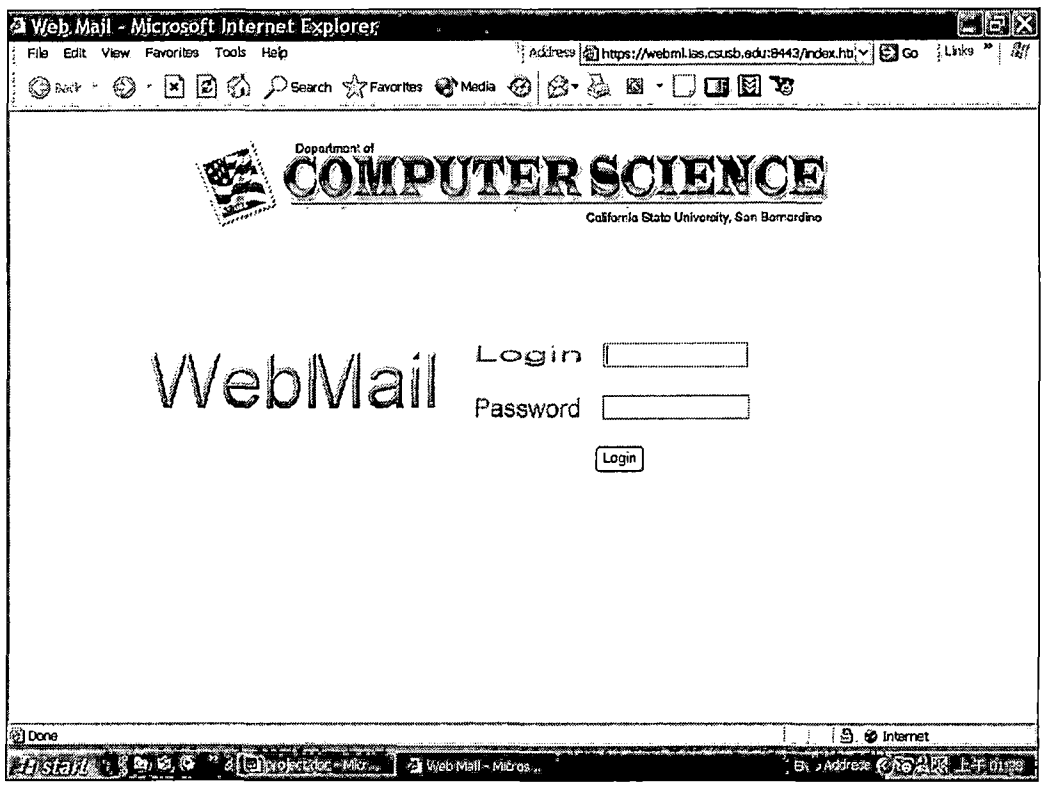

Figure 40. Login Page

The user logs in by providing the user's CSCI account id and password. After verifying the user id and password, the JSP program will send the page to main menu. If the user id or password is error, the program will show the error message and the user can re-login.

4.2.2 Inbox Folder

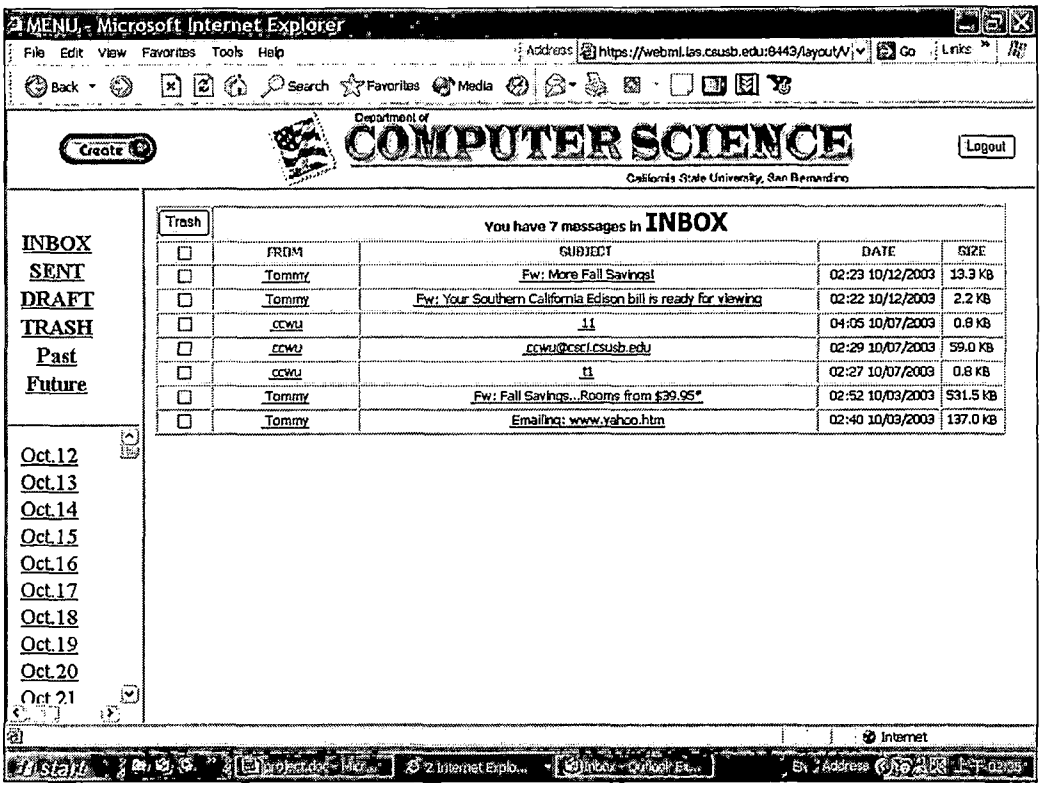

Figure 41. Messages in the Inbox Folder

After logging in, the WebMail go directly to the "Inbox" folder. The "Inbox" folder shows all message headers information in INBOX. The information includes: where is the message form, the message subject, the message sent date and the message size. The User can click on each subject to view the message contents.

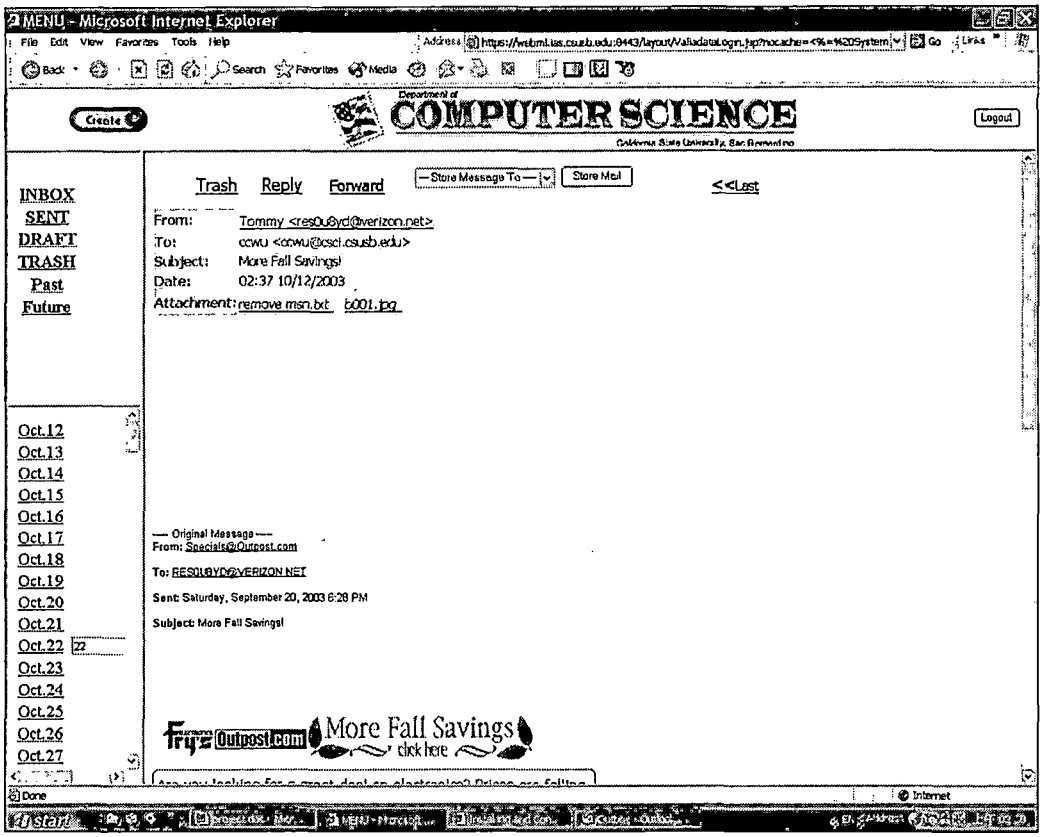

Figure 42. View the Message Content

A message with html document can be viewed directly in line with the message from, to, subject, date and attachments information. You can press "Last" or "Next" to view the last or next message. Or press the hyperlink of the sender's address to write a message to him/her. To put the message to trash, just click "Trash" hyperlink, then the message will be moved to the trash folder.

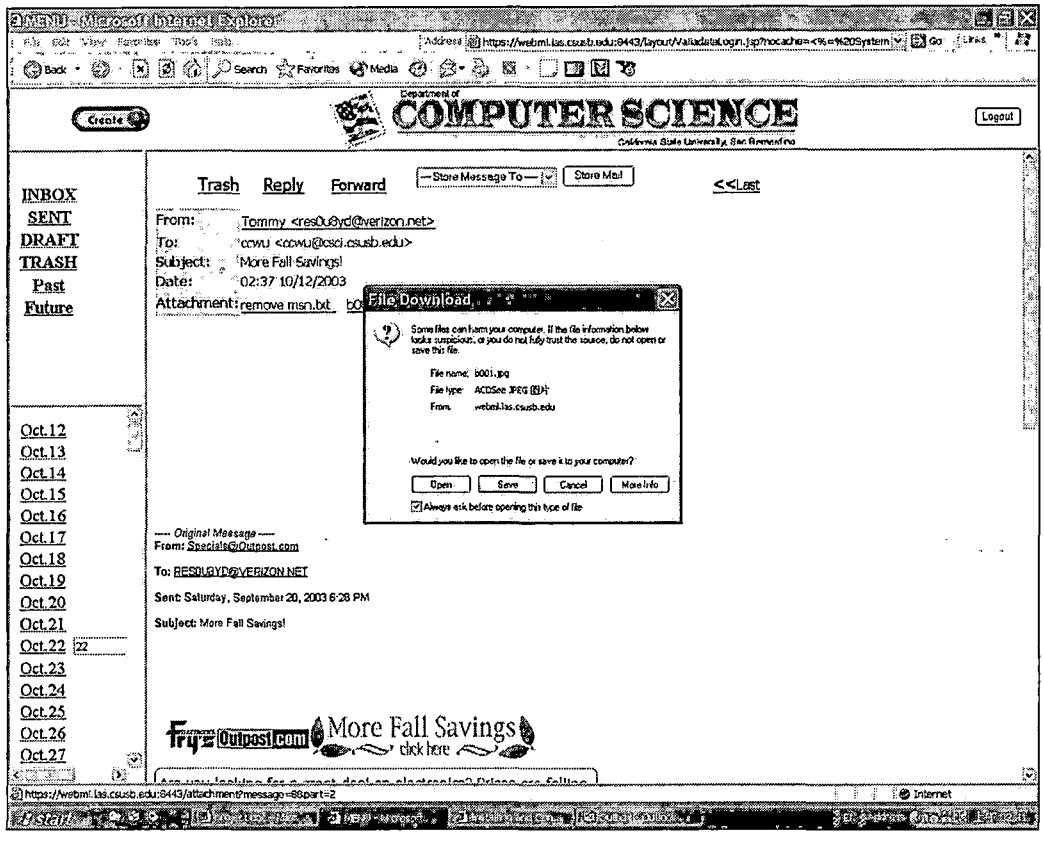

Figure 43. Handle the Message Attachments

Just click on the hyperlink of each attachment, you can select to open the attached file to view or save it to your computer. If you want to put this message into your trash folder, just click "Trash" hyperlink. The message will be moved to your trash folder. If you need reply or forward the message, just click "Reply" or "Forward" hyperlink. Then you can compose this message with its original or new content and attachments.

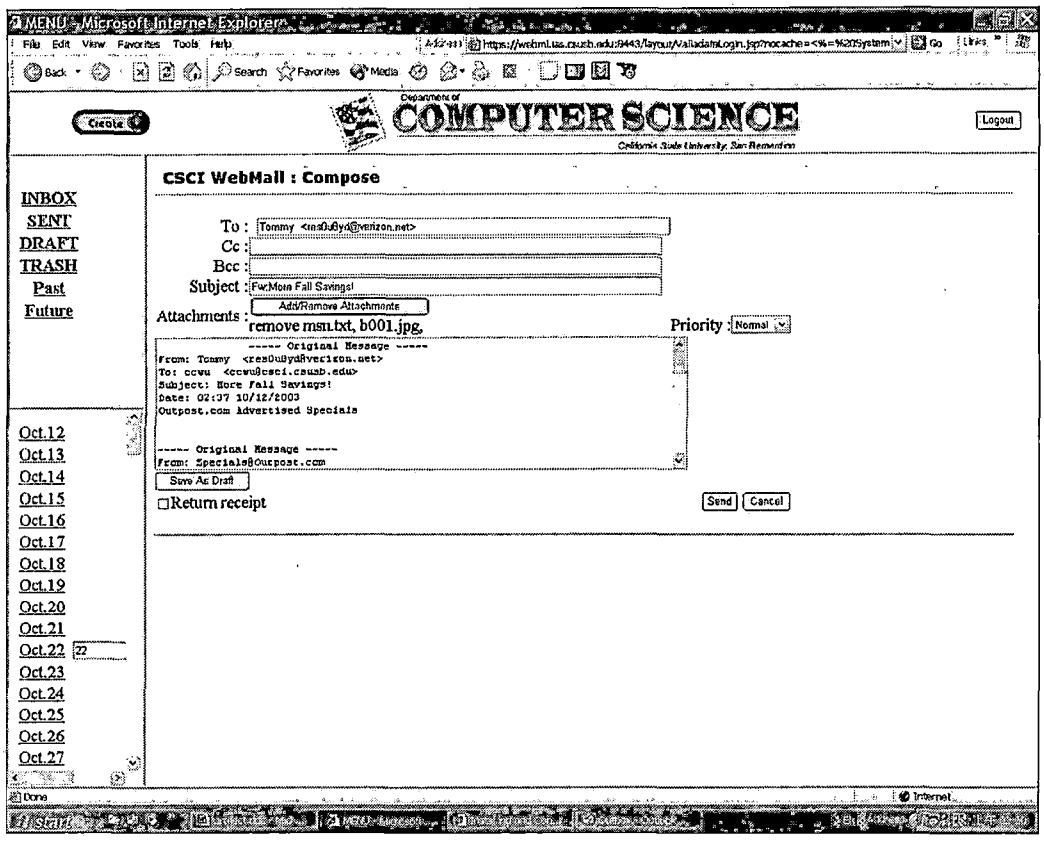

Figure 44. Message Reply

When you click "Reply" or "Forward", a window for composing a message will show up. For message reply, A "Reply To" address is already input in the "TO" entry. You can re-edit this message and add new attachments in it.

4.2.3 Sent Folder

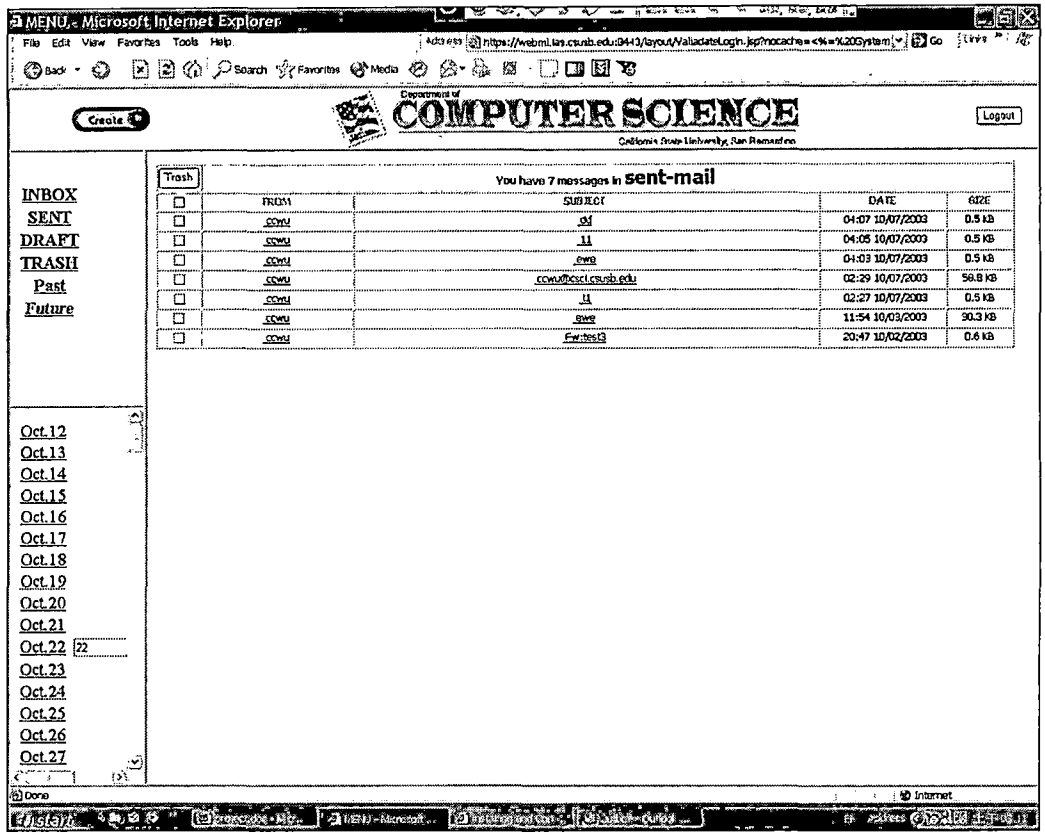

Figure 45. Messages in the Sent Folder

Every messages you sent out are automatically stored to the sent folder. You can view each messages like what you do in the Inbox folder or select messages to put into the trash folder.

4.2.4 Create a New Message

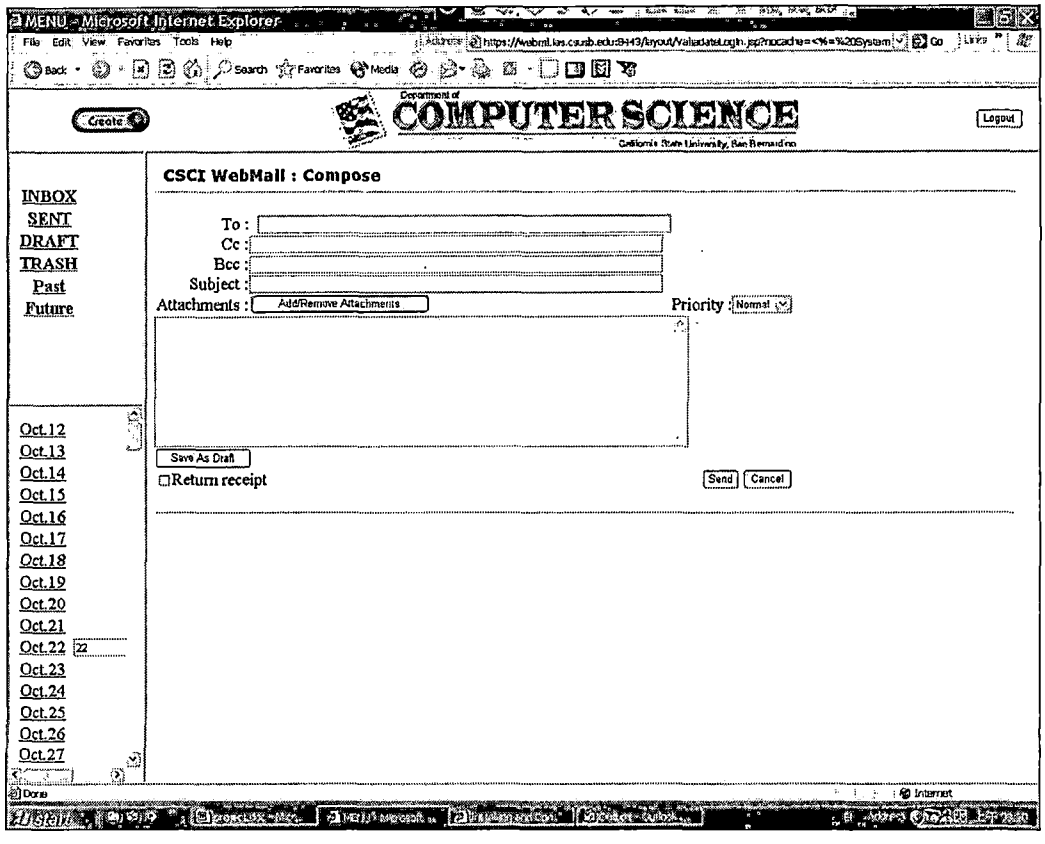

Figure 46. Create A New Message

You can compose a new message with address to, cc and bcc. If you need input more than one addresses, just use ";" to separate each e-mail address. You can also set its priority and ask to return a receipt. To attach files you just click the "Add/Remove Attachments" button to add or remove attachments.

