2004

Designing secure, JAVA based online registration systems to meet peak load performance targets

Tang-Li Chen

Follow this and additional works at: https://scholarworks.lib.csusb.edu/etd-project

Part of the Software Engineering Commons

Recommended Citation
Chen, Tang-Li, "Designing secure, JAVA based online registration systems to meet peak load performance targets" (2004). Theses Digitization Project. 2767.
https://scholarworks.lib.csusb.edu/etd-project/2767

This Project is brought to you for free and open access by the John M. Pfau Library at CSUSB ScholarWorks. It has been accepted for inclusion in Theses Digitization Project by an authorized administrator of CSUSB ScholarWorks. For more information, please contact scholarworks@csusb.edu.
DESIGNING SECURE, JAVA BASED ONLINE REGISTRATION SYSTEMS TO MEET PEAK LOAD PERFORMANCE TARGETS

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
Tang-Li Chen
March 2004
DESIGNING SECURE, JAVA BASED ONLINE REGISTRATION SYSTEMS TO MEET PEAK LOAD PERFORMANCE TARGETS

A Project
Presented to the
Faculty of
California State University,
San Bernardino

by
Tang-Li Chen
March 2004

Approved by:

Dr. David Turner, Chair, Computer Science

Dr. Ernesto Gomez

Dr. Owen Murphy

Date 3/12/2004
ABSTRACT

This project "Designing Secure, Java Based Online Registration Systems to Meet Peak Load Performance Targets" is a simulation of a Web-based exposition management system plus a performance testing procedure to examine this web application. The website provides users with information about an exposition. It allows visitors to buy tickets to attend the exposition, and it allows exhibitors to register for a presentation booth. The system allows both visitors and exhibitors to pay over the Internet. Furthermore, this site offers a secure environment to keep sensitive information protected by adopting SSL (Secure Socket Layer). This site is unlike other sites like eBay and Amazon, because it exists just for a couple months until the exposition ends. In general there will be several thousand people visiting the exposition. Therefore, more then a thousand people may browse the web site during a peak period. Therefore, it is important that the web application can handle this load. To see if the Web application would be able to handle this amount of traffic, a testing procedure was developed to measure performance statistics, including throughput, delay, and failure rates.
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 success. And thanks to my two project committee professors Dr. Gomez and Dr. Murphy for their suggestions in developing my project.

Also, I would like to express my appreciation to my family. Thanks to my parents, who provide all I need to study overseas.

Lastly, I want to thank everyone else who has helped me with learning the principles of software development and their application to this project. With their help, my master's project was possible.
# TABLE OF CONTENTS

ABSTRACT ................................................................. iii
ACKNOWLEDGMENTS ................................................... iv
LIST OF TABLES ........................................................ viii
LIST OF FIGURES ....................................................... ix

CHAPTER ONE: INTRODUCTION ...................................... 1
  1.1 Purpose of this Project .................................... 2
  1.2 Scope of Project ........................................... 2
    1.2.1 Phase I ............................................... 2
    1.2.2 Phase II ............................................ 3
  1.3 Organization of Chapters ................................. 3

CHAPTER TWO: TECHNOLOGIES ..................................... 4
  2.1 Java Language ............................................... 4
  2.2 Java Servlet and JSP ...................................... 5
    2.2.1 Java Servlet ....................................... 5
    2.2.2 JSP ................................................ 6
  2.3 Database Connection Pooling .............................. 7
  2.4 PostgreSQL ................................................ 8
  2.5 Secure Sockets Layer ..................................... 9

CHAPTER THREE: EXPO SERVER DESIGN ........................... 10
  3.1 Hardware Interfaces ....................................... 10
  3.2 Software Interfaces ...................................... 12
  3.3 Expo Server Design ....................................... 13
    3.3.1 Description ........................................ 13
    3.3.2 Expo Server Function .............................. 14
    3.3.3 Use Case Diagram ................................ 15
**LIST OF TABLES**

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 5.1</td>
<td>Performance Metrics</td>
<td>41</td>
</tr>
<tr>
<td>Table 5.2</td>
<td>User Distribution</td>
<td>43</td>
</tr>
<tr>
<td>Table 5.3</td>
<td>User-Page Distribution</td>
<td>45</td>
</tr>
<tr>
<td>Table 5.4</td>
<td>User-Page Distribution (100 users) in Case A</td>
<td>45</td>
</tr>
<tr>
<td>Table 5.5</td>
<td>User-Page Distribution (100 users) in Case B</td>
<td>46</td>
</tr>
<tr>
<td>Table 5.6</td>
<td>Page Category Percentage</td>
<td>46</td>
</tr>
<tr>
<td>Table 5.7</td>
<td>Result at Load 2400 in Case B</td>
<td>48</td>
</tr>
<tr>
<td>Table 5.8</td>
<td>Results of Response Time</td>
<td>50</td>
</tr>
<tr>
<td>Table 5.9</td>
<td>Result of Throughput</td>
<td>53</td>
</tr>
<tr>
<td>Table 5.10</td>
<td>Result of Failure Rate</td>
<td>55</td>
</tr>
<tr>
<td>Table 5.11</td>
<td>Result of Failure Rate (2)</td>
<td>58</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

Figure 2.1  Flow Chart of the Java Program Executed ........................................... 5
Figure 2.2  The Difference between CGI and Servlet .............................................. 6
Figure 2.3  JDBC and Database Connection Pooling ................................................ 8
Figure 3.1  System Deployment Diagram ............................................................... 11
Figure 3.2  Architecture of the Expo Server ............................................................ 13
Figure 3.3  Use Case Diagram ..................................................................................... 16
Figure 3.4  Payment Processing Architecture ............................................................. 17
Figure 4.1  Class Diagram .......................................................................................... 22
Figure 4.2  Main Page .................................................................................................. 23
Figure 4.3  Visitor Page (Registration Form for Visitor) ........................................... 25
Figure 4.4  Receipt Page (for Visitor) ......................................................................... 26
Figure 4.5  Failure Page (for Visitor) ......................................................................... 27
Figure 4.6  Information Page for Exhibitor ................................................................. 28
Figure 4.7  Exhibitor Page (Registration Form for Exhibitor) ................................... 29
Figure 4.8  Receipt Page (for Exhibitor) ..................................................................... 30
Figure 4.9  Failure Page (for Exhibitor) ..................................................................... 31
Figure 4.10 Login Page (for Exhibitor) ...................................................................... 32
Figure 4.11 Modify Profile Page .................................................................................. 33
Figure 4.12 Contact Us Page ....................................................................................... 34
Figure 5.1  Basic Configuration .................................................................................... 36
Figure 5.2  Advanced Configuration ............................................................................ 37
Figure 5.3  Class Diagram .......................................................................................... 40
CHAPTER ONE

INTRODUCTION

There are a lot of expos around the world. Generally, the visitors would buy the tickets at the front of the entrance for attending the expo. The exhibitors would need to call or fax to ask the relevant information then they can make the decision about the time and space. This method wastes a lot of time and labors. Recently, Internet develops very fast. Shopping on the Internet becomes very popular. The goal of this project "Designing Secure, Java Based Online Registration Systems to Meet Peak Load Performance Targets" is to build a simulation of the expo application and construct a performance testing procedure to examine this website. The functions of this website are to provide the users the information about the expo, and allow users to pay the fee on the Internet. Furthermore, this site offers a secure environment to keep sensitive information private by adopting SSL (Secure Socket Layer).

Exposition takes place for a short period and there will be a lot of people attending the exposition. The goal of the expo server is to service users who are interested in the exposition. Because in the short period, more than a thousand people may browse the server. It becomes
important that the server can handle this load. So the testing procedure will diagnose the performance of this web, including the throughput, average page delay, and percent failures, to see if the web server can handle the traffic.

1.1 Purpose of this Project

The purpose of this project is to build an expo server that offers a friendly and simple interface to let users easily use and buy the tickets and register accounts online. Also, the expo server constructs a lightly secure environment, therefore, users can process any request without fear of outside attacks. And another purpose is to build a web-performance testing procedure to test this expo server's performance to meet peak load performance targets.

1.2 Scope of Project

The project was separated into two phases:

1.2.1 Phase I

The first phase is an expo server which is a working web-based online application system with XML configuration, Java Servlet, JSP and PostgreSQL database. The users of the expo server can select either the visitors or exhibitors' section. The visitor can gain
information about the expo and purchase the tickets on the expo server. The exhibitor can access the information about the expo and order the booths on the server.

1.2.2 Phase II

The second phase is a web-performance testing procedure written by Java language. The testing procedure diagnoses the performance of the expo server including the average throughput, average response time and failure rate.

1.3 Organization of Chapters

Chapter Two introduces the technologies used in the project. Chapter Three discusses the details of expo server design. Chapter Four focuses on the expo server implementation. Chapter Five focuses on the performance testing. Chapter Six presents the conclusion and future work.
CHAPTER TWO

TECHNOLOGIES

This chapter I will briefly introduce the technologies I used in this project.

2.1 Java Language

Java is one of the most popular programming languages in the world. Programmers use Java language to ensure their applications can migrate onto different operating systems such as Microsoft Windows, Linux family, etc. without re-writing any part of code. Because the program is not executed directly on the computer. It is executed on the Java virtual machine (JVM) and then JVM uses another program to execute this program on the computer. Figure 2.1 shows the flow chart of the Java program executed.
2.2 Java Servlet and JSP

2.2.1 Java Servlet

A CGI (Common Gateway Interface) program is stored and executed on the web server in response to a request from a user. In CGI, the system creates a new child process for each request to run a CGI program. This method wastes finite server resource and limits the number of
Figure 2.3 JDBC and Database Connection Pooling

2.4 PostgreSQL

Both PostgreSQL and MySQL are very popular open source database management systems. Although MySQL is faster, PostgreSQL is more standard, strict, and powerful. For instances, MySQL doesn’t support the foreign key mechanism and transactions don’t support well. Thus, PostgreSQL is the better solution to build a reliable and integrated database system.
2.5 Secure Sockets Layer

The Secure Sockets Layer (SSL) is a commonly-used protocol for managing the security of a message transmission on the Internet. HTTPS (Hypertext Transfer Protocol over Secure Socket Layer, or HTTP over SSL) is a Web protocol developed by Netscape and built into its browser that encrypts and decrypts user page requests as well as the pages that are returned by the Web server. HTTPS is really just the use of Netscape's Secure Socket Layer (SSL) as a sub layer under its regular HTTP application layering. HTTPS uses port 8443 instead of HTTP port 8080 in its interactions with the lower layer, TCP/IP. SSL uses a 128-bit key size for the RC4 stream encryption algorithm, which is considered an adequate degree of encryption for commercial exchange. This prevents anyone monitoring Internet data transmissions from easily capturing this data.

Even though setting up and using SSL is not difficult. Considering the web performance issue, employing HTTPS is costing more time than without HTTPS. For this reason, it is undesirable to deploy an entire web application under SSL. For fastest performance, it is best to deploy a web application under HTTP and employ HTTPS only for those pages and processes that transmit sensitive data.
project. Hence, there is no need to discuss the hardware interfaces for the system.
3.2 Software Interfaces

The expo server focuses on on-line application. All the software components are open-source, and put together to accomplish a web-based application system. The software interfaces used in this project are summarized as following:

- Operating system: Linux/Unix (Red Hat 9.0).
- Web Server/Container: Jakarta Tomcat Server 4.1.29.
- Java Compiler: J2SDK 1.4.2.
- Web Browser: Microsoft IE, Netscape
- Database Server: PostgreSQL Server 7.3
- Database Connection Poll: Jakarta Commons DBCP 1.1.
- Languages: HTML/JAVA/JSP/XML.
3.3 Expo Server Design

3.3.1 Description

The components in the Figure 3.2 are described as follows:

- A visitor is a person who wants to attend the expo.
- An exhibitor is a company that wants to present its products at the expo.
- An expo web server is a web server that shows the expo information to users and deals the users’ requests, including buying the tickets,
ordering the booths and modifying the profile online.

