2005

# Wiki message linking

Hongping Yang

# WIKI MESSAGE LINKING

---

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

Hongping Yang

March 2005

# WIKI MESSAGE LINKING

A Project

Presented to the

Faculty of

California State University

San Bernardino

by

Hongping Yang

March 2005

Approved by:

[redacted]

Dr. David Turner, Chair, Computer Science

3/8/2005

Date

[redacted]

Dr. Ernesto Gomez

[redacted]

Dr. Kerstin Voigt

ABSTRACT

WIKI MESSAGE LINKING (WML) is a piece of server software that allows registered users to freely create and edit web page content using any web browser. It is a free communication tool which will be used as a group communication tool and repository of user's work. Members can edit their works online and the group leader can review the works and correct the works directly if necessary from the WML. And furthermore, WML supports HTML markup.

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. David Turner who offered me this project and directed me through this entire effort. I also thank my other committee members, Dr. Kerstin Voigt and Dr. Ernesto Gomez for their valuable input.

If there were not supports from my family, I would not finish this study. I thank my wife Xuejun and my son Tonghui. They are my best friends.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

# CHAPTER ONE

# INTRODUCTION

Wiki Message Linking (WML) is a group communication tool that will be used between group members and the group leader. By using WML, group web pages are created and modified easily. From the WML, new pages are created and owned by their creators. The group leader and the page owner can modify the page, and the owner may give permission for other people to modify the web page; members can edit their works online and the leader can review the works and correct the works directly from the WML. Some of the announcements like plans or schedules may also be put on WML as read-only pages. For safety, it is suggested that when each new page is created, the page owner sets the page as read-only.

## 1.1 Purpose of This Project

The purpose of this project is to design, build and implement a fast communication assistant system for group work. All the pages and user information will be stored in a PostgreSQL database and retrieved by JAVA Servlet and JDBC. The main purpose of this project is to provide an easy-to-edit and web-based communication environment for the members and the leader in a group. Moreover, the system

offers the authorization function to make sure that the pages are secure from malicious editing. In the system, all the users can create any pages and delete their own pages easily.

## 1.2 Project Products

This project would lead to the following products:

- Implementation of Wiki Message Linking: a working web site with JSP programs, Java programs and PostgreSQL database, which would achieve the needs of a communication board of a group work. All browser requests will be checked by the security system in order to keep the information on the pages correct and secure.
- Users manual: an implementation manual was written for the user.
- Systems Manual: a project report (this report) is available with design details and specifications.

CHAPTER TWO

WIKI MESSAGE LINKING

ARCHITECTURE

This project, Wiki Message Linking (WML), implements a web system to provide an environment for the members and the group leader in a group to share information. Thus, the components needed to implement WML are a database server, a web server, graphical user interface components, and a database interface Application Programming Interface (API) to programmatically access the database. The following figure describes the interaction among the components used in WML.

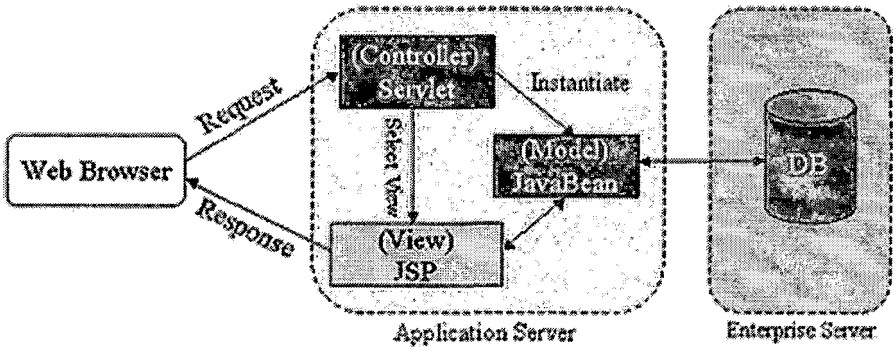


Figure 1. Wiki Message Linking Architecture

The components used to build WML 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., the 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) and Java Servlet. JSP was used because it can use javabeans, which provide a reusable way for all programs and java container, Tomcat can be installed under Windows or Linux. Also, it is easy to process whole user input from the HTML forms. The reason that Java Servlets were used is that it has the advantages of portability and efficiency. By using Java Servlets, the Servlet can be executed in any other web server which is the special property of Java Servlets, "write once, serve anywhere". Moreover, Java provides a convenient function, Java Database Connector (JDBC), to connect to the database.

The database used for WML is PostgreSQL. PostgreSQL is a real multi-user database and is royalty-free open source software. To use it, simply activate PostgreSQL in Linux. Also, the availability of the JDBC driver for PostgreSQL is another important reason to choose it. Moreover, the same

code could be used to link with another database by changing the JDBC driver, thereby making it database independent.

## 2.1 Software Interfaces

- Internet browser: Netscape or Internet Explorer.
- Operating system: Windows 98/Me/2000/XP or Unix/Linux.
- Database: PostgreSQL.
- Compiler: JDK 1.4.1.
- Language: HTML / JAVA / JavaScript / JSP.
- Database connector: JDBC.
- JSP Container/Web server: Jakarta Tomcat.

# CHAPTER THREE

# DATABASE DESIGN

## 3.1 Data Analysis

The data for designing and implementing the schema of the database depends on the properties of pages and users. The page data needed by the wiki page are title, body, ownerid, modified setting, and publish setting. The user data needed by the system are the user id, user name, password and e-mail address. The pages and users are connected by the relation of user id.

## 3.2 Database Schema Conceptual Model - Entity Relationship Diagram

In designing the schema for the WML database, two distinct parts have been identified. The first includes page part, which includes page's title, body, owner id, modified setting and publish setting. The second includes user id, user name, password and e-mail address. All the entities and attributes are detailed in Figure 2.

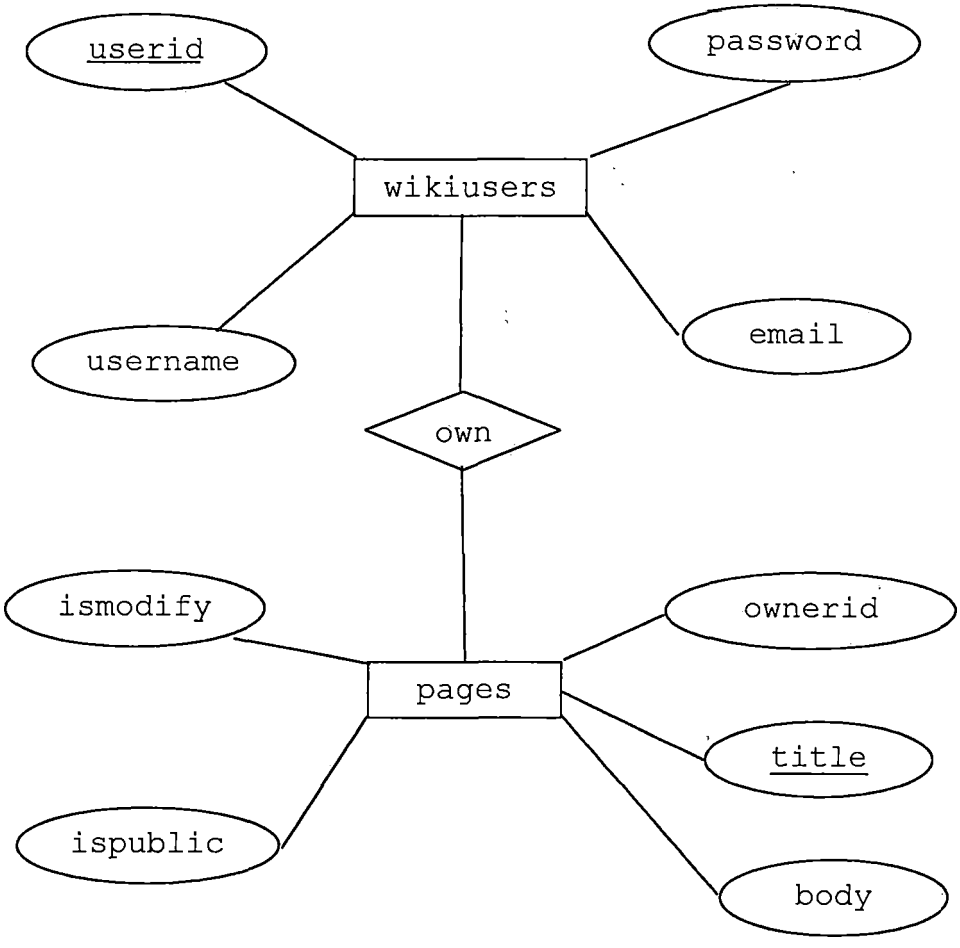


Figure 2. Entity Relationship Diagram

### 3.3 Database Schema Logical Model - Relational Schema

The conceptual model ER diagram maps into the following relational table design. In the following tables, underlined fields indicate the primary key.

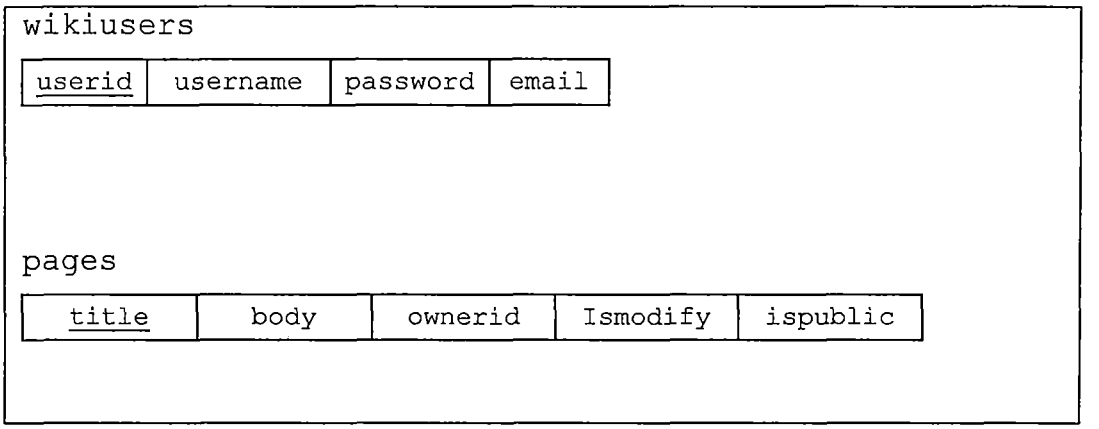


Figure 3. Wiki Message Linking Database Relational Schema

### 3.4 Data Type and Details

The logical model establishes the following detailed design in PostgreSQL database. The following tables describe data type, length, primary key, null or non-null keys.

Table 1. Structure of Table Wikiusers

| field | type | null | key | default | Extra |
|---|---|---|---|---|---|
| userid | varchar(30) | | PRI | | |
| username | varchar(30) | YES | | | |
| password | varchar(30) | | | | |
| email | varchar(30) | | | | |

Table 2. Structure of Table Pages

| field | Type | null | key | Default | Extra |
|---|---|---|---|---|---|
| title | varchar(255) | | PRI | | |
| body | Text | | | | |
| ismodify | varchar(1) | | | 1 | |
| ispublic | varchar(1) | | | 1 | |
| ownerid | varchar(255) | | | | |

CHAPTER FOUR

PROJECT IMPLEMENTATION

Wiki Message Liking is designed to perform 5 different functions for 3 different users. The following Figure 4 is the Use Case Diagram of this project.

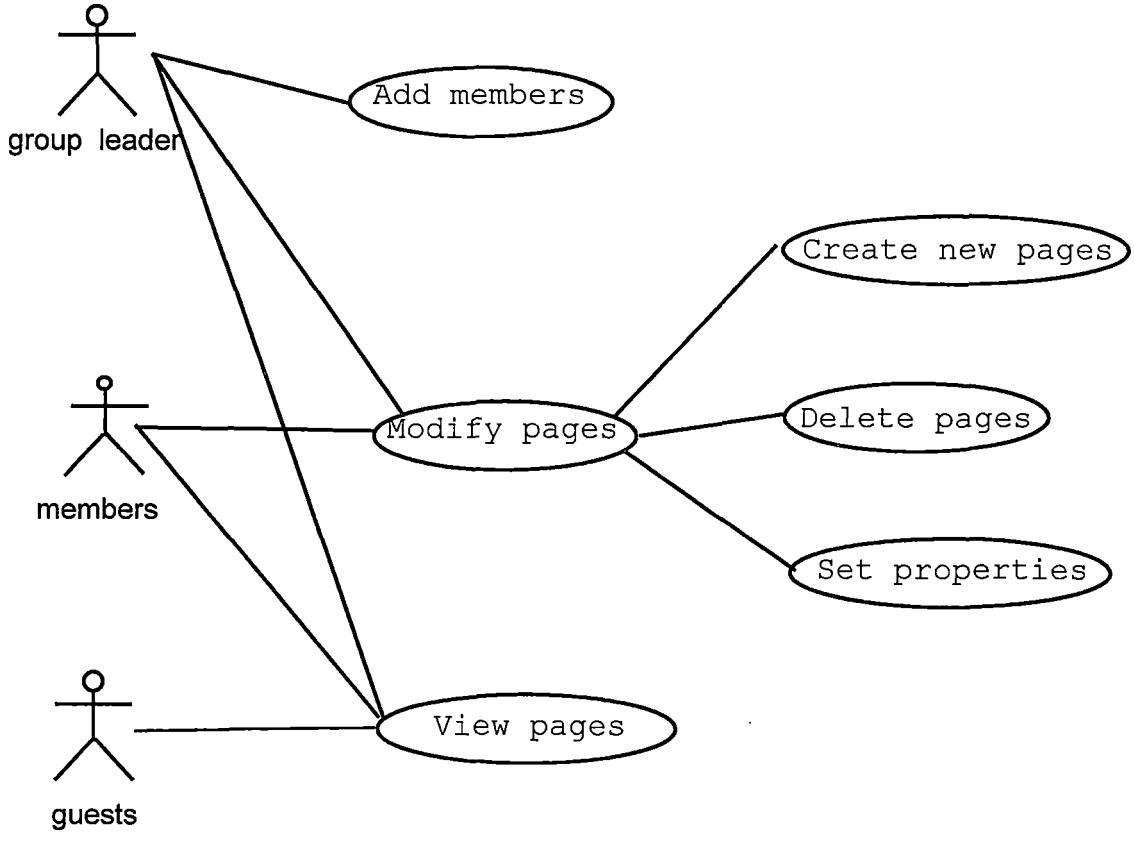


Figure 4. Use Case Diagram

## 4.1 Wiki Message Linking Graphical User Interface Design

Wiki Message Liking GUI is easy to use. The GUI is written using Java Server Page. All the functions that the user has are placed in the menu part which is the upper region of the layout page. And, the contents part is placed at the body part which is in the lower region of the layout page. All the inputs which are not acceptable by the system will be reported by an error message box. The following sub sections explain the GUI.

### 4.1.1 Wiki Message Linking Home Page

This page will be the first page that all the users will see when they enter WML. The home pages will be owned by the administrator (group leader) of WML and the properties of "visible by others" and "modifiable by users" will be set to be true as default value, which means everyone except guest can edit the page. The default content of the homepage will be empty. All of the users including guests can browse this page. In order to edit the page, click the "Edit" function in the menu at the top of the page then the Edit Servlet will forward to edit page. The user then can use html markup language to edit the page.