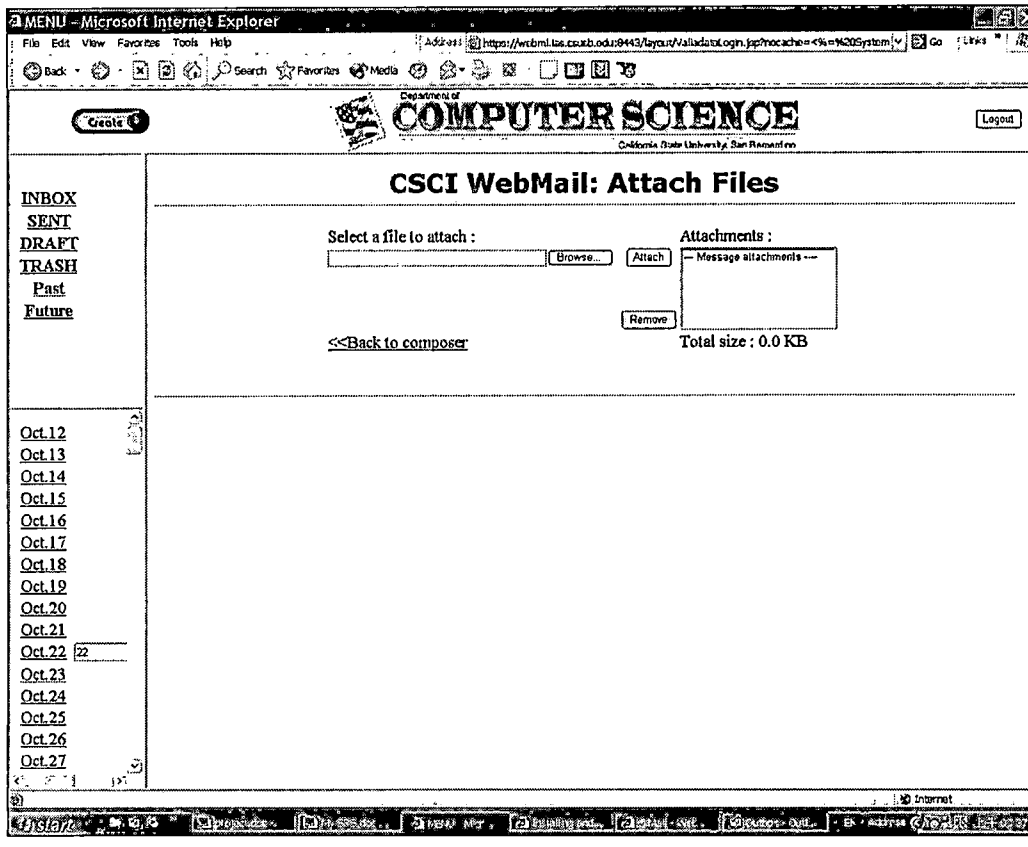

Figure 47. Select A File to Attach

To select any files form your computer, you click "Browser" button to view files in your computer, then select the file you want to attach.

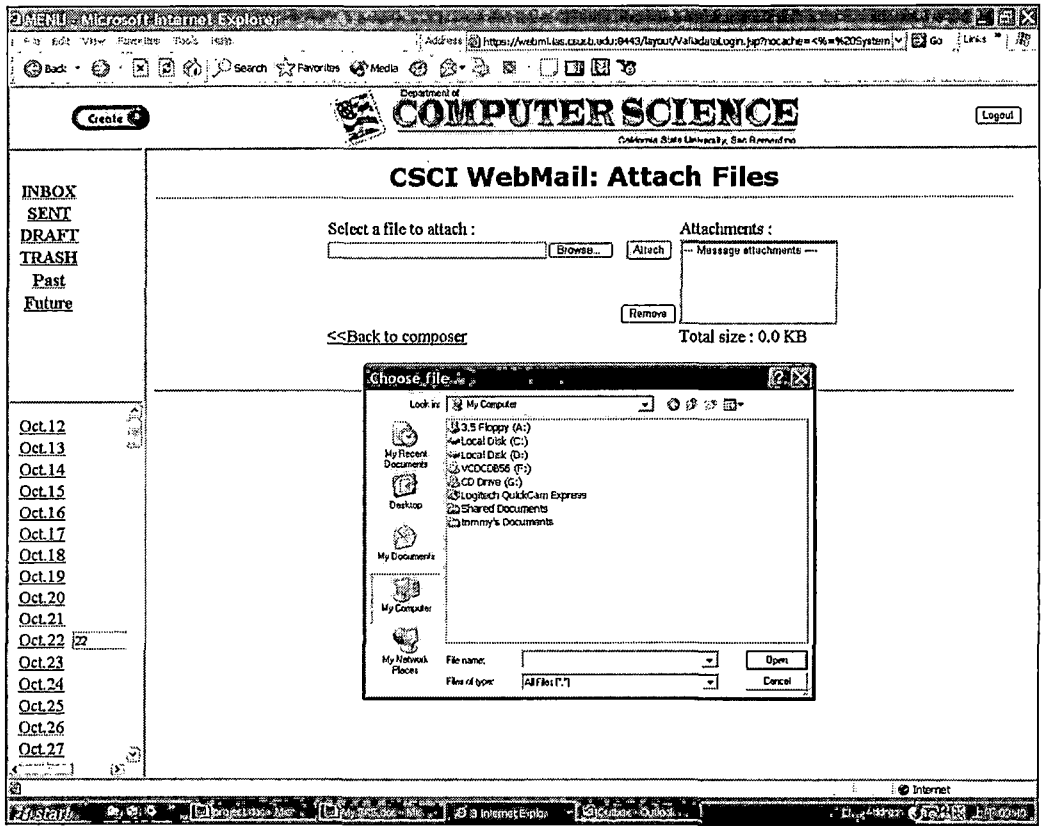

Figure 48. Browse your Computer to Select A File to Attach

After selecting the file you want to attach, you click "open" then "Attach" button, then the file in your computer will be uploaded to web server to package with your message contents.

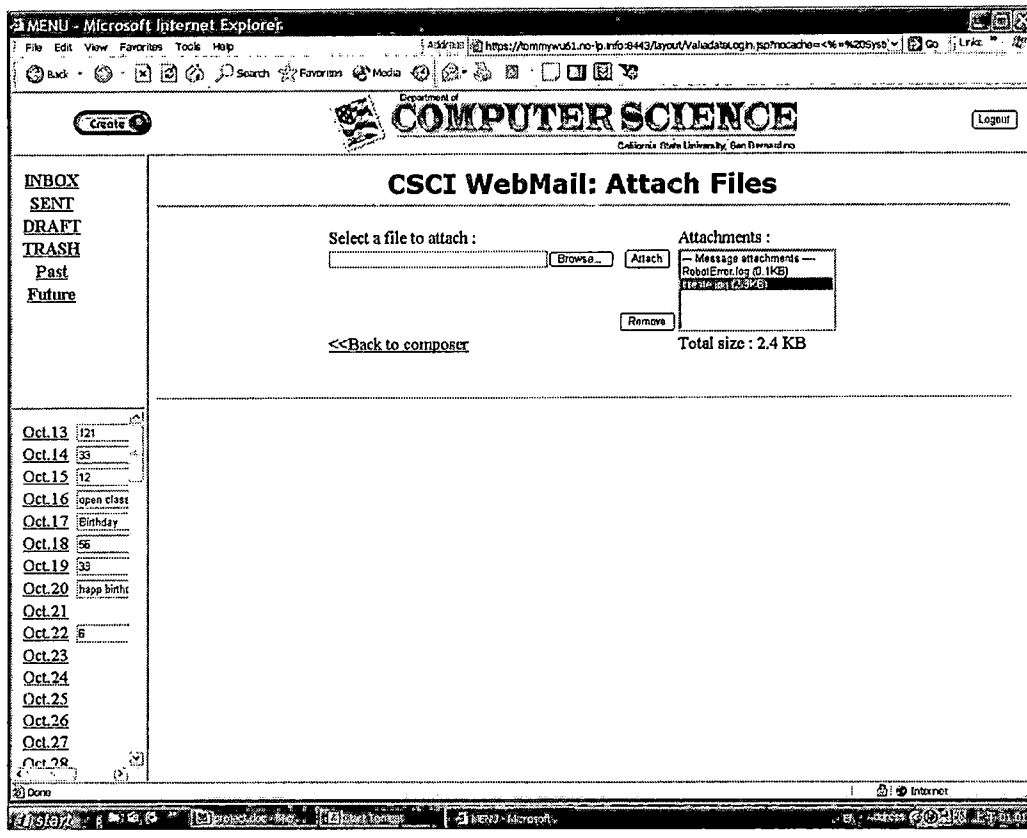

Figure 49. Remove Attached Files

After attaching files, you can also select files to remove. Just click "remove" button to remove that attached file. The total size of the attached files will show below. The maximum size of total attached files is 5 MB for each message. After composing your message, you click "Send" button, then the message with your attached files will be sent immediately.

4.2.5 Draft Folder

After composing your message, you can also save it to the draft folder. All the message contents including attached files will be stored to the draft folder.

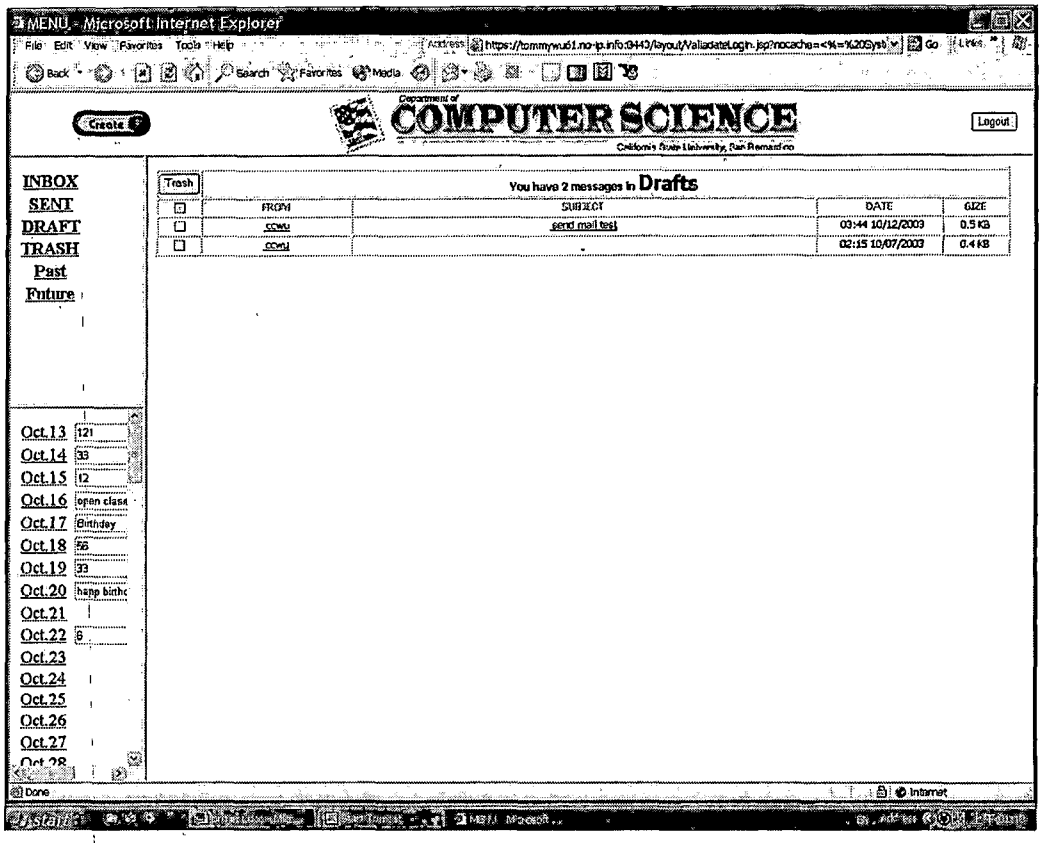

Figure 50. Messages in the Draft Folder

You can view each messages in the draft folder just like what you do in the inbox folder or select messages to put into the trash folder.

4.2.6 Trash Folder

It's better for you to delete the messages in your trash folder for a period of time. If your messages are in

manage your messages by day. When you view a message in any folder, you can save this message to any day of the future 183 days.

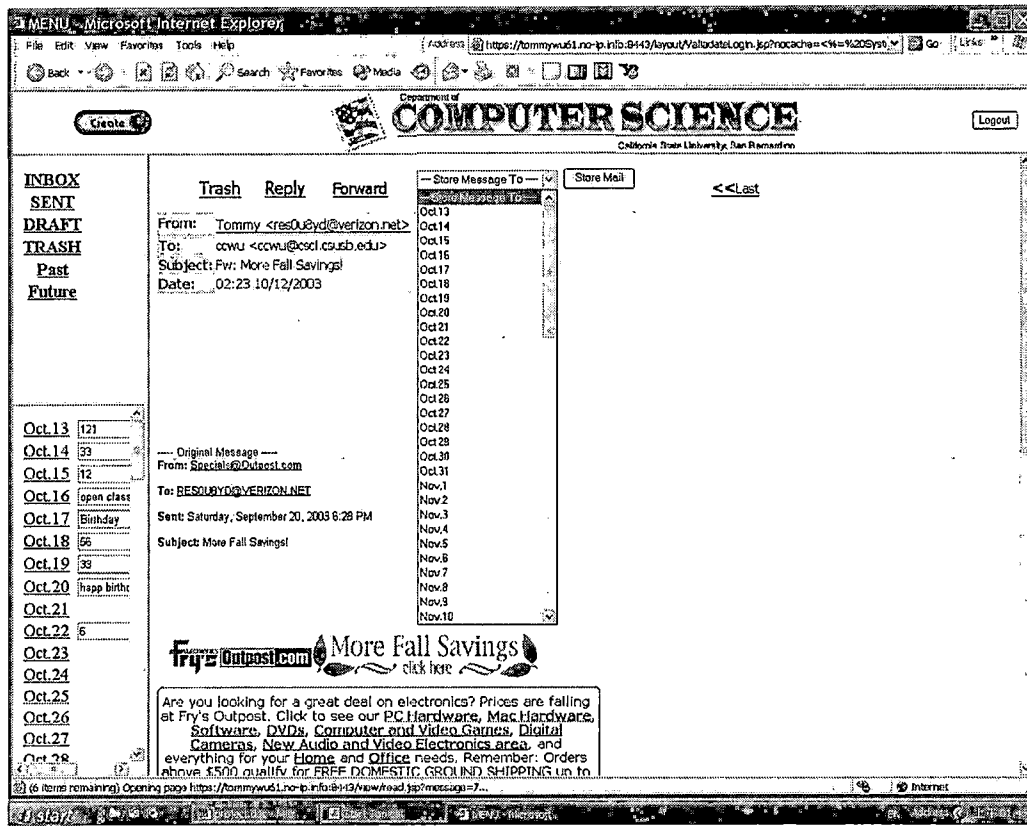

Figure 52. Store the Message by Day

You can select any day from the lists, then click "Store Mail" button. The message including its attachments will be moved to the day you selected immediately.

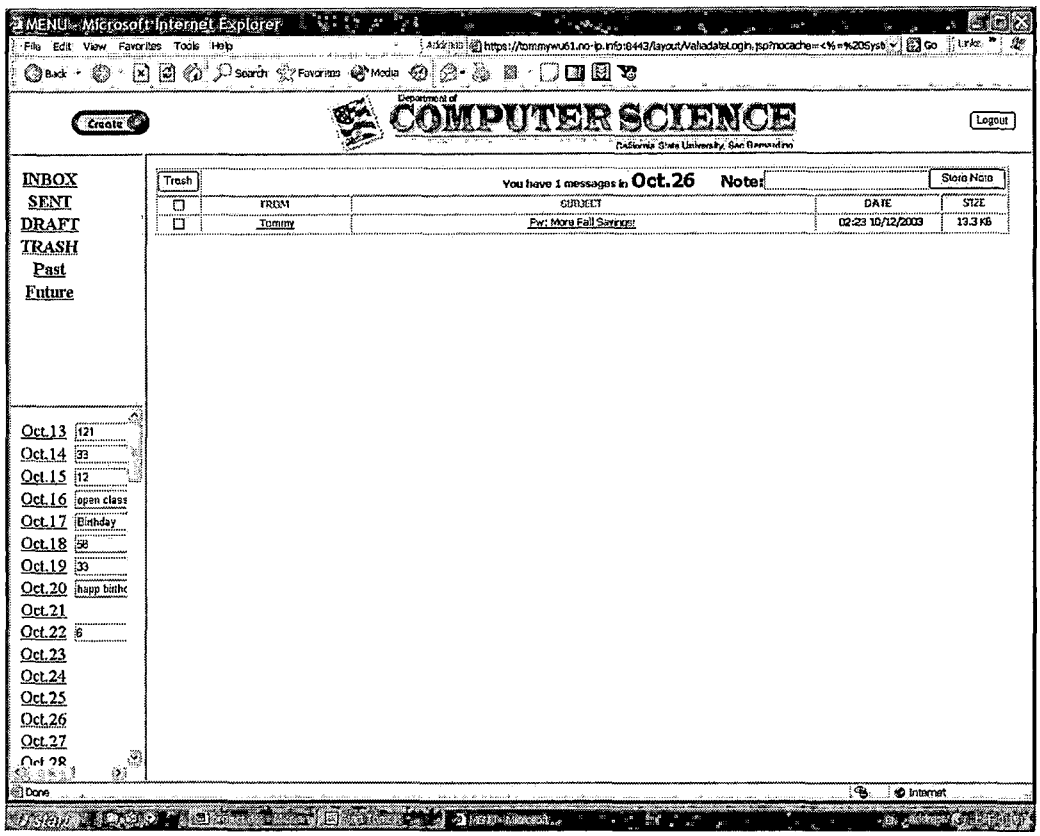

Figure 53. Messages in Oct. 26

After you storing messages to any day you selected, you can click the hyperlink of that day to view all messages in that day. Of course, you can click each subject hyperlinks to view its message contents.

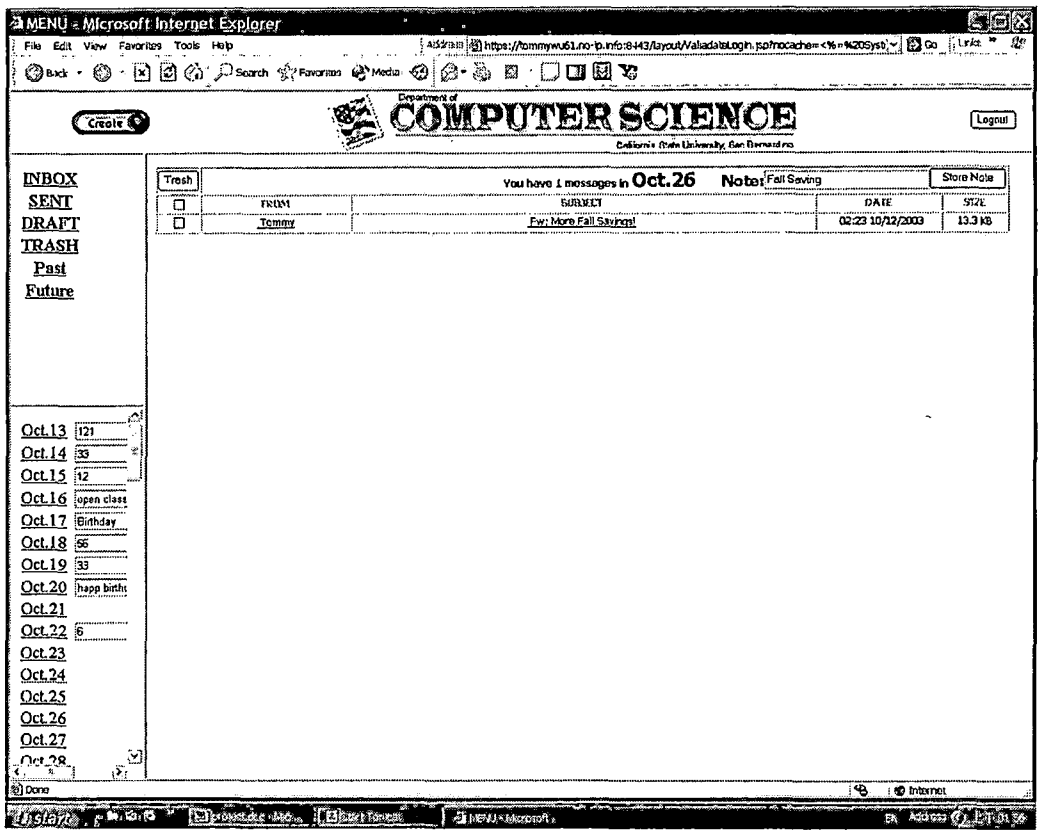

Figure 54. Make A Note on Oct.26

Or, you can make a note for that day, then press "Store Note" button to store your note.

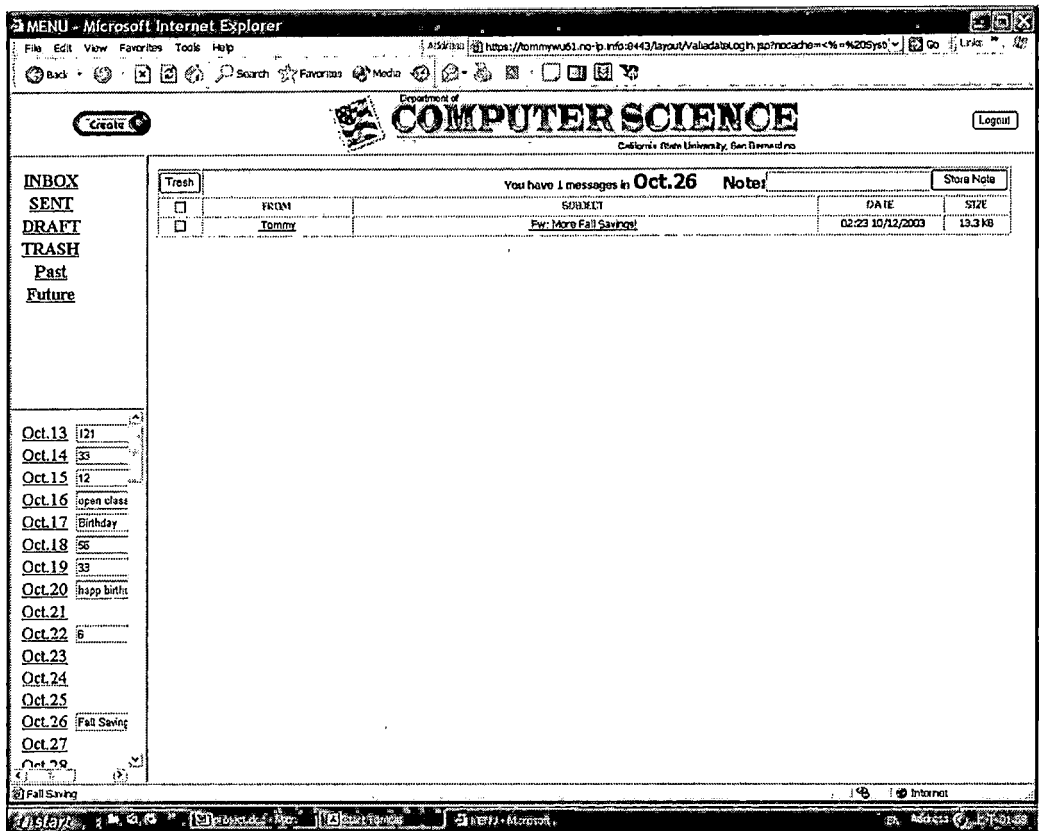

Figure 55. The Note "Fall Saving" on Oct.26

After you press "Store Note" button, the note will show on that day immediately. If the note is too long to show it all, just move your cursor to that note. A full note will show in the window status bar of your explorer.