- A Payment Gateway server is a server that checks users’ credit card information and charges the online payment of the users.
- A database is for expo server to save the visitors’ and exhibitors’ information.

3.3.2 Expo Server Function

- Online Registration: In general, users will separate into visitors and exhibitors. The server offers two different registration forms for visitors and exhibitors. A visitor fills out the form, and chooses how many tickets he/she wants and chooses the time. An exhibitor will be told the rules, the price and the space. Then the exhibitor makes a decision and fills out the form. After, the staff will process the exhibitor’s registration form.

- Get Information: This part includes all the information about the expo, for example, the days, the time, and the products. The visitor can get all the information on website without logging in. The exhibitor also can get the
information on the website without logging in. If the exhibitor wants to check his status, he needs to login. This is to protect the exhibitor’s privacy; only the exhibitor can see his own personal information.

- Modify Account: Exhibitors can login to the server to check their status and change their profile. If they forget their profile, they can view their profile on the site. Also, if they change their address or phone number, they can edit their address or phone number on internet.

- Online Payment: A user can buy tickets or spaces online. When the user fills out the form (including the credit card number), the server will pass the credit card number to a PGS (Payment Gateway Server) for transacting the money. Later on, the server will send a webpage with the response message like accepted or rejected back to the user.

3.3.3 Use Case Diagram

Figure 3.3 shows the Use Case Diagram for the expo web server. The server offers different interfaces for different operations to users.
The Payment Gateway server is a server that deals with the online payment. In this project, expo server offers the users the online payment service. So expo server would connect to Payment Gateway server in order to offer the online payment service to users. I use the Concord EFSnet Web Payment Services as my project’s Payment Gateway Server. Concord EFSnet Web Payment Services offer a free test account that allows me to submit the test transaction and access the EFSnet Merchant Services Web site for transaction management.
3.4.1 EFSnet Architecture

Figure 3.4 Payment Processing Architecture

At the front of Figure 3.4, the merchant gathers transaction information from a customer and submits the transaction to EFSnet using the CGI (Common Gateway Interface style) to exchange transaction data over the Internet using the HTTPS protocol. Once received by the EFSnet servers, the transaction request is first logged and then validated for required fields. Based on two of the fields, StoreID and StoreKey, contained in the transaction request, the EFSnet merchant profile is scanned to determine which Concord processor network will be used for the transaction. At the processor, additional
information is included with the transaction, such as merchant account ID numbers and other static fields that are needed by the card association and financial institutions. When a transaction response returns through the system, it is logged by both the EFSnet servers and by Concord’s processors. Logging of transactions on EFSnet servers permits merchants to access transaction histories, either programmatically or by logging into the EFSnet Merchant Services Web site. In addition to querying and viewing transactions, the Merchant Services site facilitates the creation of new transactions with the EFSnet Virtual Terminal or acting on existing transactions (i.e. voiding, refunding, and settling).

3.4.2 EFSnet Test Server Environment

The test environment allows developers to freely test and certify their application with EFSnet prior to usage with the live production environment. Since EFSnet is a server side solution, applications must send transaction information to EFSnet using a publicly accessible Internet address (URL): https://testefsnet.concordebiz.com/efsned.dll. In order to test transactions without incurring card association transaction fees, they have developed a processor simulator known as the "NULL processor". The NULL processor simulates responses that
occur when connecting to a live Concord processor but does not actually pass requests through the financial networks. All EFSnet test accounts send transactions to a "NULL" processor instead of an actual processor platform.
CHAPTER FOUR
EXPO SERVER IMPLEMENTATION

4.1 Expo Server Class Design

The class descriptions for the expo server are summarized as following:

1. Database: A database access object manager handles all database access operations.

2. Visitor: A database access object is used to access the database related to visitor table.

3. Exhibitor: A database access object is used to access the database related to exhibitor table.

4. ProcessV: A Java Servlet between expo server and Payment Gateway server. It gets the request from Visitor page and passes the visitor’s information to Payment Gateway server. When Payment Gateway server sends back the response message, it values the message, and then it forwards control to an appropriate page.

5. ProcessE: A Java Servlet between expo server and Payment Gateway server. It gets the request from Exhibitor page and passes the exhibitor’s information to Payment Gateway server. When Payment Gateway server sends back the response
message, it values the message, and then it forwards control to an appropriate page.

6. Editfile: A Java Servlet between Login page and Modify Profile page. It gets the request from Login page then checks the data of request message if the data is the same as data from database. Then it forwards control to an appropriate page.

7. Update: A Java Servlet between Modify Profile page and Update Profile page. It gets the data from request message and saves the changed data into database. And it forwards control to an appropriate page.

8. Logout: A Java Servlet, it removes all the session that used before and forwards control to a Logout page.

The class diagram is shown in Figure 4.1.
Figure 4.1 Class Diagram
4.2 Graphical User Interface

4.2.1 Main Page

This page offers the expo information including the place, time, exhibit profile and so on.

There are four buttons above this page. (1) Visitor: this link is for a visitor who wants to attend the expo. It connects to Figure 4.3 Registration Form for Visitor. (2) Exhibitor: this link is for companies which want to be exhibited in the expo. It connects to Figure 4.6 Information for Exhibitor. (3) Account: this link is for
the exhibitor who has registered the expo. It connects to Figure 4.10 Login page. (4) Contact Us: this link is for all users. It connects to Figure 4.12 Contact us page.

4.2.2 Visitor Page

A visitor can fill this form to buy the tickets online. When a visitor clicks the submit button, server will pass the credit card information to Payment Gateway Server. Payment Gateway Server checks the validity of the information, and then sends the response to the server. If the response is approved, expo server sends the receipt page to the visitor. See Figure 4.4 Receipt Page for Visitor. If the response is invalid, expo server sends the failure page to the visitor. See Figure 4.5 Failure Page for Visitor.
4.2.3 Receipt Page for Visitor

This page is a receipt that shows the visitor’s name, confirmed number, show time, number of tickets and total amounts.
Figure 4.4 Receipt Page (for Visitor)

4.2.4 Failure Page for Visitor

This page tells the visitor that his/her credit card have a problem. In this page, there is a link that the visitor can use that link to go back to registration form page and refill the form.
This page includes all the information that exhibitors need to know. When an exhibitor decides to join in the expo, the exhibitor can click the Apply online button to register online.
Information for Exhibitor

Show Time: May 1-5, 2004 from 9:00 a.m. to 5:00 p.m.
Exhibit Profile: Electric Appliances

SPACE COST
1. One booth:
   Standard booth (3m * 3m): US$3,100
2. Two booths:
   Standard booth (3m * 3m): US$6,100
3. Three or more booths (raw space only):
   Standard booth (3m * 3m): US$3,000 per booth
4. Each booth will be provided with 100V/500W electricity free of charge.
   Additional power supply will be at exhibitor’s expense.
5. All the exhibitors will be charged US$500.00 for deposit.
   No matter how many booths they order.

Figure 4.6 Information Page for Exhibitor

4.2.6 Exhibitor Page

An exhibitor can fill this form to order the number of booths that he/she needs to present the products in expo. When an exhibitor clicks the submit button, server will pass the credit card information to Payment Gateway Server. Payment Gateway Server checks the validity of the information, then sends the response to the server. If the response is approved, expo server sends the receipt page to the exhibitor. See Figure 4.8 Receipt Page for Exhibitor. If the response is invalid, expo server sends
the failure page to the exhibitor. See Figure 4.9 Failure Page for Exhibitor.

![Registration Form for Exhibitor](image)

Figure 4.7 Exhibitor Page (Registration Form for Exhibitor)

4.2.7 Receipt Page for Exhibitor

This page is a receipt that shows the exhibitor’s username, company, show time, number of booths and deposit.
Dear Bob Flick,

Your online payment has been accepted.

Username: bob
Company: General Electric Company
Date: 2/4/04
ShowTime: May 1-5, 2004 from 9:00 a.m. to 5:00 p.m.
Booth(s): 2 booth(s)
Location: 6000 Grade Blvd. San Bernardino CA 92888
Deposit: $500.00

If you have any questions, please contact us by:
Phone Number: (909)111-4567
Fax: (909)111-5678
E-mail Address: exhibitorexpo@hotmail.com

Figure 4.8 Receipt Page (for Exhibitor)

4.2.8 Failure Page for Exhibitor

This page tells the exhibitor that his/her credit card have a problem. In this page, there is a link that the exhibitor can use that link to go back to registration form page and refill the form.
Dear Bob Frick,

Your credit card has a problem. We cannot continue your online payment. Wrong credit card information, please refill the form.

Register for Exhibitors

Figure 4.9 Failure Page (for Exhibitor)

4.2.9 Login Page for Exhibitor

This page is only for exhibitors. When an exhibitor registers the expo, the exhibitor will get the username and password. Using the username and password, the exhibitor can login to Figure 4.11 Modify profile page.
Figure 4.10 Login Page (for Exhibitor)

4.2.10 Modify Profile Page

This page is for exhibitors to change their information including password, phone number, and address and so on.
4.2.11 Contact Us Page

This page offers a couple ways for visitors and exhibitors to contact with staffs of expo.
Contact Us

Show Time: May 1-5, 2004 from 9:00 a.m. to 5:00 p.m.
Exhibit Profile: Electric Appliances

Question about visitors: you can use either ways
Phone Number: (909)123-4567
Fax: (909)234-5678
E-mail Address: expol@hotmail.com

Question about exhibitors: you can use either ways
Phone Number: (909)987-4567
Fax: (909)987-5678
E-mail Address: expo2@hotmail.com

Figure 4.12 Contact Us Page
Like I mentioned before, web-performance is a big issue in the expo website. If it can not handle the traffic, then the expo may lose the potential visitors or lose the information of users. In this project, I’ll implement a web performance testing procedure to examine the performance of this system. To see if the system is stable and to see how many pages it can handle.

5.1 Web-Performance Testing Procedure

The web performance testing procedure will examine the system’s throughput, average response time, and percent failures under various loads (pages per minute). Throughput indicates the average number of transactions completed within a period of time. Response time is the time elapsed from the time a request is sent from the browser to the time that the response is received by the browser. It also includes the connection and shutdown times when a connection is not persistent. Percent failure means the percentage of web page requests that fail. Failure case happens when testing procedure sends a request to expo website and expo website returns an exception to a browser.
5.1.1 Two Configurations

Two configurations will be tested: a basic implementation, and an advanced implementation. And these two configurations will run in Linux.

5.1.1.1 Basic Configuration. In the basic implementation, the database server and Tomcat will run in the same machine. All HTTP requests will be handled by the web application running in Tomcat. See the Figure 5.1 Basic Configuration. In this configuration, I ran the Tomcat and database on the ors.ias.csusb.edu machine.