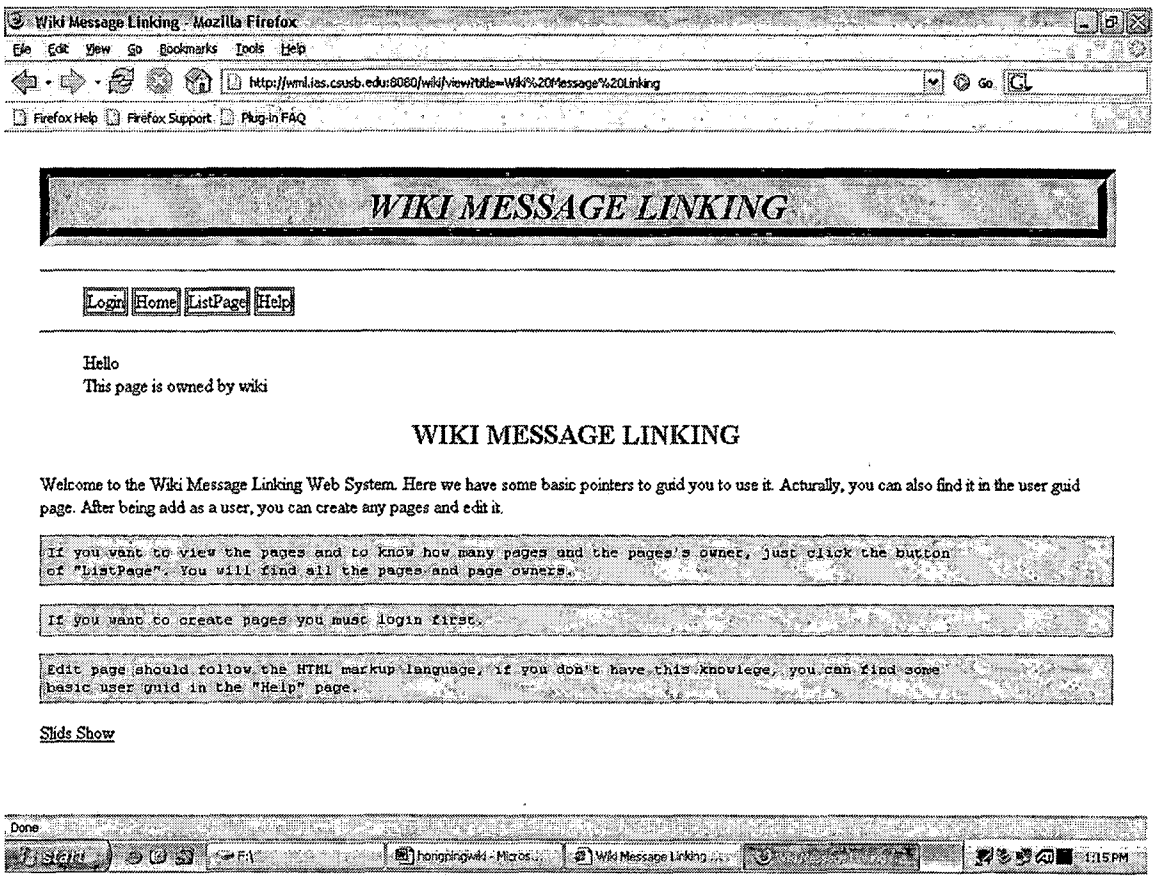


Figure 5. Home Page of Wiki Message Linking

To edit a page of the WML, the user must not be a guest and must have the privilege to edit the page. Otherwise, the user will not be given an edit link.

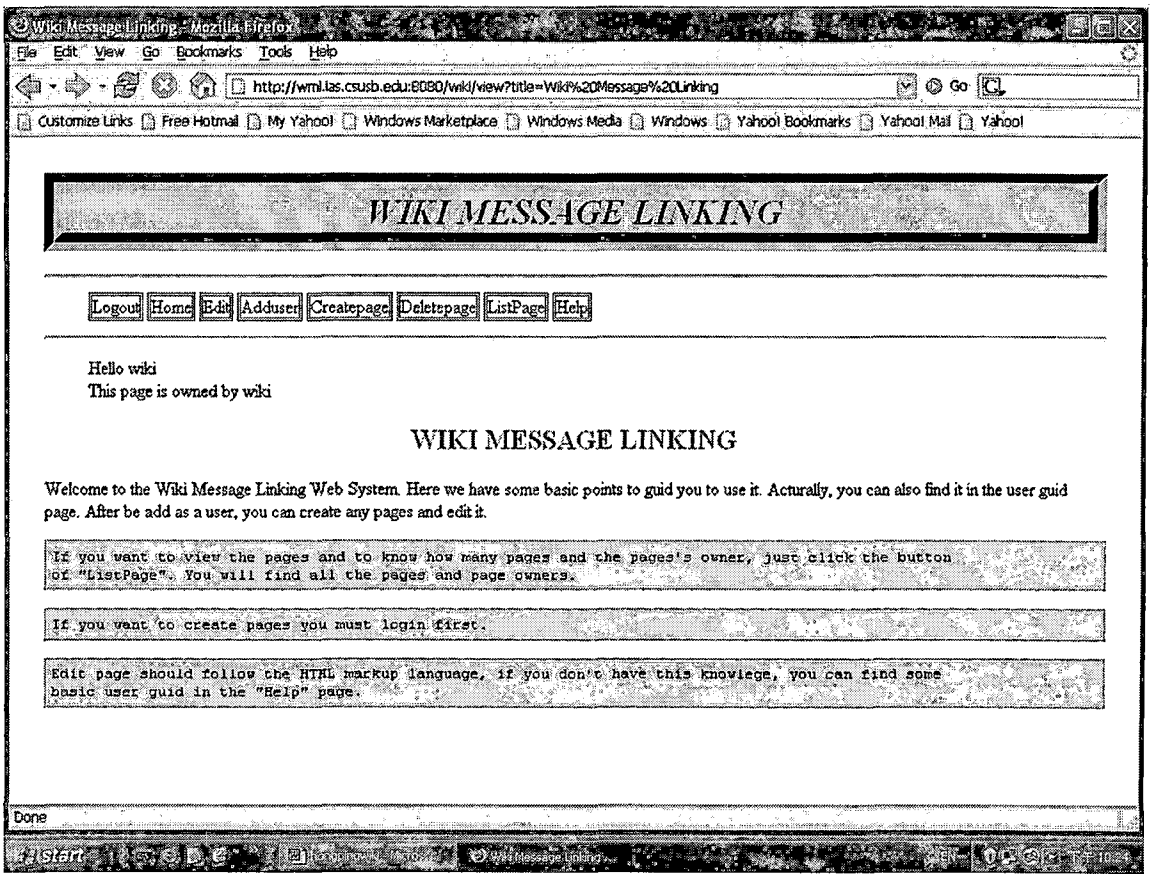


Figure 6. Home Page Login by the Group Leader

### 4.1.2 Login Page

The user logs in by providing a user id and a password that are created by the group leader (administrator) who is the administrator of the group. After the servlet verifies the user id and password, it director or forwards a request page for the user. Moreover, the user information will be saved in the session for later use, and the session will be killed when the browser is closed or when the user is idle for 1800 seconds (30 minutes) which is the default

session life time set up by Tomcat Server. The system will display different menu items based on the privileges granted to the user. If the user id or password is wrong, the program will show an error message and the user can re-login. For guests, there is no need to check the database. The users who do not login will be treated as guests who can only browse the pages of the system.

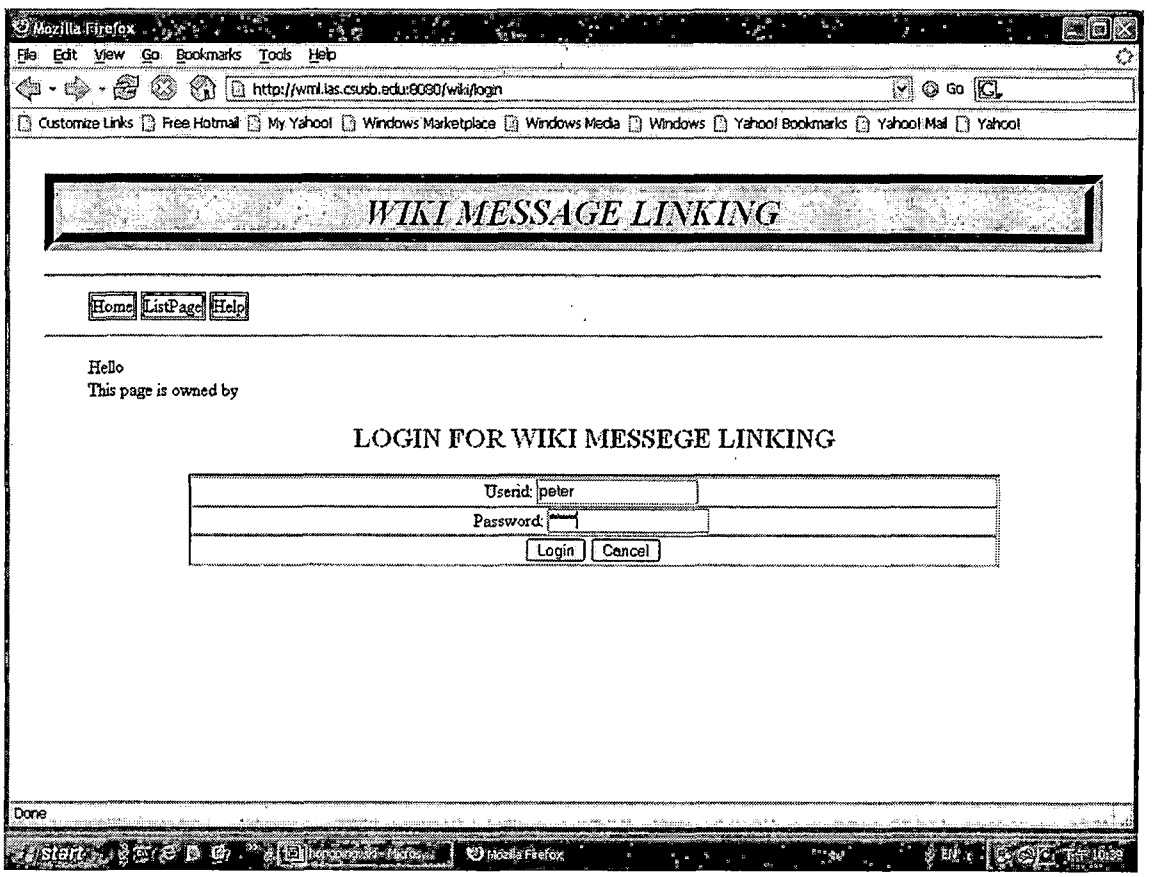


Figure 7. Login Page

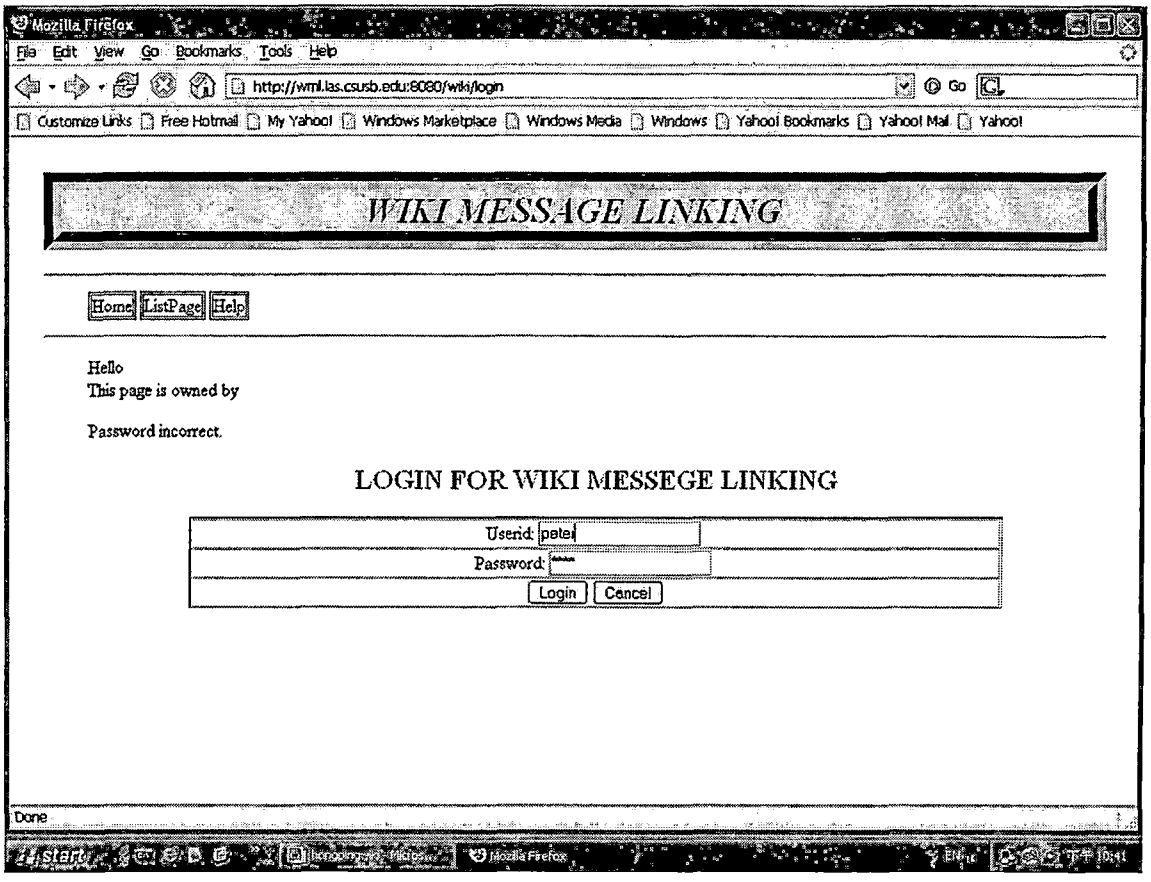


Figure 8. Login Error at Login Page

### 4.1.3 Add Users

A WML member is the UML user who has the privilege to modify wiki pages. The userid and password are created by the group leader and signed to the member. The group leader login first, then click the menu "Adduser", then the adduser page will show out, then he/she can input the userid and password for the new added user. The user must

be created by the administrator before the first login of a user.

