2004

Java/XML-based Trading Information Processing System for produce wholesale market

Ching-Ling Yang

JAVA/XML-BASED TRADING INFORMATION PROCESSING SYSTEM FOR PRODUCE WHOLESALE MARKET

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

Ching-Ling Yang

June 2004

JAVA/XML-BASED TRADING INFORMATION PROCESSING SYSTEM FOR PRODUCE WHOLESALE MARKET

A Project

Presented to the

Faculty of

California State University,

San Bernardino

by

Ching-Ling Yang

June 2004

Approved by:

[redacted]

Dr. David Turner, Chair, Computer Science — Date 5/24/2004

[redacted]

Dr. Richard Botting

[redacted]

Dr. Ernesto Gomez

ABSTRACT

This project is a Java/XML-based Trading Information Processing System (TIPS) to provide HTML-based web pages and a friendly user interface for all buyers and sellers. The application is written using JSP, Java servlets, JavaScript and Java language. All buyers can create their own individual accounts and login to the main page. From the main page, they can view all the produce in the Catalog page. The main page will link to the following pages: Catalog, Shopping Cart, View all orders, Contact Us, and Logout. The most important design in this project is: Design a user-friendly interface for retrieving the enquired data easily.

Internally, data is retrieved in the form of XML documents through a data access layer. These XML documents are transformed into HTML using XSL templates. XML-RPC is a remote procedure call protocol that works over the Internet. An XML-RPC message is an HTTP-POST request. The body of the request is in XML. A procedure executes on the server and the value it returns is also formatted in XML. XML-RPC provides layers of abstraction that make it simple to connect different kinds of computing systems without needing to create new standards for every application.

ACKNOWLEDGMENTS

I would like to express my special gratitude to my project advisor Dr. Turner, who gave me lots of valuable guidance to make my project a whole. And thanks to my two project committee professors Dr. Botting and Dr. Gomez for their strong support in developing my project.

In the beginning, I spent lots of time reading about the Java language and XML to write this project. After several months of effort, I learned more and more from Java network programming skills. Such experiences helped me build a JSP web application using not only JSP and JavaBeans, but also Java servlets and XSL.

Finally, I want to thank everyone else who has helped me with, shown me the way on, or given me the opportunity in the software development. With their existence, my master's project became possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION

TIPS (Trading Information Processing System) was designed for an on-line Java/XML-Based E-Commerce site for a produce retailer, which provides a user-friendly interface for all users. It is a simulated trading tool that could be developed as a trading E-commerce interface between all member retailers and the seller in a produce wholesale market. After a member finishes a trade, then he or she will get an order number and the order result will be saved in an XML file. The members also can use their order number to check the status of the order. From the TIPS, users may also click link to "view all orders," which will show all orders for a member. Users can view order details in a normal HTML page or in an XSL page. The purpose of this was to investigate the the implementation details of incorporating this new XML-based approach in Web applications. When viewing the HTML Page, the Web system will use SOAP to comunicate with a server process that is responsible for managing member orders. When viewing the XSL page, the system also invokes a SOAP method to get the result, but uses an XSL template to transform the XML data in the SOAP message into HTML,

which is then returned to the browser. Furthermore, TIPS's catalog is also an XML file that the seller can easily update frequently.

1.1 Purpose of this Project

The purpose of this project is to investigate the use of the emerging XML technologies to improve online Business to Business (B2B) supply chain processes. This project does this investigation by implementing a prototype online system for produce suppliers. A full-scale implementation is not attempted; instead, the project implements a system that satisfies a subset of the use case scenarios for a full-scale B2B produce supplier system. Internally, some events are simulated. Data is retrieved in the form of XML documents through a data access layer. These XML documents are transformed into HTML using XSL templates. XML-RPC is a Remote Procedure Call protocol that works over the Internet. An XML-RPC message is an HTTP-POST request. The body of the request is in XML. A procedure executes on the server and the value it returns is also formatted in XML.

1.2 Project Products

This project leads to the following products:

- Implementation of TIPS: a working web site with

JSP, Java programs and MySQL database.

- Users manual: an implementation manual will be available for the user.
- System documentation: a project documentation (this report), which is available with system design, specifications, project implementation and testing reports.

1.3 Organization of Chapters

The organization of this documentation is divided into eight chapters. Chapter Two describes the system architecture of this project. Chapter Three discusses the details of database design. Chapter Four focus on the project implementation. Chapter Five involves some the security issues of the project. Chapter Six provides a system validation report. Chapter Seven introduces maintenance manual in this project. The last chapter presents the conclusion and future directions.

CHAPTER TWO

SYSTEM ARCHITECTURE

This project, Trading Information Processing System (TIPS), implements a web-page system to provide a friendly interface for all buyers and sellers to handle their need. There are a web server and a database server in this system. The web server connects to Internet by TCP/IP to exchange information with the web server under HTTP. Also the system accesses a database by communicating with a database server using JDBC.

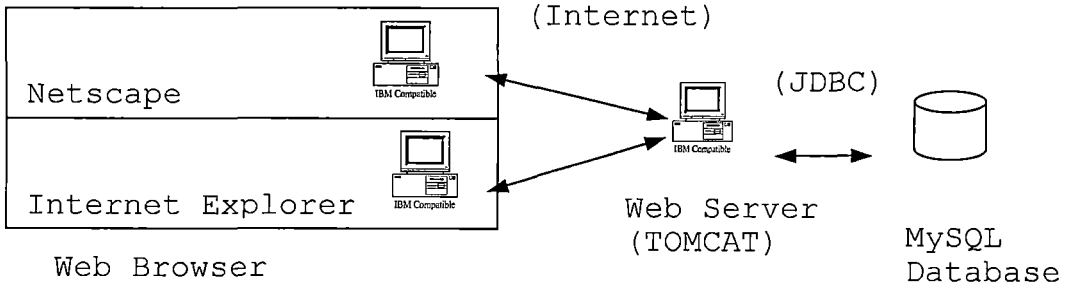

Figure 1. Trading Information Processing System Architecture

In order to choose implementation components conforming to the criteria of shareware, standard, and independent, this project uses Tomcat server as web server, and MySQL server as database server. The other components, such as the web browsers, depend on what kind of browsers the end users use.

The user interface components are built by using HTML 4.01 forms, frames and JavaScript. And the applications are launched using Java Server Pages (JSP) and Java Servlets. JSP was used because it can use java beans, which provide a reusable way to develop system components, and the JSP/servlet application 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 was 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 choice available to TIPS is MySQL. MySQL is a real multi-user database and free. Also, the availability of the JDBC driver for MySQL is the most important reason to choose it. Moreover, the same code could be used to link with another database by changing the JDBC driver, thereby making it database independent.

Simple Object Access Protocol (SOAP) is a lightweight XML-based protocol, developed by W3C, for the exchange of information in a decentralized, distributed environment.

It is an XML-based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined datatypes, and a convention for representing remote procedure calls and responses.

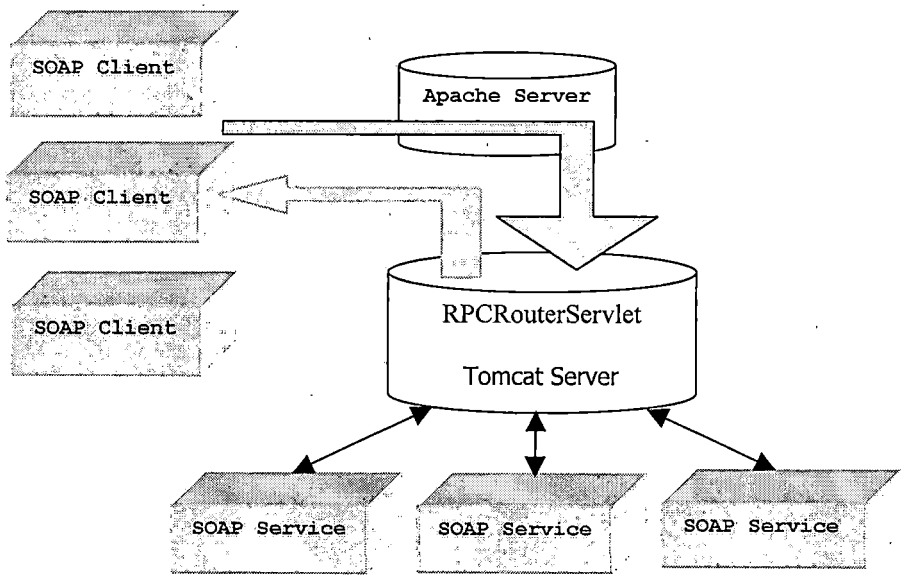

Figure 2. Simple Object Access Protocol (SOAP) Architecture

2.1 Hardware Interfaces

The TIPS project employs Windows operating system to maintain all hardware interfaces. Hardware components that can be coordinated with Windows system are suitable to this project. Hence, there is no need to discuss the hardware interfaces for the system.

2.2 Software Interfaces

The software interfaces used in this project are summarized as following:

- Internet browser: Netscape or Internet Explorer.
- Operating system: Windows 98/Me/2000/XP or Unix/Linux.
- Database: MySQL.
- 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

Since the application is a web-page interface in the web server, most parts of the data are stored in the database server. The data for designing and implementing the schema of the database depends on the properties of members (customers of the web site). The member's data needed by the system include the username, password, e-mail address, and telephone number. If you want check your order status, and then you need to click the "view all orders" link. If you want to know more detail about your order, then click the "check HTML page" link, which brings you to a web page with order details. All the data is saved into the database so that the web application can be shutdown and restarted with loss of data.

3.2 Database Schema Conceptual Model - ER Diagram

In designing the schema for the TIPS database, two distinct parts have been identified. First is the information part, which includes personal information. All the entities and attributes are described in Figure 2. Second is the deliver part, including functions to create

a new account, view all users, view all order numbers, modify an order, and manage stock. All the entities and attributes are detailed in Figure 3.

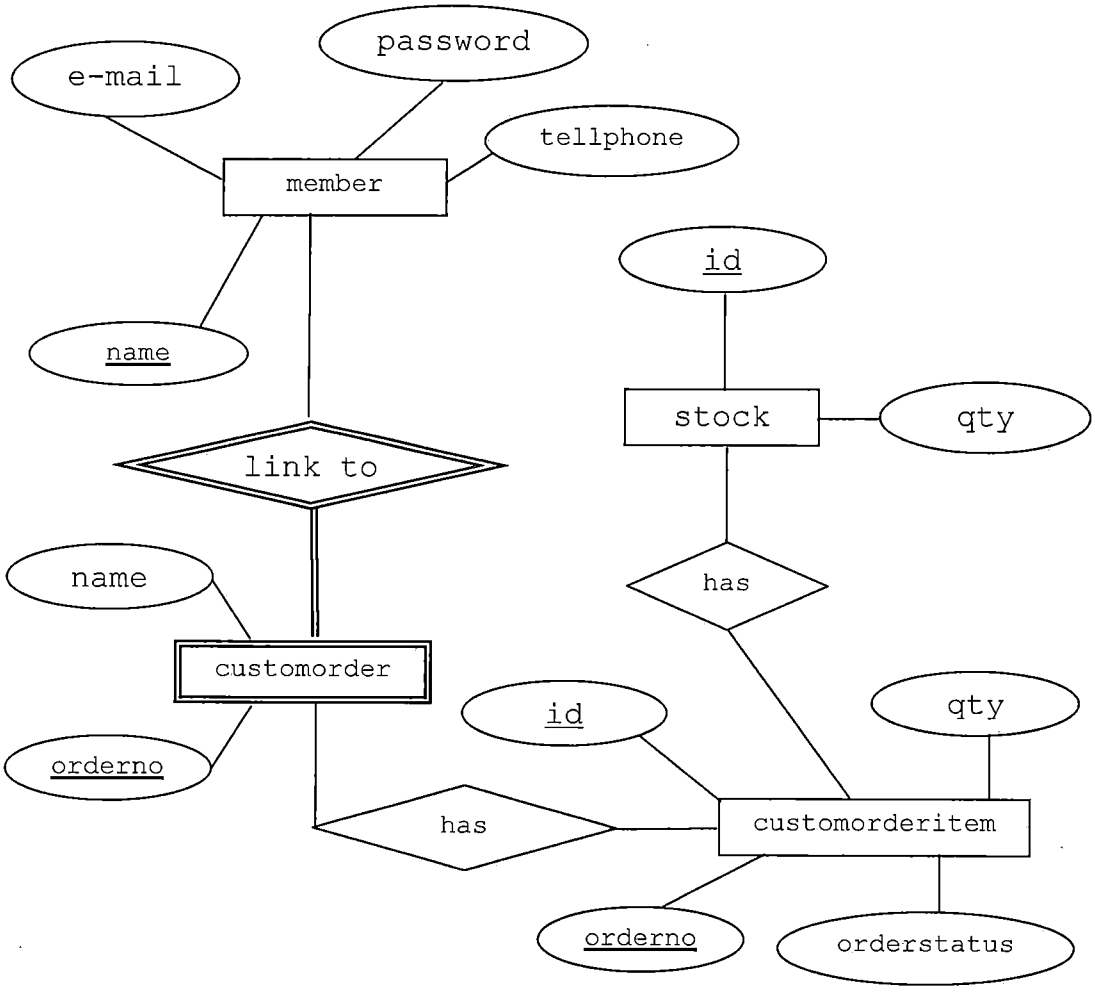

Figure 3. E-R Diagram

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, and fields has * symbol indicate the foreign key.

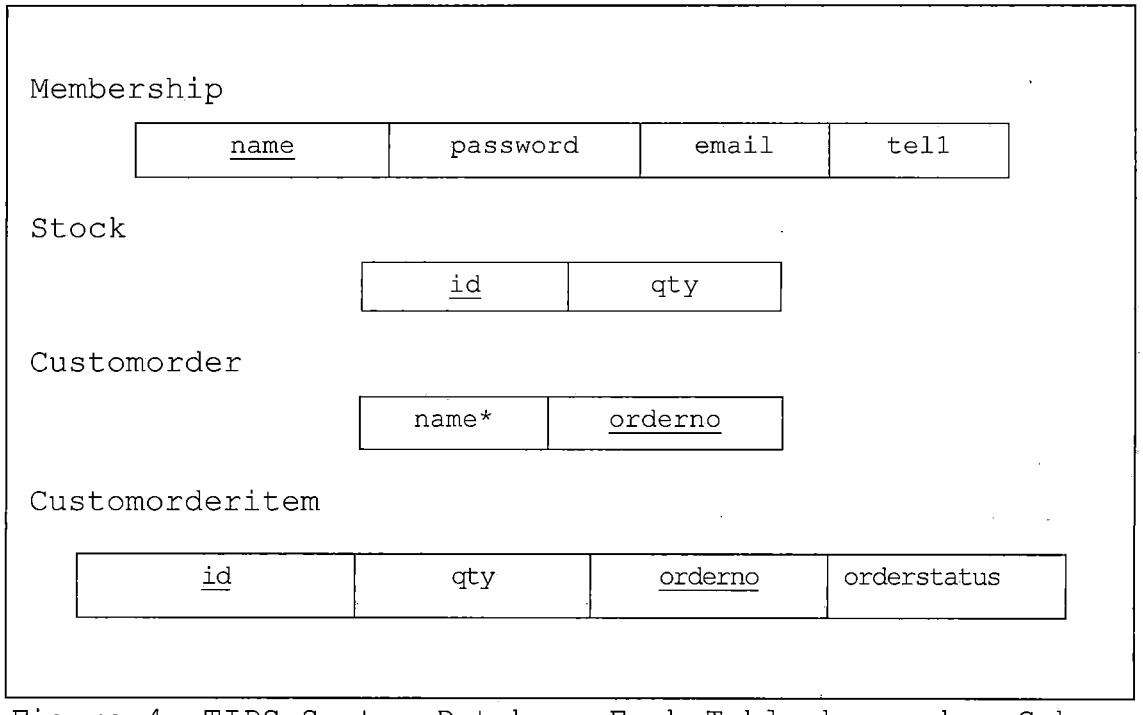

Figure 4. TIPS System Database Each Table has a key Schema

3.4 Data Type and Details

The logical model established into the following detail design in MySQL database. The following tables describe data type, length, primary key or not, null or non-null key, and extra information, such as auto_increment. Since the only data stored in the web Server is the order status. There are four tables in the web server database. The following are the table describe data type, length, primary key or not, null or non-null key.

Table 1. Structure of Table Membership

field	type	null	key	default	extra
name	varchar(20)	Yes	Pri	Null	
password	varchar(20)	Yes		Null	
email	varchar(60)	Yes		Null	
tell	varchar(20)	Yes		Null	

Table 2. Structure of Table Stock

field	type	null	key	default	extra
id	varchar(20)	Yes	Pri	Null	
qty	int(11)	Yes		Null	

Table 3. Structure of Table Customorder

field	type	null	key	default	extra
name	varchar(20)	Yes	For	Null	
orderno	varchar(15)		Pri	Null	

Table 4. Structure of Table Customorderitem

field	type	null	key	default	extra
id	varchar(20)	Yes	Pri	Null	
qty	int(11)	Yes		Null	
orderno	varchar(15)	Yes	Pri	Null	
orderstatus	varchar(60)	Yes		Null	

CHAPTER FOUR

PROJECT IMPLEMENTATION

A TIP is designed to perform 7 different functions for 2 different users. The following Figure 5 is the Use Case Diagram of this project.

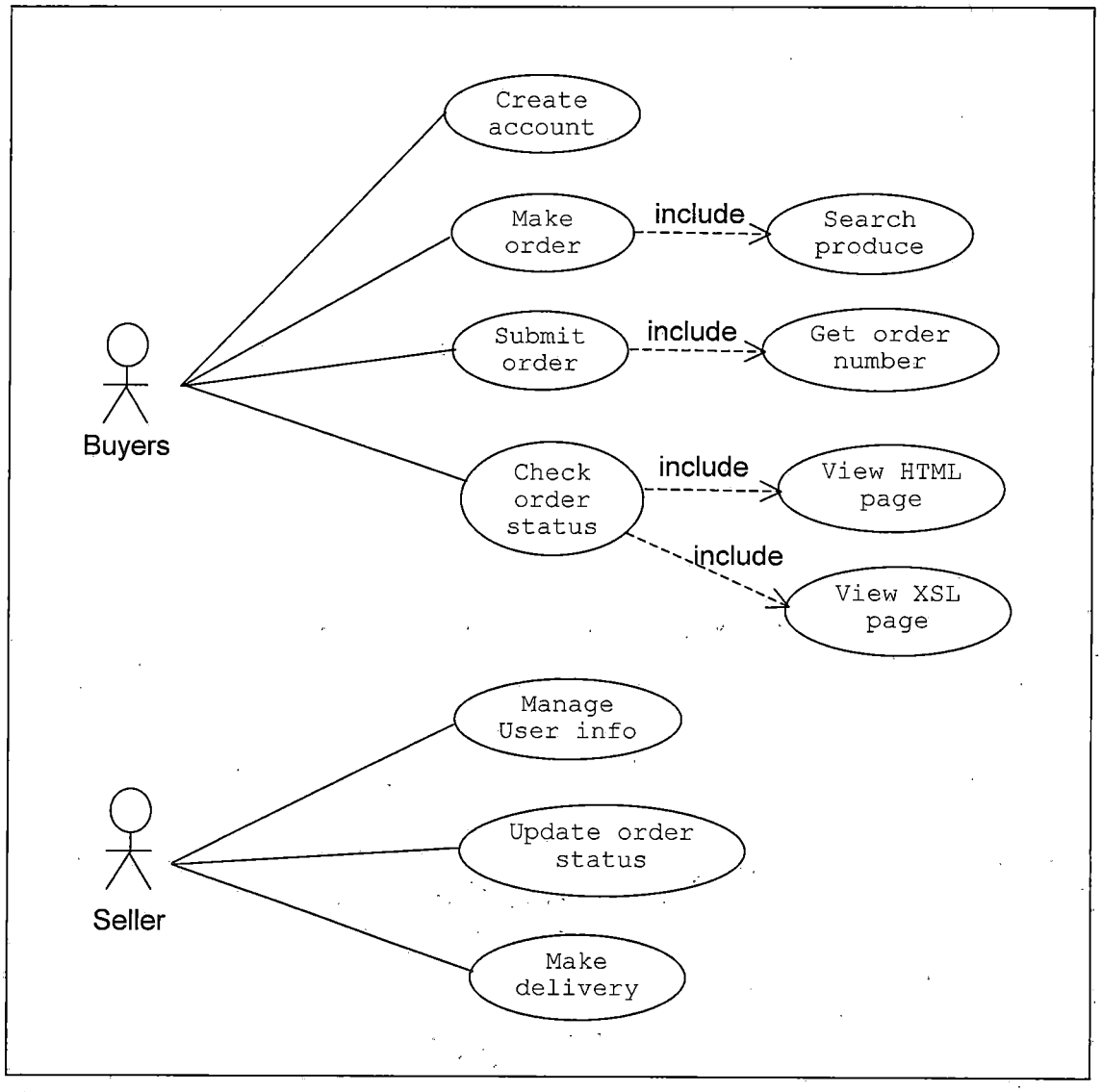

Figure 5. Use Case Diagram

4.1 Trading Information Processing System Graphical User Interface Design

TIPS GUI is easy for everyone to use. The GUI is written using Hyper Text Markup Language (HTML) Version 6.0 forms and frames. Also, it also uses JavaScript to check the accuracy of the user's input. Hence, the TIPS GUI is executable under Internet Explorer 5.0 or greater. All the inputs that are not acceptable by the system will be reported by an error message page. Hence, the TIPS GUI is executable with browsers that support JavaScript. The following sub-sections explain the GUI work and details.