![Diagram](image.png)

Figure 5.1 Basic Configuration

5.1.1.2 Advanced Configuration. The advanced implementation will rely on two computers. Apache web server will run on one machine, and Tomcat will run on the other machine. All HTTP requests will be sent to the Apache web server. Requests for static HTML files, images and dynamic content will be routed to Tomcat running on the other machine. Thus, Apache will run as a reverse proxy for the web application running in Tomcat. Additionally, SSL connection requests will be handled by
Apache. When a connecting client submits an HTTPS request to the site, the browser and Apache will establish an SSL session. The browser will then submit an HTTP request through this secure channel. When Apache receives the request, it routes the request to Tomcat through an unencrypted TCP connection. See the Figure 5.2 Advanced Configuration.

The performance gains from the advanced implementation will be compared to the basic single machine implementation.

In this configuration, I used wiki2.ias.csusb.edu as apache server and ors.ias.csusb.edu as Tomcat server.

![Diagram](image)

Figure 5.2 Advanced Configuration

5.1.2 Testing Procedure Implementation

The class descriptions for the testing procedure are summarized as following:
1. SampleCreator: A Java class that produces the threads to handle the request connections.

2. RequestCreator: A Java class that makes the connection to the page which we want to test and counts the response time.

3. NormalPage: A Java class that holds the normal page information including the link, number of pages per minute and running time.

4. LoginPage: A Java class that extends the class NormalPage. It holds the Login page information including the link, number of pages per minute, running time, username and password.

5. VRegPage: A Java class that extends Thread. It holds the Visitor page information including the link, number of pages per minute and running time. It generates the data of the visitors automatically.

6. ERegPage: A Java class that extends Thread. It holds the Exhibitor page information including the link, number of pages per minute and running time. It generates the data of the exhibitors automatically.

7. RequestObject: A JavaBean object that stores the response time of tested page.
8. **AverageValueObject**: A JavaBean object that stores the average response time and average throughput of the tested pages.

9. **DiagramDraw**: A Java class that performs the visual interface which displays the result of this testing.

10. **Interface**: A Java class that performs a window which contents DiagramDraw class.

11. **Tester**: A main class that reads the data from testing.properties file tests the expo server pages and outputs the results into result.txt and history.txt files.

The class diagram is shown in Figure 5.3.
Figure 5.3 Class Diagram
5.2 Assumptions

5.2.1 Performance Metrics

Like I mention before, this testing procedure examines the system’s throughput, response time and percent failure in two configurations. Without the coordinate performance-testing unit, we can’t compare the web performance between these two configurations. Table 5.1 Performance Metrics shows the each unit I use for each testing performance.

Table 5.1 Performance Metrics

<table>
<thead>
<tr>
<th>Name of Performance</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Percent Failure</td>
<td>percentage</td>
</tr>
<tr>
<td>Response Time (RT)</td>
<td>millisecond (ms)</td>
</tr>
<tr>
<td>Average Response Time</td>
<td>ms/Page</td>
</tr>
<tr>
<td>Average Normal Page RT</td>
<td>ms/NormalPage</td>
</tr>
<tr>
<td>Average Login Page RT</td>
<td>ms/LoginPage</td>
</tr>
<tr>
<td>Average Visitor Registration Page RT</td>
<td>ms/VisitorPage</td>
</tr>
<tr>
<td>Average Exhibitor Registration Page RT</td>
<td>ms/ExhibitorPage</td>
</tr>
<tr>
<td>Throughput (T)</td>
<td>Success-Page/min</td>
</tr>
<tr>
<td>Normal Page T</td>
<td>Success-NormalPage/min</td>
</tr>
<tr>
<td>Login Page T</td>
<td>Success-LoginPage/min</td>
</tr>
<tr>
<td>Visitor Page T</td>
<td>Success-VisitorPage/min</td>
</tr>
<tr>
<td>Exhibitor Page T</td>
<td>Success-ExhibitorPage/min</td>
</tr>
</tbody>
</table>
5.2.2 Scenarios

There are many situations that will happen in the expo server. But I suppose that there are only four situations happening.

1. Normal users: Normal users are the people who are not interested in the expo. They visit the main page and leave the server.

2. Visiting users: Visiting users are the people who want to attend the expo. They visit the main page. They go to Visitor page and buy the tickets. After they get the response (receipt page), they leave the server.

3. Exhibiting users: Exhibiting users are the users that want to be the exhibitors of the expo. They visit the main page, and they are interested. They go to Information for Exhibitor page and get the information. Then they go the Exhibitor Page to order the booths. After they get the response (receipt page), they leave the server.

4. Login users: Login users are the exhibitors that have registered the expo server. They go to main page and want to check their profiles. They go to Login Page and login server. They check their profile on Modify Profile Page. After they make
the changes, they go to Logout Page and logout the server. Then they leave the server.

5.2.3 Distribution Assumption

Now we have four situations. And we need to distribute the rate of these four situations. Suppose there are 100 users that access the server. I test two cases, A and B. In case A, I assume that 90 users are normal users, 8 users are visiting users, 1 user is exhibiting user and 1 user is login user. In case B, I assume that 20 users are normal users, 60 users are visiting users, 15 users are exhibiting user and 5 users are login user. See Table 5.2 User Distribution.

Table 5.2 User Distribution

<table>
<thead>
<tr>
<th>Scenarios</th>
<th>Case A</th>
<th>Case B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal user</td>
<td>90</td>
<td>20</td>
</tr>
<tr>
<td>Visiting user</td>
<td>8</td>
<td>60</td>
</tr>
<tr>
<td>Exhibiting user</td>
<td>.1</td>
<td>15</td>
</tr>
<tr>
<td>Login user</td>
<td>1</td>
<td>5</td>
</tr>
</tbody>
</table>

Now, we know the rate of these four users. The testing procedure is used to testing the server's pages. So we need to know how many pages the procedure should test. Before that, I want to explain how I separate the all pages into four categories: (1) NormalPage: it's a
simple page. Users only request the page without filling any value. It includes main page, receipt page, information page and logout page. (2) VRegPage (Visitor Registration Page): it’s a page with registration form. Users need to fill all the information that it requests. It includes visitor page. (3) ERegPage (Exhibitor Registration Page): it’s a page with registration form. Users need to fill all the information that it requests. It includes exhibitor page. (4) Loginpage: Users need to fill the username and password to login server. In order to simplify the categories of pages, login page and modify profile page belong to LoginPage.

From Table 5.3 User-Page Distribution, we can know a Normal user only requests 1 NormalPage, a Visiting user requests 2 NormalPage and 1 VRegPage, a Exhibiting user requests 3 NormalPage and 1 ERegPage, and a Login user requests 2 NormalPage and 2 LoginPage.
Table 5.3 User-Page Distribution

<table>
<thead>
<tr>
<th>User</th>
<th>NormalPage</th>
<th>VRegPage</th>
<th>ERegPage</th>
<th>LoginPage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal user</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Visiting user</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Exhibiting user</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Login user</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

Suppose that there are 100 users visiting the expo server. Table 5.4 User-Page Distribution (100 users) in Case A shows the total number of each page that are

Table 5.4 User-Page Distribution (100 users) in Case A

<table>
<thead>
<tr>
<th>User</th>
<th>NormalPage</th>
<th>VRegPage</th>
<th>ERegPage</th>
<th>LoginPage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal user</td>
<td>90</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Visiting user</td>
<td>16</td>
<td>8</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Exhibiting user</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Login user</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Total page</td>
<td>111</td>
<td>8</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>
requested by 100 users in Case A. Table 5.5 User-Page Distribution (100 users) in Case B shows the total number of each page that are requested by 100 users in Case B.

Table 5.5 User-Page Distribution (100 users) in Case B

<table>
<thead>
<tr>
<th>User</th>
<th>NormalPage</th>
<th>VRegPage</th>
<th>ERegPage</th>
<th>LoginPage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal user</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Visiting user</td>
<td>120</td>
<td>60</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Exhibiting user</td>
<td>45</td>
<td>0</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>Login user</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>Total page</td>
<td>195</td>
<td>60</td>
<td>15</td>
<td>10</td>
</tr>
</tbody>
</table>

From Table 5.4 and Table 5.5, we can count the percentage of the each page category. Table 5.6, Page Category Percentage, shows the results.

Table 5.6 Page Category Percentage

<table>
<thead>
<tr>
<th>Page Category</th>
<th>Case A</th>
<th>Case B</th>
</tr>
</thead>
<tbody>
<tr>
<td>NormalPage</td>
<td>91 %</td>
<td>70 %</td>
</tr>
<tr>
<td>VRegPage</td>
<td>6 %</td>
<td>21 %</td>
</tr>
<tr>
<td>ERegPage</td>
<td>1 %</td>
<td>5 %</td>
</tr>
<tr>
<td>LoginPage</td>
<td>2 %</td>
<td>4 %</td>
</tr>
</tbody>
</table>
From Table 5.6 we can know that in case A there are 91 NormalPage requests, 6 VRegPage requests, 1 ERegPage request and 2 LoginPage requests in 100 requests sent by users, and in case B, there are 70 NormalPage requests, 21 VRegPage requests, 5 ERegPage requests and 4 LoginPage requests in 100 requests sent by users. With Table 5.6, we know the percentage of each page category in case A and B. Now we can test the expo server using these rates.

5.3 Results

From the assumptions in Section 5.2, we know the percentage of each category page. We can start to examine the expo web site. In order to avoid the unstable connection and other factors, I tested each load (requests per minute) twenty minutes and counted their averages. Also, in order to avoid a bottleneck occurring in the traffic simulator, I used twelve machines. First, I ran the testing procedure on the six machines, and then I ran the testing procedure on the twelve machines at load 2400 in case B to see if there are a lot of differences between the two data. The following results were obtained from result.txt file of the tester performed. All the measures are averages calculated from a range of actual results.
without unusual data. Unusual data are those response times which are abnormal (too small or too big).

In the table 5.7, it shows the response time and the failure rate of running the case B at load 2400 in 6 machines and 12 machines. It includes the two averages and two standard deviations, and first one is calculated by

Table 5.7 Result at Load 2400 in Case B

<table>
<thead>
<tr>
<th>Load 2400 (20 mins)</th>
<th>Response time in 6 machines</th>
<th>Failure rate in 6 machines</th>
<th>Response time in 12 machines</th>
<th>Failure rate in 12 machines</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12082</td>
<td>22</td>
<td>10459</td>
<td>22</td>
</tr>
<tr>
<td>2</td>
<td>9676</td>
<td>22</td>
<td>12198</td>
<td>21</td>
</tr>
<tr>
<td>3</td>
<td>10576</td>
<td>21</td>
<td>10292</td>
<td>23</td>
</tr>
<tr>
<td>4</td>
<td>9361</td>
<td>32</td>
<td>11149</td>
<td>22</td>
</tr>
<tr>
<td>5</td>
<td>10891</td>
<td>23</td>
<td>9632</td>
<td>36</td>
</tr>
<tr>
<td>6</td>
<td>12314</td>
<td>19</td>
<td>10434</td>
<td>22</td>
</tr>
<tr>
<td>7</td>
<td>11529</td>
<td>29</td>
<td>11456</td>
<td>21</td>
</tr>
<tr>
<td>8</td>
<td>10647</td>
<td>22</td>
<td>8666</td>
<td>24</td>
</tr>
<tr>
<td>9</td>
<td>11304</td>
<td>22</td>
<td>12154</td>
<td>19</td>
</tr>
<tr>
<td>10</td>
<td>17529</td>
<td>24</td>
<td>9450</td>
<td>22</td>
</tr>
<tr>
<td>Average</td>
<td>10650</td>
<td>23</td>
<td>10589</td>
<td>23</td>
</tr>
<tr>
<td>Standard Deviation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Abnormal 11</td>
<td>1295</td>
<td>3.9</td>
<td>1161</td>
<td>4.7</td>
</tr>
<tr>
<td>Abnormal 12</td>
<td></td>
<td></td>
<td>19632</td>
<td>76</td>
</tr>
<tr>
<td>Average</td>
<td>11275</td>
<td>24</td>
<td>11016</td>
<td>28</td>
</tr>
<tr>
<td>Standard Deviation</td>
<td></td>
<td></td>
<td>3120</td>
<td>15.8</td>
</tr>
</tbody>
</table>
ten normal data and second one is calculated by normal and abnormal data. We can see there is no big difference between 6 machines and 12 machines in response time and failure rate. It indicates that it is sufficient to use 6 machines to run the testing instead of using 12 machines.