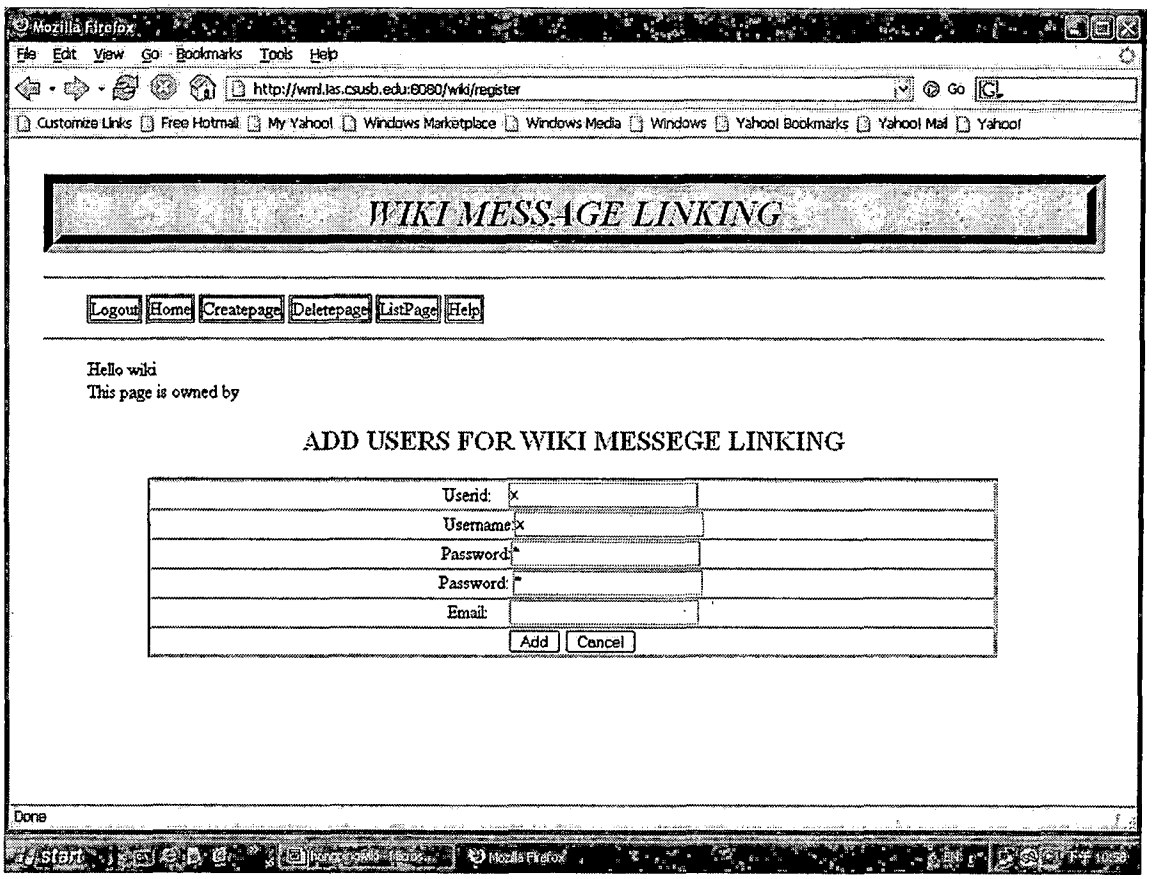


Figure 9. Group Leader Add Users for Wiki Message Linking

### 4.1.4 Edit Page

This is the most important page that the system has. To edit the page, browse to the page you want to edit and click on the link "edit" in the menu frame to access the edit page. There is one thing that users need to know. To edit a page, the user needs to be logged in and has the

privilege to edit the page. If the user who is editing the page is the owner of the page, there will be two more check boxes shown at the bottom of the text field named "modifiable by users" and "visible by others". These are the page attributes that can be set by the owner of the page to grant or deny to other users the ability to modify the page. The users also can get basic html mark up language knowledge in the help page to guid how to edit the page.

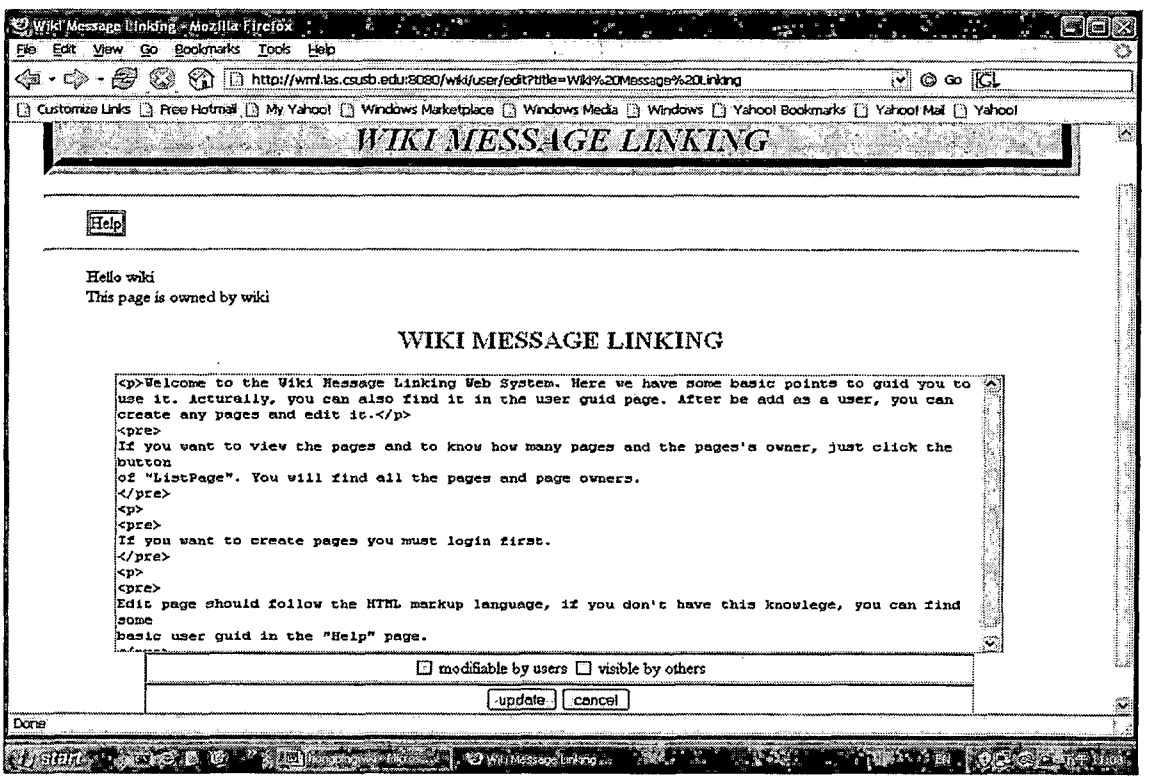


Figure 10. Edit Page

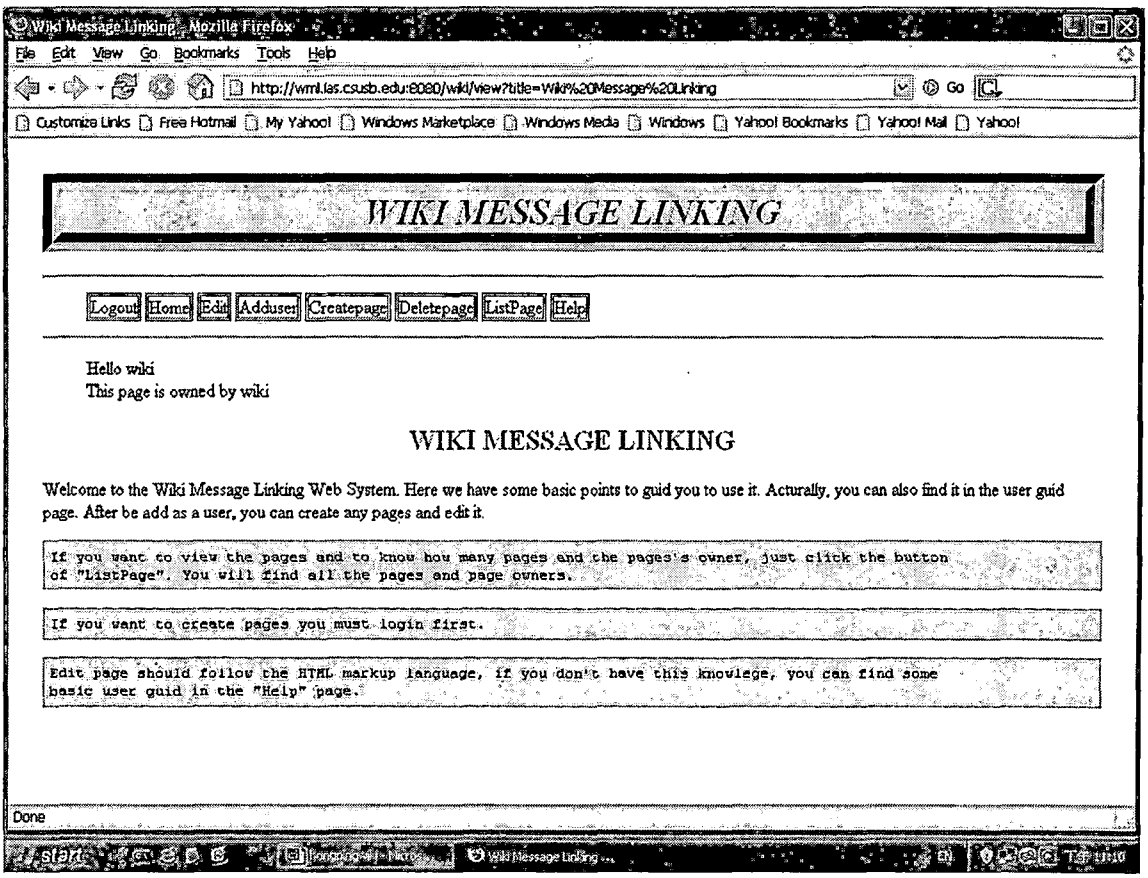


Figure 11. Page Content after the Edit in Figure 10

### 4.1.5 Create Pages

After login, member users can create pages which are owned by the login user. Click the menu "Createpage", the create page will show out, input the page title's name, then click "Add", a new page can be created. If the page is already exited, the system will remind you try to give another name for your new page.

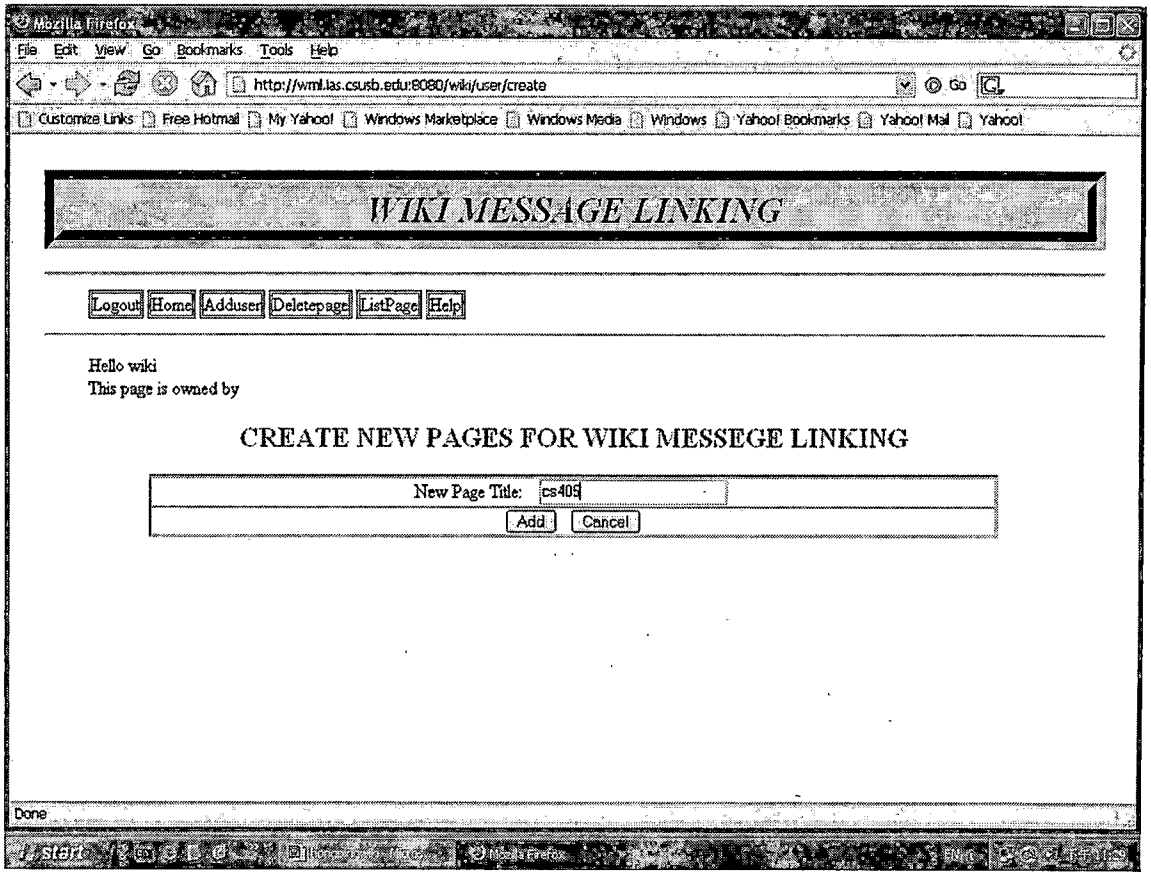


Figure 12. Member User Creates Page for Wiki Message Linking

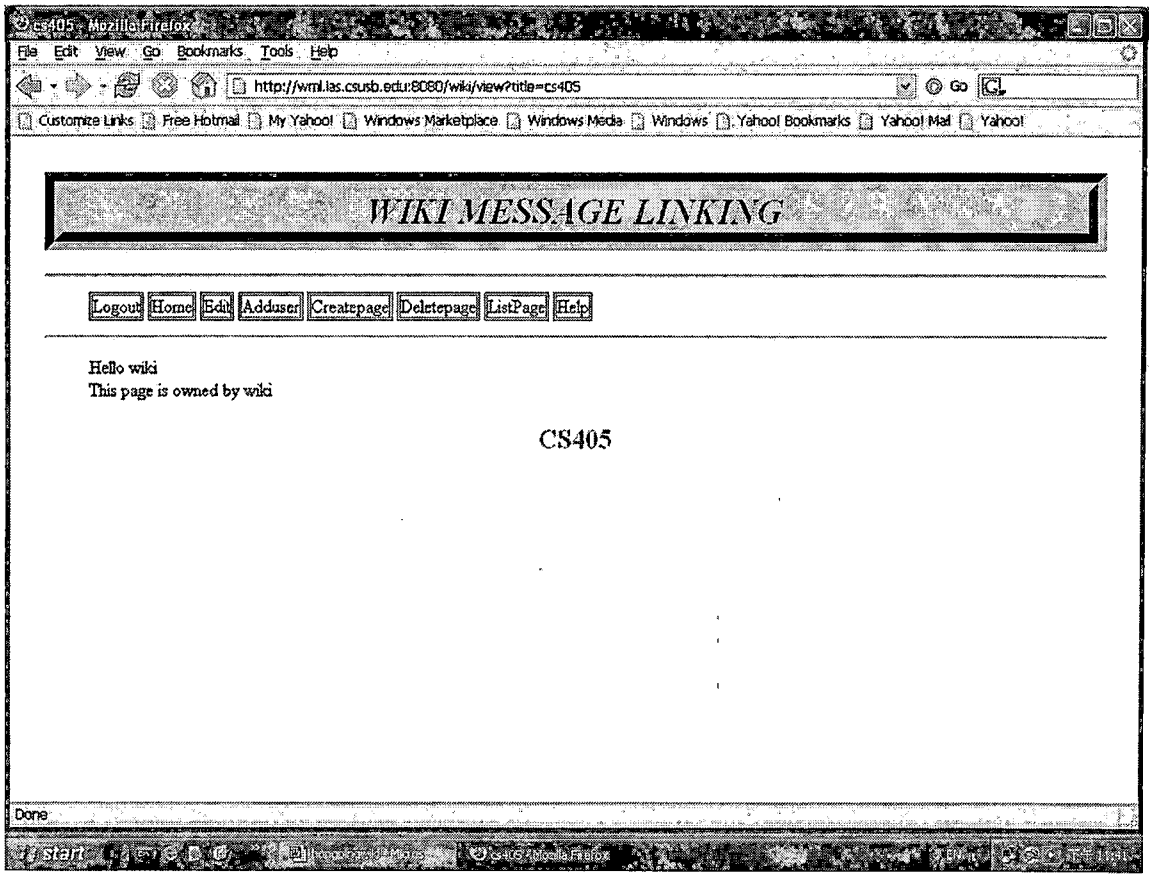


Figure 13. User "Wiki" Just Creates A New Page "Cs405"

### 4.1.6 Delete Pages

Any exiting pages can be deleted by the group leader and the page owner itself. Other users can not delete the page which is not belonged to them. After login, click the "Deletepage" menu, the delete page will show out, then input the page title's name which you want to delete, and click "send" button, the existing page is being deleted.