4.1.1 Trading Information Processing System Home Page

This page will be the first page that all the users will see when they enter TIPS. The user logs in by providing a username and a password. After verifying the username and password, the JSP program will send the page to main menu. Also, the program will record the username into the session for later use. The menu program will display a different user menu depending on the user role (authority value). If the username or password is incorrect, the program will show an error message and the user can re-login.

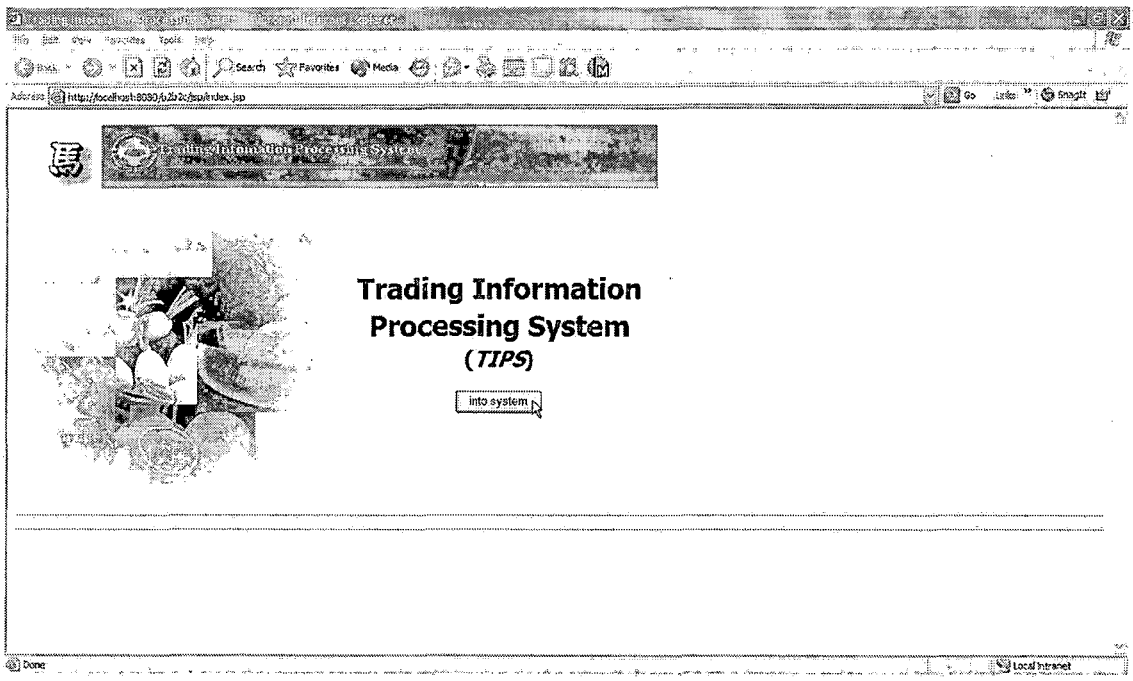

Figure 6. Home Page of Trading Information Processing System

Users enter the system through the TIPs page (Figure 6). After the users click "into system" button then users can go to the next page.

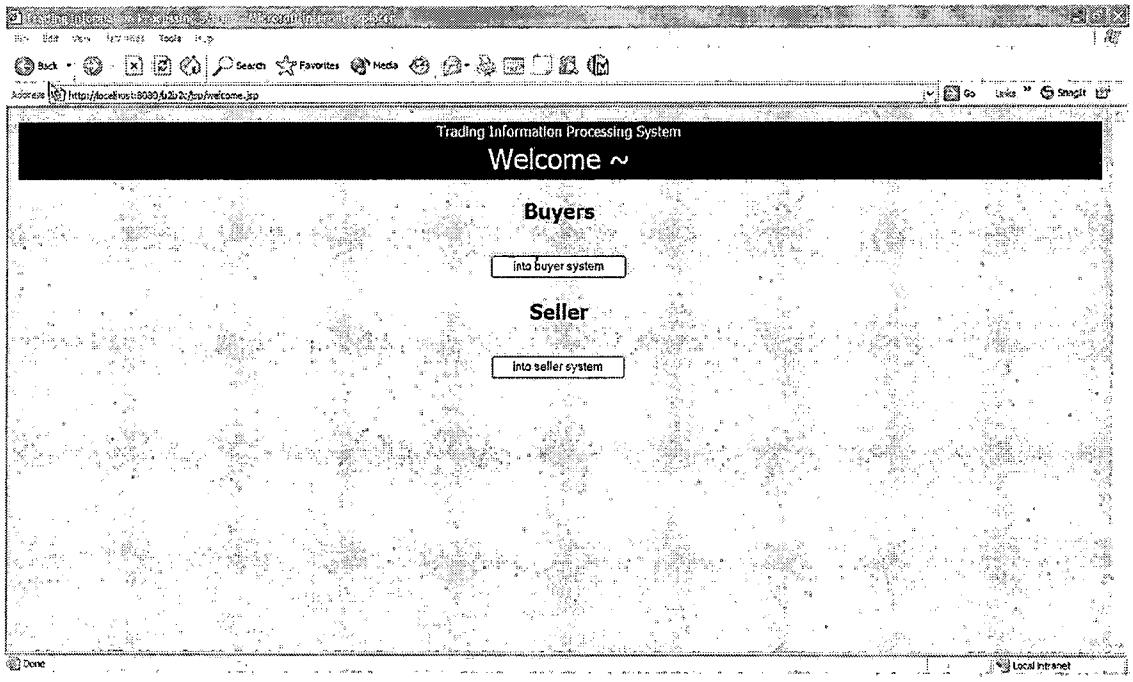

Figure 7. Choose to be a Buyer or a Seller

After clicking "into system" the system will display a welcome page; then users can choose between buyer and seller (Figure 7). If the user chooses buyer, then the system will display the next page.

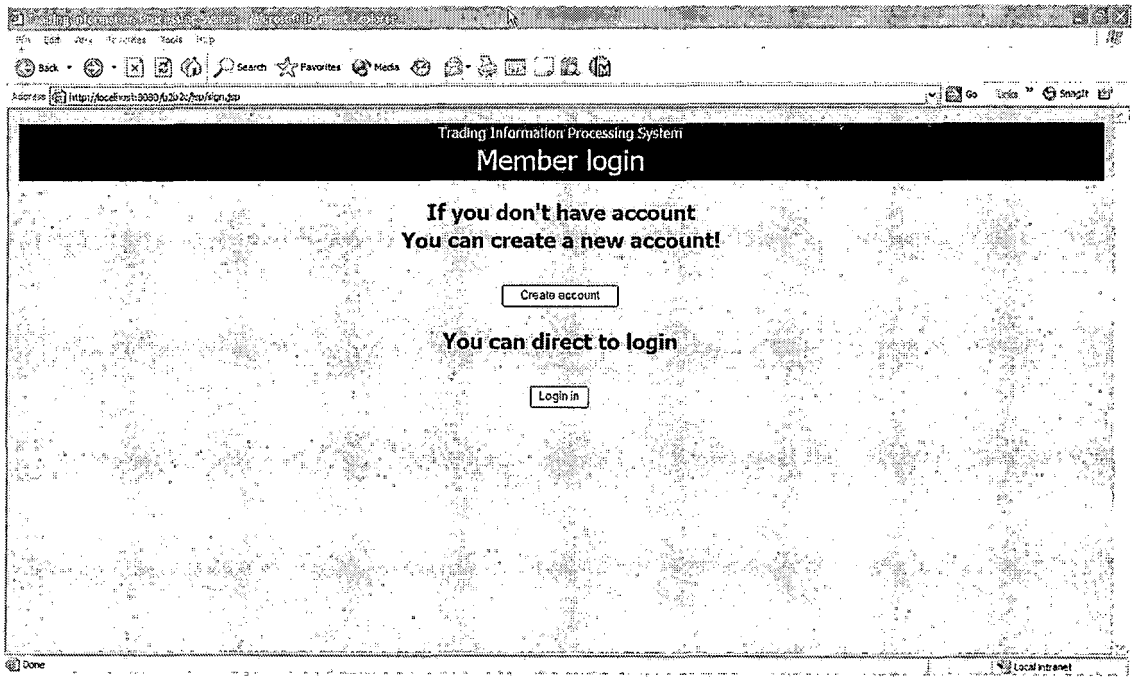

Figure 8. Member Login Page

If user doesn't have an account, then they can create their own account, otherwise the user goes to the login page (Figure 8).

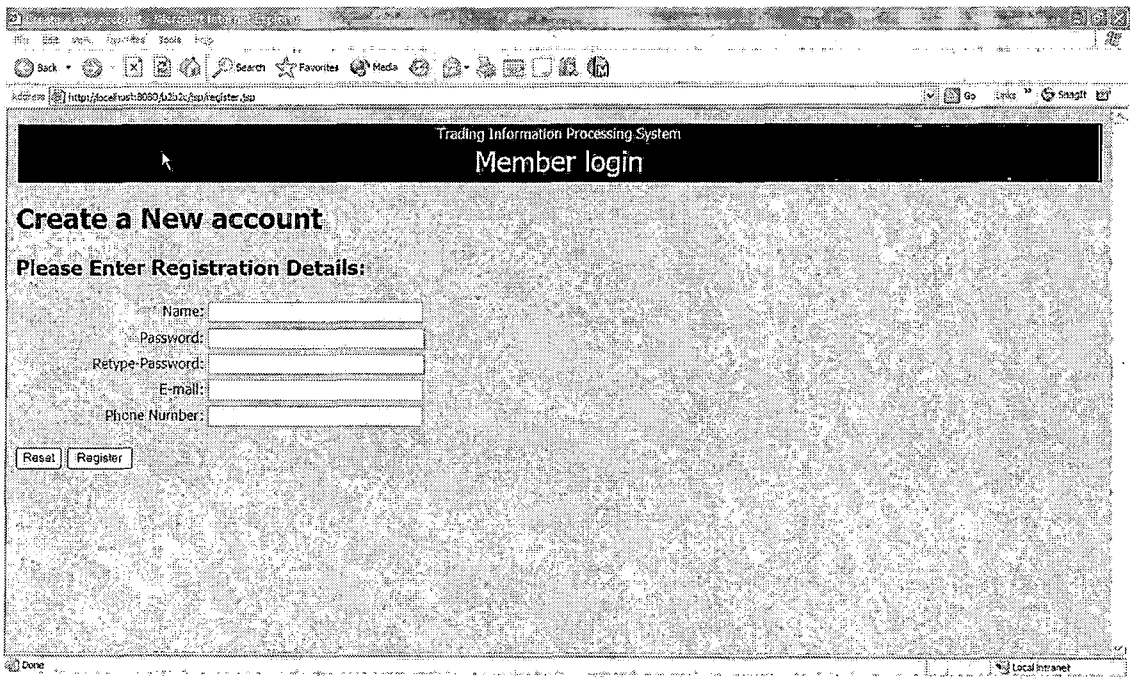

Figure 9. Create Account Page

If users do not have account, they can create a new account in the Create account Page by providing a name, password, E-mail, and phone number (Figure 9).

4.1.2 Trading Information Processing System Login

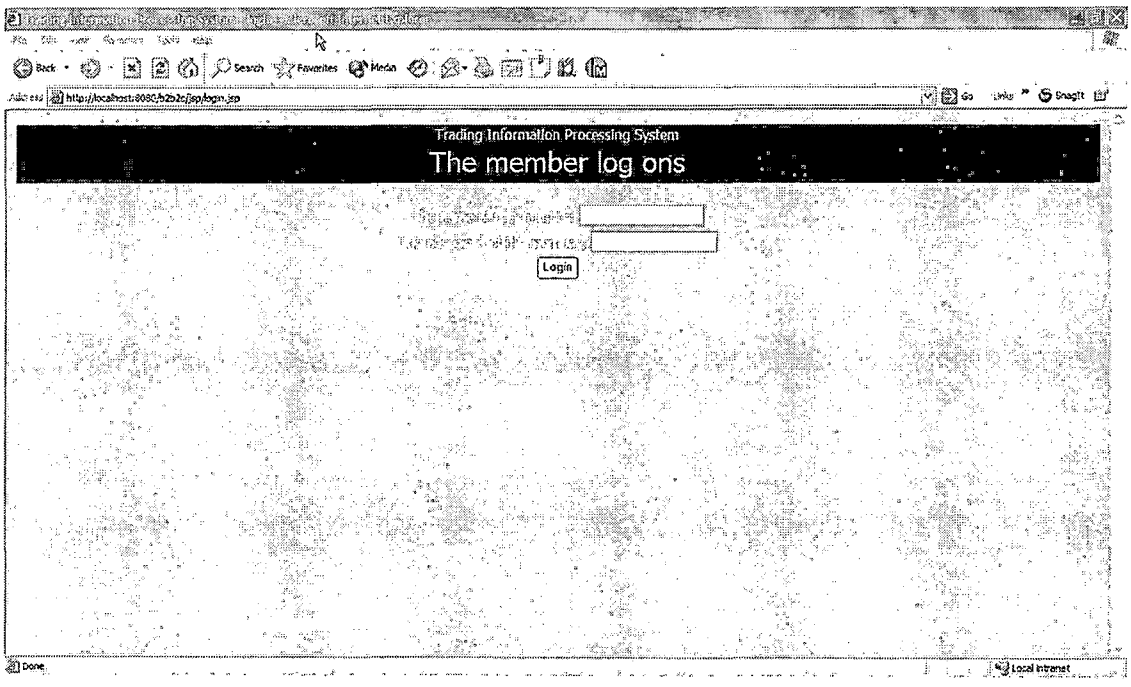

Figure 10. User Login Page

The user logs in by providing a username and a password that are created by him or her self. After the servlet verifies the username and password (Figure 10), it forwards to a jsp page, which will show the main page. 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 name or password is wrong, the

system will show an error message and the user can re-login (Figure 11).

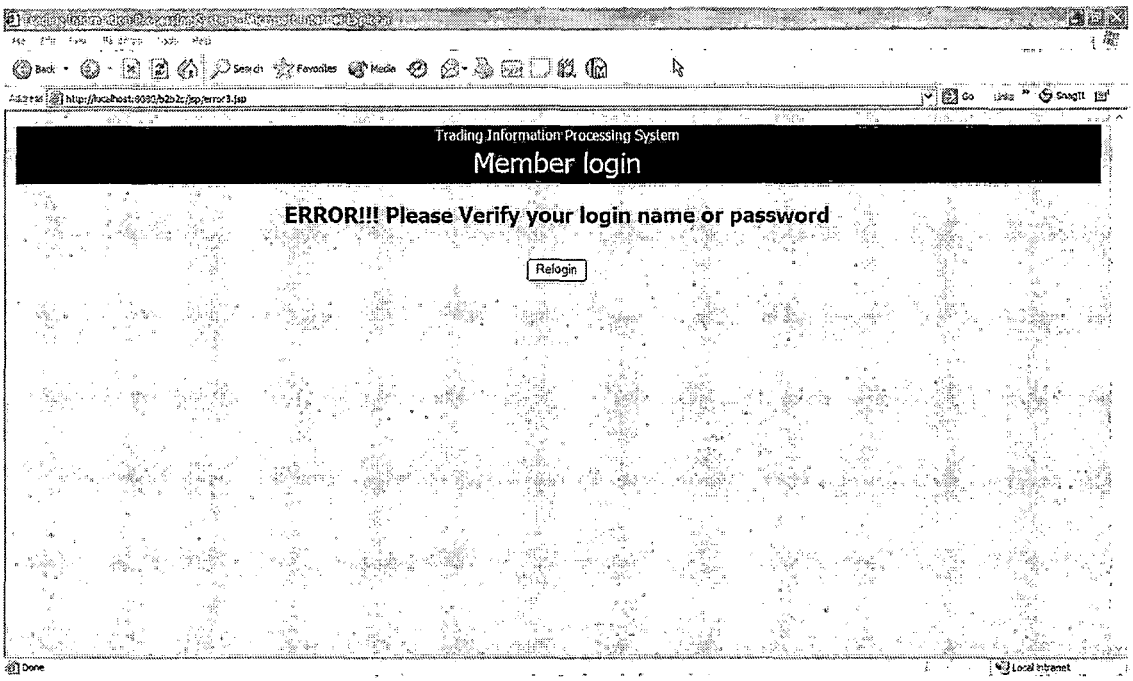

Figure 11. Login Name or Password Error

Moreover, the system does not save the password in a cookie, so that the next time the user visits TIPS, the system will show login page again.

4.1.3 On-Line Trade Transaction

This function is the most important part in this project. After trading and the JSP program will check the required information before submit it. Then the JSP program will save all the information into session. Also, after the data save and store in database server.