5.3.1 Response Time

This measure indicates how long it takes for a page to be generated from a request to response receipt.

In session 5.1, I mentioned that I will measure the testing performance in two configurations. The result is showed on Table 5.8. In Table 5.8, I measured the response times from load 300-3000. I also calculated the standard deviation of each load. Figure 5.4 is the graphic result of Table 5.8. We can see: (1) the response time of case A is distinctly smaller than the response time of case B. In case A, there are more normal pages and normal pages took less time to transact. In case B, there are more login pages, visitor pages and exhibitor pages. These pages access the database, and visitor and exhibitor pages also connect to payment gateway server. So these pages in case B took more time to transact. (2) Before load 1800, it shows that web performance of case B in advanced implementation are better than web performance of case B in basic implementation. In case A, there is no much
difference between the performance in basic configuration and the performance in advanced configuration. That because for https protocol take more time to process. In advanced configuration, apache is a proxy web server which deals with the SSL and forwards the requests to Tomcat. Between Apache and Tomcat, there are just simple html documents transmitted over http protocol. Also, we found the Apache can handle more https pages and quicker than Tomcat. In case A, there are more http pages (normal pages), and these pages don’t need to encrypt and decrypt. These pages cost less time to transact in basic and advanced configurations. So it doesn’t make much difference of the response time in two configurations.

Table 5.8 Results of Response Time

<table>
<thead>
<tr>
<th>load</th>
<th>case A</th>
<th>stdev</th>
<th>case B</th>
<th>stdev</th>
<th>case A (adv)</th>
<th>stdev</th>
<th>case B (adv)</th>
<th>stdev</th>
</tr>
</thead>
<tbody>
<tr>
<td>300</td>
<td>188</td>
<td>37.9</td>
<td>382</td>
<td>39.4</td>
<td>196</td>
<td>32.2</td>
<td>355</td>
<td>106.0</td>
</tr>
<tr>
<td>600</td>
<td>168</td>
<td>12.2</td>
<td>531</td>
<td>45.3</td>
<td>145</td>
<td>37.9</td>
<td>376</td>
<td>69.2</td>
</tr>
<tr>
<td>900</td>
<td>180</td>
<td>20.6</td>
<td>13674</td>
<td>3296.8</td>
<td>191</td>
<td>55.5</td>
<td>335</td>
<td>35.1</td>
</tr>
<tr>
<td>1200</td>
<td>220</td>
<td>36.1</td>
<td>15535</td>
<td>3076.2</td>
<td>163</td>
<td>43.9</td>
<td>442</td>
<td>124.7</td>
</tr>
<tr>
<td>1500</td>
<td>259</td>
<td>32.4</td>
<td>14733</td>
<td>1160.7</td>
<td>259</td>
<td>57.2</td>
<td>592</td>
<td>110.6</td>
</tr>
<tr>
<td>1800</td>
<td>261</td>
<td>10.7</td>
<td>13245</td>
<td>908.3</td>
<td>282</td>
<td>132.8</td>
<td>886</td>
<td>557.7</td>
</tr>
<tr>
<td>2100</td>
<td>709</td>
<td>450.9</td>
<td>11051</td>
<td>1690.7</td>
<td>215</td>
<td>94.5</td>
<td>27077</td>
<td>9817.8</td>
</tr>
<tr>
<td>2400</td>
<td>8931</td>
<td>2736.8</td>
<td>10650</td>
<td>1295.0</td>
<td>194</td>
<td>46.1</td>
<td>39129</td>
<td>3924.8</td>
</tr>
<tr>
<td>2700</td>
<td>10897</td>
<td>2735.4</td>
<td>10021</td>
<td>737.8</td>
<td>231</td>
<td>45.7</td>
<td>46532</td>
<td>3003.3</td>
</tr>
<tr>
<td>3000</td>
<td>11170</td>
<td>2056.9</td>
<td>11367</td>
<td>2660.4</td>
<td>408</td>
<td>174.2</td>
<td>47144</td>
<td>3478.8</td>
</tr>
</tbody>
</table>
(3) After load 2100, the response time of case B in advanced implementation increased fast but the response time in basic implementation didn’t keep increasing. But in case A, the situation is reverse, response time in basic implementation increased a lot. It could be that https pages cost a lot of encryption and decryption between proxy and web browser in advanced configuration. When pages load increase to a certain load, proxy and tomcat servers can not handle. So the response time

![Response Time Graph](image)

Figure 5.4 Graphic Result of Response Time
increases fast in case B, but not in case A in advanced configuration. In basic implementation, response times of case A increase largely at load 2400.

5.3.2 Throughput

Throughput measures how many pages can be generated within a particular period of time.

Table 5.9 is the throughput of case A and case B connecting to the basic and advanced configurations. The Figure 5.5 is the graphic result of Table 5.9.

I tested page load 300-5100. From Figure 5.5, it shows that success cases decrease from load 3000 and more success pages in case A than in Case B at any page load. In case A, there are more success pages in advanced configuration than in basic configuration. But in case B, there are more success pages in basic configuration than in advanced configuration, unlike I expect.

In fact, there are a lot of factors that affect the condition of Internet. The connection is not stable all the time. The result is unlike I except. It shows there is no rule in this graph. Only we can know when page load gets bigger the failure cases happen.
Table 5.9 Result of Throughput

<table>
<thead>
<tr>
<th>Load</th>
<th>Case A (basic)</th>
<th>Case B (basic)</th>
<th>Case A (adv)</th>
<th>Case B (adv)</th>
</tr>
</thead>
<tbody>
<tr>
<td>300</td>
<td>300</td>
<td>300</td>
<td>300</td>
<td>300</td>
</tr>
<tr>
<td>600</td>
<td>600</td>
<td>600</td>
<td>600</td>
<td>600</td>
</tr>
<tr>
<td>900</td>
<td>900</td>
<td>877</td>
<td>900</td>
<td>900</td>
</tr>
<tr>
<td>1200</td>
<td>1200</td>
<td>1059</td>
<td>1200</td>
<td>1200</td>
</tr>
<tr>
<td>1500</td>
<td>1500</td>
<td>1270</td>
<td>1500</td>
<td>1500</td>
</tr>
<tr>
<td>1800</td>
<td>1800</td>
<td>1441</td>
<td>1800</td>
<td>1800</td>
</tr>
<tr>
<td>2100</td>
<td>2042</td>
<td>1584</td>
<td>2100</td>
<td>1393</td>
</tr>
<tr>
<td>2400</td>
<td>2272</td>
<td>1834</td>
<td>2400</td>
<td>1582</td>
</tr>
<tr>
<td>2700</td>
<td>2471</td>
<td>2096</td>
<td>2699</td>
<td>1875</td>
</tr>
<tr>
<td>3000</td>
<td>2631</td>
<td>2263</td>
<td>3000</td>
<td>2017</td>
</tr>
<tr>
<td>3300</td>
<td>2617</td>
<td>2187</td>
<td>3299</td>
<td>629</td>
</tr>
<tr>
<td>3600</td>
<td>2844</td>
<td>2253</td>
<td>3600</td>
<td>1466</td>
</tr>
<tr>
<td>3900</td>
<td>3418</td>
<td>1223</td>
<td>3872</td>
<td>417</td>
</tr>
<tr>
<td>4200</td>
<td>2069</td>
<td>2627</td>
<td>3763</td>
<td>428</td>
</tr>
<tr>
<td>4500</td>
<td>3167</td>
<td>1688</td>
<td>4262</td>
<td>416</td>
</tr>
<tr>
<td>4800</td>
<td>3059</td>
<td>2185</td>
<td>4474</td>
<td>528</td>
</tr>
<tr>
<td>5100</td>
<td>2369</td>
<td>2099</td>
<td>4845</td>
<td>2071</td>
</tr>
</tbody>
</table>
5.3.3 Failure Rate

Failure Rate measures the rate of fail cases (pages didn’t receipt a response) divided into total cases.

Table 5.10 is the failure rates of case A and case B connecting to the basic and advanced configurations. The Figure 5.6 is the graphic result of Table 5.10. It shows that in case B, the failure cases happen at load 900 and in case A, the failure case happen at load 2100. Before page load 2100, the performance of advanced implementation
is better than basic implementation. But after page load 2100, the performance of case B in advanced is worse than the performance of case B in basic. The same results are showed in response time and throughput.

In the basic configuration, Tomcat can handle maximum requests around 3300 pages per minute in case A and 2400 pages per minute in case B. In the advanced configuration, server can handle maximum requests around 2400 pages per minute.

Table 5.10 Result of Failure Rate

<table>
<thead>
<tr>
<th>Load</th>
<th>Case A (basic)</th>
<th>Case B (basic)</th>
<th>Case A (adv)</th>
<th>Case B (adv)</th>
</tr>
</thead>
<tbody>
<tr>
<td>300</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>600</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>900</td>
<td>0.00</td>
<td>2.60</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>1200</td>
<td>0.00</td>
<td>11.79</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>1500</td>
<td>0.00</td>
<td>15.36</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>1800</td>
<td>0.00</td>
<td>19.93</td>
<td>0.00</td>
<td>0.01</td>
</tr>
<tr>
<td>2100</td>
<td>2.76</td>
<td>24.59</td>
<td>0.00</td>
<td>33.65</td>
</tr>
<tr>
<td>2400</td>
<td>5.35</td>
<td>23.60</td>
<td>0.00</td>
<td>34.08</td>
</tr>
<tr>
<td>2700</td>
<td>8.50</td>
<td>22.39</td>
<td>2.20</td>
<td>30.54</td>
</tr>
<tr>
<td>3000</td>
<td>12.30</td>
<td>24.58</td>
<td>0.00</td>
<td>32.76</td>
</tr>
<tr>
<td>3300</td>
<td>20.68</td>
<td>33.72</td>
<td>0.02</td>
<td>80.94</td>
</tr>
<tr>
<td>3600</td>
<td>20.99</td>
<td>66.04</td>
<td>0.00</td>
<td>59.29</td>
</tr>
<tr>
<td>3900</td>
<td>12.37</td>
<td>42.23</td>
<td>0.71</td>
<td>93.59</td>
</tr>
<tr>
<td>4200</td>
<td>28.88</td>
<td>37.46</td>
<td>10.40</td>
<td>93.88</td>
</tr>
<tr>
<td>4500</td>
<td>29.60</td>
<td>62.49</td>
<td>5.29</td>
<td>94.45</td>
</tr>
<tr>
<td>4800</td>
<td>36.27</td>
<td>54.49</td>
<td>6.79</td>
<td>93.40</td>
</tr>
<tr>
<td>5100</td>
<td>53.55</td>
<td>58.84</td>
<td>4.99</td>
<td>75.64</td>
</tr>
</tbody>
</table>
Figure 5.6 Graphic Result of Failure Rate

Table 5.11 is the failure rates of case B connecting to the basic configuration in running time 60 and 120 minutes at load 300, 600, 1200, 2100 and 3300. The Figure 5.7 is the graphic result of Table 5.11.