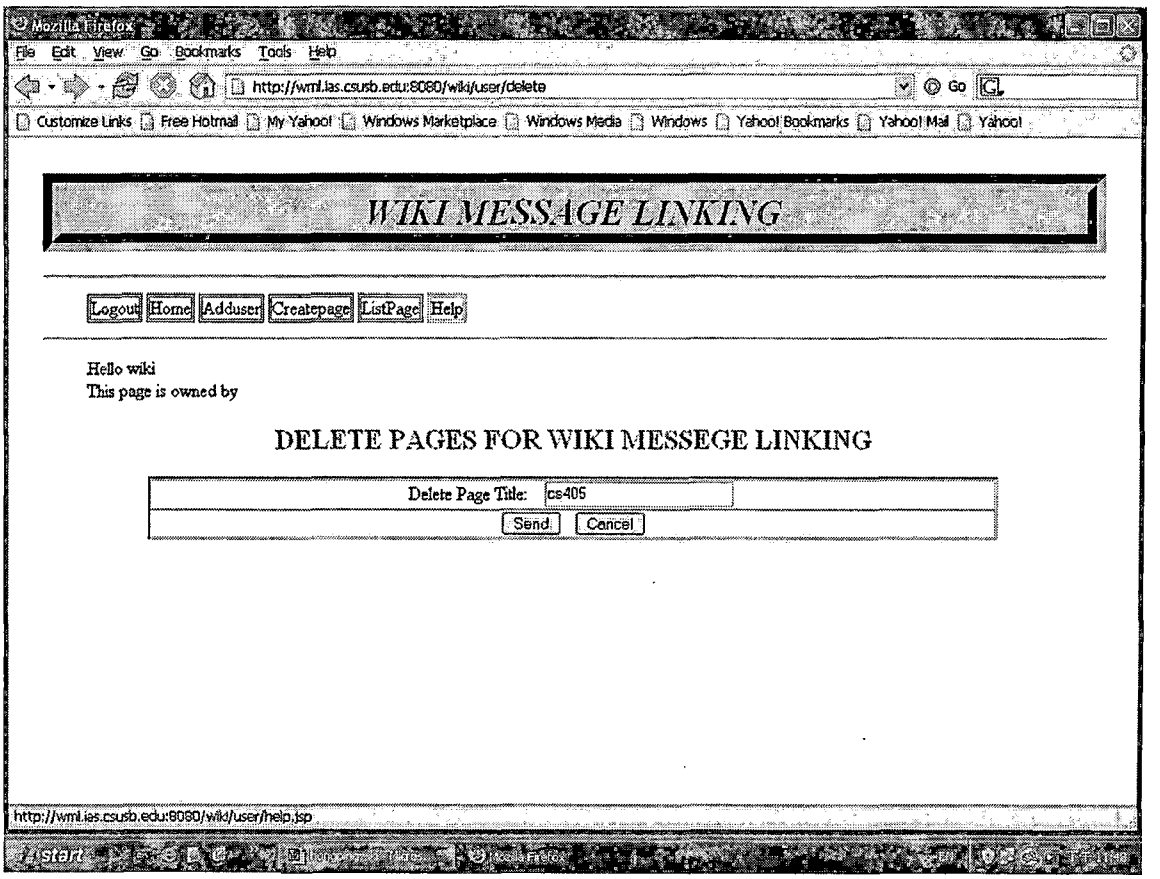


Figure 14. User "Wiki" Will Delete the Page "Cs405"

### 4.1.7 List Pages

This is a convienent way for all the users to check out there are how many pages and their owners. On this page, all the pages and their owners are listed in a table, and there are links for the pages. If you want to view the pages, just click the page link and the page will show out.

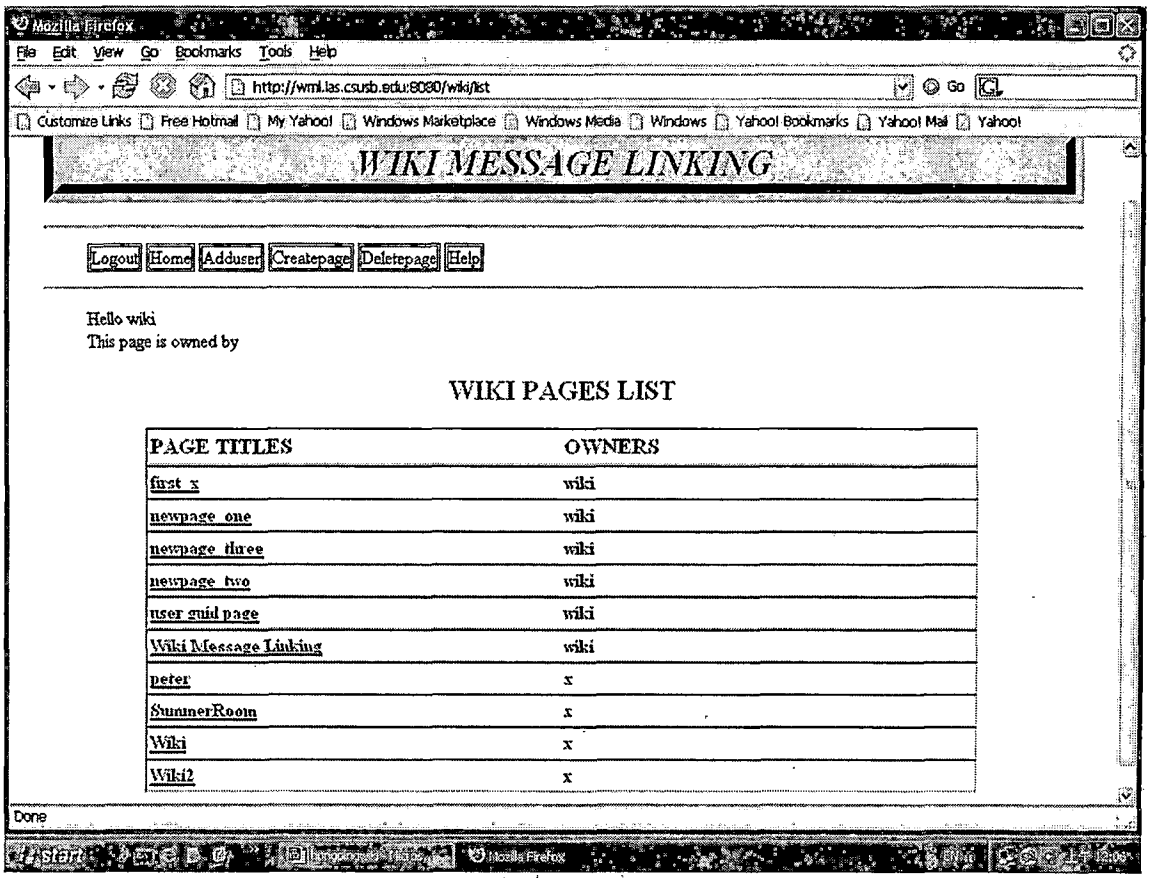


Figure 15. All the Pages and Their Owners of Wiki Message Linking

### 4.1.8 Help Page

In order to help users know more about the system, the system also offers the help page for all users including the guest users. The help page tells the users how to use the system and also shows the users how to use the mark up language to edit the wiki page. To view the help page, click the link "Help" in the menu, then the help page will display.

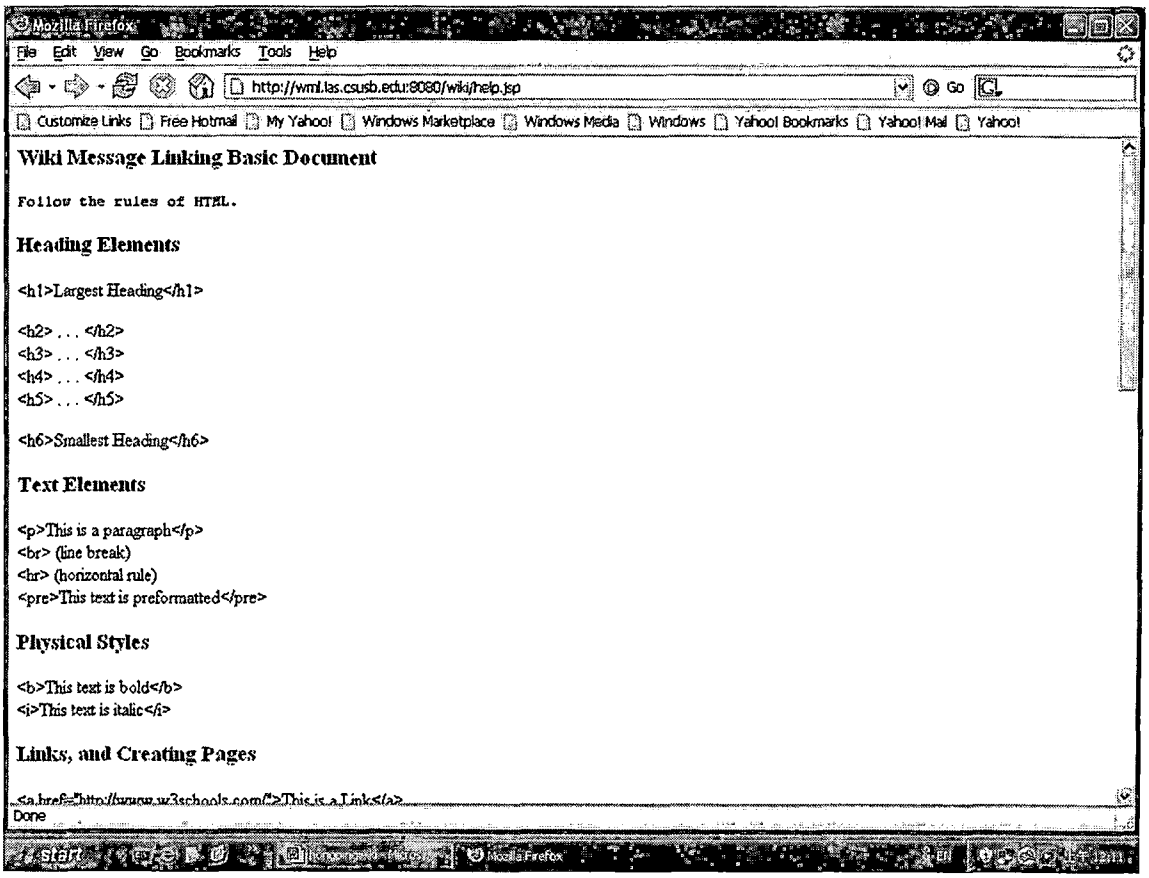


Figure 16. Help Page

# CHAPTER FIVE

# SYSTEM VALIDATION

The system validation test is a kind of test process which can ensure that our program meets the expectation of the user. The purpose of the system validation is to provide a high degree of assurance that a specific process will consistently produce a result which meets predetermined specifications and quality attributes. This can also guarantee the system performance and reliability.

## 5.1 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, etc. The unit testing results of WML are shown in Table 3.

Table 3. Unit Test Results (GUI)

| Forms | Tests Performed | Results |
|---|---|---|
| Home Page | • Check all the menus shown properly by the user.<br>• Check all the links work as expected. | Pass |
| Login Page | • Check all the button work properly.<br>• Verify the page can get the error message and work properly by the | Pass |

| Forms | Tests Performed | Results |
|---|---|---|
| | message. | |
| Logout Page | • Check all the button work properly.<br>• Verify the user remove from session after logout.<br>• Check the page redirect to proper page after logout. | Pass |
| Edit Page | • Verify the set properties work properly.<br>• Check the edit using html work properly.<br>• Check all the button work properly. | Pass |
| Create Page | • Verify page is created correct.<br>• Verify the error messages show out when the page already exist in the system. | Pass |
| Delete Page | • Verify page is deleted correct.<br>• Verify the error messages show out when the invalid data are input. | Pass |
| List Pages | • Verify all the pages and their ownerid are listed as expected.<br>• Check all the pages links work properly.<br>• Check all the menu work correct. | Pass |
| Help Page | • Verify the context in the help page is correct.<br>• Make sure the links work properly. | Pass |

Table 4. Unit Test Results (Class: Servlets)

| Forms | Tests Performed | Results |
|---|---|---|
| ParentServlet | • Make sure servlet is init and layout page is init as RequestDispatcher. | Pass |
| RegisterServlet | • Make sure it checks if the user going to create is exist.<br>• Make sure it is stored in database correctly. | Pass |
| LoginServlet | • Make sure it checks the users when login. Error message will give out if try to login with non-member users. | Pass |
| LogoutServlet | • Make sure the session is closed when user logout. | Pass |
| ViewServlet | • Make sure correct jsp page is forward.<br>• Menu bar work properly | Pass |
| CreateServlet | • Make sure correct jsp page is forward.<br>• Make sure only group leader can do this. | Pass |
| DeleteServlet | • Make sure correct jsp page is forward.<br>• Only group leader and page owner can do this. | Pass |
| EditServlet | • Make sure correct jsp page is forward.<br>• Make sure html work properly. | Pass |
| UpdateServlet | • Make sure correct jsp page is forward.<br>• Make sure the update date stored in database properly. | Pass |
| ListPageServlet | • Make sure correct jsp | Pass |

| Forms | Tests Performed | Results |
|---|---|---|
| | page is forward.<br>• All pages links work properly. | |

## 5.2 System Testing

System testing is the testing process that uses real data, which the system is intended to manipulate, to test the system. Test the system by using a variety of data to see the overall result.

System testing of WML system begins with the following steps:

Table 5. System Test Results

| System Testing | Results |
|---|---|
| 1. Install WML system into server. | Pass |
| 2. Start up all servers such as Tomcat server, PostgreSQL database server. | Pass |
| 3. Running testing by using real data on all forms and reports. | Pass |

# CHAPTER SIX

# MAINTENANCE MANUAL

It is very important to have a maintenance manual with a system no matter how small the system is. The maintenance manual records any information that can be used to setup the system or backup the system. In order to make sure the system works smoothly and meets the expectation of the users, it is very important to follow the maintenance manual step by step carefully. In Wiki Message Linking, there are 3 major issues: Software Installation, Variable Installation, and Wiki Message Linking Installation.

## 6.1 Software Installation

In WML, it requires RedHat, PostgreSQL, JSDK, Ant, TOMCAT, JSTL, and JDBC to run the programs. Following will detail the installation of those softwares.

### 6.1.1 RedHat Installation

RedHat is a linux base operating system which is offerd freely and be downloaded from internet. The reason we choose RedHat is it offers better performance than a Microsoft operating system. Following are the steps to install RedHat onto your machine.

1. Download a latest version of the RedHat operating systems from

http://ftp.redhat.com/pub/redhat/linux/9/en/iso/i386/ and burn the iso files into CDs.

2. Install the operating system by inserting CD 1 into the CD-ROM and start up the machine which is going to install the operating system.
3. Select Disk Druid to manually configure partition. Create swap(256MB), /boot(90MB) and /(all remaining storage) three disk partitions.
4. Use GRUB boot loader and set no GRUB password.
5. Network configuration, turn off bootp/dhcp, keep on "activate on boot" and set the server IP Address:139.182.137.64, Netmask:255.255.252.0, Gateway:139.182.136.1, DNS:139.182.136.136/139.182.137.137/139.182.2.1
6. After all the necessary files are copied into the computer and install it, the machine will restart and Redhat is installed.
7. Firewall configuration, open the file /etc/sysconfig/iptables, opens port 22 for ssh daemon, port 8080 for tomcate server. Restart the daemon "/sbin/service iptables restart" in order for to take effect.
8. Network parameters, open the file /etc/hosts, add 139.182.137.44 wml.ias.csusb.edu. Restart the

network daemon "/sbin/service network restart" to make all these changes effective.

9. Create user account, do "/user/sbin/useradd hoyang", "passwd hoyang"

### 6.1.2 PostgreSQL Installation

PostgreSQL is the database system we use in WML; it is preconfigured under Redhat linux to run as user postgres. The reason that we choose PostgreSQL as WML's database system is because it also provides a JDBC driver to easily connect from a JAVA program, thus it's a good choice for designing this project.