4.2.8 Past Folder

In case you need to check your past daily folders, just click "Past" hyperlink. A decreasing date lists will list from today to the past 183 days.

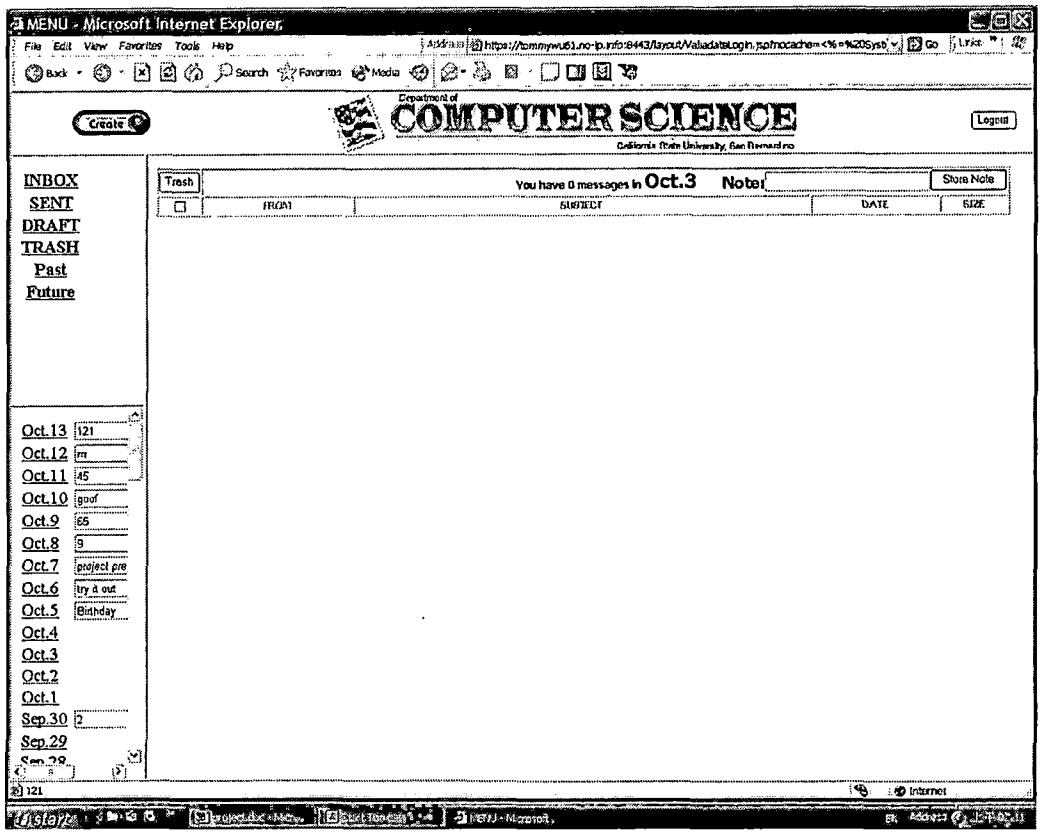

Figure 56. A Decreasing Date List

You can also check messages on each day, or make note for that passing day. Of course, it will show up immediately too. To return to check your future daily folder, just click the "Future" hyperlink.

4.2.9 Error Handle

Every inputs in the web page will have an error check. The error checks include username, password, e-mail address and other information.

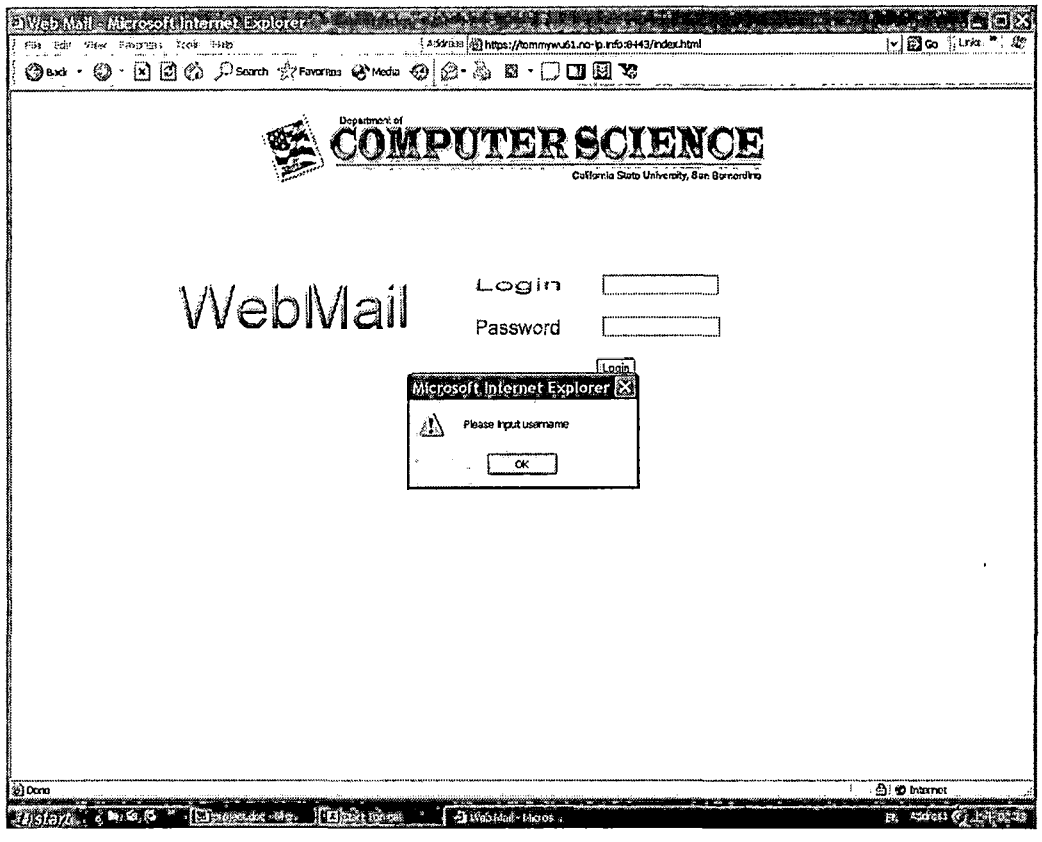

Figure 57. Error Messages for Blank Username Input

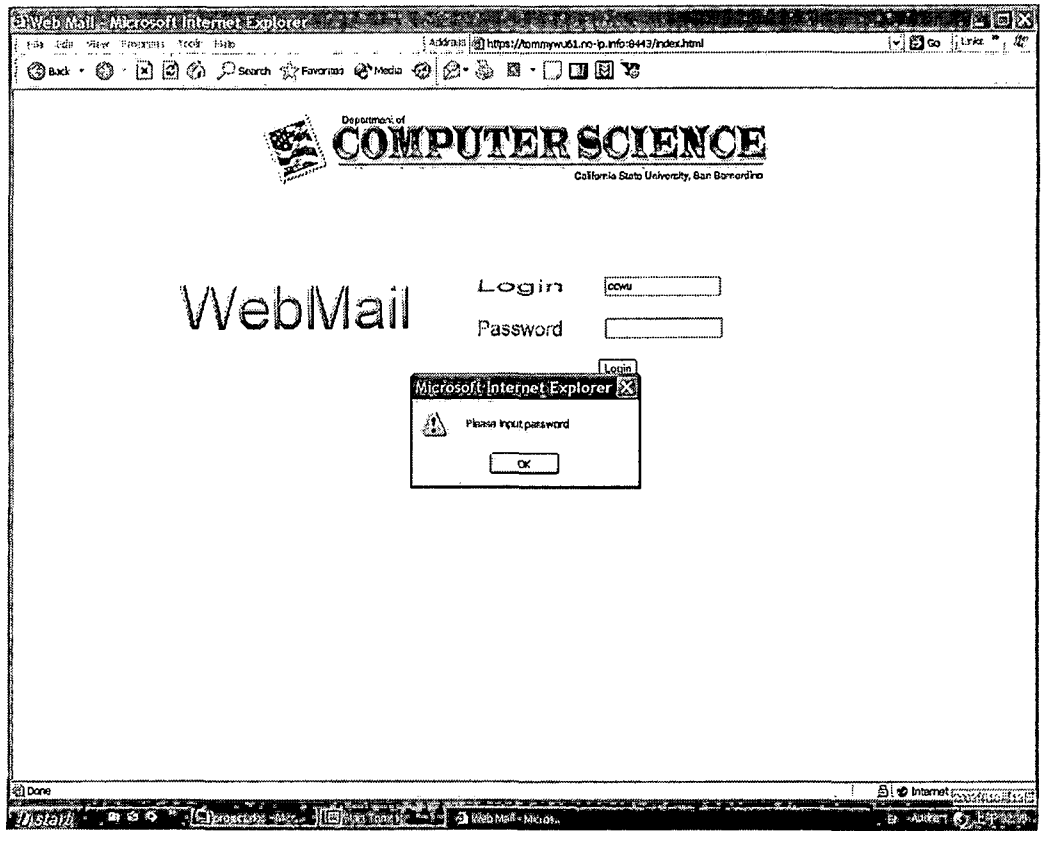

Figure 58. Error Messages for Blank Password Input

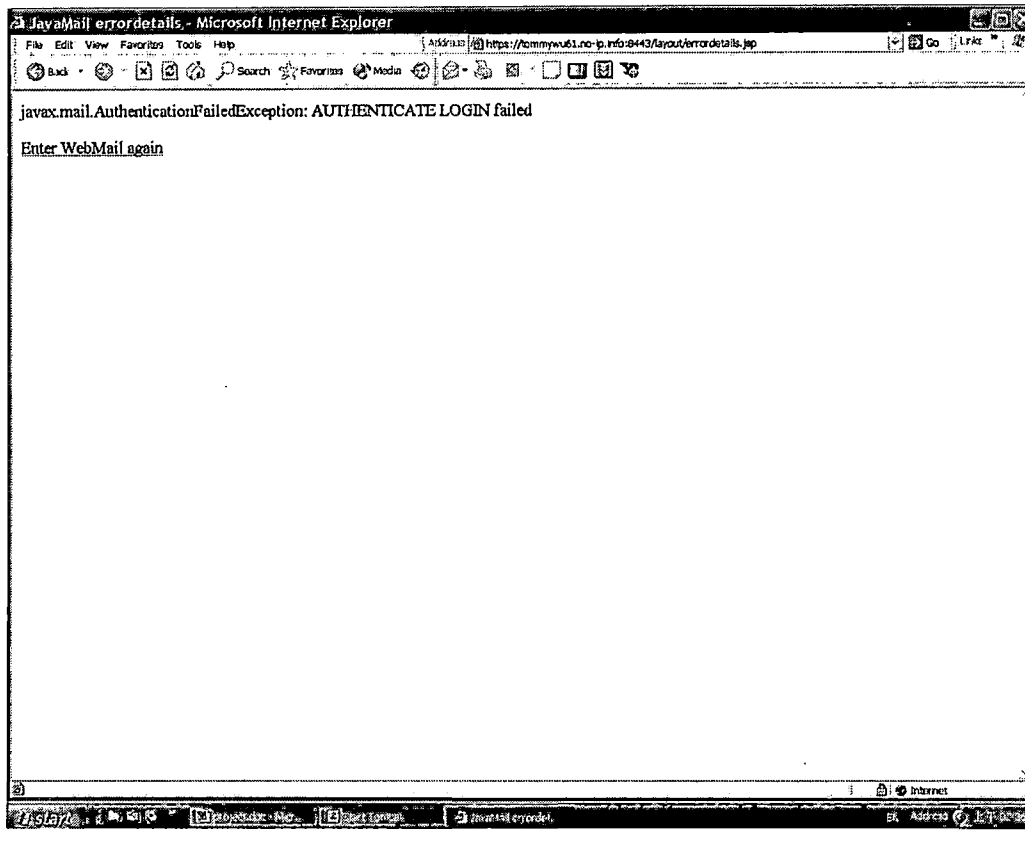

Figure 59. Error Messages for Incorrect Username or Password

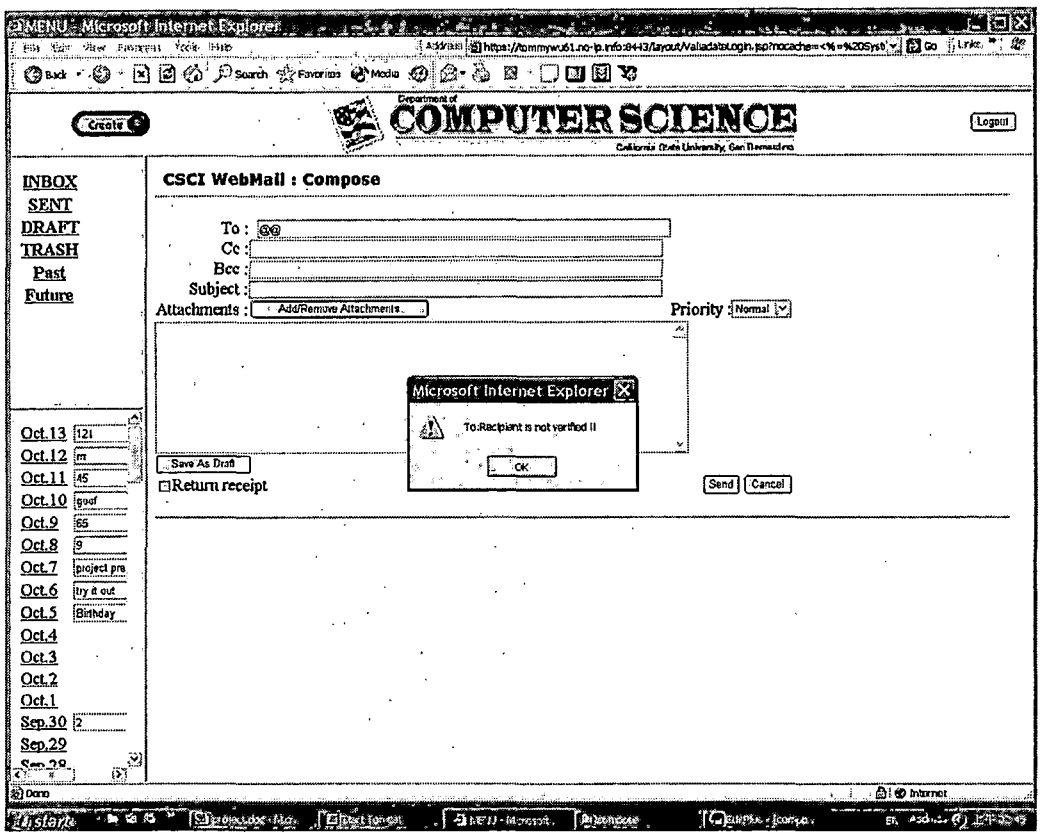

Figure 60. Error Messages for Blank or Error Address Inputs

An e-mail address should look like --@--, if the user input an e-mail address not in this form, an error message will pop up immediately.

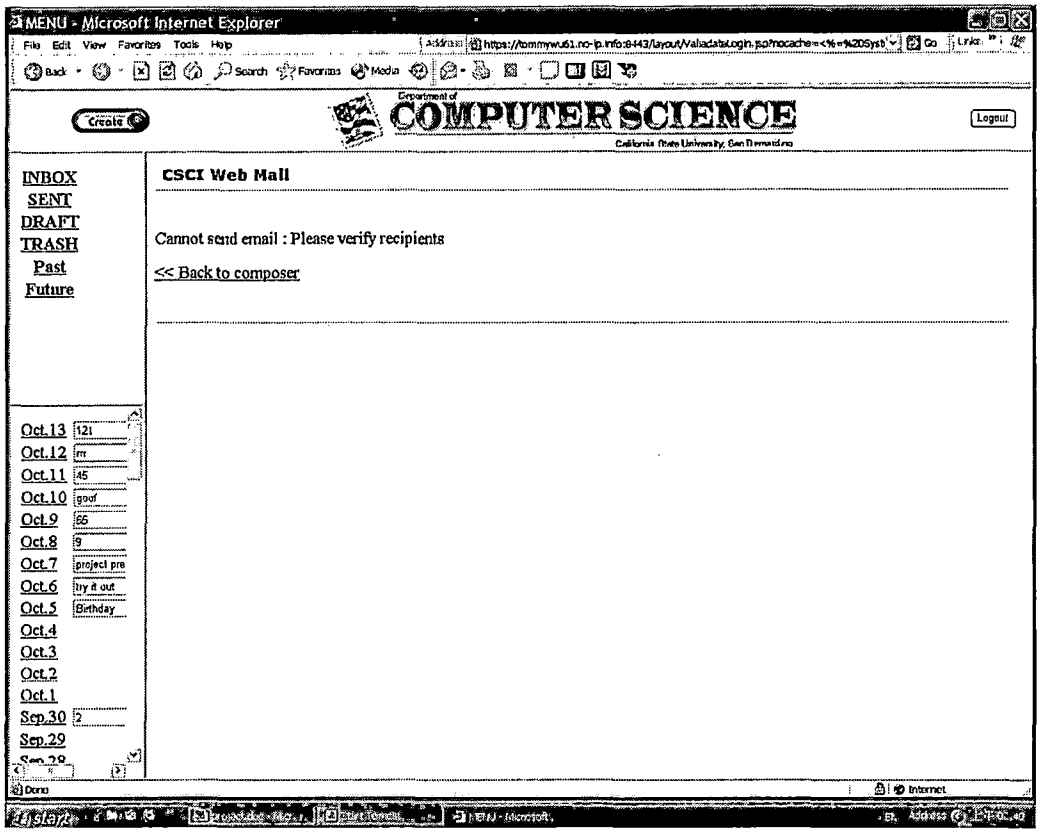

Figure 61. Error Messages for Incorrect Address Format

After you click "Send" button to send a message, if the e-mail address has other problem, a error message will show up, then you need press "<<Back to composer" hyperlink to check the e-mail address again.

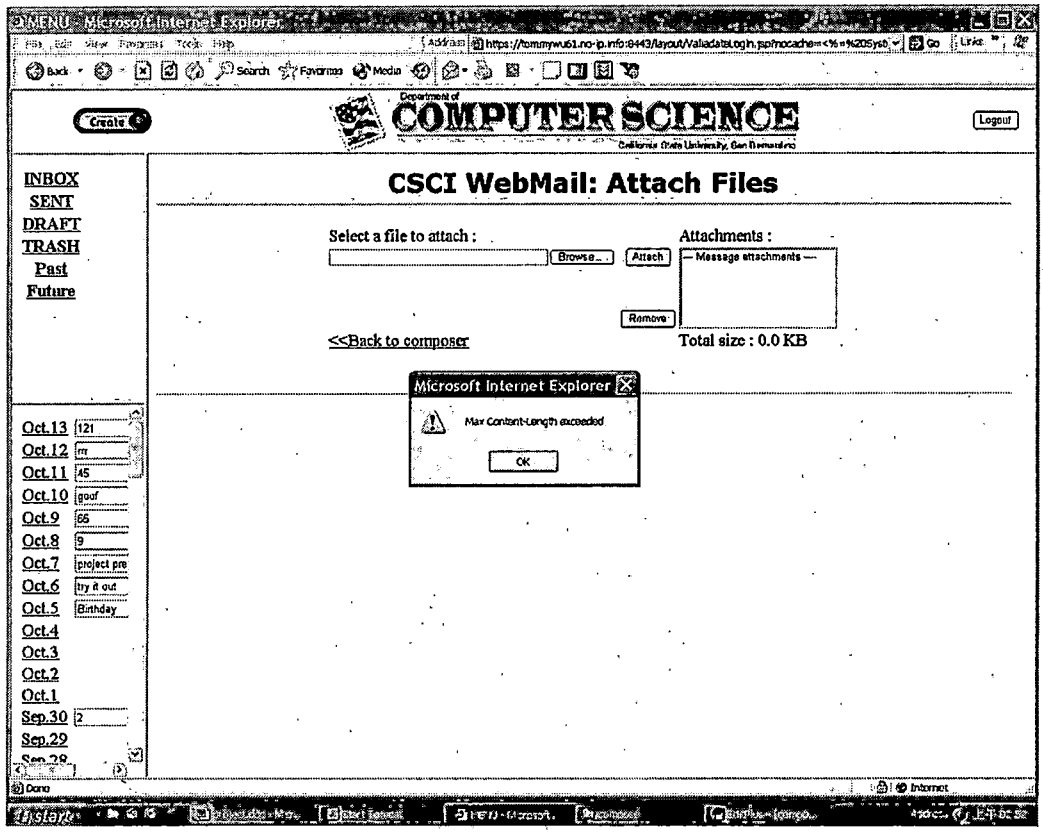

Figure 62. Error Messages for Exceeding the Maximum Size of one Attachment

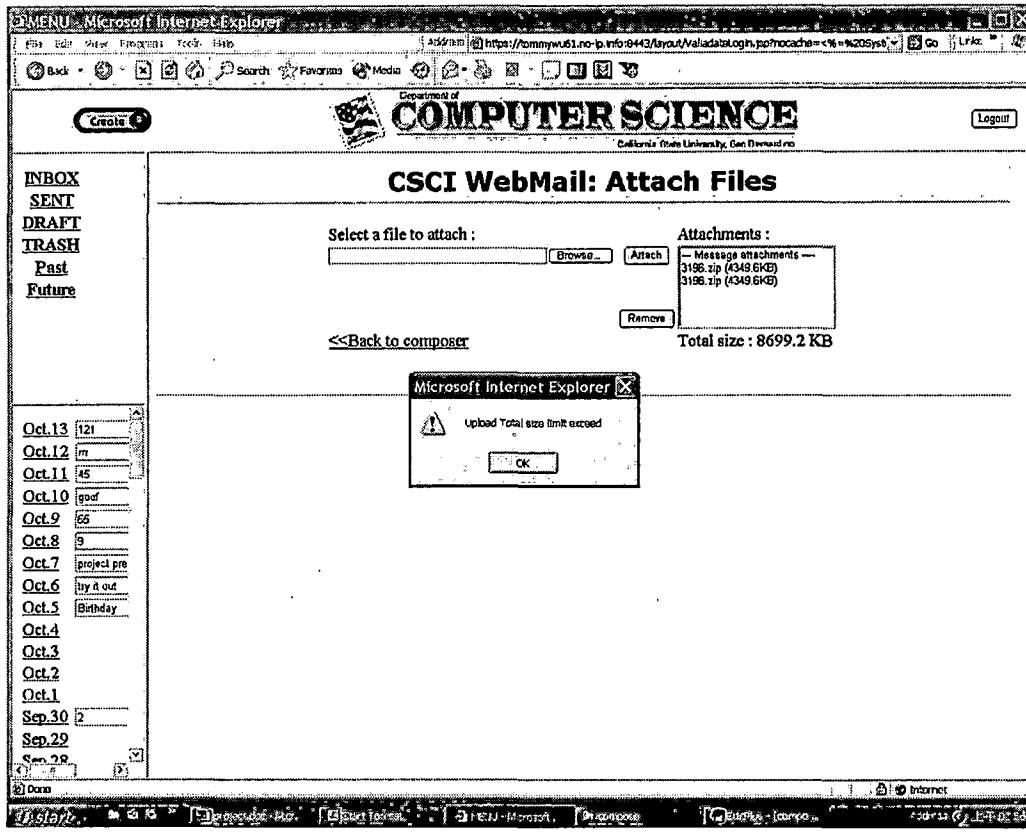

Figure 63. Error Messages for Exceeding the Total Maximum Size of the Attachments