In Figure 5.7, it shows that the failure rate is almost 100% at load 2100 at running time 120 minutes (2 hours). At load 1200, the failure rates at running time 60 and 120 are slightly larger than the failure rate at running time 20.

When running time is increased, the failure rate would be affected. The affect would be more apparent in
large page load than in small page load. I tested several cases to determine the effect of running time on the performance metrics. Table 5.11 and Figure 5.7 summarize my results. These results indicate that at lower load levels (less than 2100 pages per minute) the performance metrics are approximately equal. However, for larger load levels, there is a significant discrepancy between the short and long running times. If the site expects to get a relatively short-term burst of traffic (20 minutes or less), then the 20-minute test data is sufficient to measure the expected performance at load levels greater than 2000. However, if the site expects high load levels for a period longer than 20 minutes, then the 20-minute load performance metrics should not be used.
Table 5.11 Result of Failure Rate (2)

<table>
<thead>
<tr>
<th>Load</th>
<th>20 mins</th>
<th>60 mins</th>
<th>120 mins</th>
</tr>
</thead>
<tbody>
<tr>
<td>300</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>600</td>
<td>0</td>
<td>0</td>
<td>1.79</td>
</tr>
<tr>
<td>900</td>
<td>2.60</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1200</td>
<td>11.79</td>
<td>20.04</td>
<td>19.14</td>
</tr>
<tr>
<td>1500</td>
<td>15.36</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1800</td>
<td>19.93</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2100</td>
<td>24.59</td>
<td>66.71</td>
<td>82.83</td>
</tr>
<tr>
<td>2400</td>
<td>23.60</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2700</td>
<td>22.39</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3000</td>
<td>24.58</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3300</td>
<td>33.72</td>
<td>31.90</td>
<td>90.33</td>
</tr>
<tr>
<td>3600</td>
<td>66.04</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3900</td>
<td>42.23</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4200</td>
<td>37.46</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4500</td>
<td>62.49</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4800</td>
<td>54.49</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5100</td>
<td>58.83</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Figure 5.7 Graphic Result of Failure Rate (2)
CHAPTER SIX
CONCLUSION AND FUTURE DIRECTIONS

6.1 Conclusion

There are a lot of factors that affect the web performance, for example SSL, network distance and CPU memory. In my project, I compare two configurations and two different rates of requests. Definitely, a normal page has better performance because data is transferred over the HTTP protocol without any secure protection.

In the comparison of basic and advanced configuration, we know: (1) before page load of 2100 in case A, there is not much difference between the performance in basic configuration and the performance in advanced configuration. But in case B, web performance in advanced implementation is much better than web performance in basic implementation. (2) After page load 2100, in case A the performance in advanced configuration is much better than the performance in basic configuration. But in case B, web performance in advanced implementation is worse than web performance in basic implementation.

As we know that Apache handles the SSL and Tomcat handles expo server in the advanced implementation. For security issue, the result demonstrates that Apache deals
with HTTPS protocol better than Tomcat. But when page load is getting big to a certain load, the performance in advanced configuration would become worse than in basic configuration. In a real world, it's rare to find a server that needs to handle more than two thousand requests in one minute. Therefore for the real case, the advanced configuration (in which Apache is a proxy server that deals with SSL connections) can handle a larger number of page flows (failure rate is smaller in advanced than in basic under page load 2100). In the advanced configuration, Apache deals with front-end tasks and Tomcat handles the backend jobs. This configuration reduces the system load on both web servers; furthermore, the whole system becomes more stable and have better performance than just let Tomcat handle everything (supposing the page load under 2100 pages per minute).

6.2 Future Directions

The web-performance testing procedure is an application only can be used to testing the expo server. For the future direction, web-performance testing procedure can be expanded to examine any web server's performance.
The analysis performed in this project can be expanded to better analyze and locate the system components that are responsible for limiting the ability of the expo site to handle additional traffic. Additionally, the current version of the testing procedure can not determine the break point that makes the server completely fail. Testing at higher load levels is needed to determine this.

And considering the expo server, it can be developed more functionalities, and the user interface can be made friendlier.
APPENDIX A

APACHE SETTING
APACHE SETTING

Apache is the most popular open source web server. In previous version (1.x), the regular application doesn’t support https. There are two solutions for this problem, either add a module called mod-ssl or install a special version called Apache-SSL. Fortunately, mod-ssl is included in the latest version (2.x). In the case, we only need to open this option while configuring the apache. Also, we can use a module called mod-proxy to let apache web server be a proxy server. Later on, I will show how to setup the system step by step.

1. Download Apache and OpenSSL:
   Apache 2.0.48: The latest state release version of Apache HTTP server. (http://www.apache.org)
   OpenSSL 0.9.7c: The latest version of OpenSSL which is toolkit implementing the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS v1) protocols. (http://www.openssl.com)

2. Configure and Install OpenSSL
   #tar -zxvf openssl-0.9.7c.tar.gz
   #cd openssl-0.9.7c
   #./config #default configuration (/usr/local/ssl)
   #make
   #make test
   #make install

3. Configure and Install Apache
   #tar -zxvf httpd-2.0.48.tar.gz
   #cd httpd-2.0.48
   #./configure --enable-proxy
   --enable-proxy-connect
   --enable-proxy-http
   --enable-ssl
   #configure ssl module
   --with-ssl=/usr/local/ssl #SSL/TSL toolkit
#make
#make install

4. Build Server Certificate

```
#cd /usr/local/apache2/conf/ssl.key/
#openssl genrsa -des -out server.key 1024
#openssl rsa -in server.key -out server.key
#openssl req -new -key server.key -out server.csr
#openssl x509 -in server.csr -req -signkey server.key -out server.crt
#mv server.csr ../ssl.csr
#mv server.crt ../ssl.crt
```

5. Modify Apache Configuration (httpd.conf)

```
#Dynamic Shared Object (DSO) Support

# To be able to use the functionality of a module which was built as a DSO you
# have to place corresponding 'LoadModule' lines at this location so the
directives contained in it are actually available _before_ they are used.
# Statically compiled modules (those listed by `httpd -l`) do not need
# to be loaded here.
#
# Example:
# LoadModule foo_module modules/mod_foo.so

ProxyRequests On
ProxyPass /test http://ors.ias.csusb.edu:8080/expo
ProxyPassReverse /test http://ors.ias.csusb.edu:8080/expo
```

Proxy Setting and server mapping URL
6. Startup the Apache server with SSL

    # cd /usr/local/apache2/bin/apachectl startssl

7. Check if the proxy and ssl work
APPENDIX B

WEB APPLICATION DESCRIPTOR FILE
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd">

<web-app>
    <display-name>Welcome to Tomcat</display-name>
    <description>
        Welcome to Tomcat
    </description>
    <listener>
        <listener-class>project.expo.Visitor</listener-class>
    </listener>
    <listener>
        <listener-class>project.expo.Exhibitor</listener-class>
    </listener>

    <servlet>
        <servlet-name>processV</servlet-name>
        <servlet-class>project.expo.ProcessV</servlet-class>
    </servlet>

    <servlet>
        <servlet-name>processE</servlet-name>
        <servlet-class>project.expo.ProcessE</servlet-class>
    </servlet>

    <servlet>
        <servlet-name>editfile</servlet-name>
        <servlet-class>project.expo.Editfile</servlet-class>
    </servlet>

    <servlet>
        <servlet-name>update</servlet-name>
        <servlet-class>project.expo.Update</servlet-class>
    </servlet>
</web-app>
<servlet>
  <servlet-name>Logout</servlet-name>
  <servlet-class>project.expo.Logout</servlet-class>
</servlet>

<servlet-mapping>
  <servlet-name>processV</servlet-name>
  <url-pattern>/processV.html</url-pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>processE</servlet-name>
  <url-pattern>/processE.html</url-pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>editfile</servlet-name>
  <url-pattern>/editfile.html</url-pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>update</servlet-name>
  <url-pattern>/update.html</url-pattern>
</servlet-mapping>

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

<welcome-file-list>
  <welcome-file>/page/index.jsp</welcome-file>
</welcome-file-list>