To install PostgreSQL, follow the following steps:

1. Because PostgreSQL may install on to RedHat when the operating system is installed, the first thing we have to do is to check if the PostgreSQL is already in the operating system. Using the command to check if PostgreSQL exist in the operating system:

   rpm -q postgresql

   If PostgreSQL is not installd in the operating system, then use rpm to install it.

2. In order to create a database user, "wiki", and database, "wiki" for WML use, have the following commands executed.

```
su postgres
initdb -D /var/bin/pgsql/data
createuser wiki
createdb wiki
```

where at the first command, the postgres is the default user for PostgresSQL. Starting the database by using the command "initdb" with the directory "/var/bin/pgsql/data/" which is the default database directory will start up the database system. Login postgres as the supervisor and create a database user, "wiki", and the database "wiki".

3. There are still some steps needed to setup the environment values.

In the user's environment setup file /etc/profile.d/*.sh, add the following line:

```
export PGDATA=/var/lib/pgsql/data
```

Open the file /var/lib/pgsql/data/postgresql.conf and uncomment the line:

```
tcpip_socket = true
```

In order to have the database system starup at the system start, have the command executed:

```
/sbin/chkconfig --level 3 postgresql on
```

And, the last step is to startup the database system immediately now without restart the system:

/sbin/service postgresql start

After having the steps above executed, the database system is ready to go and now we have to install JAVA platform, JAVA 2 Platform, Standard Edition (J2SE).

### 6.1.3 JAVA 2 Platform, Standard Edition (J2SE)

J2SE is the compiler program for JSP and JAVA Servlet 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 Linux (all languages, including English) to the directory /usr/java, then execute the following commands:

```
chmod +x j2sdk-1_4_2_01-linux-i586-rpm.bin
./j2sdk-1_4_2_01-linux-i586-rpm.bin
rpm -ivh j2sdk-1_4_2_01-linux-i586.rpm
```

And, set the environment variables in the file /etc/profile.d/*.sh:

```
JAVA_HOME=/usr/java/j2sdk1.4.2_01
PATH=${PATH}:${JAVA_HOME}/bin
Export JAVA_HOME
```

6.1.4 Tomcat

TOMCAT is one of the apache jakarta projects, which is a web container to process JSP and JAVA Servlet programs, and to serve static web pages. At the tomcat's official download ftp server at http://ftp.epix.net/apache/jakarta/tomcat-4/v4.1.27-beta/bin/ to download the file of tomcat server for linux jakarta-tomcat-4.1.27.tar.gz to /usr/java/ and extract it to the hard drive.j

tar -xzvf jarkata-tomcat-4.1.27.tar.gz

Also, we modify the file /usr/java/jarkata-tomcat-4.1.27/conf/server.xml by add the following setting in the file:

```
<Service name="Tomcat-Standalone">
  <Connector
className="org.apache.catalina.connector.http.HttpConnector
"
             port="8080"
             minProcessors="5"
             maxProcessors="75"
             enableLookups="false"
             redirectPort="8443"
             acceptCount="10"
             debug="0"
```

```
                connectionTimeout="60000" />
        <Engine name="Standalone" defaultHost="localhost"
debug="0">
          <Host name="localhost"
                debug="0"
                appBase="webapps"
                unpackWARs="true"
                autoDeploy="true"
                deployXML="true"
                liveDeploy="true">
            <Context path="" docBase="ROOT" debug="0" />
          </Host>
        </Engine>
      </Service>
    </Server>
```

And, set the environment variable by adding the following lines in the file /etc/profile.d/myenv.sh:

```
CATALINA_HOME=/usr/java/jarkata-tomcat-4.1.27
PATH=${PATH}:${JAVA_HOME}/bin:${CATALINA_HOME}/bin
export CATALINA_HOME
```

Add the following lines in the file /etc/rc.local to have the tomcat run when the system boots:

```
Export JAVA_HOME=/usr/java/j2sdk1.4.2_01
export CATALINA_HOME=/usr/java/jarkata-tomcat-4.1.27
```

${CATALINA_HOME}/bin/startup.sh

### 6.1.5 Install Ant

Go to an.apache.org download apache-ant-1.5.4-bin.tar.gz file. tar -zxvf apache-ant-1.5.4-bin.tar.gz. Modify /etc/profile.d/myenv.sh:

ANT_HOME=/usr/java/apache-ant-1.5.4,

PATH=${ANT_HOME}/bin,

Export ANT_HOME

### 6.1.6 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 PostgreSQL, and information about this can be obtained at the PostgreSQL homepage at http://jdbc.postgresql.org/download/. Download pg73b1jdbc3.jar and copy the file to /usr/java/jakarta-tomcat-4.1.17/common/lib/.

## 6.2 Variables Modification

In WML, we have to change some environment variables in the linux system and server.xml in Tomcat server configuration directory.

### 6.2.1 System Variables

1. Open the file "server.xml" in the directory "/usr/java/jakarta-tomcat-4.1.18/conf" via "vi" or any other editor.
2. Scroll down until you see the context area we added in at chapter 7.4.1.
3. The variable "path" in Context indicates the context path of the web application. The default value would be "/wiki".
4. The variable "docBase" in Context is the files directory for the web application. The default value would be "/pub/wiki".
5. The variable "variable" in Logger is the absolute or relative pathname of a directory in which log files created by this logger will be placed. The default value would be "/pub/wiki/logs".
6. Now, let's look down at the parameter setting for WML.

7. The parameter "contextPath" indicate the context path for WML which would be the same as the value of path.

8. The parameter "homepage" sets the name of the home page of WML. The default value would be "Wiki Message Linking".

9. The parameter "username" is the user name who can access the database system. Usually, this value would be the administrator of WML.

10. The parameter "password" is the password corresponding to the user name at 9. If there is no special setting in database system, leave the value to be empty.

## 6.3 Wiki Message Linking Installation/Migration

1. All the JSP programs and HTML programs are stored in

   /hoyang/wiki/web/WEB-INF/jsp

2. All the classes are stored in

   /hoyang/wiki/WEB-INF/classes

3. Place the web.xml for WML in

   /hoyang/wiki/web/WEB-INF

## 6.4 Backup and Restore

Backup is a very important action needed for any system to prevent losing data. No one can say a system works very well and will never have a problem. So, now let's talk about the backup process for WML. There are two steps to back up WML. One is to backup the system files. The other step is to backup the database which is used by WML.

### 6.4.1 System Backup

All the WML system files are located in the directory "/hoyang/wiki" and all its subdirectory. Thus, in order to backup the system files, all we need to do is to backup the files in the directory. The method here I suggest is to compress the directory of "/hoyang/wiki" including its subdirectory to compress files for future use by the compress program "tar". Using the following command to backup the system files:

```
tar -cf WML.tar /hoyang/wiki
```

### 6.4.2 Database Backup

To backup the database system, we use pg_dump command to backup the database used by WML. The following command is used to backup the database:

```
pg_dump wiki | gzip > WML.zip
```

After executing the backup command above, the file WML.zip would be the backup file of the database.

### 6.4.3 System Restore

To restore the system file, simply extract the backup file by using the following command:

```
tar -xzvf WML.tar /
```

By the command above, all the WML system files will restore into the directory /hoyang/wiki and complete the restore system process.

### 6.4.4 Database Restore

To restore the database needed for the WML, go to the directory where your database backup file is in, and execute the following commands:

```
createdb wiki
gunzip -c WML.zip | psql WML
```

After the commands are executed, the database is restored to the database system. Then, restart tomcat, the WML will be completely restored.

# CHAPTER SEVEN

# CONCLUSION AND FUTURE DIRECTIONS

## 7.1 Conclusion

Wiki Message Liking provides a very good communication environment for the group communication. For the group leader, WML offer a very good environment to post the announcement for group. For the members, WML provides a good environment that all the members can share any ideas or problems they have on WML. The most important of all, WML use html language to edit pages.

Wiki Message Liking offers a page security function. Only the user who has the privilege to modify the page can edit the page. By this function, all the pages can prevent being edited by malicious editing. Thus, WML is definitely a good system which will help the group leader and the members have a better communication environment.

## 7.2 Future Directions

Wiki Message Liking is a system offered to be used for any groups. The system is not created to be used for any specific group, so any people can use the system as its group communication board. In the future, if there is any function needed for the system, the changes needed for the system is in the class of Servlet. Add some new Servlets

with some new functions are fairly easy. The JSP page style is controlled by CSS style sheet, just edit the CSS sheet, then the page style will be changed simply.

Because the system is widely accepted, the system can be used as a general purpose communication board. Simply speaking, this is a collaborative tool that can be used in many contexts.