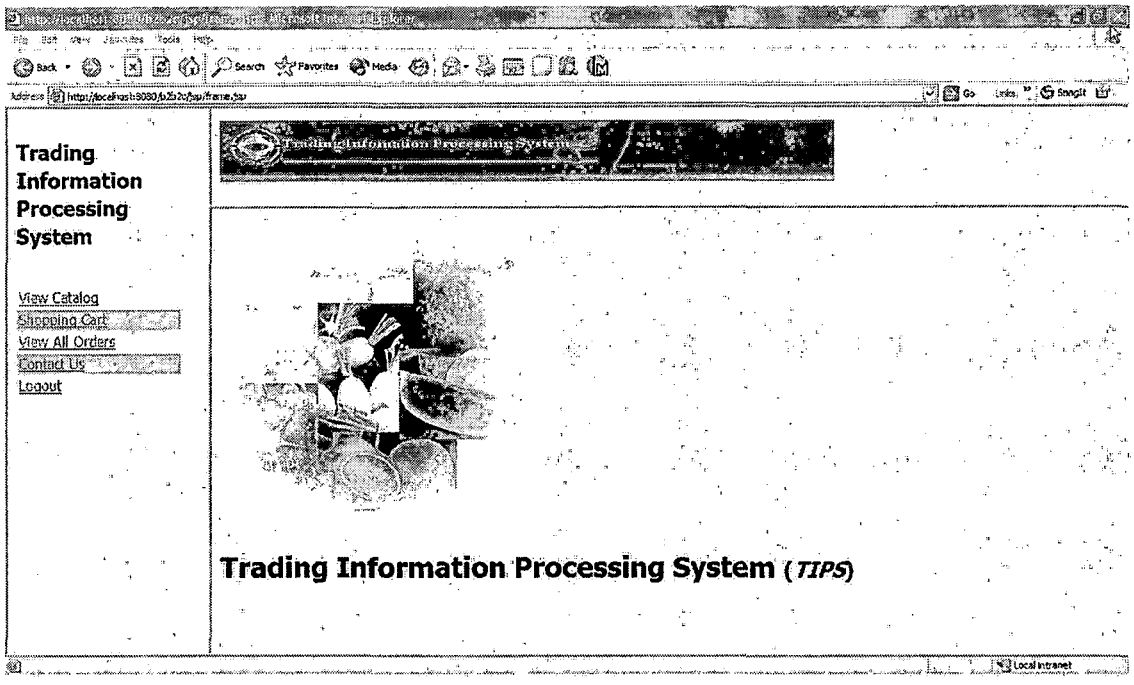

Figure 12. Main Page for Trading Information Processing System

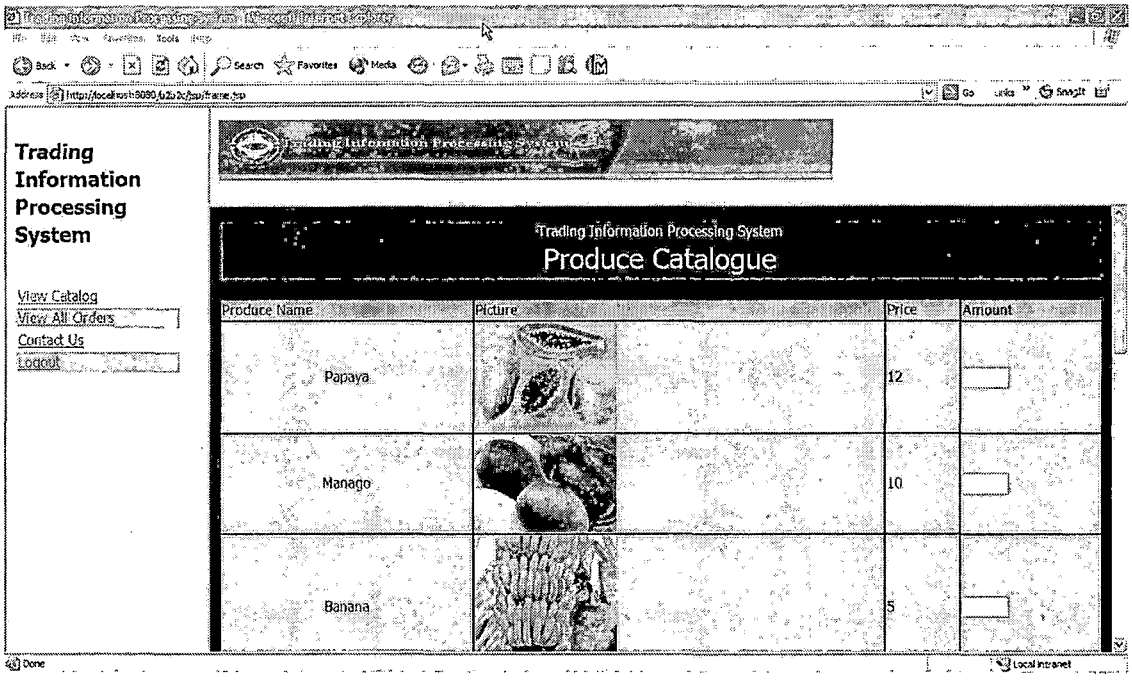

Figure 13. View Catalog Page

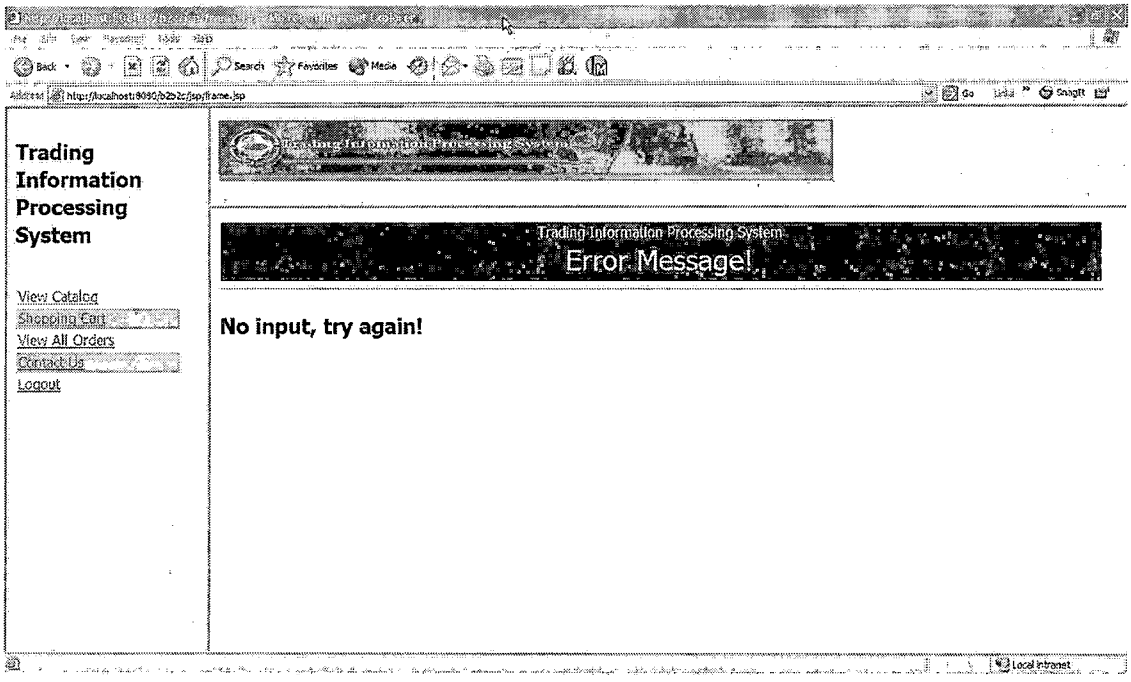

Figure 14. Shopping Cart Page for No Input

If no input at Shopping Cart then after three seconds will directly forward to Catalog page (Figure 14).

Figure 15. Shopping Cart Page

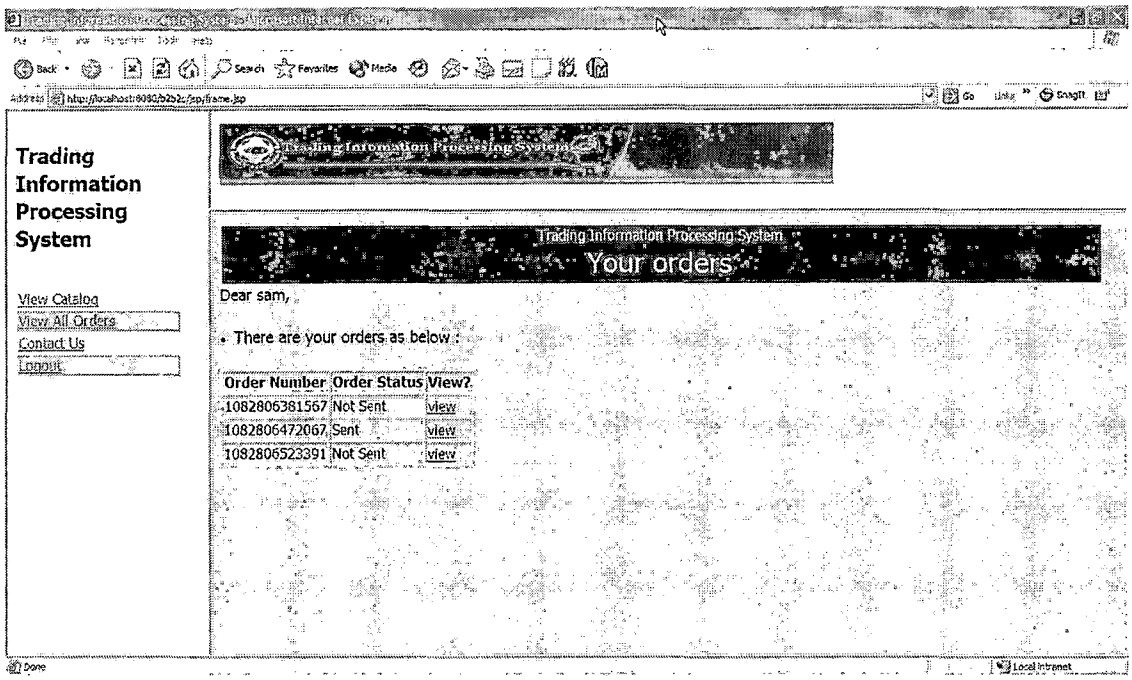

Figure 16. View All Orders

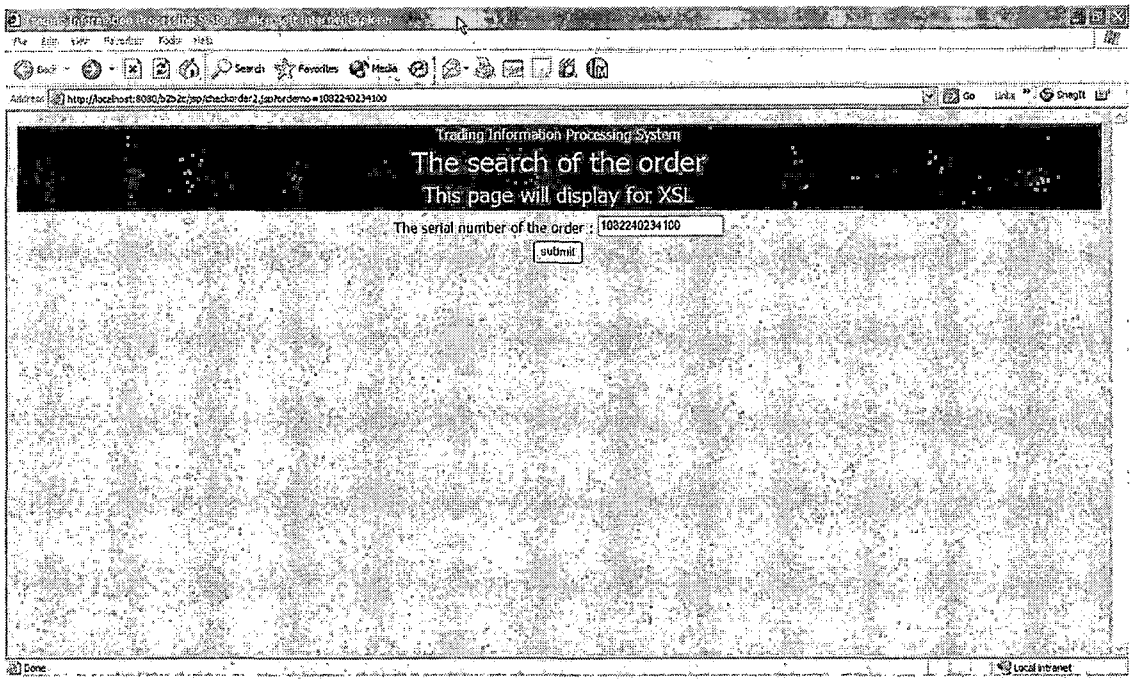

Figure 17. Check Order Page for XSL

In the "view all orders" page if you choose "view as XSL page" (Figure 17), the system will display another page using the XML/XSL technique. The web page links to the JSP script "processcheckorder2.jsp", which displays the order obtained from the database, including the product name, quantity and the order status, which is retrieved by invoking a SOAP Web Service. After SOAP Web Service transacts and transfer file to "checkorder.xsl" file and display whole data to HTML page. Eventually, the browser will show all information on "order number.xml" page to the members.

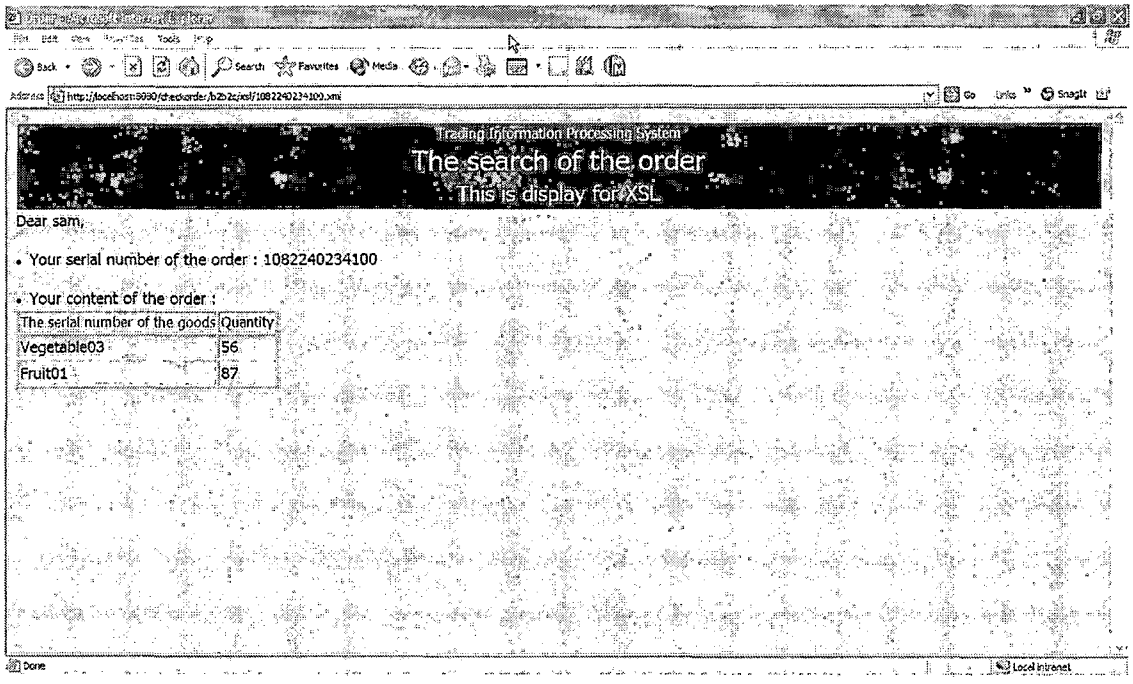

Figure 18. Display for XSL Page

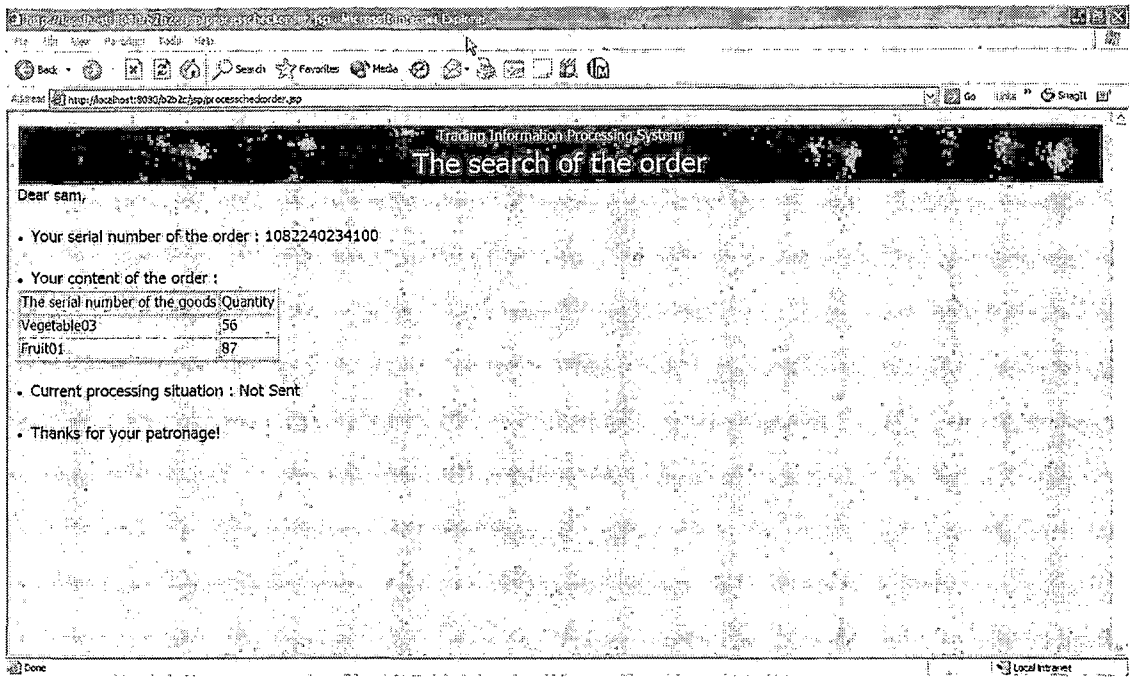

Figure 19. Display for HTML Page

In the "view all orders" page if you choose "view as HTML page" (Figure 19), the system will display another page using the SOAP method. The web page links the JSP script "processcheckorder.jsp", which displays the order obtained from the database, including the product name, quantity and the order status, which is retrieved by invoking a SOAP Web Service. Finally, the browser will show all information on HTML to the members.

4.1.4 Contact Us for Trading Information Processing System

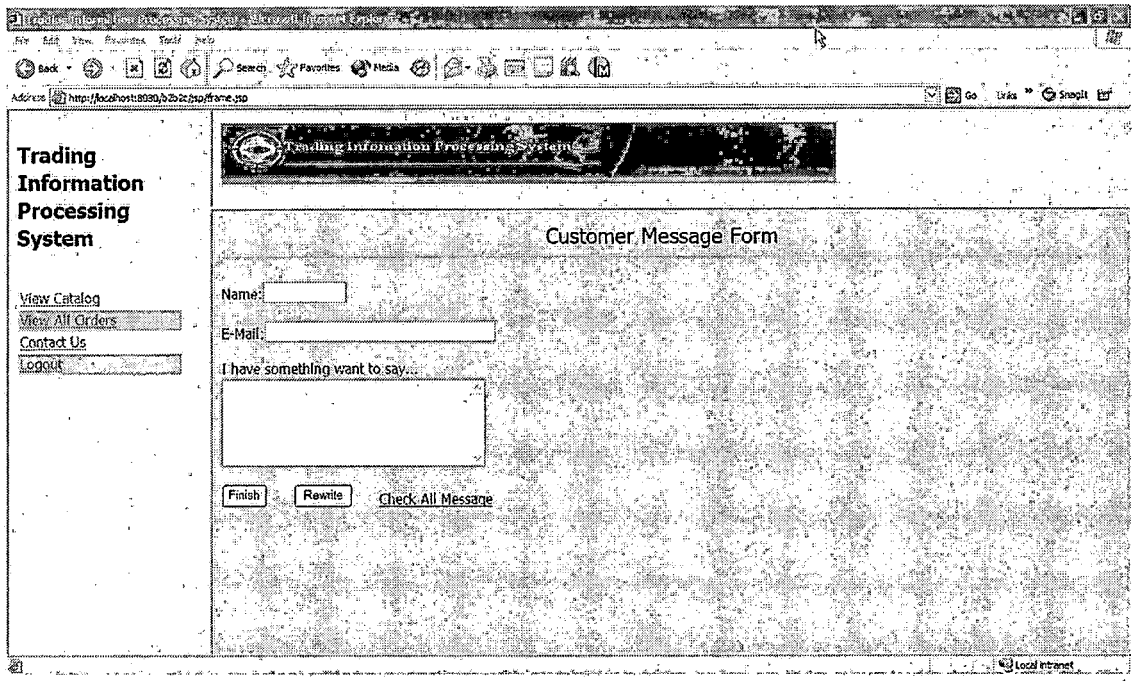

Figure 20. Contact Us Page

If users have any suggestions, they can give some advice on this page for future improvements. Some possible suggestions could include comments regarding the ease of use of the online order process, the ability to track the status of an order, the condition in which a shipment arrived, and the satisfaction with quality of produce.