</web-app>
APPENDIX C

JSP FILES
<html>
<head>
<title>Expo Server Page - Index</title>
</head>
<body background="picExpo/12.jpg">
<br>
<center>
<img border="0" src="picExpo/21.jpg" width="313" height="111">
</center>
<br><p><a href="https://ors.ias.csusb.edu:8443/expo/page/visitor.jsp">Visitor Page</a>
<a href="exhibitor.jsp">Exhibitor Page</a>
<a href="https://ors.ias.csusb.edu:8443/expo/page/login.jsp">Login Page</a>
<a href="contact.jsp">Contact Page</a></p>
<br>
<table align="center">
<tr><td><img border="0" src="picExpo/expo1.gif" width="216" height="132"></td></tr>
<tr><td><img border="0" src="picExpo/expo2.gif" width="216" height="132"></td></tr>
<br><p><table align="center">
<tr><td><font face="David" size="4">SHOW SCHEDULE</font></td></tr>
<tr><td><font size="5" face="Monotype Corsiva" color="#CC0000">May 1-5, 2004</font></td></tr>
<tr><td><font color="#CC0000" face="Monotype Corsiva" size="5">OPEN HOUR: 9:00 a.m. - 5:00 p.m.</font></td></tr>
<tr><td><font face="David" size="4">EXHIBIT PROFILE</font></td></tr>
<tr><td><font color="#CC0000" face="Monotype Corsiva" size="5">Electric Appliances</font></td></tr>
<tr><td><font face="David" size="4">PLACE</font></td></tr>
<tr><td><font color="#CC0000" face="Monotype Corsiva" size="5">6000 Grade Blvd.</font></td></tr>
<tr><td><font face="David" size="4"></font></td></tr>
<tr><td><font color="#CC0000" face="Monotype Corsiva" size="5">San Bernardino CA 92888</font></td></tr>
</table>
</p></table>
Filename: visitor.jsp

```jsp
<%@page import = "java.util.*"%>
<%@page import = "project.expo.*"%

<html>
<head>
<title>Visitor Page</title>
</head>
<body background= "picExpo/12.jpg">
  <form action= "https://ors.ias.csusb.edu:8443/expo/processV.html" method = "get">
    <br>
    <center><p>
      <font size=6 face="Cataneo BT" color="#800000">Registration Form for Visitor</font></p>
    </center>
    <br>
    <table align="center">
      <tr><p>
        <td><font color="#800000" size="4" face="Cataneo BT">Show Time</font></td>
        <td><font color="#800000" size="4" face="Cataneo BT">May 1-5, 2004 from 9:00 a.m. to 5:00 p.m.</font></td></tr>
      <tr><p>
        <td><font face="Cataneo BT" size="4" color="#800000">Price</font></td>
        <td><font face="Cataneo BT" size="4" color="#800000">US$ 10.00 per person per day.</font></td></tr>
    </table>
    <br>
    <% if (session.getAttribute("fname") == null) {
    %>
    <table align = "center">
      <tr><p>
        <td><font size="4" face="Georgia" color="#000080">First Name</font></td>
        <td><input type="text" name="firstname" size="15" /></td></tr>
    </table>
    <% %>
  </form>
</body>
</html>
```
<table>
<thead>
<tr>
<th>Last Name</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>City</td>
<td>State</td>
</tr>
<tr>
<td>Zip Code</td>
<td>Phone</td>
</tr>
<tr>
<td>E-mail</td>
<td>Credit Card</td>
</tr>
<tr>
<td>Choose the Day</td>
<td>Expiration Month</td>
</tr>
<tr>
<td></td>
<td>Expiration Year</td>
</tr>
</tbody>
</table>

```html
<font size="4" face="Georgia" color="#000080"">Last Name</font></td>
<input type="text" name="lastname" size="15" />
</td></tr>
<tr><p>
<font size="4" face="Georgia" color="#000080"">Address</font></td>
<input type="text" name="address" size="40" />
</td></p></tr>
<tr><p>
<font size="4" face="Georgia" color="#000080"">City</font></td>
<input type="text" name="city" size="15" />
</td></p></tr>
<tr><p>
<font size="4" face="Georgia" color="#000080"">State</font></td>
<input type="text" name="state" size="4" />
</td></p></tr>
<tr><p>
<font size="4" face="Georgia" color="#000080"">Zip Code</font></td>
<input type="text" name="zipcode" size="6" />
</td></p></tr>
<tr><p>
<font size="4" face="Georgia" color="#000080"">Phone</font></td>
<input type="text" name="phonenumber" size="10" />
</td></p></tr>
<tr><p>
<font size="4" face="Georgia" color="#000080"">E-mail</font></td>
<input type="text" name="email" size="20" />
</td></p></tr>
<tr><p>
<font size="4" face="Georgia" color="#000080"">Credit Card</font></td>
<input type="text" name="creditnumber" size="20" />
</td></p></tr>
<tr><p>
<font size="4" face="Georgia" color="#000080"">Expiration Month</font>
<select name="day">
<option selected>05/01/04</option>
<option>05/02/04</option>
<option>05/03/04</option>
<option>05/04/04</option>
<option>05/05/04</option>
</select>
</td></p></tr>
<tr><p>
<font size="4" face="Georgia" color="#000080"">Expiration Year</font></td>
<input type="text" name="expYear" size="2" />
</td></p></tr>
```
<tr><p><td><font size="4" face="Georgia" color="#000080">How Many</font></td></p></tr><tr><td><input type="number" name="ticket" size="2"></td></tr>

```java
%
else {
%
```
<table><tr><td><font size="4" face="Georgia" color="#000080">Expiration Month<br>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&n...
Information for Exhibitor

Show Time:
May 1-5, 2004 from 9:00 a.m. to 5:00 p.m.

Exhibit Profile:
Electric Appliances

SPA CE COST
1. One booth (Standard booth (3m * 3m)): US$3,100
2. Two booths (Standard booth (3m * 6m)): US$6,100
3. Three or more booths (raw space only) (Standard booth (3m * 3m)): US$3,000 per booth
4. Each booth will be provided with 100V/500W electricity free of charge.
5. All the exhibitors will be charged US$500.00 for deposit.
6. No matter how many booths they order.

Apply online
Home Page
Filename: exhibitor2.jsp

<%@page import = "java.util.*"%
<%@page import = "project.expo.*"%>

<html>
<head>
<title>Exhibitor Page</title>
</head>
<body background="picExpo/12.jpg">

<form action="https://ors.ias.csusb.edu:8443/expo/processE.html" method="get">

<br>
<center>
<font size="6" face="Cataneo BT"><u>Registration Form for Exhibitor</u></font></center>
<br>

<table align="center">
<tr><p>
<td><font color="#800000" size="4" face="Cataneo BT">Show Time:&nbsp;</font></td>
<td><font color="#800000" size="4" face="Cataneo BT">May 1-5, 2004 from 9:00 a.m. to 5:00 p.m. </font></td></p></tr>
<tr><p>
<td><font face="Cataneo BT" size="4" color="#800000">Exhibit Profile:&nbsp;&nbsp;</font></td>
<td><font face="Cataneo BT" size="4" color="#800000">Electric Appliances</font></td></p></tr>
</table>

<br><br>

<% 
if( session.getAttribute("username") == null ) {
%

<table align = "center">
<tr><p>
<td><font size="4" face="Georgia" color="#000080">UserName</font></td>
<td><input type="text" name="username" size="15"></td></p></tr>
<tr><p>
<td><font size="4" face="Georgia" color="#000080">Password</font></td>
<td><input type="password" name="password" size="15"></td></p></tr>
<tr><p>
<td><font size="4" face="Georgia" color="#000080">Password again</font></td>
<td><input type="password" name="passwordagain" size="15"></td></p></tr>
<tr><p>
</tr>

%>
</table>

</font>

</form>
</body>
</html>
<table>
<thead>
<tr>
<th><strong>Text</strong></th>
<th><strong>Type</strong></th>
<th><strong>Size</strong></th>
<th><strong>Color</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>First Name</td>
<td>Input</td>
<td>15</td>
<td>#000080</td>
</tr>
<tr>
<td>Last Name</td>
<td>Input</td>
<td>15</td>
<td>#000080</td>
</tr>
<tr>
<td>Company</td>
<td>Input</td>
<td>20</td>
<td>#000080</td>
</tr>
<tr>
<td>Product</td>
<td>Input</td>
<td>30</td>
<td>#000080</td>
</tr>
<tr>
<td>Address</td>
<td>Input</td>
<td>40</td>
<td>#000080</td>
</tr>
<tr>
<td>City</td>
<td>Input</td>
<td>15</td>
<td>#000080</td>
</tr>
<tr>
<td>State</td>
<td>Input</td>
<td>4</td>
<td>#000080</td>
</tr>
<tr>
<td>Zip Code</td>
<td>Input</td>
<td>6</td>
<td>#000080</td>
</tr>
<tr>
<td>Phone</td>
<td>Input</td>
<td>10</td>
<td>#000080</td>
</tr>
<tr>
<td>E-mail</td>
<td>Input</td>
<td>20</td>
<td>#000080</td>
</tr>
<tr>
<td>Credit Card</td>
<td>Input</td>
<td>20</td>
<td>#000080</td>
</tr>
<tr>
<td>Expiration Month</td>
<td>Input</td>
<td>2</td>
<td>#000080</td>
</tr>
<tr>
<td>Expiration Year</td>
<td>Input</td>
<td>2</td>
<td>#000080</td>
</tr>
<tr>
<td>How Many Booths</td>
<td>&lt;input type=&quot;number&quot; name=&quot;booth&quot; size=&quot;2&quot;&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td>------------------------------------------</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>UserName</th>
<th>&lt;input type=&quot;text&quot; name=&quot;username&quot; size=&quot;15&quot; value=&quot;%=(String)session.getAttribute('username') %&quot;&gt;</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Password</th>
<th>&lt;input type=&quot;password&quot; name=&quot;password&quot; size=&quot;15&quot;&gt;</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Password again</th>
<th>&lt;input type=&quot;password&quot; name=&quot;passwordagain&quot; size=&quot;15&quot;&gt;</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>First Name</th>
<th>&lt;input type=&quot;text&quot; name=&quot;firstname&quot; size=&quot;15&quot; value=&quot;%=(String)session.getAttribute('fname') %&quot;&gt;</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Last Name</th>
<th>&lt;input type=&quot;text&quot; name=&quot;lastname&quot; size=&quot;15&quot; value=&quot;%=(String)session.getAttribute('lname') %&quot;&gt;</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Company</th>
<th>&lt;input type=&quot;text&quot; name=&quot;company&quot; size=&quot;20&quot; value=&quot;%=(String)session.getAttribute('company') %&quot;&gt;</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Product</th>
<th>&lt;input type=&quot;text&quot; name=&quot;product&quot; size=&quot;30&quot; value=&quot;%=(String)session.getAttribute('product') %&quot;&gt;</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>&lt;input type=&quot;text&quot; name=&quot;address&quot; size=&quot;40&quot; value=&quot;%=(String)session.getAttribute('address') %&quot;&gt;</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>City</th>
<th>&lt;input type=&quot;text&quot; name=&quot;city&quot; size=&quot;15&quot; value=&quot;%=(String)session.getAttribute('city') %&quot;&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field</td>
<td>Value</td>
</tr>
<tr>
<td>--------------</td>
<td>-------</td>
</tr>
<tr>
<td>State</td>
<td></td>
</tr>
<tr>
<td>Zip Code</td>
<td></td>
</tr>
<tr>
<td>Phone</td>
<td></td>
</tr>
<tr>
<td>E-mail</td>
<td></td>
</tr>
<tr>
<td>Credit Card</td>
<td></td>
</tr>
<tr>
<td>Expiration Month</td>
<td></td>
</tr>
<tr>
<td>Expiration Year</td>
<td></td>
</tr>
<tr>
<td>How Many Booths</td>
<td></td>
</tr>
</tbody>
</table>

```java
<tr><p>
<td><font size="4" face="Georgia" color="#000080">State</font></td>
<td><input type="text" name="state" size="4" value=""><%=(String )session.getAttribute("state") %"></td></p></tr>
<tr><p>
<td><font size="4" face="Georgia" color="#000080">Zip Code</font></td>
<td><input type="text" name="zipcode" size="6" value=""><%=(String )session.getAttribute("zip") %"></td></p></tr>
<tr><p>
<td><font size="4" face="Georgia" color="#000080">Phone</font></td>
<td><input type="text" name="phorienumber" size="10" value=""><%=(String)session.getAttribute("phone") %"></td></p></tr>
<tr><p>
<td><font size="4" face="Georgia" color="#000080">E-mail</font></td>
<td><input type="text" name="email" size="20" value=""><%=(String )session.getAttribute("email") %"></td></p></tr>
<tr><p>
<td><font size="4" face="Georgia" color="#000080">Credit Card</font></td>
<td><input type="text" name="creditnumber" size="20" value=""><%=(String)session.getAttribute("credit") %"></td></p></tr>
<tr><p>
<td><font size="4" face="Georgia" color="#000080">Expiration Month</font></td>
<td><input type="text" name="expMonth" size="2" value=""><%=(String )session.getAttribute("expMonth") %"></td></p></tr>
<tr><p>
<td><font size="4" face="Georgia" color="#000080">Expiration Year</font></td>
<td><input type="text" name="expYear" size="2" value=""><%=(String )session.getAttribute("expYear") %"></td></p></tr>
<tr><p>
<td><font size="4" face="Georgia" color="#000080">How Many Booths</font></td>
<td><input type="number" name="booth" size="2" value=""><%=(String )session.getAttribute("sbooth") %"></td></p></tr>
```
<%@page import = "project.expo.*"%>
<html>
<head>
<title>Expo Web Server - login</title>
</head>
<body background="picExpo/12.jpg">
<form action="https://ors.ias.csusb.edu:8443/expo/editfile.html" method="get">
<br><br><br>
<center>
<font size="6" face="Georgia">Login Page</font>
</center>
<br><br>
<table align="center">
<tr>
<th align="right">
USERNAME: &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
</th>
<td align="left">
<input type="text" name="username" maxlength="18" size="16">
</td>
</tr>
<tr>
<th align="right">
PASSWORD: &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
</th>
<td align="left">
<input type="password" name="password" maxlength="18" size="16">
</td>
</tr>
</table>
<br><br>
<center>
<input type="submit" value="SUBMIT">&nbsp;
<input type="reset" value="RESET">
</center>
<center>
<form>
</form>
</body>
<html>
<head>
title=Edit Profile</title></head>
<body background="page/picExpo/12.jpg">
<br>
<font size=6 face="Cataneo BT">Modify Profile</font>
<br/>
<form action="https://ors.ias.csusb.edu:8443/expo/update.html" method="get">
<center>
<br>
<%Exhibitor user = (Exhibitor)session.getAttribute("user"); %>
<table align="center" width="82"
<font size="4" face="Georgia" color="#000080">UserName</font></td>
<td><%=user.getUsername()%></td></p></tr>
<font size="4" face="Georgia" color="#000080">New Password</font></td>
<td><input type="password" name="password" size="15"></p></tr>
<font size="4" face="Georgia" color="#000080">New Password again</font></td>
<td><input type="password" name="passwordagain" size="15"></p></tr>
<font size="4" face="Georgia" color="#000080">First Name</font></td>
<td><input type="text" name="firstname" size="15" value="<%=user.getFirstname()%>"></td></p>
<font size="4" face="Georgia" color="#000080">Last Name</font></td>
<td><input type="text" name="lastname" size="15" value="<%=user.getLastname()%>"></td></p>
<font size="4" face="Georgia" color="#000080">Company</font></td>
<td><input type="text" name="company" size="30" value="<%=user.getCompany()%>"></td></p>
<font size="4" face="Georgia" color="#000080">Product</font></td>
<td><input type="text" name="product" size="30" value="<%=user.getProduct()%>"></td></p>
<font size="4" face="Georgia" color="#000080">Address</font></td>
<td><input type="text" name="address" size="40" value="<%=user.getAddress()%>"></td></p>
<font size="4" face="Georgia" color="#000080">City</font></td>
<td><input type="text" name="city" size="15" value="<%=user.getCity()%>"></td></p>
<font size="4" face="Georgia" color="#000080">State</font></td>
<td><input type="text" name="state" size="4" value="<%=user.getState()%>"></td></p>
</form>
</body>
</html>
<center>Congratulation</center><br/><br>'■ 1 ■ Q' ■ J■<table><tr><td>Dear <b>firstname lastname</b><br><br>Your online payment has been successfully charged.<br><br>Confirmed Number: <b>confirmednum</b><br>Date: <b>today</b><br>ShowTime: <b>day</b> from 9:00 a.m. to 5:00 p.m.<br>Ticket(s): <b>sticket</b> ticket(s)<br>Amount: $<br>If you have any questions, please contact us by:<br>Phone Number: (909)000-4567<br>Fax: (909)000-5678<br>E-mail Address: vitorexpo@hotmail.com<br></td></tr><tr><td><a href="http://ors.ias.csusb.edu:8080/expo/page/index.jsp">Home Page</a></td></tr><tr><td></td></tr></table><br><br>Filename: receiptE.jsp