# APPENDIX A

# WIKI MESSAGE LINKING CLASSES

# SOURSE CODE

```
package wiki;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.IOException;
import java.util.Vector;
public class RegisterServlet extends ParentServlet
{
   protected void doGet(HttpServletRequest request, HttpServletResponse
                        response) throws ServletException, IOException
   {
      // Contruct menu of operations.
      Vector menu = new Vector();
      menu.add(new MenuItem("Logout", "/logout"));
      menu.add(new MenuItem("Home", "/view?title=Wiki Message Linking"));
      menu.add(new MenuItem("Createpage", "/user/create"));
      menu.add(new MenuItem("Deletepage", "/user/delete"));
      menu.add(new MenuItem("ListPage", "/list"));
      menu.add(new MenuItem("Help", "help.jsp "));
      request.setAttribute("menu", menu);
      request.setAttribute("body", "/WEB-INF/jsp/register.jsp");
      layoutPage.forward(request, response);
   }
   protected void doPost(HttpServletRequest request, HttpServletResponse
                        response)  throws ServletException, IOException
   {
      HttpSession session = request.getSession();
      String title = session.getAttribute("title").toString();
      if (request.getParameter("cancel") != null){
          //cancel button clicked.
          response.sendRedirect(Constants.viewUrl + title);
          return;
      }
      String userid = request.getParameter("userid");
      if (userid == null || userid.length() == 0) {
         request.setAttribute("error", "User ID is a required.");
         doGet(request, response);
         return;
      }
      String password = request.getParameter("password");
```

```
        if (password == null) password = "";
        String password2 = request.getParameter("password2");
        if (password2 == null) password2 = "";
        if (!password.equals(password2)) {
           request.setAttribute("error", "Passwords don't match.");
           doGet(request, response);
           return;
        }
        String username = request.getParameter("username");
        if (username == null) username = "";
        String email = request.getParameter("email");
        if (email == null) email = "";
        User user = null;
        try {
           user = User.create(userid, username, password, email);
        } catch (Exception e) {
           throw new ServletException(e.toString());
        }
        if (user == null) {
           request.setAttribute("error", "User ID already exists, please
                                 try another.");
           doGet(request, response);
           return;
        }
       // Contruct menu of operations.
        Vector menu = new Vector();
        menu.add(new MenuItem("Help","/help.jsp "));
        request.setAttribute("menu", menu);
        session.setAttribute("user", user);
        session.setAttribute("userid", userid);
        String reply = "Hello" + userid;
        session.setAttribute("reply", reply);
        response.sendRedirect(request.getContextPath() + "/login");

    }
}
```

```
package wiki;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.IOException;
import java.sql.SQLException;

public class ParentServlet extends HttpServlet
{
   RequestDispatcher layoutPage;
   public void init() throws ServletException
   {
      HttpSession session;
      ServletContext context = getServletContext();
      layoutPage = context.getRequestDispatcher("/WEB-
                                    INF/jsp/layout.jsp");
      if (layoutPage == null) {
      throw new ServletException("/WEB-INF/jsp/layout.jsp not found");
      }
   }
   protected Page findPage(String title)  throws ServletException
   {
      Page pg = null;
      try {
        pg = Page.find(title);
      } catch (SQLException e){
        throw new ServletException(e);
      }
      return pg;
   }
}
```

```
package wiki;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.IOException;
import java.util.*;
import java.util.Vector;
public class LoginServlet extends ParentServlet
{
   protected void doGet(HttpServletRequest request, HttpServletResponse
                        response) throws ServletException, IOException
   {
      // Contruct menu of operations.
      Vector menu = new Vector();
      menu.add(new MenuItem("Home", "/view?title=Wiki Message Linking"));
      menu.add(new MenuItem("ListPage", "/list"));
      menu.add(new MenuItem("Help","/help.jsp "));
      request.setAttribute("menu", menu);
      request.setAttribute("body", "/WEB-INF/jsp/login.jsp");
      layoutPage.forward(request, response);

   }

   protected void doPost(HttpServletRequest request, HttpServletResponse
                        response) throws ServletException, IOException
   {
      HttpSession session = request.getSession();
      String title = session.getAttribute("title").toString();
      if (request.getParameter("cancel") != null){
          //cancel button clicked.
          response.sendRedirect(Constants.viewUrl + title);
          return;
      }
      String userid = request.getParameter("userid");
      if (userid == null) {
         request.setAttribute("error", "User ID is a required.");
         doGet(request, response);
         return;
      }
      String password = request.getParameter("password");
```

```
        if (password == null) password = "";

        User user = null;
        try {
           user = User.find(userid);
        } catch (Exception e) {
           throw new ServletException(e.toString());
        }
        if (user == null) {
           request.setAttribute("error", "User ID doesn't exist.");
           doGet(request, response);
           return;
        }
        if (!user.getPassword().equals(password)) {
           request.setAttribute("error", "Password incorrect.");
           doGet(request, response);
           return;
        }

        session.setAttribute("user", user);
        session.setAttribute("userid", userid);
        Page pg=findPage(title);
        if(pg==null)
        response.sendRedirect(request.getContextPath() + "/view?title=" +
                              Constants.homePageTitle);
        else
        response.sendRedirect(request.getContextPath() + "/view?title=" +
                              title);
    }
}
```

```
package wiki;
import javax.naming.*;
import java.sql.*;
import javax.sql.DataSource;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;

public class User
{
   static private DataSource ds;
   static
   {
      try {
         InitialContext ic = new InitialContext();
         Context tomcatContext = (Context) ic.lookup("java:comp/env");
         ds = (DataSource) tomcatContext.lookup("database");
         if (ds == null) throw new RuntimeException("no DataSource");
      } catch (Exception e) {
         throw new RuntimeException(e);
      }
   }
   private String userid;
   private String username;
   private String password;
   private String email;
   public User(String userid, String username, String password, String
               email)
   {
      this.userid = userid;
      this.username = username;
      this.password = password;
      this.email = email;
   }

   public String getUserid() { return userid; }
   public String getUsername() {return username;}
   public String getPassword() { return password; }
   public String getEmail() {return email;}
```

```
/**
 * @return null if userid doesn't exist
 */
public static User find(String userid)
throws SQLException
{
   User user = null;
   Connection connection = null;
   PreparedStatement ps = null;
   ResultSet rs = null;
   try {
      connection = ds.getConnection();
      String sql =   "select * from " +
                     "wikiusers" +
                     " where userid = ?";
      ps = connection.prepareStatement(sql);
      ps.setString(1, userid);
      rs = ps.executeQuery();
      if (!rs.first()) return null;
      String username = rs.getString("username");
      String password = rs.getString("password");
      String email = rs.getString("email");
      user = new User(userid, username, password, email);
   } finally {
      if (rs != null) rs.close();
      if (ps != null) ps.close();
      if (connection != null) connection.close();
   }
   return user;
}
/**
 * @return null if userid already exists in users table.
 */
public static User create(String userid, String username, String
                    password, String email) throws SQLException
{
   if (userid == null) return null;
   if (password == null) return null;
```

```
        User user = null;
        Connection connection = null;
        PreparedStatement ps = null;
        try {
           connection = ds.getConnection();
           String sql =   "insert into " +
                          "wikiusers" +
                          " (userid, username, password, email) values
                           (?, ?, ?, ?)";
           ps = connection.prepareStatement(sql);
           ps.setString(1, userid);
           ps.setString(2, username);
           ps.setString(3, password);
           ps.setString(4, email);
           int rs = ps.executeUpdate();
           if (rs == 0) return null;
        } catch (SQLException e) {
           // We assume the exception is due to user already existing
           return null;
        } finally {
           if (ps != null) ps.close();
           if (connection != null) connection.close();
        }
        return new User(userid, username, password, email);
     }

}
```

```
package wiki;

public class Constants
{
   public static final String jndiContainerContext = "java:comp/env";
   public static final String jndiDatabaseName = "database";
   public static final String userTableName = "wikiusers";
   public final static String homePageTitle = "Wiki Message Linking";
   public final static String viewUrl =
"http://wml.ias.csusb.edu:8080/wiki/view?title=";
}
```

```
package wiki;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.ServletContext;
import javax.servlet.RequestDispatcher;
import javax.servlet.http.*;
import java.util.Vector;
public class CreateServlet extends ParentServlet
{
   protected void doGet(HttpServletRequest request, HttpServletResponse
                        response) throws ServletException, IOException
   {// Contruct menu of operations.
      Vector menu = new Vector();
      menu.add(new MenuItem("Logout", "/logout"));
      menu.add(new MenuItem("Home", "/view?title=Wiki Message Linking"));
      menu.add(new MenuItem("Adduser", "/register"));
      menu.add(new MenuItem("Deletepage", "/user/delete"));
      menu.add(new MenuItem("ListPage", "/list"));
      menu.add(new MenuItem("Help", "help.jsp "));
      request.setAttribute("menu", menu);
      request.setAttribute("body", "/WEB-INF/jsp/create.jsp");
      layoutPage.forward(request, response);
   }
   protected void doPost( HttpServletRequest request,
                          HttpServletResponse response)
                          throws ServletException, IOException
   {
     HttpSession session = request.getSession();
     String title = request.getParameter("title");
     if (title == null) {
        request.setAttribute("error", "CreateServlet: title parameter
                             missing");
        doGet(request, response);
        return;
     }
     if (request.getParameter("cancel") != null){
        //Cancel button clicked.
        response.sendRedirect(Constants.viewUrl +
         Constants.homePageTitle);
```

```
            return;
        }
        Page pg = findPage(title);
        if (pg != null){
            request.setAttribute("error", "CreateServlet: The page is
                                already created, please try another one");
            doGet(request, response);
            return;
        }
        String userid = session.getAttribute("userid").toString();
        try {
            Page.createPage(title, userid);
        } catch (Exception e) {
            throw new ServletException(e);
        }
        User user = (User)session.getAttribute("user");
        session.setAttribute("user", user);
        request.setAttribute("userid", userid);
        session.setAttribute("title", title);
        response.sendRedirect(request.getContextPath() +"/view?title=" +
                                title);
    }
}
```

```
package wiki;
`import javax.servlet.*;
import java.io.IOException;
import java.util.Vector;
public class ViewServlet extends ParentServlet
{
   protected void doGet(HttpServletRequest request, HttpServletResponse
                        response) throws ServletException, IOException
   {
      HttpSession session = request.getSession();
      String title = request.getParameter("title");
      session.setAttribute("title", title);
      User user = (User)session.getAttribute("user");
      session.setAttribute("user", user);
      String userid = request.getParameter("userid");
      Page pg = null;
      try {
          pg =findPage(title);
      } catch(Exception e){
          throw new ServletException(e.toString());
      }
      if (pg == null)
      {     if(session.getAttribute("user")!=null)
            {   response.sendRedirect(request.getContextPath() +
                                      "/user/create");
                return;
            } else{
                request.setAttribute("error", "If you want create this
                                      new page, please login.");
                request.setAttribute("body", "/WEB-INF/jsp/login.jsp");
                layoutPage.forward(request, response);
                return;
            }
      }
      String ownerid = pg.getOwnerid();
      request.setAttribute("pg", pg);
      request.setAttribute("ownerid", ownerid);
      request.setAttribute("body", "/WEB-INF/jsp/view.jsp");
     // Contruct menu of operations.
```

```
        Vector menu = new Vector();
        if (session.getAttribute("user") == null)
        {
            menu.add(new MenuItem("Login", "/login"));
            menu.add(new MenuItem("Home", "/view?title=Wiki Message
                                    Linking"));
            menu.add(new MenuItem("ListPage", "/list"));
            menu.add(new MenuItem("Help","/help.jsp "));
        }
        else
        {
            if(session.getAttribute("userid").toString().equals("wiki"))
            {
                menu.add(new MenuItem("Logout", "/logout"));
                menu.add(new MenuItem("Home", "/view?title=Wiki Message
                                        Linking"));
                menu.add(new MenuItem("Edit", "/user/edit?title="+title));
                menu.add(new MenuItem("Adduser", "/register"));
                menu.add(new MenuItem("Createpage", "/user/create"));
                menu.add(new MenuItem("Deletepage", "/user/delete"));
                menu.add(new MenuItem("ListPage", "/list"));
                menu.add(new MenuItem("Help", "/help.jsp "));
            }
            else
    if(session.getAttribute("userid").toString().equals(ownerid))
            {
                menu.add(new MenuItem("Logout", "/logout"));
                menu.add(new MenuItem("Home", "/view?title=Wiki Message
                                        Linking"));
                menu.add(new MenuItem("Edit", "/user/edit?title="+title));
                menu.add(new MenuItem("Createpage", "/user/create"));
                menu.add(new MenuItem("Deletepage", "/user/delete"));
                menu.add(new MenuItem("ListPage", "/list"));
                menu.add(new MenuItem("Help", "/help.jsp "));

            }
            else
            {
                if(pg.getIsmodify())
```

```
            {
                menu.add(new MenuItem("Logout", "/logout"));
                menu.add(new MenuItem("Home", "/view?title=Wiki
                                    Message Linking"));
                menu.add(new MenuItem("Edit",
                                    "/user/edit?title="+title));
                menu.add(new MenuItem("Createpage", "/user/create"));
                menu.add(new MenuItem("Deletepage", "/user/delete"));
                menu.add(new MenuItem("ListPage", "/list"));
                menu.add(new MenuItem("Help", "/help.jsp "));
            }else {
                menu.add(new MenuItem("Logout", "/logout"));
                menu.add(new MenuItem("Home", "/view?title=Wiki
                                    Message Linking"));
                menu.add(new MenuItem("Createpage", "/user/create"));
                menu.add(new MenuItem("Deletepage", "/user/delete"));
                menu.add(new MenuItem("ListPage", "/list"));
                menu.add(new MenuItem("Help", "/help.jsp "));
            }
        }
     }
     request.setAttribute("menu", menu);
    // Forward to layout.
    if(!pg.getIspublic())
    {
       if (session.getAttribute("user")==null)
       {   response.sendError(response.SC_FORBIDDEN);
           return;
       }else if
(!session.getAttribute("userid").toString().equals("wiki")&&

!session.getAttribute("userid").toString().equals(ownerid))
       {   response.sendError(response.SC_FORBIDDEN);
           return;
       }else{
           layoutPage.forward(request, response);
           return;
       }
     }
     layoutPage.forward(request, response);
   }
}
```

```
package wiki;
/**
 * Encapsulates an item to appear in a navigational menu
 * of a Web page.
 */
public class MenuItem
{
   private String title;
   private String link;
   public MenuItem(String title, String link)
   {
      this.title = title;
      this.link = link;
   }
   public String getTitle() { return title; }
   public String getLink() { return link; }
}
```

```
package wiki;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.IOException;

/**
 * Requires users to login successfully in order to access protected
resources.
 */
public class UserFilter implements Filter
{
   public void init(FilterConfig filterConfig) throws ServletException
   { }
   public void destroy() { }

   /**
    * User is logged in if a user object is stored in session.
    */
   public void doFilter(   ServletRequest req,
                           ServletResponse resp,
                           FilterChain chain)
   throws IOException, ServletException
   {
      HttpServletRequest request = (HttpServletRequest) req;
      HttpServletResponse response = (HttpServletResponse) resp;
      HttpSession session = request.getSession();
      User user = (User) session.getAttribute("user");
      if (user == null) {
         // Save the requested resource,
         // so we can forward to it after logging in.
         response.sendRedirect(request.getContextPath() + "/");
         return;
      }
      chain.doFilter(request, response);
   }
}
```

```
package wiki;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.ServletContext;
import javax.servlet.RequestDispatcher;
import javax.servlet.http.*;
import java.util.Vector;
public class DeletePageServlet extends ParentServlet
{
   protected void doGet(HttpServletRequest request, HttpServletResponse
                        response)throws ServletException, IOException
   {
     // Contruct menu of operations.
      Vector menu = new Vector();
      menu.add(new MenuItem("Logout", "/logout"));
      menu.add(new MenuItem("Home", "/view?title=Wiki Message Linking"));
      menu.add(new MenuItem("Adduser", "/register"));
      menu.add(new MenuItem("Createpage", "/user/create"));
      menu.add(new MenuItem("ListPage", "/list"));
      menu.add(new MenuItem("Help", "help.jsp "));
      request.setAttribute("menu", menu);
      request.setAttribute("body", "/WEB-INF/jsp/deletePage.jsp");
      layoutPage.forward(request, response);
   }
    protected void doPost( HttpServletRequest request,
                          HttpServletResponse response)
                          throws ServletException, IOException
   {
     String title = request.getParameter("title");
     if (title == null) {
        request.setAttribute("error", "DeletePageServlet: title
                             parameter missing");
        doGet(request, response);
        return;
     }
     if (request.getParameter("cancel") != null){
        //Cancel button clicked.
        response.sendRedirect(Constants.viewUrl + title);
        return;
```

```
        }
        Page pg = findPage(title);
        if (pg == null){
           request.setAttribute("error", "DeletePageServlet: There is no
                                page with title = " + title);
           request.setAttribute("body", "/WEB-INF/jsp/error.jsp");
           layoutPage.forward(request, response);
           return;
        }

        HttpSession session = request.getSession();
        String userid = session.getAttribute("userid").toString();
        String ownerid = pg.getOwnerid();
        if(userid.equals("wiki")||userid.equals(ownerid))
        {
            try {
                Page.deletePage(pg);
            } catch (Exception e) {
                throw new ServletException(e);
            }
        }
        else
        {