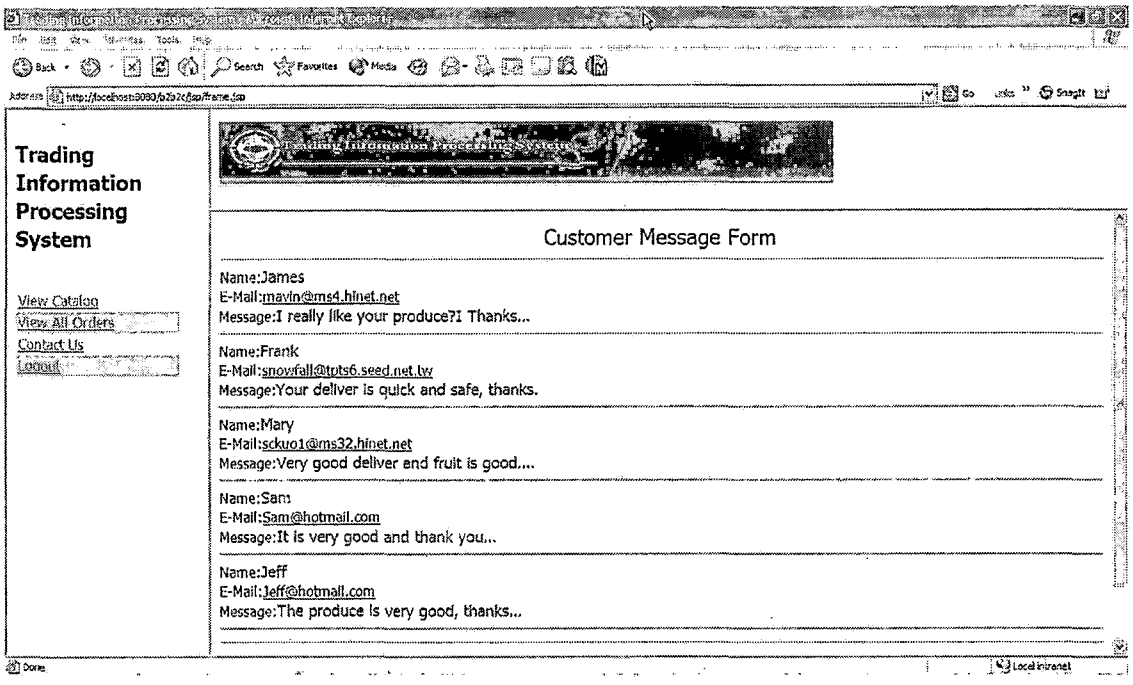

Figure 21. Customer Message Form

4.1.5 Seller Page (Administrator's function)

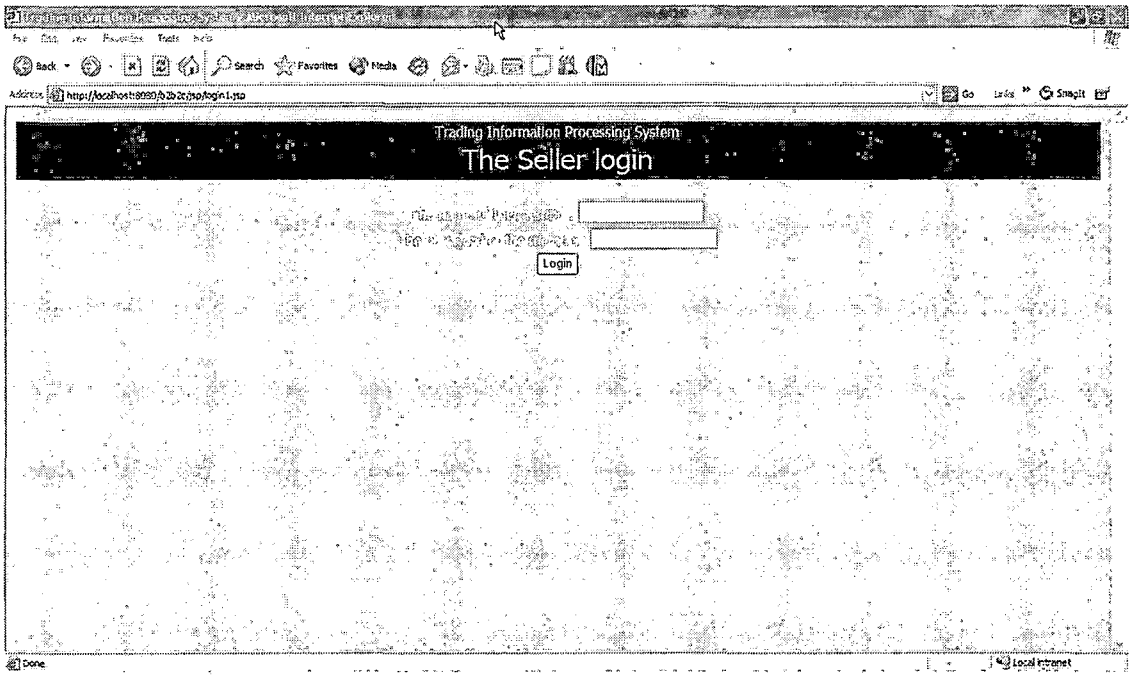

Figure 22. Seller Login Page

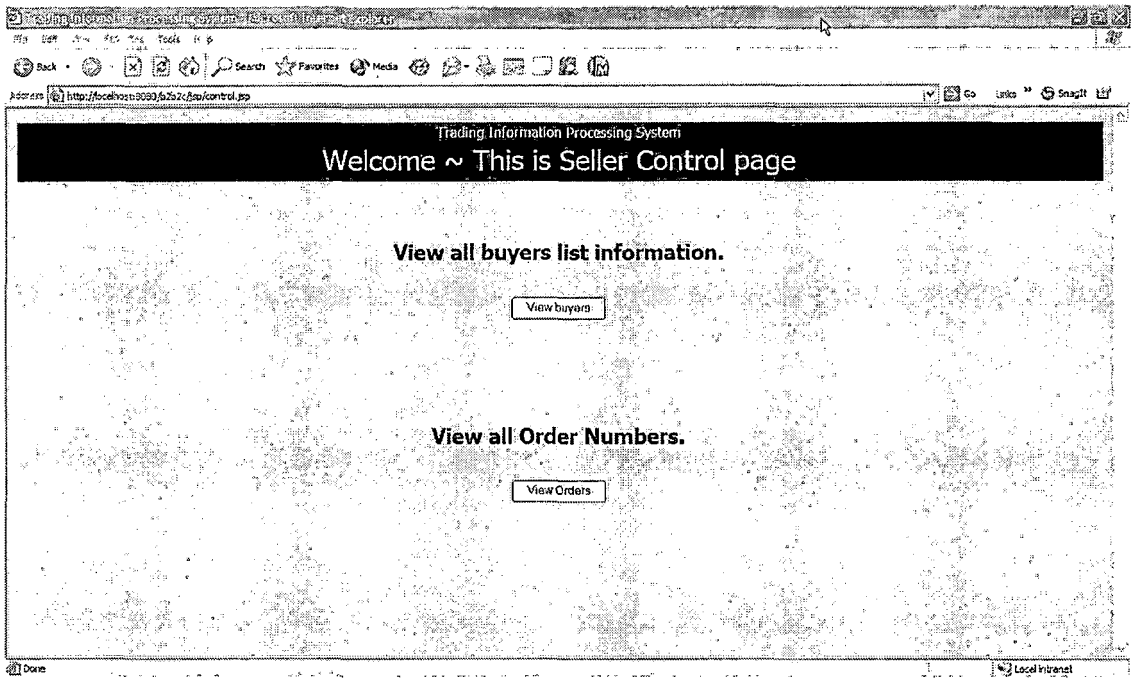

Figure 23. Seller Control Page

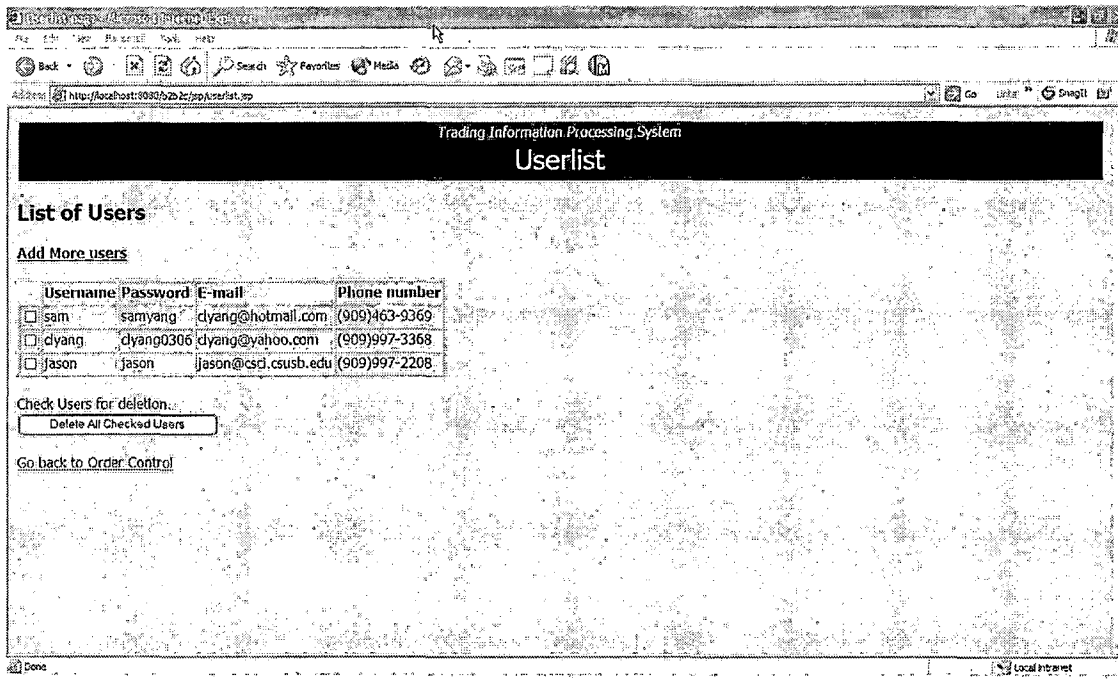

Figure 24. All Users Information Page

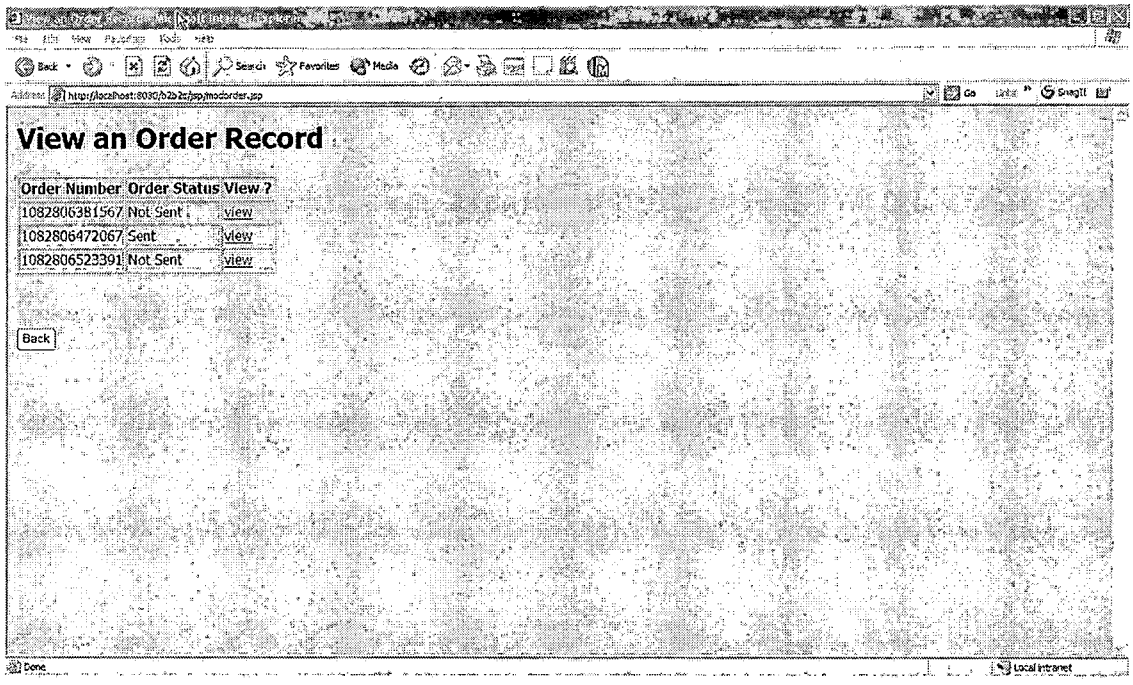

Figure 25. View All Orders Page

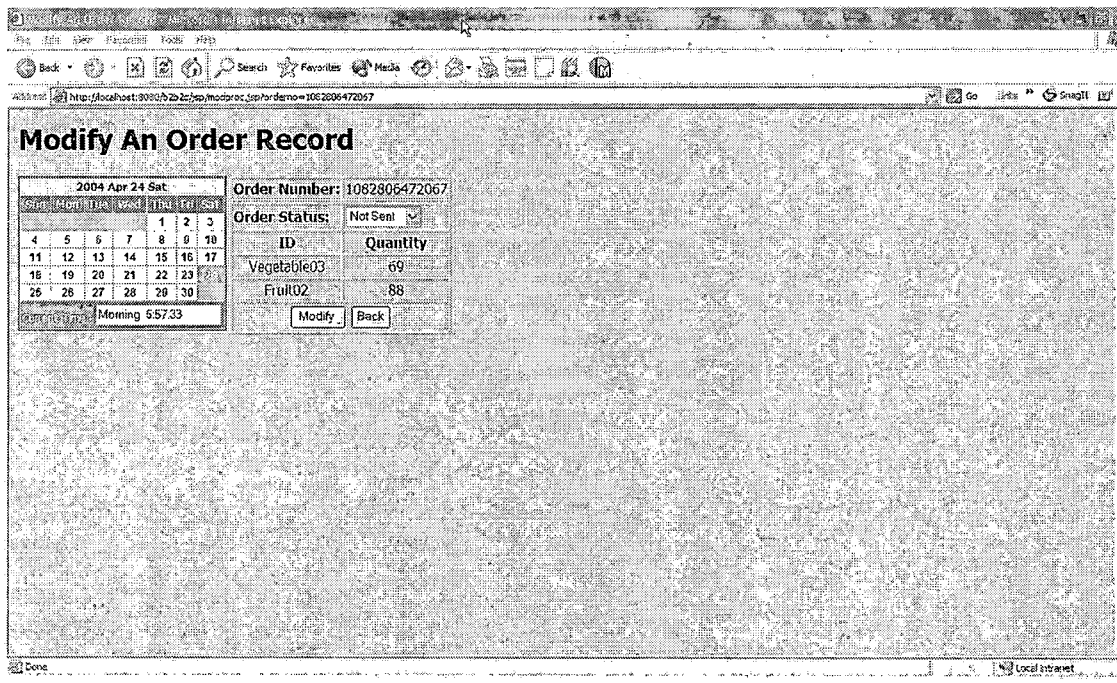

Figure 26. Modify an Order Page

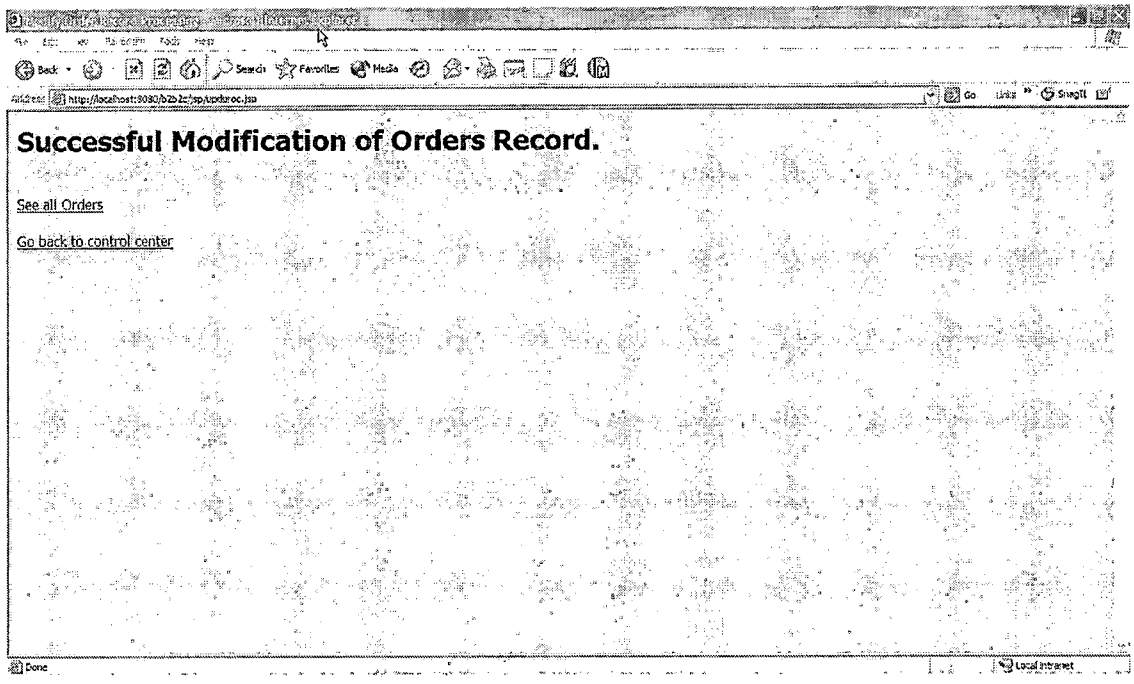

Figure 27. Successful Modification of Order Page

4.1.6 Logout for Trading Information Processing System

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

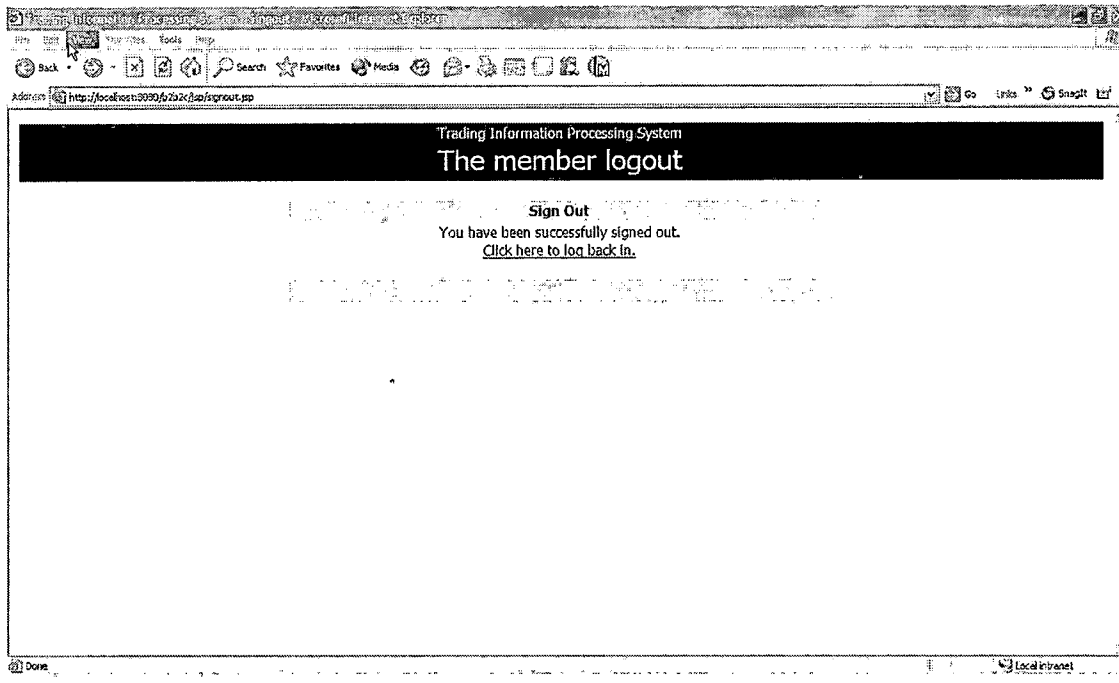

Figure 28. Logout Page

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

CHAPTER FIVE

SECURITY

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

5.1 Login Page

When the buyer enters the system, the system will require the buyer to enter username and password. If the username is not found in database or the password is incorrect for the input username, then the system will show the error message and ask the buyer to re-login again.