<%@page import = "java.util.*"%>
<%@page import = "project.expo.*"%>

<html>
<head><title>Add exhibitor - receipt</title></head>

<body background="page/picExpo/12.jpg">
<br>
<center><font size="6" face="Cataneo BT">Congratulation</font></center>
<br/>
<table>
<tr><td>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&ndash;85

<%=(String)session.getAttribute("fname") + " " + (String)session.getAttribute("lname") %></b><br><br></td></tr>
<tr><td>Your online payment has been accepted.<br><br></td></tr>
<tr><td>Username: %=(String)session.getAttribute("username") %></td></tr>
<tr><td>Company: %=(String)session.getAttribute("company") %></td></tr>
<tr><td>Date: %=(String)session.getAttribute("today") %></td></tr>
<tr><td>ShowTime: May 1-5, 2004 from 9:00 a.m. to 5:00 p.m.</td></tr>
<tr><td>Booth(s): %=(String)session.getAttribute("sbooth") %> booth(s)</td></tr>
<tr><td>Location: 6000 Grade Blvd. San Bernardino CA 92888.</td></tr>
<tr><td>Deposit: $500.00</td></tr>
<tr><td>If you have any questions, please contact us by:</td></tr>
<tr><td>Phone Number: (909)111-4567</td></tr>
<tr><td>Fax: (909)111-5678</td></tr>
<tr><td>E-mail Address: exhibitorexpo@hotmail.com</td></tr>
</table>
<br>
<a href="http://ors.ias.csusb.edu:8080/expo/page/index.jsp">Home Page</a><br>
</body>
</html>
Filename: update1.jsp

```java
<%@page import = "java.util.*"%>
<%@page import = "project.expo.*"%>

<html><head><title>Edit Profile</title></head>
<body background="page/picExpo/12.jpg">
<center>
<font size="6" face="Cataneo BT">Modify Profile</font>
</center>

- Exhibitor user = (Exhibitor)session.getAttribute("user");

<table align = "center">
<tr><td><font size="4" face="Georgia" color="#000080">UserName</font></td>
<td><%=user.getUsername()%></td></tr>
<tr><td><font size="4" face="Georgia" color="#000080">New Password</font></td>
<td><input type="password" name="password" size="15"></td></tr>
<tr><td><font size="4" face="Georgia" color="#000080">New Password again</font></td>
<td><input type="password" name="passwordagain" size="15"></td></tr>
<tr><td><font size="4" face="Georgia" color="#000080">First Name</font></td>
<td><input type="text" name="firstname" size="15" value="<%=user.getFirstName()%>"></td></tr>
<tr><td><font size="4" face="Georgia" color="#000080">Last Name</font></td>
<td><input type="text" name="lastname" size="15" value="<%=user.getLastName()%>"></td></tr>
<tr><td><font size="4" face="Georgia" color="#000080">Company</font></td>
<td><%=user.getCompany()%></td></tr>
<tr><td><font size="4" face="Georgia" color="#000080">Address</font></td>
<td><input type="text" name="address" size="40" value="<%=user.getAddress()%>"></td></tr>
<tr><td><font size="4" face="Georgia" color="#000080">City</font></td>
<td><input type="text" name="city" size="15" value="<%=user.getCity()%>"></td></tr>
<tr><td><font size="4" face="Georgia" color="#000080">State</font></td>
<td><%=user.getState()%></td></tr>
</table>
<form action="https://ors.ias.csusb.edu:8443/expo/update.html" method="get">
<br>
<br>
<br></form>
</body></html>
```
Contact Us

Show Time:

May 1-5, 2004 from 9:00 a.m. to 5:00 p.m.

Exhibit Profile:

Electric Appliances

Question about visitors:

Phone Number: (909)123-4567

Fax: (909)234-5678

E-mail Address: expo1@hotmail.com

Question about exhibitors:

Phone Number: (909)987-4567

Fax: (909)987-5678

E-mail Address: expo2@hotmail.com

Filename: logout.jsp

<%@page import = "java.util.*"%>
<%@page import = "project.expo.*"%>

<html>

</html>
<body background="page/picExpo/12.jpg">
<br><br>% Exhibitor user = (Exhibitor)session.getAttribute("user");%
<br><br><table align = "center">
<tr><p><td><font size="4" face="Georgia" color="#000080">UserName:&nbsp;&nbsp;&nbsp;&nbsp;</font></td><td><%= user.getUsername() %></td></tr>
<tr><p><td>You have logout the server. </td><td></td></tr>
<tr><p><td>You can close this window to leave the server or click &lt;a href="http://ors.ias.csusb.edu:8080/expo/page/index.jsp"&gt;Home page&lt;/a&gt; Back to Expo. </td><td></td></tr>
</table>
<br><br></body>

Filename: vError.jsp
package project.expo;
import java.io.*;
import java.util.*;
import java.text.*;
import java.text.DateFormatSymbols;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.http.HttpSession;

public class Logout extends HttpServlet
{
   public void doGet(HttpServletRequest req, HttpServletResponse res)
      throws ServletException, IOException
   {
      res.setContentType("text/html");
      PrintWriter out = res.getWriter();
      HttpSession session = req.getSession();

      Exhibitor user = (Exhibitor)session.getAttribute("user");
      String username = user.getUsername();

      String alink = "/page/logout.jsp";
      RequestDispatcher dispatcher = req.getRequestDispatcher(alink);
      dispatcher.forward(req, res);
session.removeAttribute("user");
}
}

Filename: vError1.jsp
<%@page import = "java.util.*"%>
<%@page import = "project.expo."%>
<html>
<head><title>visitor information</title></head>
<body background="page/picExpo/12.jpg">
<br><br><center><font size="6" face="Cataneo BT">Forget to fill the information!</font></center>
<br/><br>
<table><tr><td>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&n

Filename: vError2.jsp
<%@page import = "java.util.*"%>
<%@page import = "project.expo."%>
<html>
<head><title>Add visitor - Invalid Card</title></head>
<body background="page/picExpo/12.jpg">
<br><br><center><font size="6" face="Cataneo BT">Sorry</font></center>
<br /><br>
<table><tr><td>You didn't purchase any tickets</td></tr>
<tr><td>Please go back to <a href="https://ors.ias.csusb.edu:8443/expo/page/visitor.jsp"<b>Registration Form</b></a> to make a purchase.</td></tr>
</table>
<br><br>
</body>
</html>
Dear <b><%=(String)session.getAttribute("fname") + " " + (String)session.getAttribute("lname") %></b>

Your credit card has a problem.<br>We can not continue your online payment.<br>Wrong credit card information, please refill the form.<br><a href="https://ors.ias.csusb.edu:8443/expo/page/visitor.jsp">Register for Visitors</a>

---

Filename: eError.jsp

```html
%@page import = "java.util.*"
%@page import = "project.expo.*"

<html>
<head><title>exhibitor information</title></head>

<body background="page/picExpo/12.jpg">
<br><br>
<center>
<font size="6" face="Cataneo BT">Forget to fill the information!</font>
</center>
<br/>
<table>
<tr><td> The username, firstname, lastname, password, company, product, email, credit card information and number of booths are requested.</td></tr>
<tr><td> You forgot to fill these information.</td></tr>
<tr><td> Please go back to <a href="https://ors.ias.csusb.edu:8443/expo/page/exhibitor2.jsp">Registration Form</a> to fill these information.</td></tr>
</table>
</body>
</html>
```
Filename: eError1.jsp

```html
<%@page import = "java.util.*"%>
<%@page import = "project.expo.*"%>

<html>
<head><title>password error</title></head>
<body background="page/picExpo/12.jpg">
<br><br>
<center>
<font size="6" face="Cataneo BT">The passwords don't match.</font>
</center>
<br /><br>
<table>
<tr><td>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&n
```
Filename: eError3.jsp

```jsp
<%@page import = "java.util.*"%>
<%@page import = "project.expo.*"%>

<html>
<head><title>Add exhibitor - Invalid Card</title></head>
<body background="page/picExpo/12.jpg">
  
  <br><br>
  <center>
    <font size="6" face="Cataneo BT">Sorry</font>
  </center>
  <br/><br>
  <table>
    <tr><td>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&n
```

Filename: eError4.jsp

```jsp
<%@page import = "java.util.*"%>
<%@page import = "project.expo.*"%>

<html>
<head><title>exhibitor information</title></head>
<body background="page/picExpo/12.jpg">
  
  <br><br>
  <center>
    <font size="6" face="Cataneo BT">Sorry</font>
  </center>
  
  <table>
    <tr><td>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&n
```
<font size="6" face="Cataneo BT">The username has been used!</font>
<br />
<table>
<tr><td>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&n
Forget fill password.