           request.setAttribute("error", "DeletePageServlet: The user don't
                                have the authorate to delete this page.");
           doGet(request, response);
           return;
        }

        User user = (User)session.getAttribute("user");
        session.setAttribute("user", user);
        request.setAttribute("userid", userid);
        response.sendRedirect(request.getContextPath() +"/view?title=" +
                              Constants.homePageTitle);
    }

}
```

```
package wiki;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.IOException;

public class LogoutServlet extends HttpServlet
{
   protected void doGet(HttpServletRequest request,
                        HttpServletResponseresponse)
                 throws ServletException, IOException
   {
      HttpSession session = request.getSession();
      session.invalidate();
      String contextPath = request.getContextPath();
      response.sendRedirect(request.getContextPath() + "/");
   }
}
```

```
package wiki;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.ServletContext;
import javax.servlet.RequestDispatcher;
import javax.servlet.http.*;
import java.util.Vector;
public class EditServlet extends ParentServlet
{
  protected void doGet(HttpServletRequest request,
                       HttpServletResponse response)
  throws ServletException, IOException
  {
     HttpSession session = request.getSession();
     String userid = session.getAttribute("userid").toString();
     String title = request.getParameter("title");
     if (title == null){
        request.setAttribute("error", "EditServlet: title parameter
                              missing");
        return;
     }
     Page pg = findPage(title);
     if (pg == null){
        request.setAttribute("error","EditServlet: There is no page with
                             title ="+title);
        return;
     }
     // Contruct menu of operations.
     Vector menu = new Vector();
     menu.add(new MenuItem("Help","/help.jsp "));
     request.setAttribute("menu", menu);
     String ownerid = pg.getOwnerid();
     request.setAttribute("userid", userid);
     request.setAttribute("ownerid", ownerid);
     request.setAttribute("pg", pg);
     request.setAttribute("body", "/WEB-INF/jsp/edit.jsp");
     layoutPage.forward(request, response);
  }
}
```

```
package wiki;
import java.sql.*;
import javax.sql.DataSource;
import java.io.IOException;
import javax.naming.InitialContext;
import javax.naming.Context;
import java.util.*;
import java.io.*;

public class Page
{
   public static final int MAX_PAGE_NAME_LENGTH = 255;
   static private DataSource ds;
   static
   {
     try {
       InitialContext ic = new InitialContext();
       Context tomcatContext = (Context)
             ic.lookup(Constants.jndiContainerContext);
       ds = (DataSource)tomcatContext.lookup(Constants.jndiDatabaseName);
       if (ds==null) throw new RuntimeException("no DataSource");
     } catch (Exception e){
       throw new RuntimeException(e);
     }
   }

   private String title;
   private String body;
   private boolean ismodify;
   private boolean ispublic;
   private String ownerid;
   private String userid;

   public Page()
   {}

   public Page(String title, String body, String ownerid, boolean
               ismodify, boolean ispublic)
   {
```

```
    this.title = title;
    this.body = body;
    this.ismodify = ismodify;
    this.ispublic = ispublic;
    this.ownerid = ownerid;
}

public String getTitle(){return title;}
public String getBody(){return body;}
public boolean getIsmodify(){return ismodify;}
public boolean getIspublic(){return ispublic;}
public String getOwnerid(){return ownerid;}
public void setBody(String body){this.body = body;}
public void setIsmodify(boolean ismodify){this.ismodify = ismodify;}
public void setIspublic(boolean ispublic){this.ispublic = ispublic;}

public static Page find (String title) throws SQLException
{
    Connection conn = null;
    Statement statement = null;
    ResultSet rs = null;
    String body = null;
    boolean ismodify = false;
    boolean ispublic = false;
    String ownerid = null;
    try {
        conn = ds.getConnection();
        statement = conn.createStatement();
        String query = "select * from pages where title='"
                       + title + "'";
        rs = statement.executeQuery(query);
        if (!rs.first()) return null;
        body = rs.getString("body");
        ismodify = rs.getBoolean("ismodify");
        ispublic = rs.getBoolean("ispublic");
        ownerid = rs.getString("ownerid");
    } finally {
        if (rs != null) rs.close();
        if (statement != null) statement.close();
```

```
            if (conn != null) conn.close();
        }
//      if (body == null)return null;
        return new Page(title, body, ownerid, ismodify, ispublic);
    }

    public static int deletePage(Page page) throws SQLException
    {
        String title = page.getTitle();
        Connection conn = null;
        Statement statement = null;
        int result;
        try {
            conn = ds.getConnection();
            statement = conn.createStatement();
            String query = "delete from pages where title='" + title + "';";
            result = statement.executeUpdate(query);
            return result;
        } finally {
            statement.close();
            conn.close();
        }
    }

    public static boolean updatePage(Page page) throws SQLException
    {
        Connection conn = null;
        Statement statement = null;
        ResultSet rs = null;
        String title = page.getTitle();
        try {
            conn = ds.getConnection();
            statement = conn.createStatement();
            String query = "select * from pages where title ='"
                           + title + "';";
            rs = statement.executeQuery(query);
            if (!rs.first())
                return false;//can't find the page with the title.
```

```
        // Write new body.
        int setismodify, setispublic;
        if(page.getIsmodify())
            setismodify = 1;
        else
            setismodify = 0;
        if(page.getIspublic())
            setispublic = 1;
        else
            setispublic = 0;

        query = "update pages set body='" +
                escapeForSqlQuery(page.getBody()) +
                                "', ismodify = '" + setismodify +
                                "', ispublic = '" + setispublic +
                                "' where title ='" + title +"';";
        statement.executeUpdate(query);
      } finally {
        rs.close();
        statement.close();
        conn.close();
    }
     return true;
}

// Create new empty request page not in the database.
public static Page createPage(String title, String userid)
                              throws SQLException
{
    Connection conn = null;
    Statement statement = null;
    ResultSet rs = null;
    Page pg = null;
    String body;
    String ownerid;
    boolean ismodify;
    boolean ispublic;
    try {
        conn = ds.getConnection();
```

```
        statement = conn.createStatement();
        String query = "insert into pages(title, body, ownerid,
                        ismodify, ispublic)values('" +
                    title + "', '', '" + userid + "', '1', '1')";
        statement.executeUpdate(query);
        query = "select * from pages where title='"+title+"';";
        rs = statement.executeQuery(query);
        ismodify = rs.getBoolean("ismodify");
        ispublic = rs.getBoolean("ispublic");
        body = "";
        ownerid = userid;

    } catch (SQLException e){
        return null;

    } finally {
        if(statement != null) statement.close();
        if(conn != null) conn.close();
    }

    return new Page(title, body, ownerid, ismodify, ispublic);
}

/**
 *Inserts spaces before the upper case letters in the title.
 *see Wiki#addSpace(String)
 *return the title with spaces
 */
public String getSpacedTitle()
{
    return Wiki.addSpaces(title);
}

/**
 * Converts the wiki marked up content to HTML
 * see wiki#transform(String)
 * return the HTML version of the page content
 */
public String getHtmlContent()
```

```
    {
      try {
          return Wiki.transform(body, userid);
      } catch(IOException e){
          return e.toString();
      }
    }

    /**
     * Escape the slashes appearing in an SQL query string.
     * param s the String that contains slashes
     * return a new String that is safe for SQL query
     */
    private static String escapeForSqlQuery(String s)
    {
       String r = s.replaceAll("\\\\", "\\\\\\\\");
       return r.replaceAll("\\'","\\\\'");
    }


}
```

```
package wiki;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.ServletContext;
import javax.servlet.RequestDispatcher;
import javax.servlet.http.*;


public class UpdateServlet extends ParentServlet
{
    protected void doPost( HttpServletRequest request,
                           HttpServletResponse response)
                  throws ServletException, IOException
   {
     String title = request.getParameter("title");
     String req_userid=request.getParameter("userid");
     if (title == null) {
        request.setAttribute("error", "UpdateServlet: title
                              parameter missing");
        request.setAttribute("body", "/WEB-
                              INF/jsp/error.jsp");
        layoutPage.forward(request, response);
        return;
     }
     if (request.getParameter("update") != null)
     {  //Update the page.
        HttpSession session = request.getSession();
        String userid =
               session.getAttribute("userid").toString();

        //Check if is the right user update the page.
        if(session.isNew()||userid.compareTo(req_userid) != 0)
        { response.sendRedirect(Constants.viewUrl + title);
          return;
        }
        Page pg = findPage(title);
        if (pg == null){
        request.setAttribute("error", "UpdataServlet: There
                         is no page with title = " + title);
        request.setAttribute("body", "/WEB-INF/jsp/error.jsp");
        layoutPage.forward(request, response);
        return;
        }
        String ownerid = pg.getOwnerid();

        //check if others can modify the page
```

```
        if(req_userid.compareTo("wiki") != 0 &&
req_userid.compareTo(ownerid) != 0 && !pg.getIsmodify())
        {
           response.sendRedirect(Constants.viewUrl + title);
           return;
        }

        String body = request.getParameter("body");
        if (body == null){
        request.setAttribute("error", "UpdataServlet:
                          content parameter missing");
        request.setAttribute("body", "/WEB-
                              INF/jsp/error.jsp");
        layoutPage.forward(request, response);
        return;
        }

        pg.setBody(body);
     String setismodify = request.getParameter("ismodify");
     String setispublic = request.getParameter("ispublic");
        boolean ismodify, ispublic;
        if(setismodify != null)
           ismodify = true;
        else
           ismodify = false;
        if(setispublic != null)
           ispublic = true;
        else
           ispublic = false;
        pg.setIsmodify(ismodify);
        pg.setIspublic(ispublic);
        try {
           Page.updatePage(pg);
        } catch (Exception e) {
            throw new ServletException(e);
        }

        User user = (User)session.getAttribute("user");
        session.setAttribute("user", user);
        session.setAttribute("userid", userid);
    }
    response.sendRedirect(request.getContextPath()
+"/view?title=" + title);
  }
}
}
```

```
package wiki;
import java.util.TreeSet;
import java.util.Vector;
import java.util.Iterator;
import java.io.StringReader;
import java.io.BufferedReader;
import java.io.IOException;
import java.util.*;
import java.io.*;
import java.sql.SQLException;
/**
 * Provides methods needed to process wiki markup.
 */
public class Wiki
{
  public static final int MAX_PAGE_NAME_LENGTH = 255;
  protected static String contextPath;
  private static String validUrlChars =
        "+-=._/*(),@'$:;&!?/~%#" +
        "0123456789" +
        "abcdefghijklmnopqrstuvwxyz" +
        "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
  public static boolean isValidUrlChar(char ch)
  {
      return validUrlChars.indexOf(ch) >= 0;
  }

  private final static String validTitleChars =
        "_." +
        "0123456789" +
        "abcdefghijklmnopqrstuvwxyz" +
        "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

  public static boolean isValidTitleChar(char ch)
  {
      return validTitleChars.indexOf(ch) >= 0;
  }
  public static String extractTitle(StringReader r) throws IOException
  {
```

```
    StringBuffer sb = new StringBuffer();
    for (int i = 0; i < MAX_PAGE_NAME_LENGTH; ++i) {
       r.mark(2);
       int c = r.read();
       if (c == -1) break;
       char ch = (char) c;
       if (!isValidTitleChar(ch)) break;
       sb.append(ch);
    }
    r.reset();
    return sb.toString();
}

public static String transform(String body, String userid)
      throws IOException
{
   StringReader sr = new StringReader(body);
   StringBuffer sb = new StringBuffer();
   while(true){
       int c = sr.read();
       if (c==-1) break;
       char ch = (char)c;
       sb.append(ch);
   }
   return sb.toString();
}

private static boolean readString(Reader r, String s)
                          throws IOException
{
    r.mark(s.length());
    for (int i = 0; i < s.length(); ++i) {
       int c = r.read();
       if (c == -1 || c != (int) s.charAt(i)) {
          r.reset();
          return false;
       }
    }
    return true;
```

```
    }

    public static String addSpaces(String title)
    {
        if (title.length()==0) return "";
        if (title.length() == 1) return title;
        StringBuffer sb = new StringBuffer();
        sb.append(title.charAt(0));
        int n = title.length() - 1;
        for (int i = 1; i < n; ++i){
            char ch0 = title.charAt(i);
            char ch1 = title.charAt(i + 1);
            if (Character.isLowerCase(ch0) && Character.isUpperCase(ch1)){
                sb.append(ch0);
                sb.append(' ');
            } else if (Character.isUpperCase(ch0)
                    &&Character.isLowerCase(ch1)){
                sb.append(' ');
                sb.append(ch0);
            } else{
                sb.append(ch0);
            }
        }
        sb.append(title.charAt(n));
        return sb.toString();
    }
}
```

```
package wiki;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.ServletContext;
import javax.servlet.RequestDispatcher;
import javax.servlet.http.*;
import java.util.Vector;
public class CreateServlet extends ParentServlet
{
   protected void doGet(HttpServletRequest request,
                        HttpServletResponse response)
   throws ServletException, IOException
   {// Contruct menu of operations.
      Vector menu = new Vector();
      menu.add(new MenuItem("Logout", "/logout"));
      menu.add(new MenuItem("Home", "/view?title=Wiki Message Linking"));
      menu.add(new MenuItem("Adduser", "/register"));
      menu.add(new MenuItem("Deletepage", "/user/delete"));
      menu.add(new MenuItem("ListPage", "/list"));
      menu.add(new MenuItem("Help", "help.jsp "));
      request.setAttribute("menu", menu);
      request.setAttribute("body", "/WEB-INF/jsp/create.jsp");
      layoutPage.forward(request, response);
   }
    protected void doPost( HttpServletRequest request,
                          HttpServletResponse response)
                          throws ServletException, IOException
   {
     HttpSession session = request.getSession();
     String title = request.getParameter("title");
     if (title == null) {
        request.setAttribute("error", "CreateServlet: title parameter
                              missing");
        doGet(request, response);
        return;
     }
    if (request.getParameter("cancel") != null){
        //Cancel button clicked.
```

```
            response.sendRedirect(Constants.viewUrl +
                                  Constants.homePageTitle);
            return;
        }
        Page pg = findPage(title);
        if (pg != null){
            request.setAttribute("error", "CreateServlet: The page is
                                 already created, please try another one");
            doGet(request, response);
            return;
        }
        String userid = session.getAttribute("userid").toString();

        try {
            Page.createPage(title, userid);
        } catch (Exception e) {
            throw new ServletException(e);
        }
        User user = (User)session.getAttribute("user");
        session.setAttribute("user", user);
        request.setAttribute("userid", userid);
        session.setAttribute("title", title);
        response.sendRedirect(request.getContextPath() +"/view?title=" +
                              title);
    }
}
```

```
package wiki;
import java.sql.*;
import javax.sql.DataSource;
import java.io.IOException;
import javax.naming.InitialContext;
import javax.naming.Context;
import java.util.*;
import java.io.*;

/**
 * Encapsulates an title to appear in a navigational title
 * of a Web page.
 */

public class ListPage
{
   static private DataSource ds;
   static
   {
     try {
       InitialContext ic = new InitialContext();
       Context tomcatContext = (Context) ic.lookup
                   (Constants.jndiContainerContext);
       ds = (DataSource)tomcatContext.lookup(Constants.jndiDatabaseName);
       if (ds==null) throw new RuntimeException("no DataSource");
     } catch (Exception e){
       throw new RuntimeException(e);
     }
   }

   private String title;
   private String ownerid;

   public ListPage(String title, String ownerid)
   {
      this.title = title;
      this.ownerid = ownerid;
   }
```

```
   public String getTitle() { return title; }
   public String getOwnerid() { return ownerid; }
/**
 * @returns non-null, possibly empty, Vector of all pages list.
 */

   static public Vector findListPage() throws Exception
   {
      Vector pagelist = new Vector();
      Connection conn = null;
      try {
         conn = ds.getConnection();
         Statement statement = conn.createStatement();
         String s = "select ownerid, title from pages order by ownerid,
                    title;";
         ResultSet rs = statement.executeQuery(s);
         while (rs.next()) {
            String title = rs.getString("title");
            String ownerid = rs.getString("ownerid");
            pagelist.add(new ListPage(title, ownerid));
         }
      } finally {
         conn.close();
      }
      return pagelist;
   }
}
```

```
package wiki;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.ServletContext;
import javax.servlet.RequestDispatcher;
import javax.servlet.http.*;
import java.util.Vector;

public class ListPageServlet extends ParentServlet
{

  protected void doGet(HttpServletRequest request,
                       HttpServletResponse response)
  throws ServletException, IOException
  {
     HttpSession session = request.getSession();
//     String userid = session.getAttribute("userid").toString();
     String title = null;
     String ownerid = null;
     // Contruct pagetitle vector.
     ListPage lp = new ListPage(title, ownerid);
     Vector pagelist = new Vector();
     try {
         pagelist = lp.findListPage();
     }catch (Exception e){
         throw new ServletException(e.toString());
     }

     request.setAttribute("pagelist", pagelist);
     Vector menu = new Vector();
     if (session.getAttribute("user") == null)
     {
          menu.add(new MenuItem("Login", "/login"));
          menu.add(new MenuItem("Home", "/view?title=Wiki Message
                                Linking"));
          menu.add(new MenuItem("Help","/help.jsp "));
      }
      else
      {
```

```
        if(session.getAttribute("userid").toString().equals("wiki"))
        {
            menu.add(new MenuItem("Logout", "/logout"));
            menu.add(new MenuItem("Home", "/view?title=Wiki Message
                                Linking"));
            menu.add(new MenuItem("Adduser", "/register"));
            menu.add(new MenuItem("Createpage", "/user/create"));
            menu.add(new MenuItem("Deletepage", "/user/delete"));
            menu.add(new MenuItem("Help", "help.jsp "));
        }
        else
        {
                menu.add(new MenuItem("Logout", "/logout"));
                menu.add(new MenuItem("Home", "/view?title=Wiki
                                    Message Linking"));
                menu.add(new MenuItem("Createpage", "/user/create"));
                menu.add(new MenuItem("Deletepage", "/user/delete"));
                menu.add(new MenuItem("Help", "/help.jsp "));
        }
     }
     request.setAttribute("menu", menu);


    request.setAttribute("body", "/WEB-INF/jsp/list.jsp");
    layoutPage.forward(request, response);
  }
}
```