5.2 Authority Variable

The authority variable is used to distinguish between member customers and the seller. After the buyer enters the login page, the system will set the authority variable into session depending on the buyer's type in the database. This variable will be checked in every page. If the authority variable is not allowed to browse this specific page, then the system will send the buyer directly to the login page. Thus, even though remembering

the program name, the buyer can't just input address into URL field without login. This variable will be removed after logout, so the next buyer has to login again to set his authority variable.

CHAPTER SIX

SYSTEM VALIDATION

The system validation test is a kind of test process that 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.

6.1 Unit Test

Unit test is the basic level of testing where individual components are tested to ensure that they operate correctly. These individual components can be object, class, program and etc. The unit testing results of TIPS are shown in Table 5.

Table 5. Unit Test Results (Forms)

Forms	Tests Performed	Results
Login Page	• Verify handling valid data input. • Check all the links and buttons work properly. • Check the error message show correctly for any error input.	Pass
Main Page	• Make sure the menu in the left frame works. • Check all the links work as expected. • Check the hyperlink is work correctly. • Check the bottoms are work	Pass

Forms	Tests Performed	Results
	correctly. • Verify the Catalog function works correctly. • Verify the Shopping Cart function works correctly. • Verify the Check order function works correctly. • Verify the Check order for XSL function works correctly. • Verify the View all orders function works correctly. • Verify the contact us function works correctly. • Verify the Logout function works correctly.	
Create a New Account Page	• Verify handling valid data input. • Check all the buttons work properly.	Pass
Catalog Page	• Check all the links work as expected. • Check all message headers are correct. • Check the produce of total quantity is correct. • Check the produce of price is correct. • Check the hyperlink works correctly. • Check the all bottoms are work correctly.	Pass
Shopping Cart	• Check all the links work as expected. • Check all message headers are correct. • Check the produce of total quantity is correct. • Check the produce of price is correct. • Check the hyperlink works correctly.	Pass
Check Order Page	• Check all the links work as expected. • Check all message headers are correct. • Check the produce of total quantity is correct. • Check the produce of price is correct. • Check the all bottoms are work correctly.	Pass

Forms	Tests Performed	Results
	• Verify the check order function work correctly. • Check the hyperlink works correctly.	
Check Order Page for XSL	• Check all the links work as expected. • Check all message headers are correct. • Check the produce of total quantity is correct. • Check the produce of price is correct. • Check the all bottoms are work correctly. • Verify the check order function work correctly. • Check the hyperlink works correctly.	Pass
View All Orders Page	• Make sure the menu in left bottom frame works and all id and quantity show correctly. • Make sure the item list show decreasingly. • Check all the links work as expected. • Check the hyperlink works correctly. • Check all message headers are correct. • Verify the view all orders function work correctly. • Check each message can read correctly.	Pass
Modify Order Status Page	• Make sure the menu in left bottom frame works and all item and quantity show correctly. • Make sure the list show increasingly. • Check all the links work as expected. • Check all message headers are correct. • Check the hyperlink works correctly. • Verify the modify function work correctly. • Check each message can read correctly.	Pass
Create a new user Page	• Make sure message with plain text and html text can be read correctly.	Pass

Forms	Tests Performed	Results
	• Make sure all bottons showed correctly. • Check "Register" and "Reset" botton is work correctly. • Check trash function works correctly. • Check the hyperlink works correctly.	
Delete a user Page	• Check all entries show correctly. • Make sure all input data stored in session. • Verify function works correctly. • Make sure checkbox is works correctly. • Verify all error messages show correctly. • Check function return works correctly. • Check all buttons work correctly.	Pass
View all order number Page	• Check all entries show correctly. • Make sure all input data stored in session. • Check all buttons work correctly. • Verify all error messages show correctly. • Check all function works correctly.	Pass
Logout Page	• Check all session is removed. • Check the explorer can not return pages. • Check the link works properly.	Pass

6.2 Subsystem Testing

Subsystem testing is the next step up in the testing process where all related units from a subsystem to do a certain task. Thus, the subsystem test process is useful for detecting interface errors and specific functions. Table 6 show subsystem test results in detail.

Table 6. Subsystem Test Results (Class: DataBase)

Subsystem	Tests Performed	Results
Create a new account for a new user	• Test all the required input. • Check all the input in session variable. • Check the proper form appears. • Make sure all users data were packaged together. • Ensure data security during transmission.	Pass
Trade information	• Test and check all the trade are correct. • Test after trade with different formatted and database.	Pass
View all Orders	• Check all database information was created correct. • Check SQL function works correctly.	
Database Maintenance Subsystem	• Test inserts / edit / update / delete function of each table. • Check all the information in database.	Pass

6.3 System Testing

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

System testing of TIPS system begins with the following steps (Table 7):

Table 7. System Test Results

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

CHAPTER SEVEN

MAINTENANCE MANUAL

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 carefully this manual step by step. In TIPS, there are three major issues: Software Installation, Variable Installation, and TIPS Installation.

7.1 Software Installation

TIPS requires JSDK, Simple Object Access Protocol (SOAP), Java servlet package, TOMCAT, MySQL, and JDBC to run the programs. The following will detail the installation of that software.

7.1.1 JAVA 2 Platform, Standard Edition (J2SE)

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

7.1.2 Simple Object Access Protocol (SOAP)

SOAP is a lightweight protocol for exchange of information in a decentralized, distributed environment. http://xml.apache.org/soap/ to download the file soap-bin-2.0.tar.gz, and extract it to C:\Jakarta-tomcat\webapps\soap

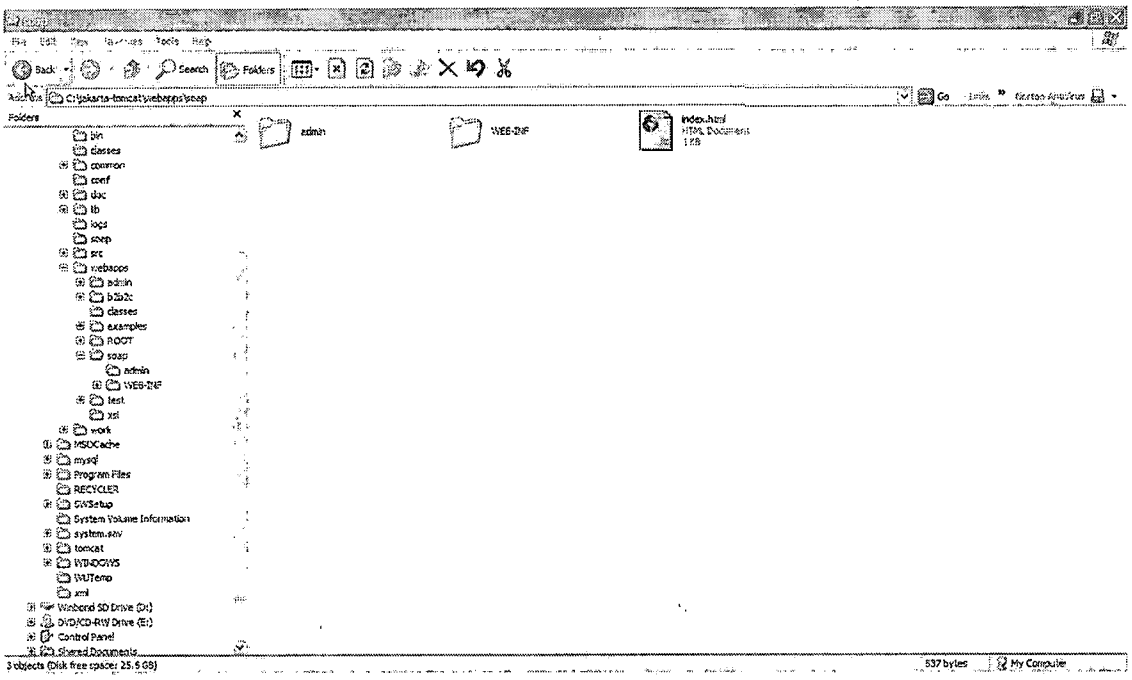

Figure 29. SOAP files

7.1.3 Tomcat

TOMCAT is one of Jakarta apache project which is a JAVA container to process JSP programs and construct a web server for web pages. First of all, we go to http://apache.mirrorcentral.com/dist/jakarta/tomcat-4/binaries/ to download the file tomcat-4.1.18.zip and extract

it to hard driver. Also, we copy C:\tomcat\bin\startup.bat and shutdown.bat to the desktop as shortcut in order to easily start and shut sown tomcat.

7.1.4 MySQL Installation

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

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

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

```
C:\mysql\bin>net start mysql
```

to start our service. Forth, we have to setup the user and password, the default user is named 'ROOT', so we have to set its password to clyang0306.

```
C:\mysql\bin>mysqladmin -u root password clyang0306
```

After that, we can try to input following command:

```
C:\mysql\bin>mysqld
mysql>select * from user
mysql>exit
```

C:\mysql\bin>

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

7.1.5 JAVA Database Connectivity (JDBC)

The API used to execute SQL statement is different for each database engine. Java programmers, however, are lucky and are freed from such database portability issues. They have a single API, the Java Database Connectivity API (JDBC), that's portable between database engines. The JDBC library provides an interface for executing SQL statements. It provides the basic functionality for data access. A number of drivers are available for MySQL, and information about this can be obtained at the MySQL homepage at http://www.mysql.com/downloads, under JDBC. For our purpose, we will use the MM.MySQL driver which is a Type-4 JDBC driver that is under the GNU Library License.

7.2 Variables Modification

In the TIPS, we have to change some environment variables in windows system, server.xml in Tomcat server, and setting for startup.bat, shutdown.bat, and JDBC.

7.2.1 System Variables

1. Go into the Control Panel and open up the System control panel application.
2. You should have a window entitled System Properties on your screen; select the Advanced tab, and click on the Environment Variable button.
3. A new window named Environment Variables should have opened. Click on the new button in the System Variables Section.
4. The New System Variable window should now be on your screen. Enter JAVA_HOME for as the name and the path to your JDK (such as c:\j2sdk1.4.1_01) for the value.
5. Repeat step 3, and enter CATALINA_HOME for as the name and the path to your TOMCAT (such as c:\tomcat) for the value.
6. Repeat step 3, and enter CLASSPATH for as the name and c:\tomcat\common\lib\servlet.jar; c:\mm.mysql.jdbc-2.0pre5\mysql_2_comp.jar; c:\mm.mysql.jdbc-2.0pre5\mysql_2_uncomp.jar; for the value. c:\tommat is the TOMCAT path and c:\mm.mysql.jdbc-2.0pre5 is the path for JDBC.

7.2.2 Batch Files Modification

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

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

7.2.3 Copying Files

Copy following two files,

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

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

to c:\tomcat\common\lib.

And

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

struts.jar

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

Before compile need to set up classpath include as following files:

xerces.jar: Apache Xerces XML parser package

soap.jar: Apache SOAP package

jdom.jar: Jdom XML parser package

servlet.jar: Sun Servlet package

sax2.jar: SAX version 2 XML parser package

7.3 Trading Information Progress System Installation/Migration

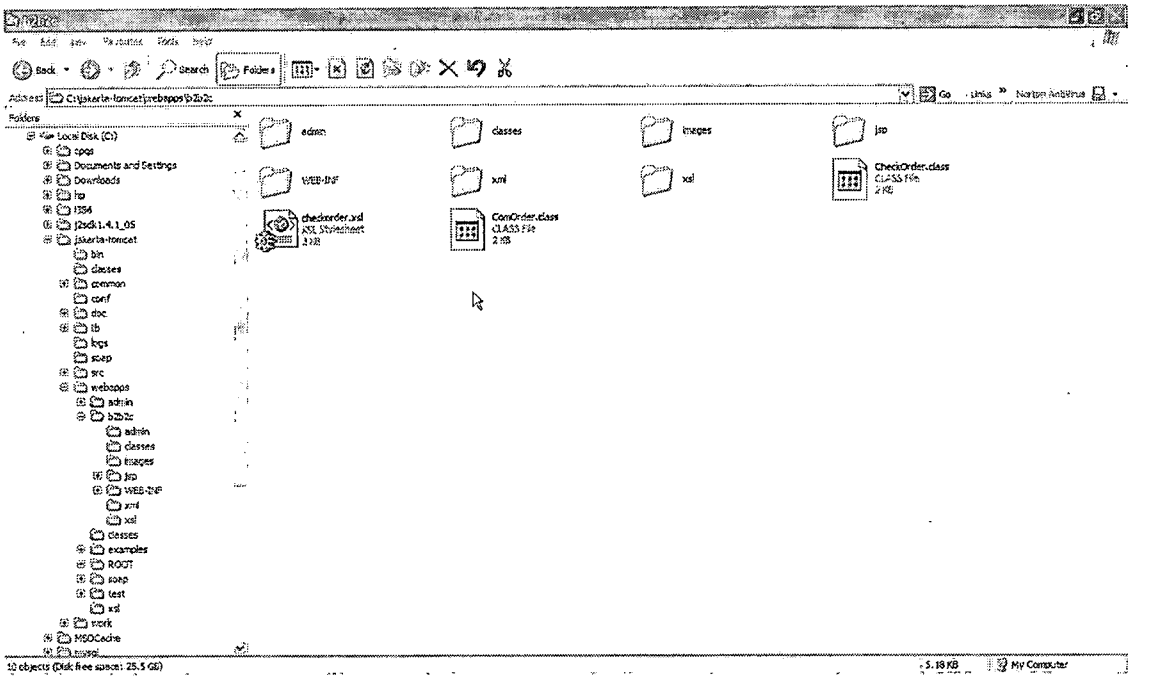

Figure 30. Project File Directory

1. All the JSP programs and HTML programs are stored in different folders %CATALINA_HOME%\webapps\b2b2c\jsp\
   For login.jsp, loginauth.jsp, index.jsp…etc.
2. All the images are stored in %CATALINA_HOME%\webapps\b2b2c\images\
3. All the classes are stored in %CATALINA_HOME%\webapps\b2b2c\WEB-INF\classes\b2b2c\

For Item.class, CartBean.class, OrderItem.class…etc.

4. All the database files are stored in %MYSQL%\data\

5. All the XML files are stored in %CATALINA_HOME%\webapps\b2b2c\xml\

6. All the XSL files are stored in %CATALINA_HOME%\webapps\b2b2c\xsl\

7. File web.xml and taglib.tld are stored in %CATALINA_HOME%\webapps\b2b2c\WEB-INF\

8. Setting Tomcat's web.xml as following below:

```
<! -- for XSLT in B2B2C -->
   <servlet>
      <servlet-name>
         checkorder
      </servlet-name>
      <servlet-class>
         Big5XSLServlet
      </servlet-class>
      <init-param>
         <param-name>stylesheet</param-name>
         <param-value>/xsl/checkorder.xsl</param-value>
      </init-param>
   </servlet>
   <servlet-mapping>
         <servlet-name>
```

```
            checkorder2
        </servlet-name>
        <url-pattern>
            /checkorder/*
        </url-pattern>
    </servlet-mapping>
    </servlet>
```

9. Deploy SOAP service:

The SOAP RPC router URL for this SOAP server

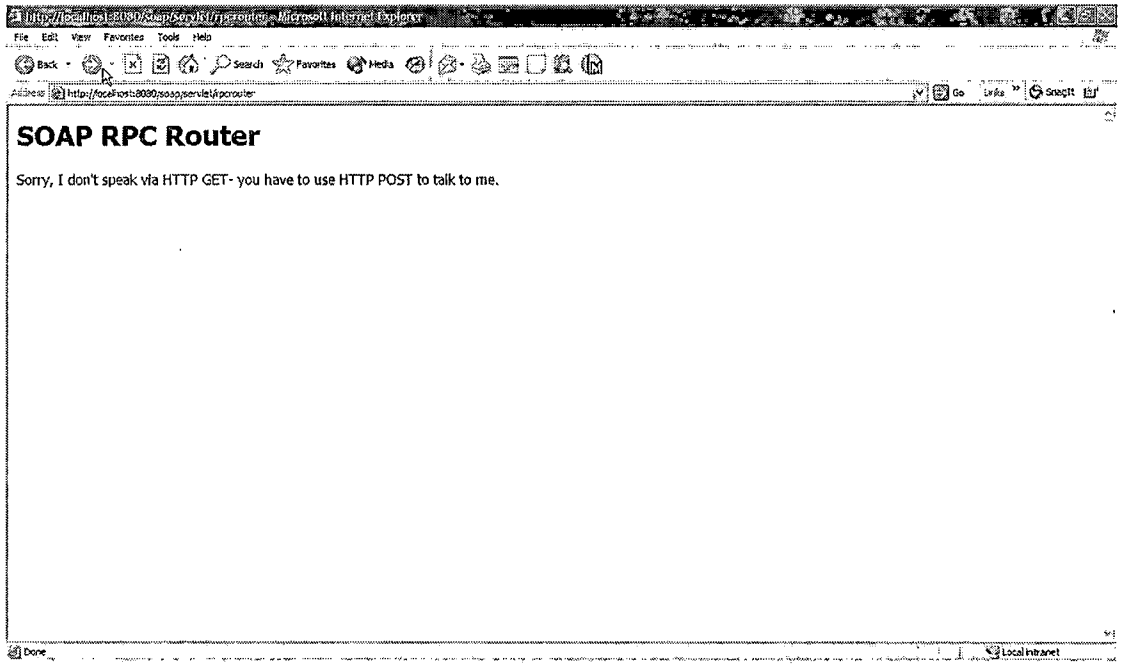

Figure 31. RPC Router Servlet

Although this looks like an error, it does indicate that things are working correctly.