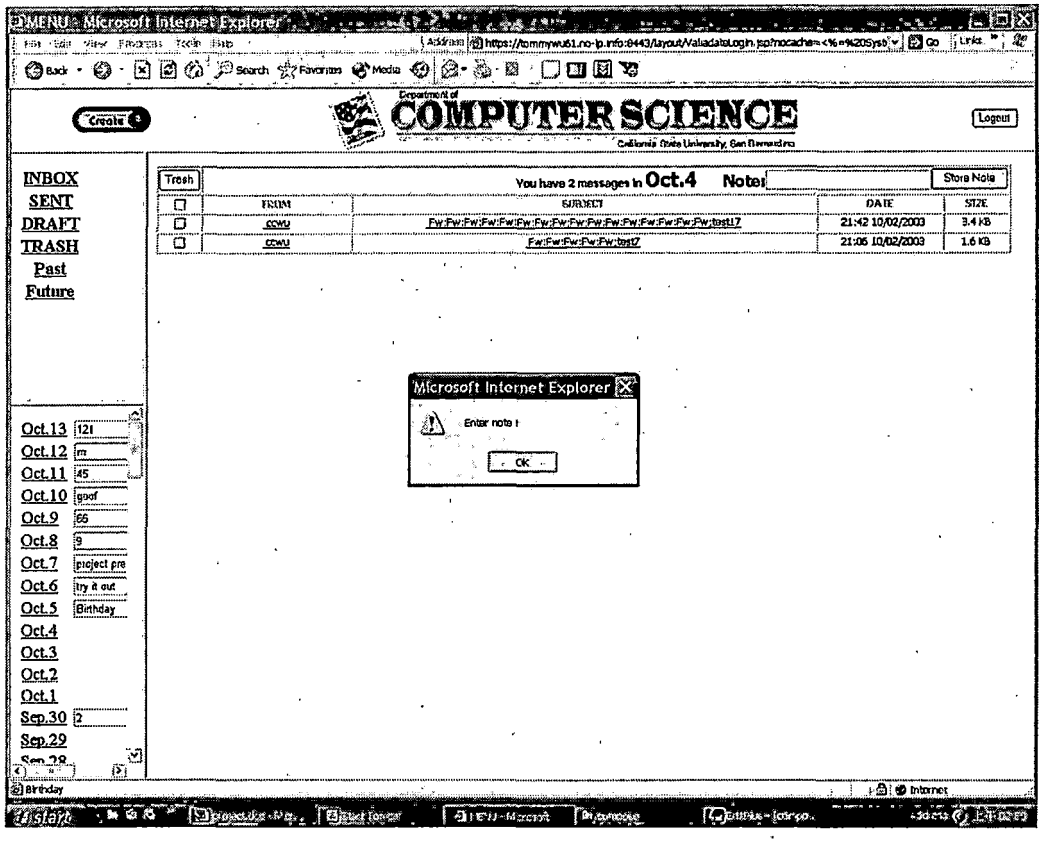

Figure 64. Error Messages for Blank Note Input

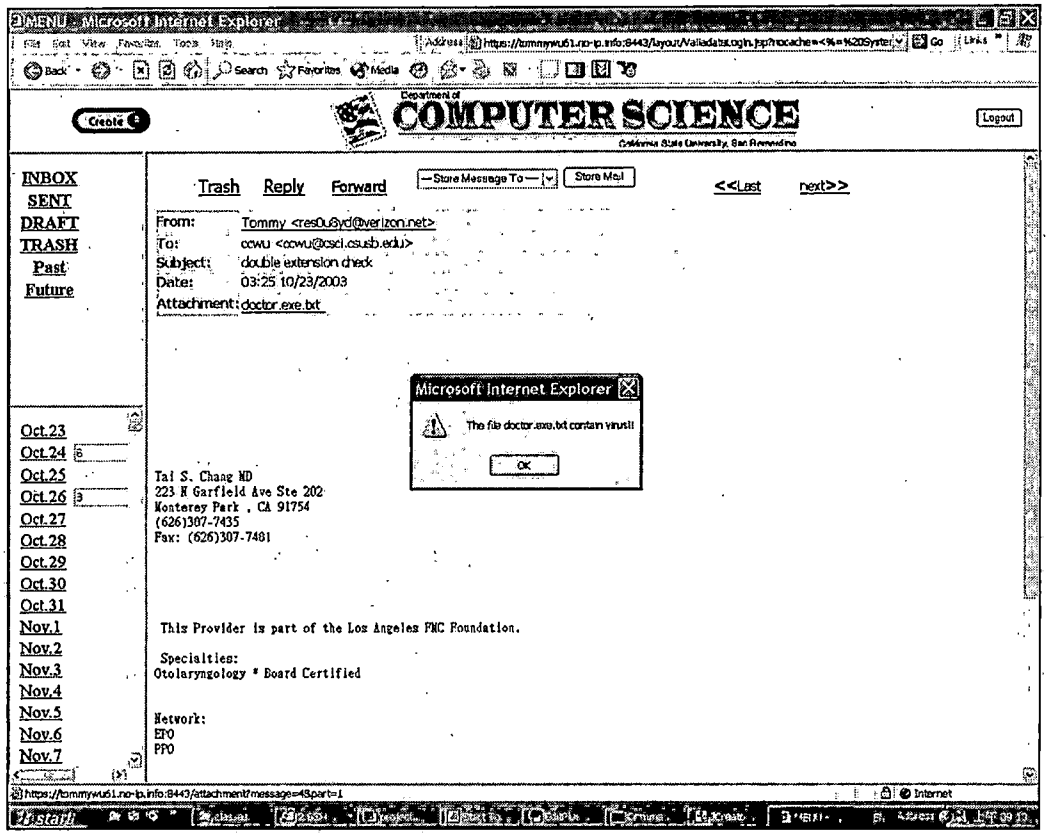

Figure 65. Virus Alert Message1

Stop users from downloading any file with a double extension with form ".???.???" where "?" can be any charcter.

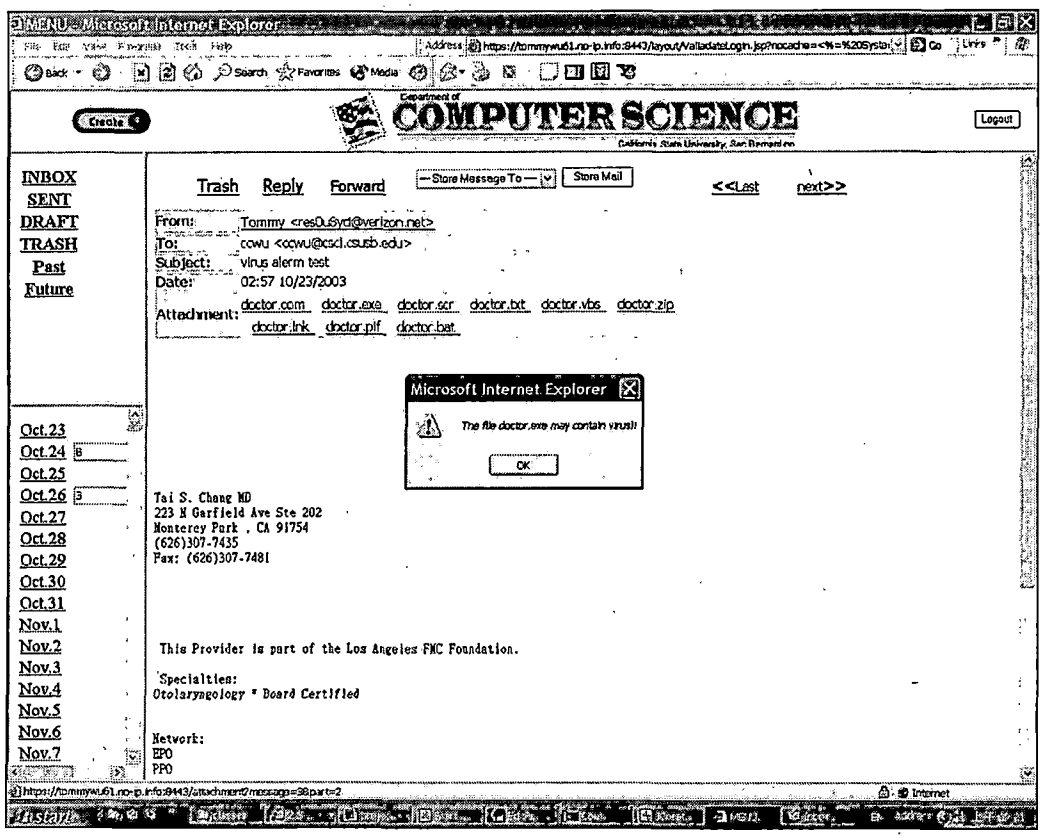

Figure 66. Virus Alert Message2

Warn people before they are about to download files with any of the following extensions: zip, pif, scr, exe, EXE, vbs, bat, lnk, com, because any of these might be a virus.

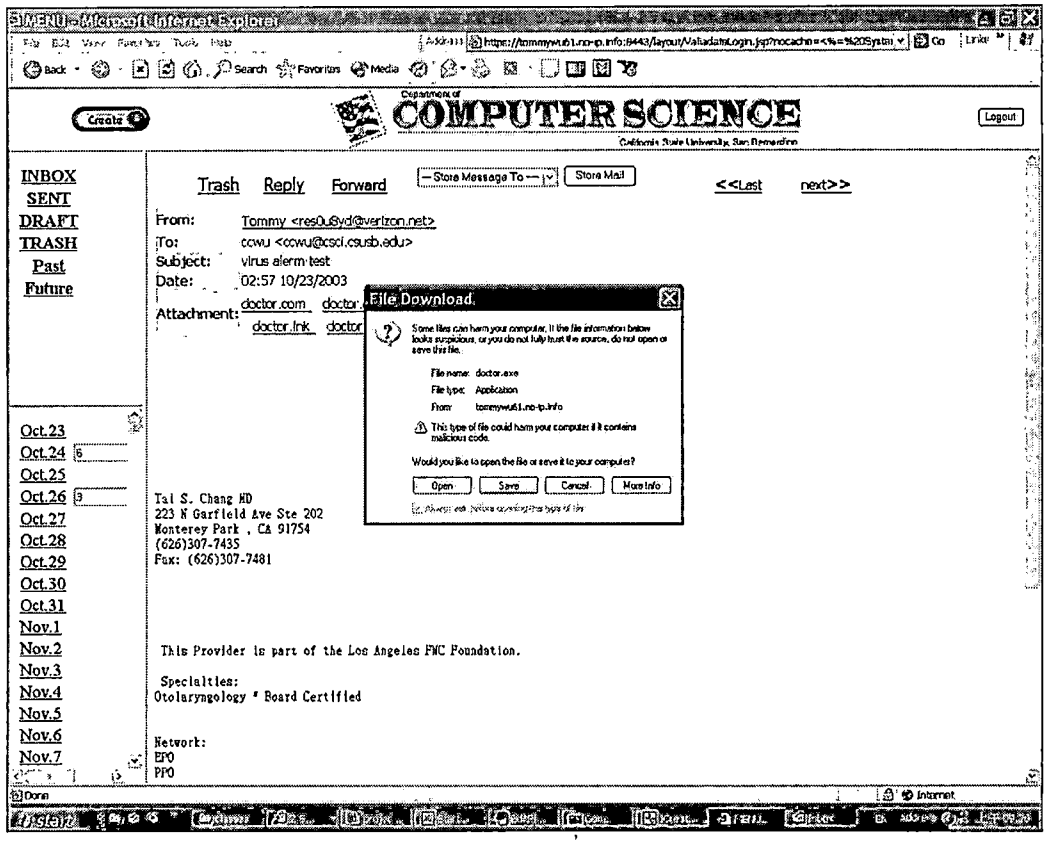

Figure 67. Some Information About A Virus Possible File

4.2.10 Logout

The user needs to logout to completely remove all data in the cookies. This procedure makes sure other users will not see your mails contents after leaving the page.

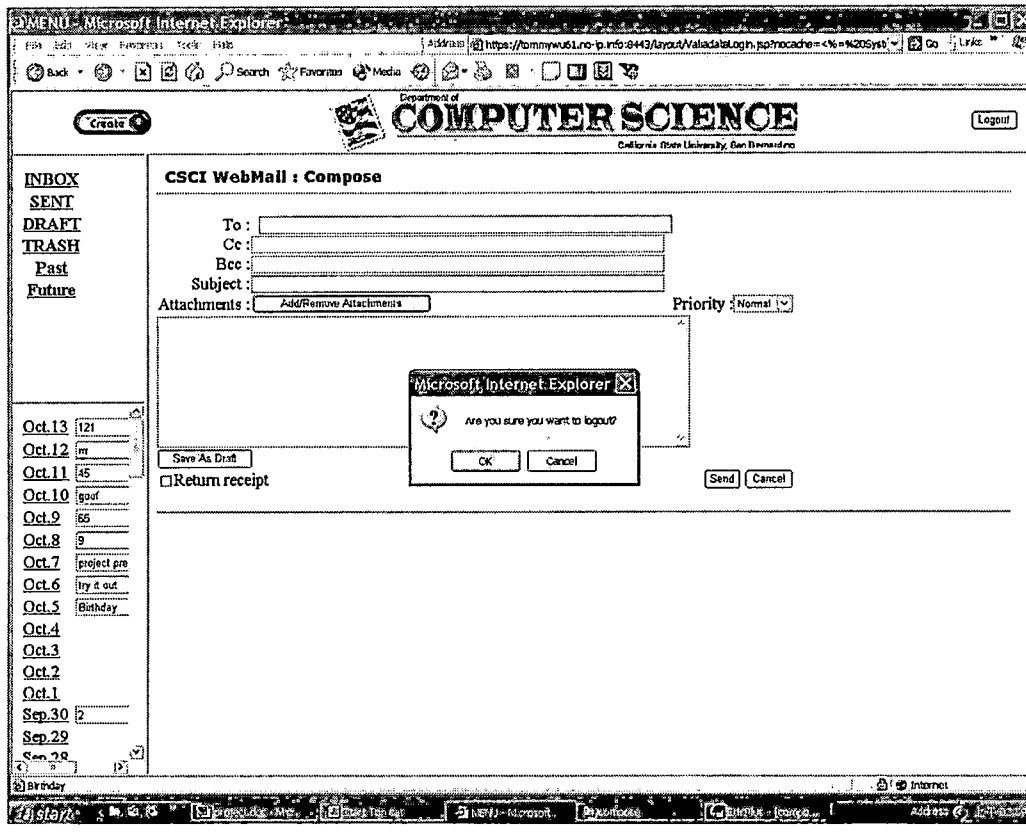

Figure 68. A Confirm Message for Logging Out

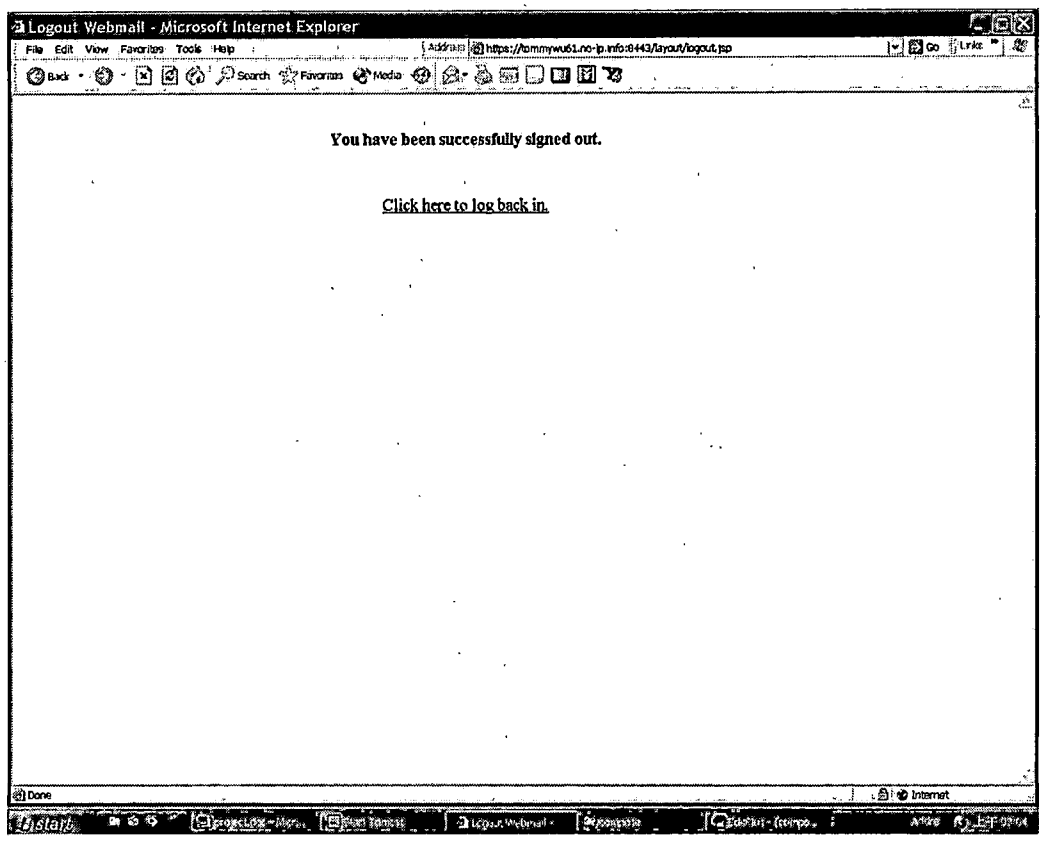

Figure 69. Logout Page

After you logging out, if you press "Back" button on your browser, you can not go back to that page again. You need re-login to enter your mail account. If you try press "Back" button, the webpage will automatically forward to the login page.

CHAPTER FIVE

SECURITY

Since this system includes all the users' private information, so data security is pretty important part. In this project, we have some security methods ensuring the information secured.

5.1 Login Page

When the user enters the WebMail page, the system will require the user to enter user name and password. Then the WebMail client will connect to CSCI mail server using this username and password. If the user name or the password is incorrect, then the system will show the error message and ask the user to re-login.

5.2 Secure Sockets Layer between User and WebMail Client

SSL, or Secure Socket Layer, is a technology which allows web browsers and web servers to communicate over a secured connection. This means that the data being sent is encrypted by one side, transmitted, then decrypted by the other side before processing. This is a two-way process, meaning that both the server AND the browser encrypt all traffic before sending out data.

Actually there are two servers in this project, one is the WebMail client, and the other one is the CSCI mail server. The WebMail client use SSL to communicate with user clients. All data is encrypted by the user client or the WebMail client, transmitted, then decrypted by the WebMail client or user client before processing.

Another important aspect of the SSL protocol is Authentication. This means that during your initial attempt to communicate with a web server over a secure connection, that server will present your web browser with a set of credentials, in the form of a "Certificate", as proof the site is who and what it claims to be. In certain cases, the server may also request a Certificate from your web browser, asking for proof that you are who you claim to be. This is known as "Client Authentication," although in practice this is used more for business-to-business (B2B) transactions than with individual users. Most SSL-enabled web servers do not request Client Authentication.

The first time a user attempts to access a secured page on your site, he or she is typically presented with a dialog containing the details of the certificate (such as the company and contact name), and asked if he or she wishes to accept the Certificate as valid and continue

with the transaction. Some browsers will provide an option for permanently accepting a given Certificate as valid, in which case the user will not be bothered with a prompt each time they visit your site. Other browsers do not provide this option. Once approved by the user, a Certificate will be considered valid for at least the entire browser session.

Also, while the SSL protocol was designed to be as efficient as securely possible, encryption/decryption is a computationally expensive process from a performance standpoint. It is not strictly necessary to run an entire web application over SSL, and indeed a developer can pick and choose which pages require a secure connection and which do not. In this project, all the application pages including payment part are secured. Those pages within an application can be requested over a secure socket by simply prefixing the address with https: instead of http:. In Tomcat server, after installing Java Secure Socket Extensions (J2EE) and set the Java Certificate Keystore, system developer just needs to remove comment tags of "SSL HTTP/1.1 Connector on port 8443" and restarts the Tomcat. The SSL pages will be use the port 8443, such as https://localhost:8443.

5.3 Secure JavaMail with Java Secure Socket Extension

About the security issue between the WebMail client and the CSCI mail server. JavaMail API providers lack direct SSL connection support, but luckily, the Java Secure Socket Extension (JSSE) API provides support for SSL network connections. The JSSE is a set of Java packages that enable secure Internet communications. It implements a Java version of SSL and includes functionality for data encryption, server authentication, message integrity, and optional client authentication. It uses secure connections to POP3, IMAP, and NNTP mail storages. This project combines JavaMail and JSSE API to make a SSL secure connection between the WebMail client and the CSCI mail server.

CHAPTER SIX

SYSTEM VALIDATION

Testing of system validation is the testing process to ensure that the program as implemented meets the expectation of the user. The purpose of system validation is to have assurance about the software quality and functionalities. This guarantees system performance and reliability also.

6.1 Unit Test

Unit test is the basic level of testing where individual components are tested to ensure that they operate correctly. These individual components can be object, class, program and etc. The unit testing results of WBECC are shown in Table 2.