# APPENDIX B

# JAVA SERVER PAGE AND CASCADING STYLE SHEET PAGES PRINTOUT

```
<!--register.jsp - ->

<h2 align="center">Add Users For Wiki Messege Linking</h2>

<div class="error">
<%
   String error = (String) request.getAttribute("error");
   if (error != null) {
      out.print(error);
   }
%>
</div>

<table align="center" border="1" cellspacing="0" width="600">
    <form action="register" method="post">
    <tr><td align="center">
       Userid:   
        <input type="text" name="userid" size="24"> <br>
    </td></tr>
    <tr><td align="center">
       Username:<input type="text" name="username" size="24"><br>
    </td></tr>
    <tr><td align="center">
       Password:<input type="password" name="password" size="24"> <br>
    </td></tr>
    <tr><td align="center">
       Password: <input type="password" name="password2" size="24"> <br>
    </td></tr>
    <tr><td align="center"> Email:    
        <input type="text" name="email" size="24"><br>
    </td></tr>
    <tr><td align="center">
       <input type="submit" value="Add">
            <input type="submit" value="Cancel" name="cancel">
    </td></tr>
    </form>
</table>
```

```
<!- -layout.jsp-->

<%@ taglib uri="../tld/c.tld" prefix="c" %>
<html>
<head>
<link rel='stylesheet' type='text/css' href='<c:url value="/global.css"
/>' >
<title><c:out value="${pg.spacedTitle}"/></title>
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
<META HTTP-EQUIV="Expires" CONTENT="-1">
</head>
<body>
<div>
   <h1> Wiki Message Linking </h1>
</div>
<hr>
<ul class="one">
    <li>
       <c:forEach var="item" items="${menu}">
           <a class="one" href='<c:url value="${item.link}" />'><c:out
value="${item.title}" /></a>
       </c:forEach>
    </li>
</ul>
<hr>
<div>
    <ul class="one">
        <c:out value="Hello ${userid}"/><br>
        <c:out value="This page is owned by ${ownerid}"/>
    </ul>
</div>

<div>
    <c:import url="${body}" />
</div>

</body>
</html>
```

```
<! -- login.jsp -->

<ul>
<div class="error">
<%
   String error = (String) request.getAttribute("error");
   if (error != null) {
      out.print(error);
   }
%>
</div>
    <h2 align="center">Login For Wiki Messege Linking</h2>

<table align="center" border="1" cellspacing="0" width="600">
    <form action="login" method="post">
   <tr><td align="center">
             Userid: <input type="text" name="userid" size="20"> <br>
        </td></tr>
   <tr><td align="center">
            Password: <input type="password" name="password" size="20">
<br>
        </td></tr>
   <tr><td align="center">
            <input type="submit" value="Login">
                <input type="submit" value="Cancel" name="cancel">
        </td></tr>
    </form>
</table>
```

```
<!- -view.jsp - ->

<%@ taglib uri="../tld/c.tld" prefix="c"%>

<h2 align="center">
    <c:out value="${pg.spacedTitle}"/>
</h2>
<div>
    <jsp:getProperty name="pg" property="htmlContent"/>
</div>
```

```
<! - - list.jsp - ->

<%@ taglib uri="../tld/c.tld" prefix="c" %>

    <h2 align="center">Wiki Pages List</h2>
<table align="center">
    <tr>
        <td><h3>Page Titles</h3></td>
        <td width = "50%"><h3>Owners</h3></td>
    </tr>
</table>
 <c:forEach var="item" items="${pagelist}">
<table align="center">


    <tr>
        <td><h4><a href='<c:url
value="view?title=${item.title}"/>'><c:out
value="${item.title}"/></a></h4></td>
        <td width="50%"><h4><c:out value="${item.ownerid}"/></h4></td>
    </tr>


</table>
</c:forEach>
```

```
< ! - - edit.jsp - ->

<%@ taglib uri="../tld/c.tld" prefix="c"%>
<%@ page import="wiki.*" %>
<%@ page import="java.util.*" %>
<h2 align="center"><c:out value="${pg.spacedTitle}" /></h2>
<center>
<%
   // variable for JavaScript
   String userid = request.getAttribute("userid").toString();
   Page pg=(Page)request.getAttribute("pg");
   String title = pg.getTitle();
   String ownerid = pg.getOwnerid();
   String setismodify, setispublic;
   if(pg.getIsmodify())
      setismodify = "checked";
   else
      setismodify = "";
   if(pg.getIspublic())
      setispublic = "checked";
   else
      setispublic = "";
%>
<form action='update' method='post' name="update">
   <textarea name="body" rows="20" cols="110"><jsp:getProperty name="pg"
property="body"/></textarea>

   <%
      if(ownerid.equals(userid)||userid.equals("wiki"))
      {
   %>

   <table align="center" >
   <tr>
      <td align="center">
          <input type="checkbox" name="ismodify" <%=setismodify%>
value="1" onclick="checked_modify()">
```

```
          modifiable by users

          <input type="checkbox" name="ispublic" <%=setispublic%>
value="1" onclick="checked_public()" >
          visible by others
       </td>
    </tr>
    </table>

    <%
       }
       else
       {
          if (setismodify.equals(""))
               setismodify = null;
          if (setispublic.equals(""))
               setispublic = null;
    %>
        <input type='hidden' name='ismodify' value='<%= setismodify %>'>
        <input type='hidden' name='ispublic' value='<%= setispublic %>'>
     <%
        }
     %>

    <input type = 'hidden' name='title' value='<jsp:getProperty name="pg"
property="title" />'>

    <input type = 'hidden'  name ='userid'  value = '<%=userid%>'>

    <table align="center" >
    <tr>
       <td align="center">
          <input type="submit" value="update">

          <input type="submit" value="cancel" name="cancel">
       </td>
    </tr>
    </table>
```

```
</form>
</center>
<Script Language="JavaScript">
   function checked_modify()
   {
      if (window.document.update.ismodify.checked)
         window.document.update.ispublic.checked = true;
   }
   function checked_public()
   {
      if (!window.document.update.ispublic.checked)
         window.document.update.ismodify.checked = false;
   }
</Script>
```

```
<! - - error.jsp - ->

<body>
<div class="error">
<%
   String error = (String)request.getAttribute("error");
   if (error != null){
      out.print(error);
   }
%>
</div>
</body>
```

```
<! - menuItem.jsp -->

<%@ page import="wiki.*" %>

<div>
    <ul>
        <c:forEach var="item" items="${menu}">
            <a href='<c:url value="${item.link}" />'><c:out
value="${item.title}" /></a> 
        </c:forEach>
    </ul>
</div>
```

```
<! - create.jsp -->

<%@ taglib uri="../tld/c.tld" prefix="c"%>

<h2 align="center">Create New Pages For Wiki Messege Linking</h2>

<ul>
<div class="error">
<%
   String error = (String) request.getAttribute("error");
   if (error != null) {
      out.print(error);
   }
%>
</div>
</ul>

<table align="center" border="1" cellspacing="0" width="600">
    <form action="create" method="post">
    <tr><td align="center">
       New Page Title:   
        <input type="text" name="title" value = <c:out value=
"${title}"/> size="24"> <br>
    </td></tr>
    <tr><td align="center">
       <input type="submit" value="Add">  
       <input type="submit" value="Cancel" name="cancel">
    </td></tr>
    </form>
</table>
```

```
<! - deletePage.jsp -->

<%@ taglib uri="../tld/c.tld" prefix="c"%>
<h2 align="center">Delete Pages For Wiki Messege Linking</h2>
<ul>
<div class="error">
<%
   String error = (String) request.getAttribute("error");
   if (error != null) {
      out.print(error);
   }
%>
</div>
</ul>
<table align="center" border="1" cellspacing="0" width="600">
    <form action="delete" method="post">
    <tr><td align="center">
       Delete Page Title:   
        <input type="text" name="title" value = <c:out value=
"${title}"/> size="24"> <br>
    </td></tr>
    <tr><td align="center">
       <input type="submit" value="Send">  
       <input type="submit" value="Cancel" name="cancel">
    </td></tr>
    </form>
</table>
```

/HOME/HOYANG/WIKI/WEB/GLOBAL.CSS

```
body {
  /* overflow: scroll;*/
   margin: 2em 2em 2em 2em;
}

h1 { color: blue; background:yellow; text-align: center; font-family:
"New Century Schoolbook", Western;
     font-style: italic; text-transform: uppercase;border: groove 0.5em }
h2 { color: green; text-align: center; text-transform: uppercase }
h3 {color: red; text-transform:uppercase }

UL.one {list-style:none; color:blue; text-align:left; }

hr { color:blue; }

A.one:link { width:70;border: medium double #ff0000;text-decoration:
none;text-align: center;}
A.one:visited { width:70;border: medium double #ff0000;text-decoration:
none;text-align: center;}
A.one:hover { width:70;border: medium double #66ff66;text-decoration:
none;text-align: center;}

P { background: white; text-indent: 0em }

pre {
   border: 1px dotted #000;
   background: #cce;
   padding-top: .4em;
   padding-right: .2em;
   padding-bottom: .4em;
   padding-left: .4em;
}

table {align:center; border:groove 0.1em; cellspacing:0; width:80%}
```

# APPENDIX C

# WEB.XML FILE

/home/hoyang/wiki/web/WEB-INF/web.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd">

<web-app>

   <!-- filters -->
   <filter>
      <filter-name>userFilter</filter-name>
      <filter-class>wiki.UserFilter</filter-class>
   </filter>

   <!-- filter mappings -->

   <filter-mapping>
      <filter-name>userFilter</filter-name>
      <url-pattern>/user/*</url-pattern>
   </filter-mapping>

   <!-- servlets -->

   <servlet>
      <servlet-name>login</servlet-name>
      <servlet-class>wiki.LoginServlet</servlet-class>
      <load-on-startup>1</load-on-startup>
   </servlet>

   <servlet>
      <servlet-name>logout</servlet-name>
      <servlet-class>wiki.LogoutServlet</servlet-class>
      <load-on-startup>1</load-on-startup>
   </servlet>

   <servlet>
      <servlet-name>register</servlet-name>
      <servlet-class>wiki.RegisterServlet</servlet-class>
```

```
    <load-on-startup>1</load-on-startup>
</servlet>

<servlet>
    <servlet-name>view</servlet-name>
    <servlet-class>wiki.ViewServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
</servlet>

<servlet>
    <servlet-name>edit</servlet-name>
    <servlet-class>wiki.EditServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
</servlet>

<servlet>
    <servlet-name>update</servlet-name>
    <servlet-class>wiki.UpdateServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
</servlet>

<servlet>
    <servlet-name>create</servlet-name>
    <servlet-class>wiki.CreateServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
</servlet>

<servlet>
    <servlet-name>delete</servlet-name>
    <servlet-class>wiki.DeletePageServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
</servlet>

<servlet>
    <servlet-name>list</servlet-name>
    <servlet-class>wiki.ListPageServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
</servlet>
```

```
<servlet-mapping>
   <servlet-name>login</servlet-name>
   <url-pattern>/login</url-pattern>
</servlet-mapping>

<servlet-mapping>
   <servlet-name>logout</servlet-name>
   <url-pattern>/logout</url-pattern>
</servlet-mapping>

<servlet-mapping>
   <servlet-name>register</servlet-name>
   <url-pattern>/register</url-pattern>
</servlet-mapping>

<servlet-mapping>
   <servlet-name>view</servlet-name>
   <url-pattern>/view</url-pattern>
</servlet-mapping>

<servlet-mapping>
   <servlet-name>edit</servlet-name>
   <url-pattern>/user/edit</url-pattern>
</servlet-mapping>

<servlet-mapping>
   <servlet-name>update</servlet-name>
   <url-pattern>/user/update</url-pattern>
</servlet-mapping>

<servlet-mapping>
   <servlet-name>create</servlet-name>
   <url-pattern>/user/create</url-pattern>
</servlet-mapping>

<servlet-mapping>
   <servlet-name>delete</servlet-name>
   <url-pattern>/user/delete</url-pattern>
</servlet-mapping>
```

```
    <servlet-mapping>
        <servlet-name>list</servlet-name>
        <url-pattern>/list</url-pattern>
    </servlet-mapping>

</web-app>
```

# APPENDIX D

# BUILD.XML FILE

/home/hoyang/wiki/build.xml

```
<!--
    This file assumes that you are running tomcat version 4.x.
-->
<project basedir="." default="compile">
    <property file="../build.properties" />
    <property name="context.path" value="/wiki" />

  <!--
.....................................................................-->
  <!-- task definitions -->
  <taskdef name="list"
           classname="org.apache.catalina.ant.ListTask"
           classpath="${manager.jar}" />
  <taskdef name="install"
           classname="org.apache.catalina.ant.InstallTask"
           classpath="${manager.jar}" />
  <taskdef name="remove"
           classname="org.apache.catalina.ant.RemoveTask"
           classpath="${manager.jar}" />

  <!--
.....................................................................-->
  <!-- classpath -->
  <path id="classpath">
     <pathelement location="${servlet.jar}" />
     <pathelement location="${commons-fileupload.jar}" />
  </path>
  <!--
.....................................................................-->
 <!-- minor targets -->

  <target name="delete-logs">
     <delete>
        <fileset dir=".">
           <include name="*.log" />
        </fileset>
     </delete>
```

```
    </target>

    <target name="copy-libs">
      <!-- Copy tag library descriptor. -->
      <mkdir dir="web/WEB-INF/tld" />
      <copy todir="web/WEB-INF/tld">
        <fileset file="${jstl.home}/tld/c.tld" />
      </copy>

      <!-- Copy libraries. -->
      <mkdir dir="web/WEB-INF/lib" />
      <copy todir="web/WEB-INF/lib">
        <fileset file="${jstl.home}/lib/jstl.jar" />
        <fileset file="${jstl.home}/lib/standard.jar" />
        <fileset file="${commons-fileupload.jar}" />
      </copy>
    </target>

    <!--
.....................................................................-->
    <!-- main targets -->

    <target name="list" description="List the running web apps.">
      <list   url="http://localhost:8080/manager" username="manager"
password="manager" />
    </target>

    <target name="docs" description="Publish docs to the root web app.">
      <mkdir dir="${catalina.home}/webapps/ROOT/docs"/>
      <mkdir dir="${catalina.home}/webapps/ROOT/docs/${context.path}"/>
      <copy todir="${catalina.home}/webapps/ROOT/docs/${context.path}">
        <fileset dir="docs/html" />
      </copy>
    </target>

    <target name="clean" description="Delete all derived objects.">
      <delete dir="web/WEB-INF/classes" />
      <antcall target="delete-logs" />
    </target>
```

```
   <target name="compile" description="Compile Java source code.">
      <mkdir dir="web/WEB-INF/classes" />
      <!-- compile -->
      <javac   srcdir="src"
               destdir="web/WEB-INF/classes"
               fork="no"
               classpathref="classpath" />
   </target>
   <target name="restart" description="Restart web app.">
      <antcall target="stop" />
      <antcall target="start" />
   </target>

   <target name="start" depends="compile,copy-libs" description="start
web app.">
      <antcall target="delete-logs" />
      <install url="http://localhost:8080/manager"
               username="manager"
               password="manager"
               path="${context.path}"
               config="file://${basedir}/context.xml" />
   </target>


   <target name="stop" description="Stop web app.">
      <remove  url="http://localhost:8080/manager"
               username="manager"
               password="manager"
               path="${context.path}" />
      <antcall target="delete-logs" />
   </target>
   <target name="javadoc" depends="compile">
      <mkdir dir="${catalina.home}/webapps/ROOT/docs"/>
      <mkdir dir="${catalina.home}/webapps/ROOT/docs/${context.path}"/>
      <mkdir
dir="${catalina.home}/webapps/ROOT/docs/${context.path}/javadocs"/>
      <javadoc sourcepath="src"

destdir="${catalina.home}/webapps/ROOT/docs/${context.path}/javadocs"
```

```
                packagenames="*"
                Private="true"
                Overview="${basedir}/docs/overview.html">
            <classpath refid="classpath" />
            <link href="http://139.182.134.94:8080/tomcat-docs/servletapi/"
/>
            <link href="http://139.182.134.94:8080/commons-fileupload/" />
            <link href="http://java.sun.com/j2se/1.4.1/docs/api/" />
        </javadoc>
    </target>

    <target name="createdb" depends="compile">
     <!--echo message="${postgresql.jar}"/-->
        <java classname = "wiki.CreateDatabase">
            <classpath>
                <pathelement path="web/WEB-INF/classes"/>
                <pathelement location="${postgresql.jar}"/>
            </classpath>
        </java>
    </target>

</project>
```

# APPENDIX E

## CONTEXT.XML

/home/hoyang/wiki/context.xml

```
<Context    path="/wiki"
          docBase="/home/hoyang/wiki/web"
           debug="0"
           reloadable="true"
           swallowOutput="true"
           useNaming="true">
    <Logger     className="org.apache.catalina.logger.FileLogger"
              prefix=""
              suffix=".log"
              directory="/home/hoyang/wiki"
              timestamp="true" />
    <Resource     name="database"
                  auth="Container"
                  type="javax.sql.DataSource" />
   <ResourceParams name="database">
       <parameter>
          <name>factory</name>
          <value>org.apache.commons.dbcp.BasicDataSourceFactory</value>
       </parameter>
       <parameter>
          <name>driverClassName</name>
          <value>org.postgresql.Driver</value>
       </parameter>
       <parameter>
          <name>url</name>
          <value>jdbc:postgresql://127.0.0.1/wiki</value>
       </parameter>
       <parameter>
          <name>username</name>
          <value>wiki</value>
       </parameter>
       <parameter>
          <name>password</name>
          <value>wiki</value>
       </parameter>
```

```
        <parameter>
            <name>maxActive</name>
            <value>2</value>
        </parameter>
        <parameter>
            <name>maxIdle</name>
            <value>1</value>
        </parameter>
        <parameter>
            <name>maxWait</name>
            <value>-1</value>
        </parameter>
    </ResourceParams>
</Context>
```

REFERENCES


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

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

3. Eric Freeman. JavaSpaces Principles, Patterns, and Practice. Addison Wesley, Novermber 1999.

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

5. H.M.Deitel and P.J.Deitel. JAVA 2 Platform - How to program. Prentice Hall, New Jersey, 2002.

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

7. Elmasri and Navathe. Fundamentals of Database Systems, third edition. Addison Wesley, June 2000.

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

9. PostgreSQL Reference Manual for version 7.3. http://www.postgresql.org/docs/

10. Qusay H. Mahmoud. Servlets and JSP Pages Best Practices. http://java.sun.com/developer/technicalArticles/javaserverpages/servlets_jsp/

11. Kam Hay Fung and Mark Roth. Code Conventions for JavaServer Page Technology Version 1.x. http://java.sun.com/developer/technicalArticles/javaserverpages/code_convention/

12. John Pozadzides and Liam Quinn. CSS Quick Tutorial. http://www.htmlhelp.com/reference/css/quick-tutorial.html