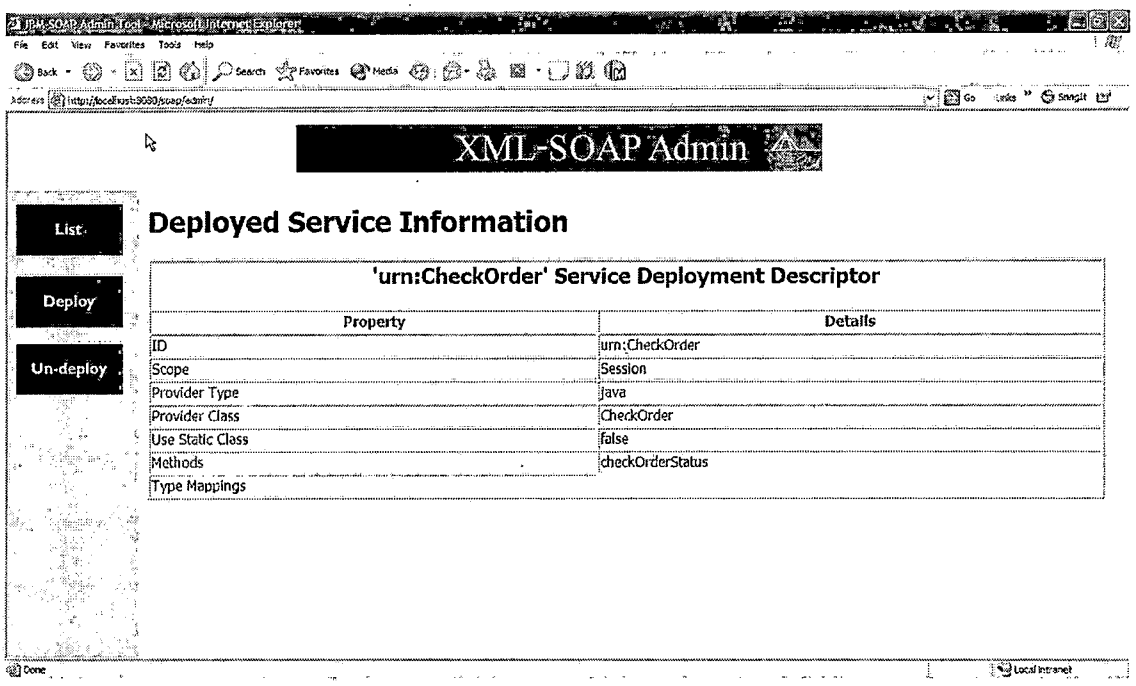

Figure 32. Deploy XML-SOAP Service

7.4 Backup

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 TIPS. There are two steps to back up TIPS. One is to backup the system files. The other step is to backup the database which is used by TIPS.

7.4.1 System Backup

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

software, such as WINZIP or WINRAR, to backup the system programs.

7.4.2 Database Backup

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

CHAPTER EIGHT

CONCLUSION AND FUTURE DIRECTIONS

8.1 Conclusion

The TIPS provides a very nice on-line enterprise E-commerce environment for the retailer members and the seller in a produce wholesale market. Java combined with XML provides developers with powerful tools and technologies for implementing Internet applications. One goal of TIPS is to design a user-friendly interface for retrieving the enquired data easily. Another goal is to investigate the use of XML technologies such as SOAP and XSL to better organize the internal architecture of the system. Internally, data is retrieved in the form of XML documents through a data access layer accessible through SOAP. These XML documents are transformed into HTML using XSL templates.

XML makes data portable. The Java platform makes code portable. The Java APIs for XML make it easy to use XML. Put these together, and one will have the perfect combination: portability of data, portability of code, and ease of use. The SOAP technique is an important part of this project, and demonstrates how to achieve separation

of concerns by loosly coupling business logic with data persistence.

8.2 Future Directions

The TIPS is a system offered to be used for various companies in E-Commerce, such as: business to business. According to electronic data interchange (EDI), which is enterprise with enterprise data interchange standard, has history almost twenty years. But EDI communication media were very expensive. Hence, the XML-EDI will very popular in the future. Java combined with XML provides developers with powerful tools and technologies for implementing Internet applications. In other words, the Java language combined with XML and SOAP technique then will provides more and more powerful technologies for implementing Internet applications.

Application integration services encompass the areas of e-commerce, remote site administrations, remote application automation, and general on-line business operation. Syndication covers the process of aggregating and reselling content, data, and services. This project TIPS is the first version build in fundamental on-line application; it may modify or improve by using Struts framework or other languages. Therefore, it can be

development more and more powerful function for various situation use.

APPENDIX

SOURCE CODE OF JAVA CLASSES

List of Classes

Big5XSLServlet.java

```
import java.io.IOException;
import java.io.File;
import java.io.Writer;
import java.io.InputStreamReader;
import java.io.InputStream;
import java.io.FileInputStream;

import java.util.Enumeration;
import javax.servlet.*;
import javax.servlet.http.*;
import org.xml.sax.*;
import com.jclark.xsl.sax.*;


public class Big5XSLServlet extends HttpServlet {
  private XSLProcessor cached;

  public void init() throws ServletException {
    String stylesheet = getInitParameter("stylesheet");

    System.out.println("Start of Big5XSLServlet....");

    if (stylesheet == null) {
      System.out.println("Big5XSLServlet: init()  stylesheet=NULL");
      throw new ServletException("missing stylesheet parameter");
    }
    cached = new XSLProcessorImpl();
    cached.setParser(createParser());
    try {

      System.out.println("Big5XSLServlet: init()   begin load xsl="+stylesheet);
      InputStream is = getServletContext().getResourceAsStream(stylesheet);
      System.out.println("Big5XSLServlet: init()   load xsl as stream ok");
      cached.loadStylesheet(new InputSource(
        new InputStreamReader(is, "Big5")          ));
      System.out.println("Big5XSLServlet: init()   load xsl ok");
    }
    catch (SAXException e) {
      System.out.println("Big5XSLServlet: init()  SAX ex:"+e.toString());

    }
    catch (IOException e) {
      System.out.println("Big5XSLServlet: init()  IO ex:"+e.toString());

    }
    catch (Exception e) {
      System.out.println("Big5XSLServlet: init()  EX:"+e.toString());

    }
  }


  public void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    System.out.println("Big5XSLServlet: doPost()");
    doGet(request, response);
  }

  public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    System.out.println("Big5XSLServlet: doGet()");
```

```
  File inputFile = new File(request.getPathTranslated());
  if (!inputFile.isFile()) {
    inputFile = new File(request.getPathTranslated() + ".xml");
    if (!inputFile.isFile()) {
          response.sendError(HttpServletResponse.SC_NOT_FOUND,
                "File not found: " + request.getPathTranslated());
          return;
    }
  }

  XSLProcessor xsl = (XSLProcessor)cached.clone();
  xsl.setParser(createParser());
  for (Enumeration e = request.getParameterNames(); e.hasMoreElements();) {
    String name = (String)e.nextElement();
    // What to do about multiple values?
    xsl.setParameter(name, request.getParameter(name));
  }


  OutputMethodHandlerImpl2 outputMethodHandler = new OutputMethodHandlerImpl2(xsl);

  xsl.setOutputMethodHandler(outputMethodHandler);
  outputMethodHandler.setDestination(new ServletDestination(response));

     System.out.println("Big5XSLServlet:  start XSL Transformation");

  try {
    xsl.parse(fileInputSource(inputFile));

  }
  catch (SAXException e) {
    System.out.println("Big5XSLServlet:  SAX Exception:"+e.toString());
    //throw new ServletException(e);
  }
}

static Parser createParser() throws ServletException {
  String parserClass = System.getProperty("com.jclark.xsl.sax.parser");
  if (parserClass == null)
    parserClass = System.getProperty("org.xml.sax.parser");
  if (parserClass == null)
    parserClass = "com.jclark.xml.sax.CommentDriver";
  try {
    return (Parser)Class.forName(parserClass).newInstance();
  }
  catch (ClassNotFoundException e) {
    throw new ServletException(e);
  }
  catch (InstantiationException e) {
    throw new ServletException(e);
  }
  catch (IllegalAccessException e) {
    throw new ServletException(e);
  }
  catch (ClassCastException e) {
    throw new ServletException(parserClass + " is not a SAX driver");
  }
}


static public InputSource fileInputSource(File file) {
  String path = file.getAbsolutePath();
```

```
    String fSep = System.getProperty("file.separator");
    InputStreamReader isr = null;

    if (fSep != null && fSep.length() == 1)
      path = path.replace(fSep.charAt(0), '/');
    if (path.length() > 0 && path.charAt(0) != '/')
      path = '/' + path;

    try {


      FileInputStream fis = new FileInputStream(file);
      isr = new InputStreamReader(fis,"Big5");


            // --- original
     // return new InputSource(new URL("file", "", path).toString());
    }
    catch (IOException e) {

            System.out.println("Big5XSLServlet:  fileInputSource() IOException:");
      System.out.println(e.toString());
    }
    catch (Exception e) {

            System.out.println("Big5XSLServlet:  fileInputSource() Exception:");
      System.out.println(e.toString());
    }

    return new InputSource(isr);

  }

}
```

CartBean.java

// this program control shopping cart's items transfer to order.xml and update database inventory

```
package b2b2c;

import java.util.Hashtable;
import java.util.Vector;
import java.util.Enumeration;

import org.jdom.Document;
import org.jdom.Element;
import org.jdom.JDOMException;
import org.jdom.input.SAXBuilder;
import org.jdom.output.XMLOutputter;

import java.io.*;
import java.net.*;
import java.sql.*;
import java.lang.System;

import javax.servlet.*;
import javax.servlet.http.*;

public class CartBean  {

   Hashtable params=new Hashtable();
```

```
// Hashtable items=new Hashtable();
Vector items = new Vector();

  private static final String DEFAULT_SAX_DRIVER_CLASS =
   "org.apache.xerces.parsers.SAXParser";


private String saxDriverClass;

private SAXBuilder builder;

private Document doc;
private Element root;

// Save order file as order.xml
private String xmldoc = ".. /xml/order.xml";


  private String orderNo;

  //... JDBC ....
  private Connection con;
  private PreparedStatement pstmt_update_stock_qty;
  private PreparedStatement pstmt_select_stock_qty;
  private PreparedStatement pstmt_insert_order;
  private PreparedStatement pstmt_insert_orderitem;

  //-------------------------------------
public CartBean() { System.out.println("start CartBean."); }


      public static void main(String args[]) {
            CatalogBean catalog = new CatalogBean();
            CartBean cart = new CartBean();
            catalog.startParse();

            cart.addItem( catalog.getItem(1) );
            cart.addItem( catalog.getItem(2) );

            cart.toXML();

      }

  public void processRequest(HttpServletRequest req) {
  Enumeration en = req.getParameterNames();
  while (en.hasMoreElements()) {
      String varName = (String)en.nextElement();
      String[] varValues = req.getParameterValues(varName);
      if (varValues.length > 1) {
            params.put(varName,  varValues);

            }
      else {
            params.put(varName, varValues[0]);
            System.out.println("req:"+varName+"="+varValues[0]);
      }
  }
  }

// transfer browser's argument to hashtable
public String getParam(String key) {
     return (String)params.get(key);
```

```
    }

    public void setParam(String key, String value) {
      if (params.containsKey(key))
          params.put(key, value);
       else
            System.out.println("no such key:"+key+" in param Hashtable");

    }

    public String[] getParamArr(String key) {
          return (String[])params.get(key);

    }
//...Start prepare CartBean
      public void init() {
// Clear Cart data
            items.clear();
      }

// add all input item amount is shopping cart
      public void addItems(Vector items) {
// items for catalog
            String qtyParamValue;
            //int qty;
            Item item;
            for (int i=0; i<items.size(); i++) {
                  qtyParamValue = getParam("qty_"+new Integer(i).toString() );

                  if (qtyParamValue!=null && !qtyParamValue.equals("")) {

                  item= (Item)items.get(i);
                   item.setQty(qtyParamValue);  // String

                  addItem(item);
                  }
            }
      }

// input items to shopping cart's vector
      public void addItem(Item item) {

            items.add(item);

            // modify warehouse database stock table
            if (con == null) {
                  createConnection();
            }

            modifyStock(item.getId(), Integer.parseInt(item.getQty()));

            System.out.println("item:"+item.getItemname());
      }

// get shopping cart all items
    public Vector getItems() {
      return items;
    }


// get shopping cart one item
```

```
    public Item getItem(int index) {
      return (Item)items.get(index);
    }

// get shopping cart items
    public int getCartSize() {
      return items.size();
    }

// get order number
      public String getOrderno() {
          return orderNo;
      }

      //.... JDBC .....
      // modify database information
      public void modifyStock(String id, int qty) {
          try {
              int oldQty = 0;
              pstmt_select_stock_qty.setString(1, id);
              ResultSet rs = pstmt_select_stock_qty.executeQuery();
              if (rs.next()) {
                  oldQty = rs.getInt(1);
                  System.out.println("old QTY="+oldQty);


              }

              int newQty = oldQty - qty;

              pstmt_update_stock_qty.setInt(1, newQty);
              pstmt_update_stock_qty.setString(2, id);
              pstmt_update_stock_qty.executeQuery();

              rs.close();

          } catch (SQLException ex) {
              System.out.println(ex.toString());
          }
          catch (Exception ex) {
              System.out.println(ex.toString());
          }
      }

// modify database items
      public void modifyOrder(String username, String email)

      {
          try {


              pstmt_insert_order.setString(1, username);
              pstmt_insert_order.setString(2, email);
              pstmt_insert_order.setString(3, orderNo);
              pstmt_insert_order.executeQuery();


          } catch (SQLException ex) {
              System.out.println(ex.toString());
          }
          catch (Exception ex) {
              System.out.println(ex.toString());
          }
```

```
    }

// modify order item's database
    public void modifyOrderItem(Item item)

    {
        try {


            pstmt_insert_orderitem.setString(1, item.getId());
            pstmt_insert_orderitem.setInt(2, Integer.parseInt(item.getQty()));
            pstmt_insert_orderitem.setString(3, orderNo);
            pstmt_insert_orderitem.executeQuery();



        } catch (SQLException ex) {
            System.out.println(ex.toString());
        }
        catch (Exception ex) {
            System.out.println(ex.toString());
        }
    }

    // prepared database connection and prepared statement
    public void createConnection() {
        try {
       Class.forName("org.gjt.mm.mysql.Driver").newInstance();

     }
     catch (Exception E) {
        System.err.println("Unable to load driver.");
        E.printStackTrace();
     }


        try {
            con =
DriverManager.getConnection("jdbc:mysql://localhost/warehouse?user=root&password=clyang0306");


            pstmt_update_stock_qty = con.prepareStatement("update stock set qty=? where id=?");


            pstmt_select_stock_qty = con.prepareStatement("select qty from stock where id=?");


            pstmt_insert_order =
              con.prepareStatement("insert into customorder (name, email, orderno) values (?,?,?)");

            pstmt_insert_orderitem =
              con.prepareStatement("insert into customorderitem (id, qty, orderno) values (?,?,?)")



        } catch (SQLException ex) {
            System.out.println(ex.toString());
        }
        catch (Exception ex) {
            System.out.println(ex.toString());
        }
```

```
    }

    // output order.xml to /xml/order.xml and close database connection
    public void close()
          throws IOException, JDOMException {

          doc.setRootElement(root);
          startOutputXML(new FileOutputStream(xmldoc));

          //.... close database .....
          try {
                pstmt_update_stock_qty.close();
                pstmt_select_stock_qty.close();
                con.close();

          } catch (SQLException ex) {
                System.out.println(ex.toString());
          }
          catch (Exception ex) {
                System.out.println(ex.toString());
          }
    }

    private void startOutputXML(OutputStream out)
     throws IOException, JDOMException {

     // create an outputter with default formatting
     XMLOutputter outputter = getXMLOutputter();
     outputter.output(doc, out);
}


    private XMLOutputter getXMLOutputter()
     throws IOException, JDOMException {

     // create an outputter with certain encoding
     XMLOutputter outputter = new XMLOutputter("  ", true, "Big5");
     outputter.setTrimText(true);
     outputter.setExpandEmptyElements(true);
     return outputter;
}

    // add item to XML document
    private void addToXML(Item item) {
          // add one element to XML file

    Element curItem =
           new Element("item");

    Element curItemname = new Element("itemname");
    curItemname.addContent(item.getItemname());

    Element curId = new Element("id");
    curId.addContent(item.getId());

    Element curQty = new Element("qty");
    curQty.addContent(item.getQty());

    curItem.addContent(curItemname);
    curItem.addContent(curId);
    curItem.addContent(curQty);

    root.addContent(curItem);
```

```
    }

// output order.xml and modify database
    public void toXML(String username, String email) {
        this.saxDriverClass = DEFAULT_SAX_DRIVER_CLASS;
     builder = new SAXBuilder(saxDriverClass);
    Item item;

 // build new XML doc
    root = new Element("order");
        doc = new Document(root);

        // add timestamp as order's serial number

        Element elemOrderNo = new Element("orderno");
        orderNo = String.valueOf( java.lang.System.currentTimeMillis() );
    elemOrderNo.addContent( orderNo );
        root.addContent(elemOrderNo);

    // system.out.println("add elements to root");
    for (int i=0; i<getCartSize(); i++) {
        item = (Item)getItem(i);
        System.out.println("add one item: "+item.getItemname());

        // save to XML file

        addToXML(item);

        // modify customorderitem table

        modifyOrderItem(item);
    }

    //..... modify customorder table
    modifyOrder(username, email);


    // system.out.println("add ok!");
    try {
        close();
      } catch (JDOMException e) {
        System.out.println(e.toString());
      } catch (IOException e) {
        System.out.println(e.toString());
      }
    }
}
```

CatalogBean.java

```
// this is catalog.jsp's Java Bean

package b2b2c;

import java.util.Vector;
import java.util.Hashtable;
import java.util.Enumeration;
import java.io.*;
import java.net.*;
```

```
import javax.servlet.*;
import javax.servlet.http.*;

public class CatalogBean {

// save browser params
   Hashtable params=new Hashtable();

// XML_Reader
   XML_Reader xr = new XML_Reader();

     String xmldoc="";
     String default_xmldoc =
          "http://localhost:8080/cart/xml/catalog.xml";

 // constructor
   public CatalogBean() { System.out.println("start CartBean.");}

     public void setXmldoc(String s) {  xmldoc = s; }
     public String getXmldoc() { return xmldoc; }

     public void startParse() {
     startParse(false);
     }

   public void startParse(boolean isValidating) {
     if (xmldoc.equals(""))
          xmldoc = default_xmldoc;

     System.out.println("xmldoc="+xmldoc);
     try {
          //.... use URL
          URL xmlURL = new URL (xmldoc);
               xr.startParse(xmlURL.openStream(), isValidating);

          } catch (MalformedURLException e) {
               System.err.println("MalformedURL Exception:"+
                    e.getMessage());
     } catch (IOException e) {
          System.out.println(e.toString());
     }
   }

   public void processRequest(HttpServletRequest req) {
      Enumeration en = req.getParameterNames();
      while (en.hasMoreElements()) {
        String varName = (String)en.nextElement();
        String[] varValues =
          req.getParameterValues(varName);
        if (varValues.length > 1) {
          params.put(varName,  varValues);

               }
        else {
          params.put(varName, varValues[0]);
          System.out.println("req:"+varName+
               "="+varValues[0]);
        }
      }
   }

   public String getParam(String key) {
        return (String)params.get(key);
```

```
    }

    public void setParam(String key, String value) {
      if (params.containsKey(key))
          params.put(key, value);
       else
            System.out.println("no such key:"+
                 key+" in param Hashtable");

    }

    public String[] getParamArr(String key) {
          return (String[])params.get(key);

    }

    public int getCatalogSize() {
          return xr.getCatalogSize();
    }

    public Item getItem(int i) {
          return xr.getItem(i);
    }

// read all items
    public Vector getItems() {
          return xr.getItems();
    }
}
```

CheckOrder.java

```
// this program is for SOAP service side

import java.util.*;
import org.apache.soap.util.xml.*;
import org.apache.soap.encoding.soapenc.*;

/**
 * SOAP service
 **/
public class CheckOrder
{

      public CheckOrder() {

            System.out.println("Start of CheckOrder Service V.1.0");

      }


      public String checkOrderStatus(String orderNo)
            throws IllegalArgumentException
      {
            String ret;


      if (orderNo == null)
      {
            throw new IllegalArgumentException
```

```
            ("Argument 'name' must not be null.");
        }

        System.out.println("order no:"+orderNo);

        ret = processCheckOrderStatus();

        ret = new Base64().encode(ret.getBytes());

        return ret;
        }

        // check order status simulation
        public String processCheckOrderStatus() {
            double r = java.lang.Math.random();
            String ret;

            if (r < 0.25)
                ret = "The order has received, the goods is in the processing.";
            else if (r >=0.25 &&  r < 0.5)
                ret = "Goods is processing and send outing.";
            else if (r >=0.5 &&  r < 0.75)
                ret = "The goods has send outed.";
            else
                ret = "The goods has sent to users.";

            System.out.println("processCheckOrderStatus(): ret="+ret);
            ret = new Base64().encode(ret.getBytes());
            System.out.println("processCheckOrderStatus(): ret(base64)="+ret);
            return ret;
        }
}
```