Table 2. Unit Test Results

Forms	Tests Performed	Results
Login page	• Verify handling valid data input. • Check all the links and buttons work properly. • Check the error message show correctly for any error input.	Pass
Inbox folder	• Make sure the menu in the left frame works. • Check all the links work as expected. • Check all message headers are correct.	Pass

Forms	Tests Performed	Results
	• Check the number of total messages is correct. • Check the message size is correct. • Check the hyperlink of sender's address works correctly. • Check the checkboxes are work correctly. • Verify the trash function works correctly. • When pressing the Inbox hyperlink, check the newest messages will show in Inbox immediately. • When pressing the "Trash" button, make sure the selected messages are moved to the Trash immediately.	
Sent folder	• Check all the links work as expected. • Check all message headers are correct. • Check the number of total messages is correct. • Check the message size is correct. • When pressing the Sent hyperlink, check the newest messages in sent folder will show immediately. • Check the hyperlink of sender's address works correctly. • Check the checkboxes are work correctly. • Verify the trash function work correctly. • For each sent message, check it shows immediately in the sent folder. • When pressing the "Trash" button, make sure the selected messages are moved to the Trash immediately.	Pass

Forms	Tests Performed	Results
Draft folder	• Check all the links work as expected. • Check all message headers are correct. • Check the number of total messages is correct. • Check the message size is correct. • When pressing the Draft hyperlink, check the newest messages in draft folder will show immediately. • Check the hyperlink of sender's address works correctly. • Check the checkboxes are work correctly. • Verify the trash function work correctly. • After pressing the "Save as Draft" button, make sure that the message is immediately stored to the Draft folder. • When pressing the "Trash" button, make sure the selected messages are immediately moved to the Trash.	Pass
Trash folder	• Check all the links work as expected. • Check all message headers are correct. • Check the number of total messages is correct. • Check the message size is correct. • Check the checkboxes are work correctly. • Verify the trash function work correctly. • For each sent message, check it shows immediately in the sent folder. • Check the hyperlink of sender's address works	Pass

Forms	Tests Performed	Results
	correctly. • When pressing the Trash hyperlink, check the newest messages in trash folder will show immediately. • When pressing the "Delete" button, make sure the selected messages are deleted immediately.	
Past folder	• Make sure the menu in left bottom frame works and all dates and notes show correctly. • Make sure the date list show decreasingly. • Check all the links work as expected. • Check the hyperlink of sender's address works correctly. • Check all message headers are correct. • Check the number of total messages is correct. • Check the message size is correct. • Check the checkboxes are work correctly. • Verify the trash function work correctly. • For each stored message, check it shows immediately in that daily folder. • When pressing the "Trash" button, make sure the selected messages are immediately moved to the Trash. • Check each message can read correctly in daily folders • When make a note for one day, make sure this note shows immediately in the left frame.	Pass

Forms	Tests Performed	Results
Future folder	• Make sure the menu in left bottom frame works and all dates and notes show correctly. • Make sure the date list show increasingly. • Check all the links work as expected. • Check all message headers are correct. • Check the number of total messages is correct. • Check the message size is correct. • Check the hyperlink of sender's address works correctly. • Check the checkboxes are work correctly. • Verify the trash function work correctly. • For each stored message, check it shows immediately in its daily folder. • When pressing the "Trash" button, make sure the selected messages are immediately moved to the Trash. • Check each message can read correctly in daily folders • When make a note for one day, make sure this note shows immediately in the left frame.	Pass
Read messages in folders	• Make sure message with plain text and html text can be read correctly. • Make sure all picture in html text can be showed correctly. • Check "Next" and "Last" hyperlink is work correctly to next or last message. • Check when reading the last message, the "Next" hyperlink will not show up.	Pass

Forms	Tests Performed	Results
	• Check when reading the first message, the "Last" hyperlink will not show up. • Check trash function works correctly. • Check reply function works correctly. • Check the hyperlink of sender's address works correctly. • Check option "Store Message to" work correctly. • Check button "Store" work correctly. • Check attached files show correctly. • Check each attached file can be download or open correctly	
Create new messages	• Check all entries show correctly. • Make sure all input data stored in session. • Verify function e-mail address check works correctly. • When two or more e-mail addresses are input, check function store e-mail address works correctly. • Check the message priority is set correctly. • Verify all error messages show correctly. • Check function return receipt works correctly. • Check all buttons work correctly. • After attaching file, when back to composing page, all data in composing page still show correctly.	Pass

Forms	Tests Performed	Results
Attach files	• Check all entries show correctly. • Make sure all input data stored in session. • Check all buttons work correctly. • Verify all error messages show correctly. • Check function files size check works correctly. • Check function total files size check works correctly. • Check function remove files works correctly.	Pass
Logout page	• Check all session is removed. • Check the explorer can not return pages. • Check the link works properly.	Pass

6.2 Subsystem Testing

Subsystem testing is the next step up in the testing process where all related units from a subsystem to do a certain task. Thus, the subsystem test process is useful for detecting interface errors and specific functions. Table 3 show subsystem test results in detail.

Table 3. Subsystem Test Results

Subsystem	Tests Performed	Results
Create and send mails	• Test all the required input. • Check all the input in session variable. • Check the proper form appears. • Make sure all message contents and attached files were packaged together. • Ensure data security during transmission by SSL.	Pass
Read and receive mails	• Test and check all the message are correct. • Test receiving e-mail with different formatted and attached files.	Pass
Create folders	• Check all daily folders were created correct. • Check	
Database Maintenance Subsystem	• Test insert / edit / edit / delete function of each table. • Check all the information in database.	Pass

6.3 System Testing

System testing is the testing process that uses real data, which the system is intended to manipulate, to test the system. First all subsystem will be integrated into one system. Then test the system by using a variety of data to see the overall result.

System testing of WBECC system begins with the following steps (Table 4):

Table 4. System Test Results

System Testing	Results
1. Install WBECC system into server.	Pass
2. Start up all services such as JSP engine, MySQL database engine.	Pass
3. Running testing by using real data on all forms and reports.	Pass

CHAPTER SEVEN

MAINTENANCE MANUAL

System maintenance is an important step to ensure that the system runs smoothly and meets the CSCI users' expectation. In the project, there are 3 major issues: Software Installation, Variables Modification, and Web-Based E-mail Client Installation /Migration.

7.1 Software Installation

In WBECC, it requires JSDK, JavaMail API, JavaBeans Activation Framework extension (JAF), com.oreilly.servlet package, TOMCAT, Secure Socket Layer (SSL), MySQL, and JDBC to run the programs. Following will detail the installation of those softwares.

7.1.1 JAVA 2 Platform, Standard Edition (J2SE)

J2SE is the compiler program for JSP programs and it's required in TOMCAT JAVA Container. Fist of all, we go to http://java.sun.com/j2se/1.4.1/download.html to download SDK Windows (all languages, including English), then install it.

7.1.2 JavaMail API

The JavaMail 1.3.1 API provides a set of abstract classes that model a mail system. The API provides a platform independent and protocol independent framework to

build Java technology-based mail and messaging applications. We need go to http://java.sun.com/products/javamail/javamail-1_3.html to download JavaMail API Implementation Version 1.3, and copy the mail.jar file into the %CATALINA_HOME%\webapps\ROOT\WEB-INF\lib\ directory.

7.1.3 JavaBeans Activation Framework (JAF) Extension

The activation.jar file in JAF 1.0.2 allows you to handle MIME (Multipurpose Internet Mail Extensions) types accessible via binary data streams. We need download JAF 1.0.2 from http://java.sun.com/products/javabeans/glasgow/jaf.html, then put activation.jar file in %CATALINA_HOME%\webapps\ROOT\WEB-INF\lib\ directory

7.1.4 The com.Oreilly.Servlet Package

The package is used to help servlets parse parameters and handle file upload in the project. We need download it from http://www.servlets.com/cos/cos-05Nov2002.zip, then copy the cos.jar file into %CATALINA_HOME%\webapps\ROOT\WEB-INF\lib\ directory

7.1.5 Tomcat

TOMCAT is one of Jakarta apache project which is a JAVA container to process JSP programs and construct a web

server for web pages. First of all, we go to http://apache.mirrorcentral.com/dist/jakarta/tomcat-4/binaries/ to download the file tomcat-4.1.18.zip and extract it to hard driver. Also, we copy C:\tomcat\bin\startup.bat and shutdown.bat to the desktop as shortcut in order to easily start and shut sown tomcat.

7.1.6 Install Secure Socket Layer Support on Tomcat

Download JSSE 1.0.2 (or later) from http://java.sun.com/products/jsse/ and copy all three JAR files (jcert.jar, jnet.jar, and jnet.jat) into the $JAVA_HOME/jre/lib/ext directory.

7.1.7 MySQL Installation

MySQL is the database system we use in the e-mail client to store daily notes. Because it also provides JDBC to easily connect by JAVA program, thus it's a good choice for designing this project. First of all, we need to download MySQL 3.23 for windows 95/98/2000/XP at http://www.mysql.com/downloads/mysql-3.23.html. After downloading the compress file, please unzip the file and install it. Second, in DOS command, we type

```
C:\mysql\bin>mysqld-nt -install
```

Then we install the MySQL service in our server. Third, we type

C:\mysql\bin>net start mysql

to start our service. Forth, we have to setup the user and password, the default user is named 'ROOT', so we have to set its password to 9841571.

C:\mysql\bin>mysqladmin -u root password 9841571

After that, we can try to input following command:

C:\mysql\bin>mysqld

mysql>select * from user

mysql>exit

C:\mysql\bin>

Then, we have already installed MySQL and it's working as expected.

7.1.8 JAVA Database Connectivity (JDBC)

The API used to execute SQL statement is different for each database engine. Java programmers, however, are lucky and are freed from such database portability issues. They have a single API, the Java Database Connectivity API (JDBC), that's portable between database engines. The JDBC library provides an interface for executing SQL statements. It provides the basic functionality for data access. A number of drivers are available for MySQL, and information about this can be obtained at the MySQL homepage at http://www.mysql.com/downloads, under JDBC. For our purpose, we will use the MM.MySQL driver which is

a Type-4 JDBC driver that is under the GNU Library License.

7.2 Variables Modification

In the e-mail client, we have to change some environment variables in windows system, server.xml in Tomcat server for SSL function, and setting for startup.bat, shutdown.bat, and JDBC.

7.2.1 System Variables

1. Go into the Control Panel and open up the System control panel application.
2. You should have a window entitled System Properties on your screen; select the Advanced tab, and click on the Environment Variable button.
3. A new window named Environment Variables should have opened. Click on the new button in the System Variables Section.
4. The New System Variable window should now be on your screen. Enter JAVA_HOME for as the name and the path to your JDK (such as c:\j2sdk1.4.1_01) for the value.

5. Repeat step 3, and enter CATALINA_HOME for as the name and the path to your TOMCAT (such as c:\tomcat) for the value.

6. Repeat step 3, and enter CLASSPATH for as the name and c:\tomcat\common\lib\servlet.jar; c:\mm.mysql.jdbc-2.0pre5\mysql_2_comp.jar; c:\mm.mysql.jdbc-2.0pre5\mysql_2_uncomp.jar; for the value. c:\tommat is the TOMCAT path and c:\mm.mysql.jdbc-2.0pre5 is the path for JDBC.

7.2.2 Batch Files Modification

There are two files should be modified. Add following two lines into c:\tomcat\bin\startup.bat and c:\tomcat\bin\shutdown.bat.

```
set JAVA_HOME = c:\j2sdk1.4.1_01
set CATALINA_HOME = c:\tomcat
```

7.2.3 Copying Files

Copy following two files,

c:\mm.mysql.jdbc-2.0pre5\mysql_2_comp.jar

c:\mm.mysql.jdbc-2.0pre5\mysql_2_uncomp.jar

to c:\tomcat\common\lib.

And

Copy c:\tomcats\server\webapps\admin\WEB-INF\lib\ struts.jar

to c:\tomcat\webapps\ROOT\WEB-INF\lib.

7.2.4 Secure Sockets Layer Configuration

1. Create a certificate keystore by executing the following command:

   %JAVA_HOME%\bin\keytool -genkey -alias tomcat -keyalg RSA and specify a password value of "changeit".

2. Uncomment the "SSL HTTP/1.1 Connector" entry in $CATALINA_HOME/conf/server.xml and tweak.

7.3 Web-based E-mail Client for Computer Science Installation/Migration

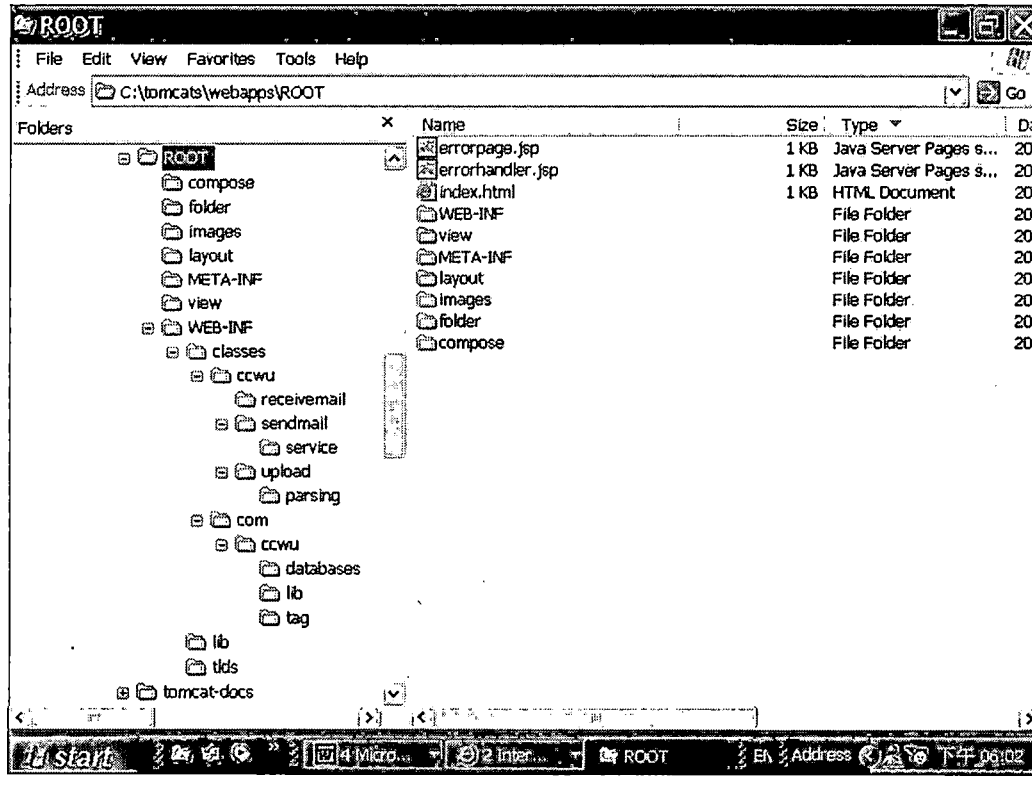

Figure 70. The Project File Directory

1. All the JSP programs and HTML programs are stored in five different folders

   %CATALINA_HOME%\webapps\ROOT\

   for index.html, errorpage.jsp, errorhandle.jsp.

   %CATALINA_HOME%\webapps\ROOT\layout\

   for index_down.html, index_up.html, down.jsp, errordetails.jsp, logout.jsp, menu.jsp, top.jsp, ValiadateLogin.jsp.

   %CATALINA_HOME%\webapps\ROOT\view\

   for right.jsp, index.jsp, Delete.jsp, read.jsp, ToTrash.jsp.

   %CATALINA_HOME%\webapps\ROOT\folder\

   for left.jsp, openFolder.jsp, Option.jsp, past.jsp, Schedule.jsp, StoreNote.jsp.

   %CATALINA_HOME%\webapps\ROOT\compose\

   for attach.jsp, compose.jsp, errorhandler.jsp, handleDraft.jsp, index.jsp, mailsemt.jsp.

2. All the images are stored in

   %CATALINA_HOME%\webapps\ROOT\images

3. All the classes are stored in

   %CATALINA_HOME%\webapps\ROOT\WEB-INF\classes\

   ccwu\receievemail\

   for AttachmentServlet.java.

   ccwu\sendmail\

for ComposeBean.java, SendMailServlet.java,

UploadServlet.java

ccwu\sendmail\service\

for ByteArrayDataSource.java, ComposeModel.java,

SendMail.java, SendMailAuthenticator.java

ccwu\upload\

for MultipartFormDataRequest.java,

UploadBean.java, UploadException.java,

UploadFile.java.

ccwu\upload\parsing\

for StrutsMultipartIterator.java,

StrutsMultipartParser.java,

StrutsUploadFile.java.

com\ccwu\databases\

for csmail.java.

ccwu\lib\

for MailUserBean.java, MonthMap.java.

ccwu\tag\

for attachmentInfo.java, deletemail.java,

ListAttachmentsTag.java,

ListAttachmentsTEI.java, ListMessagesTag.java,

ListMessagesTEI.java, MessageInfo.java,

MessageTag.java, MessageTEI.java,

MoveToTrash.java, SendTag.java, StoreTo.java.

4. All the database files are stored in %MYSQL%\data\csmail\
5. File web.xml and taglib.tld are stored in %CATALINA_HOME%\webapps\ROOT\WEB-INF\

7.4 Backup

Backup is a very important action for an administrator. We can't ensure the system will work and not stuck forever, so backup can recover the system to original status. In the project, we need backup two components, system files and database.

7.4.1 System Backup

All the programs and required graphic are stored under a directory %CATALINA_HOME%\webapps\ROOT\, including all sub-directories. Thus, the administrator just needs to copy all the files under this directory or use compressing software, such as WINZIP or WINRAR, to backup the system programs.

7.4.2 Database Backup

All the database files are stored under %MYSQL%\data\csmail\ directory including all *.frm, *.MYD, and *.MYI.

CHAPTER EIGHT

CONCLUSION AND FUTURE DIRECTIONS

8.1 Conclusion

Ray Tomlinson developed the first application for the ARPANET in 1971, consisting of a program called SNDMSG for sending mail, and a program called READMAIL for reading mail. These early e-mail programs had simple functionality and were command line driven, but established the basic model still in use today. This project develops a web-based e-mail interface for CSCI faculty, staff and students. Actually, any mail server which supports SMTP and IMAP protocol can use the project to build a web-based interface to handle e-mails. For any organizations and companies which need handle e-mails on line, the project provides a feasible way to develop an exclusive and self-feature web-based e-mail client. Moreover, the idea and design of daily messages management provide a new way to combine your e-mails and schedules. With the support of Multipurpose Internet Mail Extensions (MIME), you may store texts, files, sounds, images or even video pictures in your daily schedule lists. Finally, the WebMail interface will become not only an e-mail system, but also a daily messages management system.

8.2 Future Directions

To send and reveieve e-mails are the basic functionality of an e-mail client. Since it is an e-mail client, its functionality is limited by the e-mail server. If the e-mail server just supports pop3, not IMAP protocol to reveive mails, we can not retrieve messages in folder other than INBOX. There are no SENT, DRAFT, TRASH folders to use. Of course, we can not create other new folders to manage our messages too. Moreover, the reason applets could not be used in the project is because applets cannot communicate directly to the mail server since the mail server is not the same machine the applet loaded from. If we build a new mail server and load applets to users directly from the mail server. We don't need a mail client to help us communicate with users and the mail server. Then we may get a real efficient way to handle Web Mail. But in that case, we may not need JavaMail API any more because we may access mails directly by JDBC from the mail server database. Therefore, there still other issues need to be solved when using applets. In general, applets loaded over the net are prevented from reading and writing files on the client file system and applets loaded over the net are prevented from starting other programs on the client. Applets loaded over the net are also not allowed