CheckOrderBean.java

```
// this program is for SOAP Client side

package b2b2c;

import java.util.Hashtable;
import java.util.Enumeration;

import org.jdom.Document;
import org.jdom.Element;
import org.jdom.JDOMException;
import org.jdom.input.SAXBuilder;
import org.jdom.output.XMLOutputter;

import java.io.*;
import java.net.*;
import java.sql.*;
import java.util.Vector;
import java.lang.System;

import javax.servlet.*;
import javax.servlet.http.*;

public class CheckOrderBean  {

   Hashtable params=new Hashtable();

     private String username;
```

```
    // SOAP client
    CheckOrderClient  checker;

    //... JDBC ....
    private Connection con;
    private PreparedStatement pstmt_select_order;
  private PreparedStatement pstmt_select_orderitem;
    //-----------------------------------
  public CheckOrderBean() { System.out.println("start CheckOrderBean."); }

// initial All setting
    public void initAll(String RPCRouter) {

// initial all SOAP client and transfer RPCRouter's URL to SOAP client
        try {
            checker = new CheckOrderClient(RPCRouter);
        } catch (MalformedURLException e) {
            System.out.println(e.toString());
        }

        // connect to database and prepare the sql statement

        createConnection();

        // set init value
        this.username = null;
    }

    public void processRequest(HttpServletRequest req) {
    Enumeration en = req.getParameterNames();
    while (en.hasMoreElements()) {
        String varName = (String)en.nextElement();
        String[] varValues = req.getParameterValues(varName);
        if (varValues.length > 1) {
            params.put(varName,  varValues);

            }
        else {
            params.put(varName, varValues[0]);
            System.out.println("req:"+varName+"="+varValues[0]);
        }
    }
    }

  public String getParam(String key) {
      return (String)params.get(key);
  }

  public void setParam(String key, String value) {
    if (params.containsKey(key))
      params.put(key, value);
     else
        System.out.println("no such key:"+key+" in param Hashtable");
  }

  public String[] getParamArr(String key) {
      return (String[])params.get(key);

  }

  //.... JDBC .....
    // build connect for JDBC and database and prepare sql statement
```

```
    public void createConnection() {
        try {

        Class.forName("org.gjt.mm.mysql.Driver").newInstance();

    }
    catch (Exception E) {
        System.err.println("Unable to load driver.");
        E.printStackTrace();
    }

        try {
            con =
DriverManager.getConnection("jdbc:mysqt//localhost/warehouse?user=root&password=clyang0306");

            pstmt_select_order = con.prepareStatement("select name from customorder where orderno=?");
            pstmt_select_orderitem = con.prepareStatement("select id,qty from customorderitem where
orderno=?");


        } catch (SQLException ex) {
            System.out.println(ex.toString());
        }
        catch (Exception ex) {
            System.out.println(ex.toString());
        }
    }

    public void close()
        throws IOException {


        //.... close database .....
        try {
            pstmt_select_order.close();
            con.close();

        } catch (SQLException ex) {
            System.out.println(ex.toString());
        }
        catch (Exception ex) {
            System.out.println(ex.toString());
        }
    }

    //.....................................................
// from database order number get user name
    public String getUsernameFromDB() {
        String name=null;
        try {
            pstmt_select_order.setString(1, getParam("ordernum"));

            ResultSet rs = pstmt_select_order.executeQuery();

            if (rs != null) {
                rs.next();

                name = rs.getString(1);
                this.username = name;
                rs.close();

            }
```

```
            } catch (SQLException ex) {
                System.out.println(ex.toString());
            }
            catch (Exception ex) {
                System.out.println("getUsernameFromDB():"+ex.toString());
            }

            return name;
        }

// check order number is save in database or not
        public boolean checkOrderNo() {
            boolean ret=false;;

            if (this.username ==null) {

                if (getUsernameFromDB()!=null) {
                    System.out.println("get name from db");
                    ret = true;
                }
                else
                    System.out.println("no name in db");
            }
            else ret = true;

            return ret;
        }

// from table customorderitem do sql statement
        public Vector  getOrderitems() {
            Vector items = new Vector();
            String id;
            int qty;

            try {
                pstmt_select_orderitem.setString(1, getParam("ordernum"));

                ResultSet rs = pstmt_select_orderitem.executeQuery();

                if (rs != null) {
                    while ( rs.next() ) {
                        id = rs.getString(1);
                        qty = rs.getInt(2);

                        items.add( new OrderItem(id, qty) );
                    }

                    rs.close();
                }

            } catch (SQLException ex) {
                System.out.println(ex.toString());
            }
            catch (Exception ex) {
                System.out.println("getUsernameFromDB():"+ex.toString());
            }

            return items;
        }

// get user name
        public String getUsername() {
            return username;
```

```
    }
    // get order number
    public String getOrderno() {
        return ( getParam("ordernum") );
    }

// call SOAP client and start SOAP RPC
    public String getOrdercond() {
        String ret =
         checker.getResult("checkOrderStatus",
                                    getParam("ordernum") );
  return ret;

    }
}
```

CheckOrderBean2.java

```
// this program is for display XSL page

package b2b2c;

import java.util.Hashtable;
import java.util.Enumeration;

import org.jdom.Document;
import org.jdom.Element;
import org.jdom.JDOMException;
import org.jdom.input.SAXBuilder;
import org.jdom.output.XMLOutputter;

import java.io.*;
import java.net.*;
import java.sql.*;
import java.util.Vector;
import java.lang.System;

import javax.servlet.*;
import javax.servlet.http.*;

public class CheckOrderBean2 {

  Hashtable params=new Hashtable();

    private static final String DEFAULT_SAX_DRIVER_CLASS =
     "org.apache.xerces.parsers.SAXParser";

  private String saxDriverClass;

  private SAXBuilder builder;

  private Document doc;
  private Element root;


  private String xmlpath;
  private String xmldoc;

    private String username;
    private String orderNo;

    CheckOrderClient checker;
```

```
  //... JDBC ....
  private Connection con;
  private PreparedStatement pstmt_select_order;
private PreparedStatement pstmt_select_orderitem;
  //-----------------------------------
public CheckOrderBean2() { System.out.println("start CheckOrderBean."); }

  public void initAll(String RPCRouter,
                              String XmlPath) {

       this.xmlpath = XmlPath;

       try {
            checker = new CheckOrderClient(RPCRouter);
       } catch (MalformedURLException e) {
            System.out.println(e.toString());
       }

       createConnection();

       this.username = null;
  }


  public void procssRequest(HttpServletRequest req) {
  Enumeration en = req.getParameterNames();
  while (en.hasMoreElements()) {
       String varName = (String)en.nextElement();
       String[] varValues = req.getParameterValues(varName);
       if (varValues.length > 1) {
            params.put(varName, varValues);

            }
       else {
            params.put(varName, varValues[0]);
            System.out.println("req:"+varName+"="+varValues[0]);
       }
  }
  }

public String getParam(String key) {
      return (String)params.get(key);
}

public void setParam(String key, String value) {
  if (params.containsKey(key))
      params.put(key, value);
   else
        System.out.println("no such key:"+key+" in param Hashtable");
}

public String[] getParamArr(String key) {
      return (String[])params.get(key);

}

//.... JDBC .....


  public void createConnection() {
       try {

       Class.forName("org.gjt.mm.mysql.Driver").newInstance();
```

```
        }
        catch (Exception E) {
            System.err.println("Unable to load driver.");
            E.printStackTrace();
        }

            try {
                con =
DriverManager.getConnection("jdbc:mysql://localhost/warehouse?user=root&password=clyang0306");

                pstmt_select_order = con.prepareStatement("select name from customorder where orderno=?");
                pstmt_select_orderitem = con.prepareStatement("select id,qty from customorderitem where
orderno=?");


            } catch (SQLException ex) {
                System.out.println(ex.toString());
            }
            catch (Exception ex) {
                System.out.println(ex.toString());
            }

    }

    public String getXmlfilename() {
        return getOrderno()+".xml";
    }

    public void close()
        throws IOException, JDOMException {

        doc.setRootElement(root);

        xmldoc = xmlpath + getXmlfilename();

        System.out.println("xmldoc="+xmldoc);

        startOutputXML(new FileOutputStream(xmldoc));

        //.... close database .....
        try {
            pstmt_select_order.close();
            con.close();

        } catch (SQLException ex) {
            System.out.println(ex.toString());
        }
        catch (Exception ex) {
            System.out.println(ex.toString());
        }
    }


    public String getUsernameFromDB() {
        String name=null;
        try {
            pstmt_select_order.setString(1, getParam("ordernum"));

            ResultSet rs = pstmt_select_order.executeQuery();

            if (rs != null) {
                rs.next();
```

```
                name = rs.getString(1);
                this.username = name;
                rs.close();

            }

        } catch (SQLException ex) {
            System.out.println(ex.toString());
        }
        catch (Exception ex) {
            System.out.println("getUsernameFromDB():"+ex.toString());
        }


        return name;
}

public boolean checkOrderNo() {
        boolean ret=false;;

        if (this.username ==null) {

            if (getUsernameFromDB()!=null) {
                System.out.println("get name from db");
                ret = true;
            }
            else
                System.out.println("no name in db");
        }
        else ret = true;

        return ret;
}

public Vector  getOrderitems() {
        Vector items = new Vector();
        String id;
        int qty;

        try {
            pstmt_select_orderitem.setString(1, getParam("ordernum"));

            ResultSet rs = pstmt_select_orderitem.executeQuery();

            if (rs != null) {
                while ( rs.next() ) {
                    id = rs.getString(1);
                    qty = rs.getInt(2);

                    items.add( new OrderItem(id, qty) );
                }

                rs.close();
            }

        } catch (SQLException ex) {
            System.out.println(ex.toString());
        }
        catch (Exception ex) {
            System.out.println("getUsernameFromDB():"+ex.toString());
        }

        return items;
```

```
    }

    public String getUsername() {
        return username;
    }


    public String getOrderno() {
        return ( getParam("ordernum") );
    }

    public String getOrdercond() {
        String ret =
         checker.getResult("checkOrderStatus",
                                  getParam("ordernum") );

        //System.out.println("getOrdercond(): ret="+ret);

        return ret;
    }


    private void startOutputXML(OutputStream out)
     throws IOException, JDOMException {

     XMLOutputter outputter = getXMLOutputter();
     outputter.output(doc, out);
}


    private XMLOutputter getXMLOutputter()
     throws IOException, JDOMException {

     XMLOutputter outputter = new XMLOutputter("  ", true, "Big5");
     outputter.setTrimText(true);
     outputter.setExpandEmptyElements(true);
     return outputter;
}



    private Element addToXML(Element curItems, OrderItem item) {

    Element curItem =
          new Element("orderitem");

    Element curId = new Element("id");
    curId.addContent(item.getId());

    Element curQty = new Element("qty");
    curQty.addContent(new Integer(item.getQty()).toString());

    curItem.addContent(curId);
    curItem.addContent(curQty);

    curItems.addContent(curItem);

    return curItems;

}


    public void toXML() {
```

```
            this.saxDriverClass = DEFAULT_SAX_DRIVER_CLASS;
        builder = new SAXBuilder(saxDriverClass);

        Vector items = getOrderitems();
        OrderItem item;


        root = new Element("order");
            doc = new Document(root);

            Element curName =
             new Element("name");
            curName.addContent( getUsername() );
            root.addContent(curName);

            Element curOrderno =
             new Element("orderno");
            curOrderno.addContent( getOrderno() );
            root.addContent( curOrderno );

        Element curItems =
             new Element("orderitems");

        for (int i=0; i<items.size(); i++) {
            item = (OrderItem)items.get(i);
            System.out.println("add one item: "+item.getId());

            curItems = addToXML(curItems, item);
        }

        root.addContent( curItems );


    Element curStatus =
    new Element("status");
            curStatus.addContent( getOrdercond() );
            root.addContent( curStatus );


        try {
            close();
          } catch (JDOMException e) {
            System.out.println(e.toString());
          } catch (IOException e) {
            System.out.println(e.toString());
          }
        }
}
```

CheckOrderClient.java

```
// this program is SOAP client side
package b2b2c;

import java.io.*;
import java.util.*;
import java.net.*;
import org.apache.soap.util.xml.*;
import org.apache.soap.*;
import org.apache.soap.encoding.*;
import org.apache.soap.encoding.soapenc.*;

import org.apache.soap.rpc.*;
```

```
import org.apache.soap.transport.http.*;

/**
 * SOAP Client of CheckOrder
 **/
public class CheckOrderClient
{

    private String result = null;
    private URL url;
    private Call call;


    public CheckOrderClient(String arg) throws
        MalformedURLException
    {

        url = new URL(arg);

    }

    private void start(String method) {
        String encodingStyleURI =
        Constants.NS_URI_SOAP_ENC;


    // Build the call.
    call = new Call();

    call.setTargetObjectURI("urn:CheckOrder");
    call.setMethodName(method);
    call.setEncodingStyleURI(encodingStyleURI);

    }


    private void setParameters(String orderNo) {

    Vector params = new Vector();

     //String name1 = new Base64().encode(name.getBytes());

    params.addElement(new Parameter("orderno",
                    String.class,
              orderNo, null));


    call.setParams(params);

    }


    private void execRPC() {
        // start RPC
        Response resp;
        Object value;

        try
        {
             System.out.println("CALL: "+call.toString());

        resp = call.invoke(url, "");
        }
```

```
        catch (SOAPException e)
        {
        System.err.println
              ("Caught SOAPException ("+
         e.getFaultCode() + "): " +
         e.getMessage());
        return;
}
catch (Exception e) {
        System.out.println(e.toString());
        return;
}

if (!resp.generatedFault())
{

        System.out.println("RPC response:"+
              resp.toString());

        Parameter ret = resp.getReturnValue();
        if (ret == null) {
              System.out.println
                    ("null result value");
                    return;
        }
        else {
              value = ret.getValue();
              }

        if (value != null) {
              result = value.toString();

              System.out.println
                    ("result:"+result+"\n");
        }
        else {
                    System.out.println
                    ("null result value");
        }
}
else
{
        Fault fault = resp.getFault();

        System.err.println("Generated fault: ");
        System.out.println ("  Fault Code   = " +
              fault.getFaultCode());
        System.out.println ("  Fault String = " +
              fault.getFaultString());

}
}

public String getResult(String method,  String orderNo) {
        try {
              start(method);
              setParameters(orderNo);
              execRPC();

        } catch (Exception e) {
              System.out.println(e.toString());
        }
        //System.out.println("getResult(): result="+result);
```

```
            result = new String( new Base64().decode(result) );
            //System.out.println("getResult(): resut(base64 decode)="+result);
            return result;
        }

}
```

ComOrder.java

```
import java.util.*;
import java.io.*;

import org.apache.soap.util.xml.*;
import org.apache.soap.encoding.soapenc.*;

/**
 * SOAP service for B2B order
 *
 * @author Sam Yang
 */
public class ComOrder
{


    private String xmldoc = "../xml/comorder.xml";


        public ComOrder() {

            System.out.println("Start of ComOrder Service V.1.0");

        }


        public String xmlComOrder(String order)
            throws IllegalArgumentException
        {
            String ret;


        if (order == null)
        {
            throw new IllegalArgumentException
            ("Argument 'order' must not be null.");

        }

        //..... umark this if Big5 char is used in XML doc
            // order = new String(new Base64().decode(order));

        // System.out.println("order:"+order);

        ret = processXmlComOrder(order);

        return ret;
        }

        public String processXmlComOrder(String order) {
            System.out.println("Class ComOrder: comorder=\n"+order);

            //.... if wrong order ......
            if (order.trim().equals("")) {
                return "";
            }
```

```
            try {
                FileOutputStream fos = new FileOutputStream(xmldoc);
                fos.write(order.getBytes());
                fos.close();
            } catch (Exception e) {
                System.out.println("processXmlComOrder(): "+e.toString());
            }


            String orderNo =
                String.valueOf( java.lang.System.currentTimeMillis() );

            return orderNo;
        }
}
```

ComOrderClient.java

```
package b2b2c;

import java.io.*;
import java.util.*;
import java.net.*;
import org.apache.soap.util.xml.*;
import org.apache.soap.*;
import org.apache.soap.encoding.*;
import org.apache.soap.encoding.soapenc.*;

import org.apache.soap.rpc.*;
import org.apache.soap.transport.http.*;

/**
 * SOAP Client of ComOrder
 **/
public class ComOrderClient
{

    private String result = null;
    private URL url;
    private Call call;

    public ComOrderClient(String arg) throws
        MalformedURLException
    {

        url = new URL(arg);

    }

    private void start(String method) {
        String encodingStyleURI =
        Constants.NS_URI_SOAP_ENC;


    // Build the call.
    call = new Call();

    call.setTargetObjectURI("urn:ComOrder");
    call.setMethodName(method);
    call.setEncodingStyleURI(encodingStyleURI);

    }
```

```
private void setParameters(String order) {

Vector params = new Vector();

//String name1 = new Base64().encode(name.getBytes());

params.addElement(new Parameter("order",
                String.class,
          order, null));

call.setParams(params);

}


private void execRPC() {
     // start RPC
     Response resp;
     Object value;

     try
     {
          System.out.println("CALL: "+call.toString());

     resp = call.invoke(url, "");
     }
     catch (SOAPException e)
     {
     System.err.println
          ("Caught SOAPException ("+
               e.getFaultCode() + "): " +
               e.getMessage());
     return;
}
catch (Exception e) {
     System.out.println(e.toString());
     return;
}

if (!resp.generatedFault())
{

     System.out.println("RPC response:"+
          resp.toString());


     Parameter ret = resp.getReturnValue();
     if (ret == null) {
          System.out.println
               ("null result value");
               return;
     }
     else {
          value = ret.getValue();
          }

     if (value != null) {
          result = value.toString();

          System.out.println
               ("result:"+result+"\n");
```

```
            }
            else {
                        System.out.println
                        ("null result value");
            }
      }
      else
      {
            Fault fault = resp.getFault();

            System.err.println("Generated fault: ");
            System.out.println ("  Fault Code   = " +
                  fault.getFaultCode());
            System.out.println ("  Fault String = " +
                  fault.getFaultString());

      }
      }


      public String getResult(String method,  String order) {

            try {
                  start(method);
                  setParameters(order);
                  execRPC();

            } catch (Exception e) {
                  System.out.println(e.toString());
            }

            return result;
      }

}
```

ErrorBean.java

```
package b2b2c;

import java.util.Hashtable;
import java.util.Enumeration;

import java.io.*;

import javax.servlet.*;
import javax.servlet.http.*;

public class ErrorBean  {

   Hashtable params=new Hashtable();


     //---------------------------------
   public ErrorBean() { System.out.println("start ErrorBean."); }


     //
     public void processRequest(HttpServletRequest req) {
     Enumeration en = req.getParameterNames();
     while (en.hasMoreElements()) {
           String varName = (String)en.nextElement();
           String[] varValues = req.getParameterValues(varName);
           if (varValues.length > 1) {
```

```
                params.put(varName,  varValues);

                }
            else {
                params.put(varName, varValues[0]);
                System.out.println("req:"+varName+"="+varValues[0]);
            }
        }
        }

    public String getParam(String key){
          return (String)params.get(key);


    }

    public void setParam(String key, String value) {
      if (params.containsKey(key))
          params.put(key, value);
       else
            System.out.println("no such key:"+key+" in param Hashtable");

    }

    public String[] getParamArr(String key) {
          return (String[])params.get(key);

    }

}
```