APPENDIX

SOURCE CODE OF JAVA CLASSES

List of Classes

AttachmentServlet.java 102
AttachmentInfo.java 103
ByteArrayDataSource.java 104
ComposeBean.java 105
ComposeModel.java 106
ListAttachmentsTEI.java 110
ListAttachmentsTag.java 110
ListMessagesTEI.java 111
ListMessagesTag.java 112
MonthMap.java 113
MoveToTrash.java 114
MultipartFormDataRequest.java 115
SendMail.java 117
SendMailAuthenticator.java 120
SendMalServlet.java 121
StoreTo.java 124
StrutsMultipartIterator.java 125
StrutsMultipartParser.java 126
StrutsUploadFile.java 127
StrutsUploadBean.java 128
UploadException.java 130
Uploadfile.java 130
UploadServlet.java 131
csmail.java 133
deletemail.java 135

```
/**********************************************************
NAME     :AttachmentServlet.java
DESCRIPTION: Handle list attachments request.
REMARK  : User can click the attachment hyperlink to open or lownload the attachment.
**********************************************************/
package ccwu.receivemail;
import java.io.*;
import javax.mail.*;
import javax.mail.internet.ContentType;
import javax.servlet.*;
import javax.servlet.http.*;
import com.ccwu.lib.MailUserBean;
public class AttachmentServlet extends HttpServlet {
public AttachmentServlet() {}
public void doGet(HttpServletRequest httpservletrequest, HttpServletResponse httpservletresponse) throws
IOException, ServletException
{
HttpSession httpsession = httpservletrequest.getSession();
ServletOutputStream servletoutputstream = httpservletresponse.getOutputStream();
//For attachments download
int i = Integer.parseInt(httpservletrequest.getParameter("message"));
//Specify which message
int j = Integer.parseInt(httpservletrequest.getParameter("part"));
//Specify which attachment of the messge
MailUserBean mailuserbean = (MailUserBean)httpsession.getAttribute("mailuser");
if(true)
  try
  {
     Folder folder = mailuserbean.getFolder();
     if(!folder.isOpen())//Make sure folder is open
     try
   {
          folder.open(Folder.READ_WRITE);
     } catch (MessagingException ex)
     {
          folder.open(Folder.READ_ONLY);                    }
   Message message = folder.getMessage(i);
   Multipart multipart = (Multipart)message.getContent();
   javax.mail.BodyPart bodypart = multipart.getBodyPart(j);//get attachment
   String s = bodypart.getContentType();
   String filename = bodypart.getFileName();
   if(s == null)
   {
     servletoutputstream.println("invalid part");
      return;
   }
   ContentType contenttype = new ContentType(s);
     httpservletresponse.setContentType(
     contenttype.getBaseType());                              httpservletresponse.setHeader(
     "Content-Disposition","attachment; filename=" +     filename +";");
     //Set the file information in the pop up windoow
   InputStream inputstream = bodypart.getInputStream();
     //read the attachment
   int k;
   while((k = inputstream.read()) != -1)
     servletoutputstream.write(k);//Download the file
   servletoutputstream.flush();
   servletoutputstream.close();
   }catch(MessagingException messagingexception)
     {
     super.log("Isn't Login");
      throw new ServletException(
          messagingexception.getMessage());
   }
   else//forward page
          getServletConfig().getServletContext().getRequestDispatcher("/index.html").
          forward(httpservletrequest, httpservletresponse);
```

```
  }
}
/*************************************************************
NAME     :AttachmentInfo.java
DESCRIPTION: Store attachments information
REMARK  : Use to support list attachment tag
*************************************************************/
package com.ccwu.tag;
import java.io.IOException;
import javax.mail.MessagingException;
import javax.mail.Part;
import javax.mail.internet.ParseException;
public class AttachmentInfo
{
private Part part;
private int num;
public AttachmentInfo() {}
public String getAttachmentType() throws
MessagingException
{
      String s;
      if((s = part.getContentType()) == null)
            return "invalid part";
   else
      return s;
}

public String getContent() throws IOException, MessagingException
{
   if(hasMimeType("text/plain"))
      return (String)part.getContent();
   else//content isn't plain text
      return "";
}
public String getDescription() throws MessagingException
{
   String s;
   if((s = part.getDescription()) != null)
      return s;
   else
      return "";
}
public String getFilename() throws MessagingException
{
   String s;
      if((s = part.getFileName()) != null)
            return s;
      else
            return "";
}
public String getNum()
{
   return Integer.toString(num);
}
public boolean hasDescription() throws MessagingException {
   return part.getDescription() != null;
}
public boolean hasFilename() throws MessagingException
{
   return part.getFileName() != null;
}
public boolean hasMimeType(String s) throws MessagingException
{
   return part.isMimeType(s);
}
public boolean isInline() throws MessagingException
{//attachment can be viewed inline
```

```
    if(part.getDisposition() != null)
      return part.getDisposition().equals("inline");
    else
      return true;
}
public void setPart(int i, Part part1) throws MessagingException, ParseException
{
    part = part1;
    num = i;
}
}
/*************************************************************
NAME     :ByteArrayDataSource.java
DESCRIPTION: Store upload file
*************************************************************/
package ccwu.sendmail.service;
import java.io.*;
import javax.activation.DataSource;
public class ByteArrayDataSource implements DataSource
{
public static String DEFAULTENCODING = "iso-8859-1";
public static int INTERNALBUFFERSIZE = 4096;
private byte data[];
private String ContentType;
public ByteArrayDataSource(InputStream inputstream, String contentType)
{
    //Read InputStream then store to data[]
    if(inputstream != null)
    {
      ContentType = contentType;
      try
      {
       ByteArrayOutputStream bytearrayoutputstream = new ByteArrayOutputStream();
       byte abyte0[] = new  byte[INTERNALBUFFERSIZE];
       int i;
       while((i = inputstream.read(abyte0)) != -1)
       bytearrayoutputstream.write(abyte0, 0,i);
       data = bytearrayoutputstream.toByteArray();
      }
       catch(IOException ioexception) { }
      }
   }
public ByteArrayDataSource(byte abyte0[], String contentType)
{
    //Directly store input to data[]
    data = abyte0;
    ContentType = contentType;
}
public ByteArrayDataSource(String s, String contentType)
{
    //If input is string, transfer and store to data[]
    ContentType = contentType;
    try
    {
      data = s.getBytes(DEFAULTENCODING);
    }
    catch(UnsupportedEncodingException unsupportedencodingexception) { }
}
public ByteArrayDataSource(String s, String contentType, String codingType)
{
//Transfer string by specified coding tpye to store to data[]
    ContentType = contentType;
    try
    {
      data = s.getBytes(codingType);
    }
    catch(UnsupportedEncodingException unsupportedencodingexception) { }
```

```
}
public InputStream getInputStream() throws IOException
{
      if(data == null)
        throw new IOException("no data available");
    else
        return new ByteArrayInputStream(data);
}
public OutputStream getOutputStream() throws IOException
{
    throw new IOException("N/A");
}
public String getContentType()
{
    return ContentType;
}
public String getName()
{
    return getClass().getName();
}
}
/*************************************************************
NAME     :ComposeBean.java
DESCRIPTION: Store the information of the composing  message and attachments
*************************************************************/
package ccwu.sendmail;
import ccwu.sendmail.service.ComposeModel;
public class ComposeBean extends ComposeModel {
public static String ATTACHSEPARATOR = ", ";
public static String NOERROR = "noerror";
public static String FROMEMAIL = "fromemail";
public static String FROMNAME = "fromname";
public static String REPLYTO = "replyto";
public static String TO = "to";
public static String CC = "cc";
public static String BCC = "bcc";
public static String SUBJECT = "subject";
public static String MESSAGE = "message";
public static String PRIORITY = "priority";
public static String RETURNRECEIPT = "returnreceipt";
private String Attachments;
private int AttachmentSizeLimit;
public ComposeBean()
{
    Attachments = null;
    AttachmentSizeLimit = -1;
    setReplyto("");
    setTo("");
    setCc("");
    setBcc("");
    setSubject("");
    setMessage("");
    setXpriority(3);
    Attachments = "";
}
public String getAttachments()
{
    return Attachments;
}
public void setAttachments(String s)
{
    Attachments = s;
}
public int getAttachmentsizelimit()
{
    return AttachmentSizeLimit;
}
```

```
public void setAttachmentsizelimit(int i)
{
    AttachmentSizeLimit = i;
}
}
/*************************************************************
NAME     :ComposeModel.java
DESCRIPTION: Store the basic information of the composing  message
*************************************************************/
package ccwu.sendmail.service;
import java.io.Serializable;
import java.util.Date;
import java.util.Properties;
import javax.mail.*;
public class ComposeModel implements Serializable
{
public static String SEPARATOR = ";";
public static final String DEFAULTFROMEMAIL = "ccwu@csci.csusb.edu";
public static final String DEFAULTSUBJECT = "[DefaultSubject]";
public static final String DEFAULTMESSAGE = "Message sent by ccwu.";
public static final String DEFAULTXMAILER = "WebMailComposer";
public static final boolean DEFAULTRETURNRECEIPT = false;
public static final int HIGHEST = 1;
public static final int HIGH = 2;
public static final int NORMAL = 3;
public static final int LOW = 4;
public static final int LOWEST = 5;
public static final int DEFAULTPRIORITY = 3;
private Properties prop;
private Date date;
private String Xmailer;
private int Xpriority;
private String smtpServer;
private String username;
private String password;
private String frommail;
private String fromname;
private String replyto;
private String replyname;
private String to;
private String toname;
private String cc;
private String ccName;
private String bcc;
private String bccname;
private String subject;
private String message;
private String charset;
private boolean returnReceipt;
private String AttachFromForward;
private Message msg;
private Store store;
private boolean draft;
public ComposeModel()
{
    store = null;
    prop = null;
    date = null;
    Xmailer = null;
    Xpriority = 3;
    smtpServer = null;
    username = null;
    password = null;
    frommail = null;
    fromname = null;
    replyto = null;
    replyname = null;
```

```
    to = null;
    toname = null;
    cc = null;
    ccName = null;
    bcc = null;
    bccname = null;
    subject = null;
    message = null;
    charset = null;
    returnReceipt = false;
    frommail = "ccwu@csci.csusb.edu";
    subject = "Welcome to WebMail";
    message = "Message sent by WebMail.";
    Xmailer = "CSCI";
    Xpriority = 3;
    returnReceipt = false;
    AttachFromForward = "";
}
public void setIsDraft(boolean b)
{  //save the message as draft
    draft = b;
}
public boolean getIsDraft()
{  //If the message is draft, return true
    return draft;
}
public Store getStore()
{
    return store;
}
public void setStore(Store store1)
{
    store = store1;
}
public Properties getProperties()
{
    if(prop == null)
      return System.getProperties();
    else
      return prop;
}
public void setProperties(Properties properties)
{
    prop = properties;
}
public Date getDate()
{
    if(date == null)//set date to today
      return new Date();
    else
      return date;
}
public void setDate(Date date)
{
    date = date;
}
public String getXmailer()
{
    return Xmailer;
}
public void setXmailer(String s)
{
    Xmailer = s;
}
public int getXpriority()
{
    return Xpriority;
```

```
}
public void setXpriority(int i)
{
    Xpriority = i;
}
public boolean getReturnReceipt()
{
    return returnReceipt;
}
public void setReturnReceipt(boolean flag)
{
    returnReceipt = flag;
}
public String getSmtpserver()
{
    return smtpServer;
}
public void setSmtpserver(String s)
{
    smtpServer = s;
}
public String getUsername()
{
    return username;
}
public void setUsername(String s)
{
    username = s;
}
public String getPassword()
{
    return password;
}
public void setPassword(String s)
{
    password = s;
}
public String getFromemail()
{
    return frommail;
}
public void setFromemail()
{
    frommail = username+"@"+smtpServer;
}
public String getFromname()
{
    return fromname;
}
public void setFromname()
{
    fromname = username;
}
public String getReplyto()
{
    return replyto;
}
public void setReplyto(String s)
{
    replyto = s;
}
public String getReplytoname()
{
    return replyname;
}
public void setReplytoname(String s)
{
```

```
    replyname = s;
}
public String getTo()
{   //get to address
    return to;
}
public void setTo(String s)
{   //set to address
    to = s;
}
public String getToname()
{
    return toname;
}
public void setToname(String s)
{
    toname = s;
}
public String getCc()
{   //the address of Cc
    return cc;
}
public void setCc(String s)
{   //the address of Cc
    cc = s;
}
public String getCcname()
{
    return ccName;
}
public void setCcname(String s)
{
    ccName = s;
}
public String getBcc()
{   //the address of Bcc
    return bcc;
}
public void setBcc(String s)
{   //the address of Bcc
    bcc = s;
}
public String getBccname()
{
    return bccname;
}
public void setBccname(String s)
{
    bccname = s;
}
public String getSubject()
{
    return subject;
}
public void setSubject(String s)
{
    subject = s;
}
public String getMessage()
{   //the message plain text content
    return message;
}
public void setMessage(String s)
{   //the message plain text content
    message = s;
}
public String getCharset()
```

```
{
    return charset;
}
public void setCharset(String s)
{
    charset = s;
}
public String getAttachFromForward()
{
    return AttachFromForward;
}
public void setAttachFromForward(String s)
{
    AttachFromForward = s;
}
public boolean isForward()
{
    return AttachFromForward != "";
}
public void setMsgFromForward(Message m)
{
    msg = m;
}
public Message getMsgFromForward()
{
    return msg;
}
}
/*************************************************************
NAME     :ListAttachmentsTEI.java
DESCRIPTION: Store the extra information of the ListAttachments tag
REMARK:     map tag ID to AttachmentInfo.java
*************************************************************/
package com.ccwu.tag;
import javax.servlet.jsp.tagext.*;
public class ListAttachmentsTEI extends TagExtraInfo {
public ListAttachmentsTEI() {}
public VariableInfo[] getVariableInfo(TagData tagdata)
{
    VariableInfo variableinfo = new VariableInfo(tagdata.getId(), "AttachmentInfo",
                                                 true, 0);
    VariableInfo avariableinfo[] = {variableinfo};
    return avariableinfo;
}
}
/*************************************************************
NAME     :ListAttachmentsTag.java
DESCRIPTION: list all attachments in a message
*************************************************************/
package com.ccwu.tag;
import java.io.IOException;
import javax.mail.Message;
import javax.mail.Multipart;
import javax.servlet.jsp.*;
import javax.servlet.jsp.tagext.*;
import javax.mail.BodyPart;
public class ListAttachmentsTag extends BodyTagSupport {
private String messageinfo;
private int partNum;
private int numParts;
private AttachmentInfo attachmentinfo;
private MessageInfo messageInfo;
private Multipart multipart;
public ListAttachmentsTag() {
  partNum = 1;
  numParts = 0;
}
```

```
public int doAfterBody() throws JspException
{  // A JspWriter subclass that can be used to process body evaluations so they can // re-extracted later on
   BodyContent bodycontent = getBodyContent();
   try
   {
      bodycontent.writeOut(getPreviousOut());
   }catch(IOException ioexception)
   {
      throw new JspTagException("IterationTag: " + ioexception.getMessage());
   }
   bodycontent.clearBody();
   partNum++;//Get the next attachment
   if(partNum < numParts)//Still have attachments
   {
      getPart();
      return super.EVAL_BODY_AGAIN;
   } else //The end of attachments
   {
      reset();
      return super.EVAL_PAGE;
   }
}
public int doStartTag() throws JspException {
   messageInfo = (MessageInfo)super.pageContext.getAttribute(getMessageinfo());
   attachmentinfo = new AttachmentInfo();
   try
   {
      multipart = (Multipart)messageInfo.getMessage().getContent();
      numParts = multipart.getCount();
   }catch(Exception exception)
   {
      throw new JspException(exception.getMessage());
   }
   getPart();
   return super.EVAL_BODY_AGAIN;
}
public String getMessageinfo()
{
   return messageinfo;
}
private void getPart() throws JspException
{
   try
   {
      BodyPart p=multipart.getBodyPart(partNum);
      attachmentinfo.setPart(partNum,multipart.getBodyPart(partNum));
      super.pageContext.setAttribute(getId(), attachmentinfo);
   }catch(Exception exception) {
      throw new JspException(exception.getMessage());
   }
}
public void setMessageinfo(String s)
{
   messageinfo = s;
}
private void reset()
{
   partNum = 1;
   numParts = 0;
}
}
/*********************************************************
NAME     :ListMessagesTEI.java
DESCRIPTION: Store extra information of the listMessage tag
REMARK    : Map tag ID to MessageInfo.java
*********************************************************/
package com.ccwu.tag;
```

```
import javax.servlet.jsp.tagext.*;
public class ListMessagesTEI extends TagExtraInfo {
public ListMessagesTEI() {}
public VariableInfo[] getVariableInfo(TagData tagdata)
{
   VariableInfo variableinfo = new VariableInfo(tagdata.getId(),"MessageInfo", true,0);
   VariableInfo avariableinfo[] = {variableinfo};
   return avariableinfo;
}
}
/*************************************************************
NAME     :ListMessagesTag.java
DESCRIPTION: list all messages in a folder
*************************************************************/
package com.ccwu.tag;
import java.io.IOException;
import javax.mail.*;
import javax.mail.search.FlagTerm;
import javax.servlet.jsp.*;
import javax.servlet.jsp.tagext.*;
public class ListMessagesTag extends BodyTagSupport
{
private String folder;
private String session;
private int msgNum;
private int messageCount;
private Message message;
private Message messages[];
private MessageInfo messageinfo;
private boolean firstIn;
private boolean resetsort;
private boolean increase;
public ListMessagesTag()
{
    int msgNum = 0;
    messageCount = 0;
    firstIn = true;
}
public int doAfterBody() throws JspException
{
    BodyContent bodycontent = getBodyContent();
    try
    { //Write the contents of this BodyContent into a Writer
      bodycontent.writeOut(getPreviousOut());
      }catch(IOException ioexception)
      {
       throw new JspTagException("IterationTag: "+ioexception.getMessage());
      }
      bodycontent.clearBody();
      if(msgNum < messageCount && msgNum > -1)
      { //List messages
        getMessage();
        return super.EVAL_BODY_AGAIN;
      } else //end of list messages
      {
        reset();
        return super.EVAL_PAGE;
      }
}
public int doStartTag() throws JspException
{
    try
    {
      String s = getFolder();
      Folder folder1 =
      (Folder)super.pageContext.getAttribute(getFolder(), PageContext.SESSION_SCOPE);
      //A PageContext instance provides access to all the namespaces associated
```

```
      //with a JSP page, provides access to several page attributes
      if(!folder1.isOpen())
        try//Make sure the folder is open
        {
          folder1.open(Folder.READ_WRITE);
        }catch(MessagingException ex)
        {
          folder1.open(Folder.READ_ONLY);
        }//search undeleted messages
       FlagTerm flagterm = new FlagTerm(new Flags(javax.mail.Flags.Flag.DELETED),
                                                  false);
       messages = folder1.search(flagterm);
       messageCount = messages.length;
       if(firstIn)
       {
         msgNum = messageCount-1;
         firstIn = false;
       }
     }
     catch(Exception exception)
     {
      throw new JspException(exception.getMessage());
     }
     if((messageCount > 0) && (msgNum < messageCount))
     {  //Still have messages need to list
        getMessage();
        return super.EVAL_BODY_AGAIN;
     }else {//To the end of messages
        reset();
        return super.SKIP_BODY;
     }
}
public String getFolder()
{
    return folder;
}
private void getMessage() throws JspException
{
    message = messages[msgNum--];
    messageinfo.setMessage(message);
    super.pageContext.setAttribute(getId(), messageinfo);
}
public String getSession()
{
    return session;
}
public void setFolder(String s)
{
    folder = s;
}
public void setSession(String s)
{
    session = s;
}
private void reset()
{
    firstIn = true;
}
}
/*********************************************************
NAME     :MonthMap.java
DESCRIPTION: To transfer number to month
REMARK  : Use for the table of database in web server
*********************************************************/
package com.ccwu.lib;
import java.util.Hashtable;
public class MonthMap{
```

```
private Hashtable h, h1;
public MonthMap()
{
    h = new Hashtable();
    h1 = new Hashtable();
    h.put("1","Jan.");
    h.put("2","Feb.");
    h.put("3","Mar.");
    h.put("4","Apr.");
    h.put("5","May.");
    h.put("6","Jun.");
    h.put("7","Jul.");
    h.put("8","Aug.");
    h.put("9","Sep.");
    h.put("10","Oct.");
    h.put("11","Nov.");
    h.put("12","Dec.");
    h1.put("1","jan");
    h1.put("2","feb");
    h1.put("3","mar");
    h1.put("4","apr");
    h1.put("5","may");
    h1.put("6","jun");
    h1.put("7","jul");
    h1.put("8","aug");
    h1.put("9","sep");
    h1.put("10","oct");
    h1.put("11","nov");
    h1.put("12","december");
}
public String toMonth(int month)
{
    return (String)h.get(Integer.toString(month));
}
public String toMonth1(int month)
{
    return (String) h1.get(Integer.toString(month));
}
}
/*************************************************************
NAME     :MoveToTrash.java
DESCRIPTION: move messages to the trash folder
*************************************************************/
package com.ccwu.tag;
import java.io.IOException;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.mail.*;
import javax.mail.search.FlagTerm;
import java.util.*;
public class MoveToTrash extends HttpServlet
{
private Message messages[];
public MoveToTrash() {}
public void init() throws ServletException {}
public void doPost(HttpServletRequest httpservletrequest, HttpServletResponse
                   httpservletresponse) throws IOException,ServletException
{
    HttpSession httpsession = httpservletrequest.getSession(true);
    if(httpsession != null)
    {
      try
      {
        Folder folder = (Folder)httpsession.getAttribute("folder");
        folder.close(true);
        folder.open(Folder.READ_WRITE);
```

```
        FlagTerm flagterm = new FlagTerm(new Flags(javax.mail.Flags.Flag.DELETED)
                                                ,false);
        //seach undeleted messages
        messages = folder.search(flagterm);
        Store store = (Store)
        httpsession.getAttribute("store");
        Folder csmail = store.getFolder("mail");
        if (!csmail.exists())//login first time
           csmail.create(Folder.HOLDS_FOLDERS);
        Folder f = csmail.getFolder("Trash");
        if (!f.exists()//View trash folder first time
           f.create(Folder.HOLDS_MESSAGES);
        for(int i=0; i<messages.length; i++)
        {
           if(httpservletrequest.getParameter(Integer.toString(i+1))!= null)
           {
            f.appendMessages(new Message[] {messages[i]});//Add messages to Trash
            messages[i].setFlag(Flags.Flag.DELETED, true);
           //Delete messages in the original folder
           }
        }
        folder.close(true);
        folder.open(Folder.READ_WRITE);
        getServletConfig().getServletContext().
        getRequestDispatcher("/view/right.jsp").
        forward(httpservletrequest,httpservletresponse);//forward page
       } catch(MessagingException messagingexception)
       {
        throw new ServletException(messagingexception.getMessage());
       }
      }
}
public void destroy() {}
}
/*********************************************************
NAME     :MultipartFormDataRequest.java
DESCRIPTION: handle muitipart request from compose.jsp
*********************************************************/
package ccwu.upload;
import java.io.IOException;
import java.util.*;
import javax.servlet.ServletRequest;
import javax.servlet.http.HttpServletRequest;
import ccwu.upload.parsing.StrutsMultipartParser;
public class MultipartFormDataRequest
{
public static String STRUTSPARSER = "org.apache.struts.upload.MultipartIterator";
public static String DEFAULTPARSER;
public static String STRUTSPARSERTEMPDIRECTORY = System.getProperty("java.io.tmpdir");
public static int MAXCONTENTLENGTHALLOWED = 0x40000000;
public static String DEFAULTENCODING = "iso-8859-1";
private Hashtable parameterName;
private Hashtable files;
public MultipartFormDataRequest(HttpServletRequest req,int maxcontentlength, String
                        parser, String encoding) throws IOException, UploadException
{
   parameterName = null;
   files = null;
   parameterName = new Hashtable();
   files = new Hashtable();
   if(req == null)
     new UploadException(UploadException.INVALIDREQUEST);
   if(parser == null)
     parser = DEFAULTPARSER;//is Set in attach.jsp
   if(parser.equalsIgnoreCase(STRUTSPARSER))//Default set
   {
     StrutsMultipartParser smp = new StrutsMultipartParser();
```

```
        smp.handleRequest(req,maxcontentlength,parameterName,files,
                STRUTSPARSERTEMPDIRECTORY, encoding);
        //Send request to StructMultipartParser.java
        //Return 2 hashtable: parameterName and files
      } else
      {
        throw new UploadException("Unknown multipart parser");
      }
  }
  public MultipartFormDataRequest(HttpServletRequest req, int maxcontentlength, String
                                    parser)throws IOException, UploadException //(3)
  {
      this(req,maxcontentlength, DEFAULTPARSER, DEFAULTENCODING);
  }
  public MultipartFormDataRequest(HttpServletRequest req, int maxcontentlength) throws
                                    IOException, UploadException //(2)
  {
      this(req, maxcontentlength, DEFAULTPARSER);
  }
  public MultipartFormDataRequest(HttpServletRequest req)throws IOException,
                                    UploadException //(1)
  {
      this(req, MAXCONTENTLENGTHALLOWED);
  }
  public Hashtable getFiles()
  {
      return files;
  }
  public Enumeration getParameterNames()
  {
      return parameterName.keys();
  }
  public String getParameter(String name)
  {
      try
      {
        Vector v = (Vector)parameterName.get(name);
        if(v == null || v.size() == 0)
          return null;
        else
          return (String)v.elementAt(v.size() - 1);
      }catch(Exception ex)
      {
        return null;
      }
  }
  public String[] getParameterValues(String name)
  {
      try
      {
        Vector v = (Vector)parameterName.get(name);
        if(v == null || v.size() == 0)
          return null;
        else
        {
          String valuesArray[] = new String[v.size()];
          v.copyInto(valuesArray);
          return valuesArray;
        }
      }catch(Exception ex)
      {
        return null;
      }
  }
  public static Boolean isMultipartFormData(HttpServletRequest req)
  {  //Check if request type is "multipart/form-data"
      String type = null;
```

```
    String type1 = req.getHeader("Content-Type");
    String type2 = req.getContentType();
    if(type1 == null && type2 != null)
      type = type2;
    else if(type2 == null && type1 != null)
      type = type1;
    else if(type1 != null && type2 != null)//Choose longer one
      type = type1.length() <= type2.length() ? type2 : type1;
    return type != null && type.toLowerCase().startsWith("multipart/form-data");
}
}
/*********************************************************
NAME     :SendMail.java
DESCRIPTION: Package all message content and attachments then send the messages
REMARK: Referenced classes of package SendMailAuthenticator.java, ByteArrayDataSource.java,
          ComposeModel.java.
*********************************************************/
package ccwu.sendmail.service;
import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.util.*;
import javax.activation.DataHandler;
import javax.mail.*;
import javax.mail.internet.*;
import ccwu.upload.UploadFile;
public class SendMail
{
public static boolean DEBUGSESSION = false;
public static String CRLF = "\n";
public SendMail()
{  //fit for different operating systems
   CRLF = System.getProperty("line.separator");
}
public void SmtpSend(ComposeModel composemodel, Vector files) throws
                        MessagingException, IOException,  AddressException
{
   Session session = null;
   String CHARSET = composemodel.getCharset();
   if(CHARSET != null && CHARSET.trim().equals(""))
//s.trim() Returns a copy of the string, with leading and trailing whitespace omitted
   CHARSET = null;
   Properties properties = composemodel.getProperties();
   properties.put("mail.smtp.auth","true");
   if(composemodel.getSmtpserver() != null)
      properties.put("mail.smtp.host", composemodel.getSmtpserver());
   if(composemodel.getUsername() != null)
      properties.put("mail.smtp.user",  composemodel.getUsername());
   SendMailAuthenticator sendmailauthenticator = new sendMailAuthenticator(
                           composemodel.getUsername(), composemodel.getPassword());
   session = Session.getInstance(properties, sendmailauthenticator);
   if(session != null)
   {
     session.setDebug(DEBUGSESSION);
     MimeMessage mimemessage = new MimeMessage(session);
     if(composemodel.getXmailer() != null)
       mimemessage.addHeader("X-Mailer",composemodel.getXmailer());
     mimemessage.addHeader("X-Priority", "".concat(String.valueOf(String.valueOf(
              composemodel.getXpriority()))));
     if(composemodel.getReturnReceipt())
     {
      if(composemodel.getIsDraft()){}
      else
      {
        InternetAddress ainternetaddress[] = ParseAddress(
           composemodel.getFromemail(), composemodel.getFromname(), CHARSET);
        if(ainternetaddress != null)
        {
```

```
            mimemessage.addHeader("Return-Receipt-To",""".concat(String.valueOf(
             String.valueOf(ainternetaddress[0].toString()))));
            mimemessage.addHeader("Disposition-Notification-To", """.concat(
                String.valueOf(String.valueOf(ainternetaddress[0].toString()))));
          }
        }
      }
      mimemessage.addHeader("MIME-Version", "1.0");
      mimemessage.addHeader("Content-Type","text/html");
      InternetAddress ainternetaddress1[] = ParseAddress(composemodel.getFromemail(),
            composemodel.getFromname(), CHARSET);
      mimemessage.setFrom(ainternetaddress1[0]);
      if(ainternetaddress1.length > 1)
      {
        InternetAddress ainternetaddress2[] = new InternetAddress[ainternetaddress1.length - 1];
        System.arraycopy(ainternetaddress1,1,ainternetaddress2,0,
                    ainternetaddress1.length- 1);
        mimemessage.addFrom(ainternetaddress2);
      }
      if(composemodel.getReplyto() != null       && !composemodel.getReplyto().equals(""))
      {
        InternetAddress AddressReplyTo[] = ParseAddress(composemodel.getReplyto(),
            composemodel.getReplytoname(), CHARSET);
        mimemessage.setReplyTo(AddressReplyTo);
      }
      if(composemodel.getTo() != null)
      {
        InternetAddress AddressTo[] = ParseAddress(composemodel.getTo(),
                                composemodel.getToname(), CHARSET);
         mimemessage.setRecipients(javax.mail.Message$RecipientType.TO, AddressTo);
      }
      if(composemodel.getCc() != null && !composemodel.getCc().equals(""))
      {
        InternetAddress AddressCc[] = ParseAddress(composemodel.getCc(),
                                    composemodel.getCcname(), CHARSET);
mimemessage.setRecipients(javax.mail.Message$RecipientType.CC, AddressCc);
      }
      if(composemodel.getBcc() != null && !composemodel.getBcc().equals(""))
      {
        InternetAddress AddressBcc[] = ParseAddress(composemodel.getBcc(),
            composemodel.getBccname(), CHARSET);
            mimemessage.setRecipients(javax.mail.Message$RecipientType.BCC,AddressBcc);
      }
      if(composemodel.getSubject() != null)
      {
        if(CHARSET != null)
            mimemessage.setSubject(composemodel.getSubject(), CHARSET);
        else
           mimemessage.setSubject(composemodel.getSubject());
      }
      MimeMultipart mimemultipart = new MimeMultipart();
      MimeBodyPart mimebodypart = new MimeBodyPart();
      String message = composemodel.getMessage();
      if(CHARSET != null)
         mimebodypart.setContent(message,"text/html;charset=gb2312");
      else
         mimebodypart.setText(String.valueOf(message));
      mimemultipart.addBodyPart(mimebodypart);
      if(composemodel.isForward())
      {
         Message m=composemodel.getMsgFromForward();
         Multipart p= (Multipart) m.getContent();
         int cnt = p.getCount();
         String disposition;
         for (int i = 0; i < cnt; i++)
         {
            disposition=p.getBodyPart(i).getDisposition();
```

```
            if(disposition != null){if(disposition.equals(Part.ATTACHMENT))
            {
               mimemultipart.addBodyPart(p.getBodyPart(i));
            }
          }
        }
        if(files != null)
        {
          for(int i = 0; i < files.size(); i++)
          {
            MimeBodyPart mimebodypart1 = new MimeBodyPart();
            UploadFile uploadfile = (UploadFile)files.elementAt(i);
            mimebodypart1.setDataHandler(new DataHandler(new ByteArrayDataSource(
                  uploadfile.getInpuStream(), uploadfile.getContentType())));
        mimebodypart1.setFileName(uploadfile.getFileName());
            mimemultipart.addBodyPart(mimebodypart1);
          }
        }
        mimemessage.setContent(mimemultipart);
        if(composemodel.getIsDraft())
        {
          Store store = composemodel.getStore();
          Folder csmail = store.getFolder("mail");
          if (!csmail.exists())//Login first time
             csmail.create(Folder.HOLDS_FOLDERS);
          Folder f = csmail.getFolder("Drafts");
          if (!f.exists())//Use the drafts folder first time
             f.create(Folder.HOLDS_MESSAGES);
          f.appendMessages(new Message[] {mimemessage});
          //add the message to the draft folder
        }else
        {
          mimemessage.setSentDate(composemodel.getDate());
          Transport.send(mimemessage);
           Store store = composemodel.getStore();
          Folder csmail = store.getFolder("mail");
          if (!csmail.exists())//Login first time
             csmail.create(Folder.HOLDS_FOLDERS);
          Folder f = csmail.getFolder("sent-mail");
          if (!f.exists())//Send messages first time
             f.create(Folder.HOLDS_MESSAGES);
          f.appendMessages(new Message[] {mimemessage});
          //Create the Sent folder and store the message to it
        }
      }
    }
    private InternetAddress[] ParseAddress(String address, String name, String charset) throws AddressException
    {  //Parse multiple Address by ";"
      InternetAddress internetaddress = null;
      StringTokenizer stringtokenizer = new StringTokenizer(address, ComposeModel.SEPARATOR);
      StringTokenizer stringtokenizer1 = null;
      if(name != null)
        stringtokenizer1 = new StringTokenizer(name, ComposeModel.SEPARATOR);
      if(stringtokenizer.hasMoreTokens())
      {
        Vector AddressVector = new Vector();
        String s1 = null;
        Object obj = null;
        do
        {
          if(!stringtokenizer.hasMoreTokens())
             break;
          String s = stringtokenizer.nextToken();
          try
          {
             if(stringtokenizer1 != null)
            s1 = stringtokenizer1.nextToken();
```

```
        }catch(NoSuchElementException nosuchelementexception)
        {
          s1 = null;
        }
        if(s != null)
        {
          if(s1 != null)
          try
          {
            if(charset != null)
              internetaddress = new InternetAddress(s, s1, charset);
            else
              internetaddress = new InternetAddress(s, s1);
          }catch(UnsupportedEncodingException unsupportedencodingexception1)
           {
            internetaddress = new InternetAddress(s);
          }
          else
            internetaddress = new InternetAddress(s);
          AddressVector.addElement(internetaddress);
          //Put all addresses to a vector
        }
      } while(true);
      InternetAddress AddressArray[] = new InternetAddress[1];
      AddressArray = (InternetAddress[])AddressVector.toArray(AddressArray);
      return AddressArray;
    }
    if(name == null)
      internetaddress = new InternetAddress(address);
    else
      try
      {
        if(charset != null)
          internetaddress = new InternetAddress(address, name, charset);
        else
          internetaddress = new InternetAddress(address, name);
      }catch(UnsupportedEncodingException unsupportedencodingexception)
      {