Item.java

```
package  b2b2c;

public class Item {

    String itemname, id, picture, price, qty;

    public Item(String itemname,
                String id,
              String picture,
              String price,
              String qty )
    {
          this.itemname = itemname;
          this.id = id;
          this.picture = picture;
          this.price = price;
          this.qty = qty;
    }

    public String getItemname() { return itemname; }
    public String getId() { return id; }
    public String getPicture() {return picture; }
    public String getPrice() {return price; }

    public String getQty() {return qty; }
    public void setQty(String s) {this.qty = s; }
}
```

LoginAuthBean.java

```
package b2b2c;

import java.util.Hashtable;
import java.util.Enumeration;


import java.io.*;
import java.net.*;
import java.sql.*;

import javax.servlet.*;
import javax.servlet.http.*;

/*
*    B2B2C
*/

public class LoginAuthBean  {

   Hashtable params=new Hashtable();


      //... JDBC ....
      private Connection con;

      private PreparedStatement pstmt_select_password;

      String password, email, tel1;


      // constructor
   public LoginAuthBean() {

      createConnection();

      password = null;
      email = null;
      tel1 = null;
      System.out.println("start LoginAuthBean.");
   }


         public static void main(String args[]) {
            //CatalogBean catalog = new CatalogBean();
            LoginAuthBean loginauth = new LoginAuthBean();



      public void processRequest(HttpServletRequest req) {
      Enumeration en = req.getParameterNames();
      while (en.hasMoreElements()) {
            String varName = (String)en.nextElement();
            String[] varValues = req.getParameterValues(varName);
            if (varValues.length > 1) {
                  params.put(varName,  varValues);

                  }
            else {
                  params.put(varName, varValues[0]);
                  System.out.println("req:"+varName+"="+varValues[0]);
            }
      }
      }
```

```
public String getParam(String key) {
      return (String)params.get(key);


}

public void setParam(String key, String value) {
  if (params.containsKey(key))
      params.put(key, value);
   else
        System.out.println("no such key:"+key+" in param Hashtable");

}

public String[] getParamArr(String key) {
      return (String[])params.get(key);

}

  public String getUsername() {System.err.println("user="+getParam("username1"));
                          return  getParam("username1"); }
  public String getemail() {return  email; }
  public String getTel1() {return  tel1; }


  //.... JDBC .....

  public boolean checkPassword() {
  System.err.println("LoginAuthBean:user="+getParam("username1"));
      return checkPassword(getParam("username1"), getParam("password1"));
  }


  public boolean checkPassword(String uname, String pword) {
      boolean ret = false;

      System.out.println("LoginAuthBean:checkPassword: uname="+uname+",pw="+pword);

      try {
          //String password;

          pstmt_select_password.setString(1, uname);
          ResultSet rs = pstmt_select_password.executeQuery();

          if (rs == null) {
              System.out.println("null result set!");
              return false;
          }

          if (rs.next()) {
              password = rs.getString(1);
              email = rs.getString(2);
              tel1 = rs.getString(3);

              System.out.println("pw(input)="+pword+", (db)="+password);
              System.out.println("email="+email+", tel1="+tel1);

              if (password.equals(pword)) {
                  ret = true;
              }
              else {
                  ret = false;
```

```
                    System.out.println("password error");
                }
            }
            else
            {
                System.out.println("No such username");
                ret = false;
            }

            rs.close();

        } catch (SQLException ex) {
            System.out.println(ex.toString());
        }
        catch (Exception ex) {
            System.out.println(ex.toString());
        }

        return ret;
    }


    public void createConnection() {
        try {
        Class.forName("org.gjt.mm.mysql.Driver").newInstance();
     }
     catch (Exception E) {
        System.err.println("Unable to load driver.");
        E.printStackTrace();
     }


        try {
            con =
DriverManager.getConnection("jdbc:mysql://bcalhost/warehouse?user=root&password=clyang0306");
            pstmt_select_password = con.prepareStatement("select password,email,tel1 from membership where
username=?");


        } catch (SQLException ex) {
            System.out.println(ex.toString());
        }
        catch (Exception ex) {
            System.out.println(ex.toString());
        }


    }


    public void close() {


        //.... close database .....
        try {

            pstmt_select_password.close();
            con.close();

        } catch (SQLException ex) {
            System.out.println(ex.toString());
        }
```

```
            catch (Exception ex) {
                System.out.println(ex.toString());
            }
        }

}
```

OrderItem.java

```
package b2b2c;

public class OrderItem {
    String id;
    int qty;

    public OrderItem(String id, int qty) {
        this.id = id;
        this.qty = qty;
    }

    public String getId() { return id;}
    public int getQty() {return qty; }
    public void setId(String id) {this.id = id; }
    public void setQty(int qty) {this.qty = qty; }
}
```

Registers.java

```
// this program is for register new user
package b2b2c.com.lib.databases;

import java.sql.*;
import java.util.*;

public class Registers {

  String error;
  Connection con;

  public Registers()  { }

  public void connect() throws ClassNotFoundException,
                               SQLException,
                               Exception {
    try {
      Class.forName("org.gjt.mm.mysql.Driver").newInstance();
      con = DriverManager.getConnection(
        "jdbc:mysql://localhost/warehouse ?user=root&password=clyang0306");
    } catch (ClassNotFoundException cnfe) {
      error = "ClassNotFoundException: Could not locate DB driver.";
      throw new ClassNotFoundException(error);
    } catch (SQLException cnfe) {
      error = "SQLException: Could not connect to database.";
      throw new SQLException(error);
    } catch (Exception e) {
      error = "Exception: An unknown error occurred while connecting " +
            "to database.";
```

```
    throw new Exception(error);
   }
  }

  public void disconnect() throws SQLException {
   try {
    if ( con != null ) {
     con.close();
    }
   } catch (SQLException sqle) {
    error = ("SQLException: Unable to close the database connection.");
    throw new SQLException(error);
   }
  }

  public ResultSet viewRegisters() throws SQLException, Exception {
   ResultSet rs = null;
   try  {
    String queryString = ("SELECT * FROM membership;");
    Statement stmt = con.createStatement();
    rs = stmt.executeQuery(queryString);
   } catch (SQLException sqle) {
    error = "SQLException: Could not execute the query.";
    throw new SQLException(error);
   } catch (Exception e) {
    error = "An exception occured while retrieving Registers.";
    throw new Exception(error);
   }
   return rs;
  }

  public void addRegisters(String username, String password,String email, String tel1)
                throws SQLException, Exception {
   if (con != null) {
    try {
     PreparedStatement updateRegisters;
     updateRegisters = con.prepareStatement(
                "insert into membership values(?,?,?,?);");
     updateRegisters.setString(1, username);
     updateRegisters.setString(2, password);
     updateRegisters.setString(3, email);
     updateRegisters.setString(4, tel1);
     updateRegisters.execute();
    } catch (SQLException sqle) {
     error = "SQLException: update failed, possible duplicate entry";
     throw new SQLException(error);
    }
   } else {
    error = "Exception: Connection to database was lost.";
    throw new Exception(error);
   }
  }

  public void removeRegisters(String [] pkeys) throws SQLException, Exception {
   if (con != null) {
    try {
     PreparedStatement delete;
     delete = con.prepareStatement("DELETE FROM membership WHERE username=?;");
     for (int i = 0; i < pkeys.length; i++) {
      delete.setString(1, pkeys[i]);
      delete.execute();
     }
    } catch (SQLException sqle) {
```

```
        error = "SQLException: update failed, possible duplicate entry";
        throw new SQLException(error);
      } catch (Exception e) {
        error = "An exception occured while deleting Registers.";
        throw new Exception(error);
      }
    } else {
      error = "Exception: Connection to database was lost.";
      throw new Exception(error);
    }
  }
}
```

StockBean.java

```
package b2b2c;

import org.jdom.Document;
import org.jdom.Element;
import org.jdom.JDOMException;
import org.jdom.input.SAXBuilder;
import org.jdom.output.XMLOutputter;

import java.io.*;
import java.net.*;
import java.sql.*;
import java.util.Hashtable;
import java.util.Enumeration;
import java.util.Vector;
//import java.lang.System;

import javax.servlet.*;
import javax.servlet.http.*;

/**
*   Bean for stockmanage.jsp and comorder.jsp
**/

public class StockBean  {

   Hashtable params=new Hashtable();

      private static final String DEFAULT_SAX_DRIVER_CLASS =
       "org.apache.xerces.parsers.SAXParser";


   private String saxDriverClass;

   private SAXBuilder builder;

   private Document doc;
   private Element root;


      // SOAP client
      ComOrderClient  sender;

      //... JDBC ....
      private Connection con;
      private PreparedStatement pstmt_select_stock;

      //----------------------------------
```

```
public StockBean() { System.out.println("start StockBean."); }


  public void initAll(String RPCRouter) {

       try {
            sender = new ComOrderClient(RPCRouter);
       } catch (MalformedURLException e) {
            System.out.println(e.toString());
       }


       createConnection();

  }

  public void processRequest(HttpServletRequest req) {
  Enumeration en = req.getParameterNames();
  while (en.hasMoreElements()) {
       String varName = (String)en.nextElement();
       String[] varValues = req.getParameterValues(varName);
       if (varValues.length > 1) {
            params.put(varName,  varValues);

            }
       else {
            params.put(varName, varValues[0]);
            System.out.println("req:"+varName+"="+varValues[0]);
       }
  }
  }


public String getParam(String key) {
      return (String)params.get(key);
}

public void setParam(String key, String value) {
  if (params.containsKey(key))
      params.put(key, value);
   else
       System.out.println("no such key:"+key+" in param Hashtable");
}

public String[] getParamArr(String key) {
      return (String[])params.get(key);

}

//.... JDBC .....

  public void createConnection() {

       try {

       Class.forName("org.gjt.mm.mysql.Driver")newInstance();

   }
   catch (Exception E) {
       System.err.println("Unable to load MySQL driver.");
       E.printStackTrace();
   }
```

```
        try {
            con =
DriverManager.getConnection("jdbc:mysql://localhost/warehouse?user=bot&password=clyang0306");

            pstmt_select_stock = con.prepareStatement("select id,qty from stock");

        } catch (SQLException ex) {
            System.out.println(ex.toString());
        }
        catch (Exception ex) {
            System.out.println(ex.toString());
        }


    }


    public void close()
        throws IOException, JDOMException {

        //.... close database .....
        try {
            pstmt_select_stock.close();
            con.close();

        } catch (SQLException ex) {
            System.out.println(ex.toString());
        }
        catch (Exception ex) {
            System.out.println(ex.toString());
        }
    }

    //.....

    public Vector  getItems() {
        Vector items = new Vector();
        String id;
        int qty;

        try {

            ResultSet rs = pstmt_select_stock.executeQuery();
            //Statement stmt = con.createStatement();
//ResultSet rs = stmt.executeQuery("SELECT id,qty from stock");

            if (rs != null) {
                while ( rs.next() ) {
                    id = rs.getString(1);
                    qty = rs.getInt(2);
                    System.out.println("add id="+id+", qty="+qty);
                    items.add( new OrderItem(id, qty) );
                }

                rs.close();
            }

        } catch (SQLException ex) {
            System.out.println(ex.toString());
        }
        catch (Exception ex) {
```

```
            System.out.println("getItems():"+ex.toString());
        }

        return items;
    }


    private Vector buildNewItems() {
        Vector items = getItems();
        Vector newItems = new Vector();
        OrderItem  item;
        int curQty;
        String sCurQty;

        for (int i=0; i<items.size(); i++) {
            sCurQty = getParam("qty_"+i);
            if (! sCurQty.trim().equals("") ) {
                curQty = Integer.parseInt( sCurQty );
                item = (OrderItem)items.get(i);
                item.setQty(curQty);
                //item = new OrderItem(item.getId(),  curQty );
                System.out.println("new item:"+item.getId()+","+curQty);
                newItems.add(item);
            }
        }

        return newItems;
    }

    public String sendComOrder() {


        Vector newItems = buildNewItems();

        toXML(newItems);


        ByteArrayOutputStream buffer = new ByteArrayOutputStream();

        try {
            buffer = (ByteArrayOutputStream)startOutputXML(buffer);
        } catch (JDOMException e) {
        System.out.println("toXML(): JDOM exception:"+e.toString());
      } catch (Exception e) {
        System.out.println("toXML(): Exception:"+e.toString());
      }

        String orderxml = buffer.toString();


        String ret =
         sender.getResult("xmlComOrder",orderxml );

        //System.out.println("sendComOrder(): ret="+ret);

        return ret;
    }


    private void addToXML(OrderItem item) {
        // add one element to  XML file
```

```
    Element curItem =
          new Element("item");

    Element curId = new Element("id");
    curId.addContent(item.getId());

    Element curQty = new Element("qty");
    curQty.addContent(new Integer(item.getQty()).toString());

    curItem.addContent(curId);
    curItem.addContent(curQty);

    root.addContent(curItem);

  }


    public void toXML(Vector items) {
          this.saxDriverClass = DEFAULT_SAX_DRIVER_CLASS;
     builder = new SAXBuilder(saxDriverClass);

    OrderItem item;
    //System.out.println("new root");

    //  build new XML doc
    root = new Element("comorder");
          doc = new Document(root);

    //System.out.println("add elements to root");
    for (int i=0; i<items.size(); i++){

          item = (OrderItem)items.get(i);
          System.out.println("add to XML: "+item.getId());

          // add each item to XML doc
          addToXML(item);


    }

    //System.out.println("orderxml="+doc.toString());

    //..... unmark this for debug (to see XML doc)
    /*
    try {
          startOutputXML(System.out);
       } catch (JDOMException e) {
          System.out.println("toXML(): JDOM exception:"+e.toString());
       } catch (Exception e) {
          System.out.println("toXML(): Exception:"+e.toString());
       }
       */
    }


    private OutputStream startOutputXML(OutputStream out)
     throws IOException, JDOMException {

     XMLOutputter outputter = getXMLOutputter();
     outputter.output(doc, out);

     return out;
  }
```

```
    private XMLOutputter getXMLOutputter()
     throws IOException, JDOMException {

     // Create an outputter with certain encoding
     XMLOutputter outputter = new XMLOutputter(" ", true, "Big5");
     outputter.setTrimText(true);
     outputter.setExpandEmptyElements(true);
     return outputter;
  }

}
```

XML_Reader.java

```
package b2b2c;

import java.io.*;
import java.lang.*;
import org.xml.sax.Parser;
import org.xml.sax.AttributeList;
import org.xml.sax.InputSource;
import org.xml.sax.HandlerBase;
import org.xml.sax.SAXException;

import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;
import javax.xml.parsers.FactoryConfigurationError;
import javax.xml.parsers.ParserConfigurationException;
import java.util.*;


public class XML_Reader extends HandlerBase
{

    private Vector items;

    private String currentElementName;

    private String itemusername,id;
    private String price;
    private String picture;
    private String ausername;
    private String avalue;


    public  XML_Reader() {
        items = new Vector();
    }

    //-----------------------------------------
    public void startParse(InputStreamReader isr)
     throws IOException {

    if (getCatalogSize() == 0)
        startParse(new InputSource( isr ), false);
  }


    public void startParse(InputStream is)
     throws IOException {
```

```
    startParse(new InputStreamReader(is, "Big5"), false);
}

public void startParse(FileReader fr)
    throws IOException {

    startParse(new InputSource( fr ), false);
}

   public void startParse(String fileusername)
        throws IOException {

    startParse(new FileReader(fileusername), false );
   }
   //.....
   public void startParse(InputStreamReader isr, boolean isValidating)
    throws IOException {

   if (getCatalogSize() == 0)
         startParse(new InputSource( isr ), isValidating);
}

   public void startParse(InputStream is, boolean isValidating)
    throws IOException {

    startParse(new InputStreamReader(is, "Big5"), isValidating);
}

public void startParse(FileReader fr, boolean isValidating)
    throws IOException {

    startParse(new InputSource( fr ), isValidating);
}

   public void startParse(String fileusername, boolean isValidating)
        throws IOException {

    startParse(new FileReader(fileusername), isValidating );
   }

   public void startParse(InputSource in, boolean isValidating)  {

        try {
             SAXParserFactory  factory =
                  SAXParserFactory.newInstance();
             SAXParser   parser  =  factory.newSAXParser();
             factory.setValidating(isValidating);
             factory.setNamespaceAware(false);

             parser.parse(in, this);
        }
        catch (SAXException e) {
             System.out.println("SAX Example:"+e.getMessage());
        }
        catch (Exception  e)  {
             System.out.println("Example:"+e.getMessage());
        }
}


   /*
   // for debug.....
```

```
public static void main (String args[])
{
    boolean isValidating =  false;
    //....debug.....
    String p = System.getProperty("javax.xml.parsers.SAXParserFactory");
    System.out.println("parser="+p);
    //.....
    try {
        XML_Reader xr = new  XML_Reader();
        xr.startParse(args[0]);
    }
    catch (IOException e) {
        System.err.println("IO Exception:"+e.getMessage());
    }
    catch (Exception e) {
        System.err.println("Exception:"+e.toString());
    }
}
*/


//
public Item getItem (int index) {
    if (index < items.size()) {
        // ...debug
        System.out.println("Index:"+ new Integer(index).toString());

        return (Item) items.elementAt(index);
    }
    else {
        // ...debug
        //System.out.println("Index > js no.");
    }

    return (Item) items.elementAt(index);
}


public Vector getItems () {
    return items;
}

public int getCatalogSize() {
    int ret;
    if (items==null) {ret=-1;}
    else {ret = items.size();}
    return ret;
}

public void startElement (String username,
            AttributeList atts) {
    int i;

    currentElementName = username;

    if (currentElementName.equals ("PICTURE")) {
        for (i = 0; i < atts.getLength(); i++) {
        ausername = atts.getName(i);
        avalue = atts.getValue(i);

        //... debug
            System.out.println("PICTURE  "+ausername+":"+avalue);
```

```
                if (ausername.equals ("SOURCE")) {
                    picture = avalue;

                }
                }
            }
        }


        public void endElement (String username) {

            if (username.equals ("ITEM")) {
                Item item = new Item(
                    itemusername, id, picture,
                    price, "0");

                items.addElement(item);

                //... debug
                // System.out.println("new JS was built");
            }

        }


        public void characters (char ch[], int start,
                    int length)       {
        String s = new String (ch, start, length);

        if (s.trim().length() == 0)
            return;

        //.. debug
        System.out.println("data:"+currentElementName+"("+s+")");

            if (currentElementName.equals ("NAME"))
                itemusername = s;
            else if (currentElementName.equals ("ID"))
                id = s;
            else if (currentElementName.equals ("PRICE"))
                price = s;
        }

}
```

REFERENCES

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

2. E. R. Harold, et al. Processing XML with Java: A Guide to SAX, DOM, JDOM, JAXP, and TrAx. Addison Wesley, November 2002.

3. F. Galbraith, et al. Beginning JSP Web Development. Wrox Press, 2001.

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

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

6. H. Maruyama, et al. XML and Java:Developing Web Applications, Second Edition. Addison Wesley, May 2002.

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

8. J. Elmasri, et al. Fundamentals of Database Systems, third edition. Addison Wesley, June 2000.

9. J. Hunter and W. Crawford. Java Servlet programming. O'Reilly, April 2001.

10. K. Arnold and J. Gosling. The Java Programming Language Second Edition. Addison Wesley, February 2000.

11. L. Pekowsky. JavaServer Pages. Addison Wesley, April 2000.

12. M. Fowler and K. Scott. UML Distilled - A brief guide to the standard object modeling language. Addison Wesley Longman, July 2001.

13. M. Geary, et al. Advanced JavaServer Pages. Prentice Hall PTR, May 2001.

14. R. Jeliffe. W3C:Extensible Markup Language(XML),W3C Recommendation, 6 October 2000.

15. S. Laurent, et al. Programming Web Services with XML-RPC. First Edition. Addison Wesley, June 2001.