        internetaddress = new InternetAddress(address);
      }
        InternetAddress ainternetaddress[] = new InternetAddress[1];
        ainternetaddress[0] = internetaddress;
        return ainternetaddress;
}
}
/*************************************************************
NAME     :SendMailAuthenticator.java
DESCRIPTION: A container to store username and password for JavaMail authenticator
REMARK     : For some mailserver which need authenticator to send mails
*************************************************************/
package ccwu.sendmail.service;
import javax.mail.Authenticator;
import javax.mail.PasswordAuthentication;
public class SendMailAuthenticator extends Authenticator
{
String Username;
String Password;
public SendMailAuthenticator(String username, String password)
{
  Username = username;
  Password = password;
}
public PasswordAuthentication getPasswordAuthentication()
{
  PasswordAuthentication passwordauthentication = new asswordAuthentication(Username, Password);
  return passwordauthentication;
}
```

```
}
/*************************************************************
NAME     :SendMailServlet.java
DESCRIPTION: Handle sending mails
*************************************************************/
package ccwu.sendmail;
import java.io.*;
import java.util.*;
import javax.mail.MessagingException;
import javax.mail.internet.AddressException;
import javax.mail.internet.InternetAddress;
import javax.servlet.*;
import javax.servlet.http.*;
import ccwu.sendmail.service.ComposeModel;
import ccwu.sendmail.service.SendMail;
import ccwu.upload.UploadBean;
import ccwu.upload.UploadException;
public class SendMailServlet extends HttpServlet
{
StringBuffer logBuffer = new StringBuffer();
public SendMailServlet() {}
public void init() throws ServletException {}
public void service(HttpServletRequest httpservletrequest,
                    HttpServletResponse httpservletresponse) throws IOException, ServletException
{ //rewrite service mothod of HttpServlet
   String s = null;
   s = httpservletrequest.getServletPath();//call from s=/compose/composenew
   s = s.substring(s.lastIndexOf("/"), s.length());//  s=/composenew
   if(s == null)
     s = "";
   String s1 = "/compose.jsp";
   httpservletrequest.setAttribute("error", ComposeBean.NOERROR);//error=noerror
   httpservletrequest.setAttribute("errormsg", ComposeBean.NOERROR);//errormsg=noerror
   HttpSession httpsession = httpservletrequest.getSession(true);
   if(httpsession != null)
   {
     if(s.equals("/sendmail"))
     {  //User press send button
        ComposeBean composebean =
             (ComposeBean)httpsession.getAttribute("composebean");
        if(httpservletrequest.getParameter("mode") != null)
        {
             composebean.setIsDraft(true);
             httpsession.setAttribute("composebean",composebean);
        }else
        {
             composebean.setIsDraft(false);
                  httpsession.setAttribute("composebean",composebean);
        }
        if(composebean != null)
         {  //Store the composing message content
           StoreData(httpservletrequest, composebean);
           if(composebean.getTo() != null)
            {
              try
              {
                 VerifyAddress(composebean.getTo());
             }catch(AddressException addressexception1)
        {
                       httpservletrequest.setAttribute("error", ComposeBean.TO);
                       httpservletrequest.setAttribute("errormsg","To:Recipient is not verified !!");
                Dispatch(httpservletrequest, httpservletresponse,
                       "/compose.jsp");
                return;
              }
               }
               if(composebean.getCc() !=null)
```

```
            {
              try
              {
                if(!composebean.getCc().equals(""))
                  VerifyAddress(composebean.getCc());
              }catch(AddressException addressexception2)
              {
                 httpservletrequest.setAttribute("error",ComposeBean.CC);
                 httpservletrequest.setAttribute("errormsg", "CC:Recipient is not verified !!");
                Dispatch(httpservletrequest, httpservletresponse,
                                          "/compose.jsp");
                return;
              }
            }
            if(composebean.getBcc() != null)
            {
              try
              {
                 if(!composebean.getBcc().equals(""))
                   VerifyAddress(composebean.getBcc());
              }catch(AddressException addressexception3)
              {
                 httpservletrequest.setAttribute("error",ComposeBean.BCC);
                 httpservletrequest.setAttribute("errormsg", "BCC:Recipient is not verified !!");
                 Dispatch(httpservletrequest, httpservletresponse, "/compose.jsp");
                 return;
              }
            }
            SendMail sendmail = new SendMail();
            UploadBean uploadbean =
                        (UploadBean)httpsession.getAttribute("upbean");
            Vector attachfiles = null;
            if(uploadbean != null)
              attachfiles = uploadbean.getMemorystore();
            try
            {  //Start to send the mail
              sendmail.SmtpSend(composebean, attachfiles);
              httpservletrequest.setAttribute("error","Email sent successfully");
              try
              {
                if(uploadbean != null)//reset upload files to null
                  uploadbean.resetStore();
              }catch(IOException ioexception)
              {
                 ioexception.printStackTrace();
              }catch(UploadException uploadexception)
              {
                 uploadexception.printStackTrace();
              }
            }catch(MessagingException messagingexception)
            {                   messagingexception.printStackTrace();
              httpservletrequest.setAttribute("error", "Error email not sent");
              httpservletrequest.setAttribute("errormsg", "Please verify recipients");
            }
            s1 = "/mailsent.jsp";//forward page to mailsent.jsp
        }
      } else
      if(s.equals("/attachform"))
    {
      ComposeBean composebean1 =(ComposeBean)httpsession.getAttribute("composebean");
      if(composebean1 != null)
        StoreData(httpservletrequest, composebean1);
      s1 = "/attach.jsp";//Store data first, then forward page to attach.jsp
    } else if(s.equals("/composenew"))
    {
      httpsession.removeAttribute("upbean");//Remove attach files
      httpsession.removeAttribute("composebean");//Remove messages
```

```
      s1 = "/compose.jsp";
    } else if(s.equals("/compose"))
    {
      s1 = "/compose.jsp";
    }//Start forward page
    Dispatch(httpservletrequest, httpservletresponse, s1);
}
private void Dispatch(HttpServletRequest httpservletrequest, HttpServletResponse httpservletresponse, String s)
throws IOException, ServletException
{
    // Add path before forward page
    String s1 = null;
    s1 = httpservletrequest.getServletPath().
              Substring(0,httpservletrequest.getServletPath().lastIndexOf("/"));
     RequestDispatcher requestdispatcher = getServletContext().getRequestDispatcher(
                        String.valueOf(s1) + String.valueOf(s));
     requestdispatcher.forward(httpservletrequest, httpservletresponse);
}
private void StoreData(HttpServletRequest httpservletrequest, ComposeBean composebean)
{   //Store message in composebean
    composebean.setFromname();
    String s2 = httpservletrequest.getParameter(ComposeBean.TO);
    if(s2 != null)
       composebean.setTo(s2);
    else
       composebean.setTo("null");
    String s3 = httpservletrequest.getParameter(ComposeBean.REPLYTO);
    if(s3 != null)
       composebean.setReplyto(s3);
    else
      composebean.setReplyto("null");
    String s4 = httpservletrequest.getParameter(ComposeBean.CC);
    if(s4 != null)
      composebean.setCc(s4);
    String s5 = httpservletrequest.getParameter(ComposeBean.BCC);
    if(s5 != null)
      composebean.setBcc(s5);
    String s6 = httpservletrequest.getParameter(ComposeBean.SUBJECT);
    if(s6 != null)
      composebean.setSubject(s6);
    String s7 = httpservletrequest.getParameter(ComposeBean.MESSAGE);
    if(s7 != null)
      composebean.setMessage(s7);
    String s8 = httpservletrequest.getParameter(ComposeBean.PRIORITY);
    int i = 3;
    if(s8 != null)
    {
      try
      {
         i = Integer.parseInt(s8);
      }catch(NumberFormatException      numberformatexception) {}
      composebean.setXpriority(i);
    }
    String s9 = httpservletrequest.getParameter(ComposeBean.RETURNRECEIPT);
    if(s9 != null)
      composebean.setReturnReceipt(true);
    else
      composebean.setReturnReceipt(false);
}
private void VerifyAddress(String s) throws     AddressException
{   //Use ";" to separate all address
    InternetAddress internetaddress = null;
    StringTokenizer stringtokenizer = new StringTokenizer(s,ComposeModel.SEPARATOR);
    if(stringtokenizer.hasMoreTokens())
    do
    {
      if(!stringtokenizer.hasMoreTokens())
```

```
            break;
          String s1 = stringtokenizer.nextToken();
          if(s1 != null)
            internetaddress = new InternetAddress(s1);
          //If fail to create the internetaddress instance throws exception
        } while(true);
        else
          internetaddress = new InternetAddress(s);
    }
    public void log(String s)//For debug in servlet
    {
        s = new Date().toString() + ": " + s;
        System.err.println(s);
        logBuffer.append(s);
        logBuffer.append("\n");
        super.log(s);
    }
    public void destroy() {}
    }
    /**********************************************************
    NAME     :StoreTo.java
    DESCRIPTION: Handle Store mails to the daily folder
    **********************************************************/
    package com.ccwu.tag;
    import java.io.IOException;
    import javax.servlet.*;
    import javax.servlet.http.*;
    import javax.mail.*;
    import javax.mail.search.FlagTerm;
    import java.util.*;
    import com.ccwu.databases.csmail;
    import java.sql.*;
    public class StoreTo extends HttpServlet
    {
    private Message messages[];
    public StoreTo() {}
    public void init() throws ServletException {}
    public void doPost(HttpServletRequest httpservletrequest, HttpServletResponse
    httpservletresponse) throws IOException, ServletException
    {
        HttpSession httpsession = httpservletrequest.getSession(true);
        String foldername[] = httpservletrequest.getParameterValues("StoreMail");
        if(httpsession != null)
        {
          try
          {  //The folder which user is viewing
             Folder folder = (Folder)httpsession.getAttribute("folder");
             folder.close(true);
             folder.open(Folder.READ_WRITE);
             FlagTerm flagterm = new FlagTerm(new Flags(javax.mail.Flags.Flag.DELETED),
                          false);
            messages = folder.search(flagterm);//Serach undeleted messages
            Store store = (Store)httpsession.getAttribute("store");
            Folder csmail = store.getFolder("mail");
            if (!csmail.exists())//Login first time   csmail.create(Folder.HOLDS_FOLDERS);
            Folder schedule = csmail.getFolder("Schedule");
            if (!schedule.exists())//Use daily folders first time schedule.create(Folder.HOLDS_FOLDERS);
            Folder f = schedule.getFolder(foldername[0]);
            if (!f.exists())
              f.create(Folder.HOLDS_MESSAGES);//Use that daily folder first time
            if(httpservletrequest.getParameter("msgnum")!= null)
            {
               int msgnum =Integer.parseInt(httpservletrequest.getParameter("msgnum"));
               f.appendMessages(new Message[] {messages[msgnum-1]});
              //Add messages to that daily folder
               messages[msgnum-1].setFlag(Flags.Flag.DELETED, true);
            }
```

```
        folder.close(true);
        folder.open(Folder.READ_WRITE);
        getServletConfig().getServletContext().getRequestDispatcher("/view/right.jsp
        ").forward(httpservletrequest,httpservletresponse);
       //return page to right.jsp
       } catch(MessagingException messagingexception)
       {
          throw new ServletException(messagingexception.getMessage());
       }
   }
}
public void destroy() {}
}
/*************************************************************
NAME     :StrutsMultipartIterator.java
DESCRIPTION: The Iterator to handle multipart
REMARK: Extend org.apache.struts.upload.MultipartIterator
*************************************************************/
package ccwu.upload.parsing;
import java.io.*;//For BufferedOutputStream
import java.util.Vector;
import javax.servlet.http.HttpServletRequest;
import org.apache.struts.upload.*;//for MultipartIterator
public class StrutsMultipartIterator extends MultipartIterator
{
public static String FILE_PREFIX = "strts";
public StrutsMultipartIterator(HttpServletRequest request, int bufferSize, long maxSize, String tempDir) throws
IOException
{
    super(request, bufferSize, maxSize, tempDir);
}
public MultipartElement getNextElement() throws IOException
{
    MultipartElement element = null;
//MultipartIterator.inputStream => MultipartBoundaryInputStream
//The InputStream to use to read the multipart data.
//MultipartElement:This class represents an element in a multipart request.
//It has a few methods for determining * whether or not the element is a String or a //file, and methods to retrieve the
data of the aforementioned element.
//Text input elements have a null content type, files have a non-null content type.
    if(!isMaxLengthExceeded() && !super.inputStream.isFinalBoundaryEncountered())
    {
      if(super.inputStream.isElementFile())
      {  //Gets whether or not the current form element being read is a file.
         element = createFileMultipartElement();
      } else
      {  //Inherited from MultipartIterator function
        String encoding = getElementEncoding();
        element = createTextMultipartElement(encoding);
      }
      super.inputStream.resetForNextBoundary();
      //Resets this stream for use with the next element, to be used after a //boundary is encountered.
    }
    return element;
}
protected MultipartElement createFileMultipartElement()       throws IOException
{
    File elementFile = createLocalFile();
    MultipartElement element = new MultipartElement(super.inputStream.
              getElementName(),super.inputStream.getElementFileName(),
              super.inputStream.getElementContentType(), elementFile);
    return element;
}
protected File createLocalFile() throws IOException
{  //Upload files from local client to Web server
   File tempFile = File.createTempFile(FILE_PREFIX,null new File(super.tempDir));
   //Creates a new empty file in the specified directory,
```

```
    //using the given prefix and suffix strings to generate its name.
    BufferedOutputStream fos = new BufferedOutputStream(new FileOutputStream(tempFile),super.diskBufferSize);
    //BufferedOutputStream(OutputStream out,int size)
    //Creates a new buffered output stream to write data
    //to the specified underlying output stream with the specified buffer size.
    //super.diskBufferSize=20480
    int read = 0;
    byte buffer[] = new byte[super.diskBufferSize];
    do
    {  //readin buffer first
       if((read=super.inputStream.read(buffer,0,super.diskBufferSize)) <= 0)
       //MultipartBoundaryInputStream.read(byte[] buffer,int offset,int length)
       //Returns -1 if it's the end of the stream or if a boundary is encountered
          break;
       fos.write(buffer, 0, read);//From buffer to tempFile
       //write(byte[] b, int off, int len)
       //Writes len bytes from the specified byte array starting at offset off to //this buffered output stream.
    } while(true);
    fos.flush();
    fos.close();
    if(tempFile.length() == (long)0)
       tempFile.delete();
    return tempFile;
   }
}
/*********************************************************
NAME     :StrutsMultipartParser.java
DESCRIPTION: To parse multipart request from compose.jsp
*********************************************************/
package ccwu.upload.parsing;
import java.io.File;
import java.io.IOException;
import java.util.*;
import javax.servlet.http.HttpServletRequest;
import ccwu.upload.UploadFile;
import org.apache.struts.upload.*;
public class StrutsMultipartParser
{
public static String MAXCONTENTLENGTHEXCEEDED = "Max Content-Length exceeded";
public static int DEFAULTBUFFERSIZE = 4096;
private Hashtable H1;
private Hashtable H2;
public StrutsMultipartParser() {}
public void handleRequest(HttpServletRequest request, int maxContentLength, Hashtable prameters, Hashtable files,
String tmpFolder, String encoding) throws IOException
{
//Input: request, maxContentLength, and encoding method
//Return: 2 hashtable: parameters and files
   StrutsMultipartIterator iterator =
   new StrutsMultipartIterator(request, DEFAULTBUFFERSIZE, maxContentLength, tmpFolder);
   H1 = files;
   H2 = new Hashtable();
   MultipartElement element;
   while((element = iterator.getNextElement()) != null)
      if(!element.isFile())
         UnWrapper(request, element);
      //Parse request then store parameter value to H2
      else
         SetFile(element);
   Enumeration e = H2.keys();
   do
   {  //Handle parameters
      if(!e.hasMoreElements())
         break;
      String paramName = (String)e.nextElement();
      String paramValues[] = (String[])H2.get(paramName);
      if(paramValues != null)
```

```
      {
          Vector vals = new Vector();
          for(int i = 0; i < paramValues.length; i++)
             vals.addElement(paramValues[i]);
          parameters.put(paramName, vals);
       }
    } while(true);
    if(iterator.isMaxLengthExceeded())
       throw new IOException(MAXCONTENTLENGTHEXCEEDED);
    else
       return;
}
private void UnWrapper(HttpServletRequest request, MultipartElement element)
{
    if(request instanceof MultipartRequestWrapper)
    {
      ((MultipartRequestWrapper)request).setParameter(
                                element.getName(),element.getValue());
      String textValues[] = (String[])H2.get(element.getName());
      if(textValues != null) {//Add one parameter value to the end of textValue[]
        String textValues2[] = new String[textValues.length + 1];
      System.arraycopy(textValues, 0, textValues2, 0, textValues.length);
      textValues2[textValues.length] = element.getValue();
      textValues = textValues2;
    } else
    {
      textValues = new String[1];
      textValues[0] = element.getValue();
    }
    H2.put(element.getName(), textValues);
}
private void SetFile(MultipartElement element)
{
    File tempFile = element.getFile();//tempFile is a file in Web server
    if(tempFile.exists())
    {
       StrutsUploadFile theFile = new StrutsUploadFile(tempFile.getAbsolutePath());
       theFile.setContentType(element.getContentType());
       theFile.setFileName(element.getFileName());
       theFile.setFileSize((int)tempFile.length());
       H1.put(element.getName(), theFile);
    }
}
}
/*********************************************************
NAME     :StrutsUploadFile.java
DESCRIPTION: Handle file upload from user to Web server
*********************************************************/
package ccwu.upload.parsing;
import java.io.*;
import ccwu.upload.UploadFile;
public class StrutsUploadFile extends UploadFile
{
private String TmpFilename;
public StrutsUploadFile()
{
    TmpFilename = null;
}
public StrutsUploadFile(String tmpFilename)
{
    TmpFilename = null;
    TmpFilename = tmpFilename;
}
public byte[] getData()
{  //Return byte format data from input stream
    InputStream is = getInpuStream();
    if(is != null)
```

```
    {
      int size = (int)getFileSize();
      if(size > 0)
      {
        byte memoryfile[] = new byte[size];
        try
        {
          is.read(memoryfile);
          is.close();
        }catch(IOException ioexception) {}
        return memoryfile;
      } else
      {
        return null;
      }
    } else
    {
      return null;
    }
}
public void reset()
{  //Delete temp file if it already exist
   if(TmpFilename != null)
   {
      File f = new File(TmpFilename);
      if(f.exists())
        f.delete();
   }
}
public InputStream getInpuStream()
{  //Get input stream from temp file name
   if(TmpFilename == null)
      return null;
   File f = new File(TmpFilename);
   if(f.exists())
     try
     {
        BufferedInputStream bufferedinputstream = new       BufferedInputStream(
                                    new FileInputStream(TmpFilename));
        return bufferedinputstream;
     }catch(FileNotFoundException ex)
     {
        InputStream inputstream = null;
        return inputstream;
     }
   else
     return null;
}
public String getTmpFilename()
{
   return TmpFilename;
}
}
/*************************************************************
NAME     :StrutsUploadBean.java
DESCRIPTION: Store upload files information
*************************************************************/
package ccwu.upload;
import java.io.*;
import java.util.*;
import java.lang.reflect.Constructor;
import java.sql.Connection;
import java.util.*;
import java.util.zip.ZipEntry;
import java.util.zip.ZipOutputStream;
public class UploadBean implements Serializable
{
```

```
public static final int MAXUPLOADEDFILES = 0xf4240;
public static final long MAXUPLOADEDFILESIZE = 0x40000000L;
public static char SEPARATOR = '/';
public static final String EMPTYENTRY = "EmptyFile";
public static int TRANSFERTBUFFERSIZE = 4096;
private int Max;
private long Filesizelimit;
private Vector MemoryStore;
public UploadBean()
{
    Max = 100000000;//Maxmum total file size
    Filesizelimit = 5300000L;// Each file size limit: about 5 MB
    MemoryStore = null;
    SEPARATOR = System.getProperty("file.separator").charAt(0);
    MemoryStore = new Vector();
}
public Vector getMemorystore()
{
    return MemoryStore;
}
public void setParser(String parserid)
{
    MultipartFormDataRequest.DEFAULTPARSER = parserid;
}
public String getParser()
{
    return MultipartFormDataRequest.DEFAULTPARSER;
}
public String getParsertmpdir()
{
    return MultipartFormDataRequest.STRUTSPARSERTEMPDIRECTORY;
}
public void setParsertmpdir(String dir)
{
    if(dir != null)
    {
      dir = dir.replace('\\', '/').replace('/', SEPARATOR);
      MultipartFormDataRequest.STRUTSPARSERTEMPDIRECTORY = dir;
    }
}
public void setMaxfiles(int max)
{
    Max = max;
}
public int getMaxfiles()
{
    return Max;
}
public long getFilesizelimit()
{
    return Filesizelimit;
}
public void setFilesizelimit(long max)
{
    Filesizelimit = max;
}
public void store(MultipartFormDataRequest mrequest, String field) throws IOException, UploadException
{
    StoreToMemory(mrequest, field);
}
public void store(MultipartFormDataRequest mrequest) throws IOException, UploadException//(1)
{   //Store upload file to Web Server when user press attach button each time
    Hashtable files = mrequest.getFiles();
    String field;
    for(Enumeration e = files.keys(); e.hasMoreElements(); store(mrequest, field))
    {
      field =(String)e.nextElement();//field="filename"
```

```
    }
}
private void StoreToMemory(MultipartFormDataRequest mrequest, String field) throws IOException,UploadException
{
    Hashtable files = mrequest.getFiles();
    UploadFile file = (UploadFile)files.get(field);
    if(file.getFileName() != null && !file.getFileName().equals("") &&
      file.getFileSize() >= (long)0)
    {
      if(file.getFileSize() > Filesizelimit)
          throw new UploadException(String.valueOf(String.valueOf((new
            StringBuffer(String.valueOf(String.valueOf(UploadException.
          UPLOADFILESIZELIMITREACHED))))).append(" ").append(file.getFileName())));

       if(MemoryStore.size() >= Max)
          throw new UploadException(UploadException.UPLOADLIMITREACHED);
       MemoryStore.addElement(file);
    }
}
public void resetStore() throws IOException, UploadException
{
    for(int f = 0; f < MemoryStore.size(); f++)
    {
       UploadFile file = (UploadFile)MemoryStore.elementAt(f);
       file.reset();
    }
    MemoryStore.removeAllElements();
}
}
/*************************************************************
NAME     :UploadException.java
DESCRIPTION: Store upload exception messages.
*************************************************************/
package ccwu.upload;
public class UploadException extends Exception
{
public static String INVALIDREQUEST = "Invalid input request";
public static String UPLOADERROR = "Upload error";
public static String FOLDERISREADONLY = "Folder is read only";
public static String CANNOTCREATEFOLDER = "Cannot create folder";
public static String UPLOADLIMITREACHED = "Upload files limit reached";
public static String UPLOADFILESIZELIMITREACHED = "Upload file size limit reached";
public static String UPLOADFILENAMEDENIED = "Upload    filename not allowed";
public static String CANNOTDELETESTORE = "Cannot delete store";
public static String UPLOADSTORENOTFOUND = "Store not found";
public UploadException() {}
public UploadException(String msg)
{
   super(msg);
}
}
/*************************************************************
NAME     :Uploadfile.java
DESCRIPTION: Store the upload file information.
*************************************************************/
package ccwu.upload;
import java.io.InputStream;
import java.io.Serializable;
public abstract class UploadFile implements Serializable
{
private String ContentType;
private String FileName;
private long FileSize;
public UploadFile(String filename, String contenttype, long size)
{
   ContentType = null;
   FileName = null;
```

```
    FileSize = -1L;
    FileName = filename;
    ContentType = contenttype;
    FileSize = size;
}
public UploadFile()
{
    this(null, null, -1L);
}
public long getFileSize()
{
    return FileSize;
}
public void setFileSize(long filesize)
{
    FileSize = filesize;
}
public String getFileName()
{
    return FileName;
}
public void setFileName(String filename)
{
    FileName = filename;
}
public String getContentType()
{
    return ContentType;
}
public void setContentType(String contentType)
{
    ContentType = contentType;
}
public abstract byte[] getData();
public abstract InputStream getInpuStream();
public abstract void reset();
}
/*********************************************************
NAME     :UploadServlet.java
DESCRIPTION: The servlet to handle upload files.
*********************************************************/
package ccwu.sendmail;
import java.io.IOException;
import java.util.*;//for Date
import java.text.DecimalFormat;
import java.text.NumberFormat;
import java.util.Vector;
import javax.servlet.*;
import javax.servlet.http.*;
import ccwu.upload.*;
public class UploadServlet extends HttpServlet
{
StringBuffer logBuffer = new StringBuffer();
public UploadServlet() {}
public void init() throws ServletException {}
public void doPost(HttpServletRequest httpservletrequest, HttpServletResponse httpservletresponse) throws
IOException, ServletException
{  //The Jsp page to upload file
   String s = "/attach.jsp";
   HttpSession httpsession = httpservletrequest.getSession(true);
   if(httpsession != null && MultipartFormDataRequest.isMultipartFormData(
                     httpservletrequest))
   try
   {
      int i = -1;
      ComposeBean composebean =(ComposeBean)httpsession.getAttribute("composebean");
      if(composebean != null)
```

```
        i = composebean.getAttachmentsizelimit();
      MultipartFormDataRequest multipartformdatarequest = null;
      if(i > 0)//User press the attach butto...start upload file
        multipartformdatarequest = new MultipartFormDataRequest(
                                          httpservletrequest, i);
      else//user didn't set the Attachmentsizelimit....start upload file
        multipartformdatarequest= new MultipartFormDataRequest(httpservletrequest);
      String s2 = null;
      s2 = httpservletrequest.getServletPath();
      s2 = s2.substring(s2.lastIndexOf("/"), s2.length());
      if(s2 == null)
        s2 = "";
      if(s2.equals("/attach"))
      {
        UploadBean uploadbean = (UploadBean)httpsession.getAttribute("upbean");
        if(uploadbean != null)
        {
          boolean flag = true;
          if(composebean != null)
          {
            Vector vector1 =  uploadbean.getMemorystore();
            if(vector1 != null)
            {
              long l = 0L;
              for(int k = 0; k < vector1.size(); k++)                    {
                UploadFile uploadfile1 = (UploadFile)vector1.elementAt(k);
                l += uploadfile1.getFileSize();
              }
              if(i > 0 && l > (long)i)//Total file size > limited file size
                flag = false;
            }
          }
          if(flag)
           {//Store attach file
             uploadbean.store(multipartformdatarequest);
          }
          else
            throw new IOException("Upload Total size limit exceed");
          setAttachInfo((ComposeBean)httpsession.getAttribute("composebean"), uploadbean);
        }
    } else if(s2.equals("/unattach"))
    {  //User presses remove button
       String as[] = multipartformdatarequest.getParameterValues("attachments");
       if(as != null)
       {
         UploadBean uploadbean1 =(UploadBean)httpsession.getAttribute("upbean");
         if(uploadbean1 != null)
         {
           Vector vector2 = uploadbean1.getMemorystore();
           for(int j = 0; j < as.length; j++)
           try
            {
             int k = Integer.parseInt(as[j]);
             UploadFile uploadfile = (UploadFile)vector2.elementAt(k);
             vector2.removeElementAt(k);
             uploadfile.reset();
           }catch(NumberFormatException      numberformatexception) { }
           setAttachInfo((ComposeBean)httpsession.getAttribute(composebean"),
                                              uploadbean1);
         }
       }
     }
  }catch(UploadException uploadexception)
  {
    httpservletresponse.sendRedirect(httpservletresponse.encodeRedirectURL(
       String.valueOf(String.valueOf((new StringBuffer("attachform?")).append(
       "uploaderrormsg").append("=").append(uploadexception.getMessage()))));
```

```
    return;
}catch(IOException ioexception)
{
    httpservletresponse.sendRedirect(httpservletresponse.encodeRedirectURL(
        String.valueOf(String.valueOf((new StringBuffer("attachform?")).append(
        "uploaderrormsg").append("=").append(ioexception.getMessage()))));
    return;
}
String s1 = null;
s1 = httpservletrequest.getServletPath().substring(0, httpservletrequest.
    getServletPath().lastIndexOf("/"));
RequestDispatcher requestdispatcher = getServletContext().getRequestDispatcher(
                            String.valueOf(s1) + String.valueOf(s));
requestdispatcher.forward(httpservletrequest, httpservletresponse);
}
private void setAttachInfo(ComposeBean composebean,      UploadBean uploadbean)
{   //Set Attachment Info to show on Web page
    if(composebean != null && uploadbean != null)
    {
        DecimalFormat decimalformat = new DecimalFormat("0.0");
        Vector vector = uploadbean.getMemorystore();
        if(vector != null)
        {
            String s = "";
            if(vector.size() > 0)
            {
                UploadFile uploadfile = (UploadFile)vector.elementAt(0);
                s = String.valueOf(String.valueOf((new StringBuffer(String.valueOf(
                    String.valueOf(uploadfile.getFileName())))).append(" (").append(
                     decimalformat.format((double)uploadfile.getFileSize()/1024D)).
                    append("KB) ")));
            }
            for(int i = 1; i < vector.size(); i++)
            {
                UploadFile uploadfile = (UploadFile)vector.elementAt(i);
                s = String.valueOf(s) + String.valueOf(String.valueOf(String.valueOf(( new
                StringBuffer(String.valueOf(String.valueOf(ComposeBean.
                    ATTACHSEPARATOR)))).append(uploadfile1.getFileName()).append("
                    (").append(decimalformat.format((double)uploadfile1.getFileSize()
                    /1024D)).append("KB) "))));
            }//To show file size
            composebean.setAttachments(s);
        }
    }
}
public void log(String s)
{   //For debug
    s = new Date().toString() + ": " + s;
    System.err.println(s);
    logBuffer.append(s);
    logBuffer.append("\n");
    super.log(s);
}
public void destroy() {}
}
/*************************************************************
NAME     :csmail.java
DESCRIPTION: The servlet to store daily note to database
*************************************************************/
package com.ccwu.databases;
import java.sql.*;
import java.util.*;
public class csmail
{
String error;
Connection con;
public csmail()  {}
```

```
public void connect() throws ClassNotFoundException, SQLException, Exception
{
   try
   {
      Class.forName("org.gjt.mm.mysql.Driver").newInstance();
      con = DriverManager.getConnection(jdbc:mysql://localhost/csmail ?user=root&password=9841571");
   } catch (ClassNotFoundException cnfe)
   {
     error = "ClassNotFoundException: Could not locate DB driver.";
     throw new ClassNotFoundException(error);
   } catch (SQLException cnfe)
   {
     error = "SQLException: Could not connect to database.";
     throw new SQLException(error);
   } catch (Exception e)
   {
     error ="Exception: An unknown error occurred while connecting " +"to database.";
     throw new Exception(error);
   }
 }
public void disconnect() throws SQLException
{
   try
   {
    if ( con != null )
    {
      con.close();
    }
   } catch (SQLException sqle)
   {
     error = ("SQLException: Unable to close the database connection.");
     throw new SQLException(error);
   }
 }
public ResultSet view(String name, String foldername) throws SQLException, Exception
{
   ResultSet rs = null;
   try
   {
     String queryString = ("SELECT Detail FROM Note WHERE User = '"+name+"'
                    AND Folder = "+foldername+";");
     Statement stmt = con.createStatement();
     rs = stmt.executeQuery(queryString);
   } catch (SQLException sqle)
   {
     error = "SQLException: Could not execute the query.";
     throw new SQLException(error);
   } catch (Exception e)
   {
     error = "An exception occured while retrieving books.";
     throw new Exception(error);
   }
   return rs;
}
public void addNote(String name, String foldername, String note) throws SQLException, Exception
{
   if (con != null)
   {
    try
    {
      PreparedStatement updatenote;
      updatenote = con.prepareStatement("insert into Note values(?, ?, ?);");
      updatenote.setString(1, name);
      updatenote.setString(2, foldername);
      updatenote.setString(3, note);
      updatenote.execute();
    } catch (SQLException sqle)
```

```
    {
      error = "SQLException: update failed, possible duplicate entry";
      throw new SQLException(error);
    }
   } else
   {
     error = "Exception: Connection to database was lost.";
     throw new Exception(error);
   }
}
public void removeNote(String username, String foldername) throws SQLException, Exception
{
   if (con != null)
   {
     try
     {
        PreparedStatement delete;
        delete = con.prepareStatement("DELETE FROM Note WHERE user='"+username+"'
            AND folder='"+foldername+"';");
        delete.execute();
     } catch (SQLException sqle)
     {
         error = "SQLException: update failed, possible duplicate entry";
         throw new SQLException(error);
     } catch (Exception e)
     {
         error = "An exception occured while deleting.";
         throw new Exception(error);
   }
   } else
   {
      error = "Exception: Connection to database was lost.";
      throw new Exception(error);
   }
 }
}
/*********************************************************
NAME     :deletemail.java
DESCRIPTION: The servlet to delete mails from the folder
*********************************************************/
package com.ccwu.tag;
import java.io.IOException;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.mail.*;
import javax.mail.search.FlagTerm;
import java.util.*;
public class deletemail extends HttpServlet
{
private Message messages[];
public deletemail() {}
public void init() throws ServletException {}
public void doPost(HttpServletRequest httpservletrequest, HttpServletResponse
httpservletresponse) throws IOException, ServletException
{
   HttpSession httpsession = httpservletrequest.getSession(true);
   if(httpsession != null)
   {
      try
     {
        Folder folder = (Folder)httpsession.getAttribute("folder");
        folder.close(true);
        folder.open(Folder.READ_WRITE);
        FlagTerm flagterm = new FlagTerm(new Flags(javax.mail.Flags.Flag.DELETED), false);
        messages = folder.search(flagterm);
        for(int i=0; i<messages.length; i++)
        {
```

```
            if(httpservletrequest.getParameter(Integer.toString(i+1))!= null)
            {
               messages[i].setFlag(Flags.Flag.DELETED, true);
            }
         }
         folder.close(true);
         folder.open(Folder.READ_WRITE);
         getServletConfig().getServletContext().
         getRequestDispatcher("/view/right.jsp").forward(
         httpservletrequest, httpservletresponse);
      } catch(MessagingException messagingexception)
      {
         throw new ServletException(messagingexception.getMessage());
      }
   }
}
public void destroy() {}
}
```

REFERENCES

1. JavaMail API Design Specification for version 1.2. September 2000. <http://java.sun.com/products/javamail/JavaMail-1.2.pdf>

2. H.M.Deitel and P.J.Deitel. Java How To Program. Prentice Hall, 2002.

3. C.S.Horstmann and G.Cornell. Core JAVA2. Prentice Hall PTR, 2001.

4. H.M.Deitel,P.J.Deitel and S.E.Santry. Advanced Java 2 Platform. Prentice Hall,2002.

5. Ken Arnold and James Gosling. The Java Programming Language Second Edition. Addison Wesley, February 2000.

6. I.F.Darwin. Java Cookbook. O'Reilly, June 2001.

7. J.Hunter with W.Crawford. Java Servlet programming. O'Reilly, April 2001.

8. Larne Pekowsky. JavaServer Pages. Addison Wesley, April 2000.

9. Falkner, Galbraith, et al. Beginning JSP Web Development. Wrox Press, 2001.

10. David M. Geary. Advanced JavaServer Pages. Prentice Hall PTR, 2001.

11. William B. Sanders. JavaScript DESIGN. New Riders, 2002.

12. Elmasri and navathe. Fundamentals of Database Systems, third edition. Addison Wesley, June 2000.

13. MySQL Reference Manual for version 4.0.4. September 2002. <http://www.mysql.com/documentation/index.html>

14. Apache Struts Framework for version 1.2. 2000-2003. <http://jakarta.apache.org/struts/api/overview-summary.html>

15. Martin Fowler with Kendall Scott. UML Distilled - A brief guide to the standard object modeling language. Addison Wesley Longman, July 2001.