Go back to <a href="https://ors.ias.csusb.edu:8443/expo/page/login.jsp">Login Page</a>

Filename: loginError2.jsp

Filename: loginError3.jsp
Incorrect password. 

Go back to Login Page

Make the purchase - number of booth can't be zero.
APPENDIX D

JAVA SOURCE FILES
Filename: Database.java
package project.expo;

import java.sql.*;
import javax.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.*;
import javax.naming.*;

public class Database implements ServletContextListener
{
    protected static DataSource ds;

    public Database() { }

    public void contextInitialized(ServletContextEvent event)
    {
        Connection con = null;
        Statement sta = null;
        ResultSet rs = null;
        String query = null;

        try
        {
            InitialContext ic = new InitialContext();
            if(ic == null) throw new Exception("No initial context.");
            Context tomcatContext = (Context) ic.lookup("java:comp/env");
            ds = (DataSource) tomcatContext.lookup("database");
            if(ds == null) throw new Exception("No datasource.");
            con = ds.getConnection();
            sta = con.createStatement();
        }
        catch (Exception e)
        {
            throw new RuntimeException(e);
        }
    }

    public void contextDestroyed(ServletContextEvent event)
    {
    }
}

Filename: Visitor.java
package project.expo;

import java.io.*;
import java.util.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class Visitor extends Database {
    private String error;
    private Connection con = null;

    public Visitor() {
    }

    /* Save the new visitor's information to database */
    public void addVisitors(String firstname, String lastname, String address, String city, String state, String zipcode, String phonenumber, String email, String creditnumber, String expdate, String day, int ticket, long confirmednum) throws SQLException {
        try {
            con = ds.getConnection();

            PreparedStatement updatevs;
            updatevs = con.prepareStatement(insert);
            updatevs.setString(1, firstname);
            updatevs.setString(2, lastname);
            updatevs.setString(3, address);
            updatevs.setString(4, city);
            updatevs.setString(5, state);
            updatevs.setString(6, zipcode);
            updatevs.setString(7, phonenumber);
            updatevs.setString(8, email);
            updatevs.setString(9, creditnumber);
            updatevs.setString(10, expdate);
            updatevs.setString(11, day);
            updatevs.setInt(12, ticket);
            updatevs.setLong(13, confirmednum);
            updatevs.executeUpdate();
            updatevs.close();
            con.close();
        }

        catch (SQLException sqle) {
            error = "SQLException: update failed, possible duplicate entry";
            throw new SQLException(error);
        }
    }
}
Filename: Exhibitor.java

package project.expo;

import java.io.*;
import java.util.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class Exhibitor extends Database {
    private String username;
    private String password;
    private String firstname;
    private String lastname;
    private String company;
    private String product;
    private String address;
    private String city;
    private String state;
    private String zipcode;
    private String phonenumber;
    private String email;
    private String creditnumber;
    private String expdate;
    private int booth;
    private String error;

    /* Save exhibitor's information to database */
public Exhibitor(String username, String password, String firstname, String lastname, String company, String product, String address, String city, String state, String zipcode, String phonenumber, String email, String creditnumber, String expdate, int booth) {
    this.username = username;
    this.password = password;
    this.firstname = firstname;
    this.lastname = lastname;
    this.company = company;
    this.product = product;
    this.address = address;
    this.city = city;
    this.state = state;
    this.zipcode = zipcode;
    this.phonenumber = phonenumber;
    this.email = email;
    this.creditnumber = creditnumber;
    this.expdate = expdate;
    this.booth = booth;
}

public String getUsername() {
    return username;
}

public String getPassword() {
    return password;
}

public String getFirstname() {
    return firstname;
}

public String getLastname() {
    return lastname;
}

public String getCompany() {
    return company;
}

public String getProduct() {
    return product;
}
public String getAddress() {
        return address;
    }

public String getCity() {
        return city;
    }

public String getState() {
        return state;
    }

public String getZipcode() {
        return zipcode;
    }

public String getPhonenumberQ {
        return phonenumber;
    }

public String getEmailQ {
        return email;
    }

public String getCreditnumberQ {
        return creditnumber;
    }

public String getExpdate() {
        return expdate;
    }

public int getBooth() {
        return booth;
    }

/* Save exhibitor's information to database */
public void addExhibitors() throws SQLException {
    try {
        con = ds.getConnection();
        
        String sql = "INSERT INTO Exhibitors (address, city, state, zipcode, phonenumber, email, creditnumber, expdate, booth) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)"; // SQL query to insert exhibitor information into the database
        
        // Execute the SQL query
        // 
        // Handle any exceptions here, such as database errors
    } catch (SQLException e) {
        // Handle the SQLException
    } finally {
        // Close the database connection
        con.close();
    }
}
public void edit(String username, boolean newpass, String password, String firstname, String lastname, String product, String address, String city, String state, String zipcode, String phonenumber, String email, int booth) throws SQLException
{
    try
    {
        con = ds.getConnection();

        /* Update the information with new password */
        if(newpass)
        {
            updatevs = con.prepareStatement(updateWP);
            updatevs.setString(1, password);
            updatevs.setString(2, firstname);
            updatevs.setString(3, lastname);
            updatevs.setString(4, product);
            updatevs.setString(5, address);
            updatevs.setString(6, city);
            updatevs.setString(7, state);
            updatevs.setString(8, zipcode);
            updatevs.setString(9, phonenumber);
            updatevs.setString(10, email);
            updatevs.setString(11, username);
            updatevs.setInt(12, booth);
            updatevs.executeUpdate();
        }
        updatevs.close();
        con.close();
    }
    catch (SQLException sqle)
    {
        error = "SQLException: insert failed, possible duplicate entry";
        throw new SQLException(error);
    }
}
updatevs = con.prepareStatement(insert);
updatevs.setString(1, username);
updatevs.setString(2, password);
updatevs.setString(3, firstname);
updatevs.setString(4, lastname);
updatevs.setString(5, company);
updatevs.setString(6, product);
updatevs.setString(7, address);
updatevs.setString(8, city);
updatevs.setString(9, state);
updatevs.setString(10, zipcode);
updatevs.setString(11, phonenumber);
updatevs.setString(12, email);
updatevs.setString(13, creditnumber);
updatevs.setString(14, expdate);
updatevs.setInt(15, booth);
updatevs.executeUpdate();
updatevs.close();
con.close();
}
else
{
    /* Update the information without new password */
    updatevs = con.prepareStatement(updateNP);
    updatevs.setString(1, firstname);
    updatevs.setString(2, lastname);
    updatevs.setString(3, product);
    updatevs.setString(4, address);
    updatevs.setString(5, city);
    updatevs.setString(6, state);
    updatevs.setString(7, zipcode);
    updatevs.setString(8, phonenumber);
    updatevs.setString(9, email);
    updatevs.setInt(10, booth);
    updatevs.setString(11, username);
    updatevs.executeUpdate();
}
updatevs.close();
con.close();
}

try
{
    con = ds.getConnectionQ;
    searchEx = con.prepareStatement(search);
    searchEx.setString(1, username);
    rs = searchEx.executeQuery();
}
catch (SQLException sqle)
{
    error = "SQLException: Could not execute the Query.";
    throw new SQLException(error);
}

finally
{
    if(rs != null) rs.close();
    if(searchEx != null) searchEx.close();
    if(con != null) con.close();
}*/
return rs;
public void killCon() throws SQLException
{
  try
  {
    if(rs != null)
    {
      rs.close();
    }
    if(con != null)
    {
      con.close();
    }
  }
  catch(SQLException sql)
  {
    error = "SQLException: Could not disconnect."
    throw new SQLException(error);
  }
}

Filename: ProcessV.java
package project.expo;
import java.io.*;
import java.lang.*;
import java.util.*;
import java.text.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.http.HttpSession;
import java.net.*;
import javax.net.ssl.*;
import java.security.*;

/* ProcessV class handles the data of visitor
charge the fee on the visitor’s credit card
save the visitor’s information to database */
public class ProcessV extends HttpServlet
{
  private static String defaultServer =
  "https://testefsnet.concordebiz.com/efsnet.dll?Method=CreditCardAuthorize&StoreID=coyote01&StoreKey=A04B300D4C279E0B0730FC86759E0B030F86750006B2821D4876388CABC4C5ED&ApplicationID=EFSnet+samples+1%2e0";

  public void doGet(HttpServletRequest req, HttpServletResponse res)
  throws ServletException, IOException
  {
    res.setContentType("text/html");
PrintWriter out = res.getWriter();
HttpSession session = req.getSession();

String fname = req.getParameter("firstname");
String lname = req.getParameter("lastname");
String email = req.getParameter("email");
String credit = req.getParameter("creditnumber");
String expMonth = req.getParameter("expMonth");
String expYear = req.getParameter("expYear");
String sticket = req.getParameter("ticket");
String address = req.getParameter("address");
String city = req.getParameter("city");
String state = req.getParameter("state");
String zip = req.getParameter("zipcode");
String phone = req.getParameter("phonenumber");
String day = req.getParameter("day");

session.setAttribute("fname", fname);
session.setAttribute("lname", lname);
session.setAttribute("email", email);
session.setAttribute("credit", credit);
session.setAttribute("expMonth", expMonth);
session.setAttribute("expYear", expYear);
session.setAttribute("sticket", sticket);
session.setAttribute("address", address);
session.setAttribute("city", city);
session.setAttribute("state", state);
session.setAttribute("zip", zip);
session.setAttribute("phone", phone);
session.setAttribute("day", day);

// check if the visitor forgets to fill requested information
if (fname.equals("") || lname.equals("") || email.equals("") || credit.equals("") ||
    expMonth.equals("") || expYear.equals("") || sticket.equals("")
)
{
    String alink = "/page/vError.jsp";
    RequestDispatcher dispatcher = req.getRequestDispatcher(alink);
    dispatcher.forward(req, res);
}

else {
    int ticket = Integer.parseInt(sticket);
    if (expMonth.length() == 1)
        expMonth = "0" + expMonth;
    if (expYear.length() >= 2)
        expYear = expYear.substring(expYear.length() - 2);
    
    // make sure the visitor buys the tickets
    if (ticket <= 0)
    {
        String alink = "/page/vError1.jsp";
        RequestDispatcher dispatcher = req.getRequestDispatcher(alink);
        dispatcher.forward(req, res);
    }
else {
    String expdate = expMonth + "/" + expYear;
    long confirmednum = 0;

    java.util.Date Today = new java.util.Date();
    int a = DateFormat.SHORT;
    DateFormat df = DateFormat.getDateInstance(a);
    String today = df.format(Today);
    session.setAttribute("today", today);

    // disable the server side authentication
    TrustManager[] trustAllCerts = new TrustManager[] {
        new X509TrustManager() {
            public java.security.cert.X509Certificate[] getAcceptedIssuers() { return null; }
            public void checkClientTrusted( java.security.cert.X509Certificate[] certs, String authType) {
            }
            public void checkServerTrusted( java.security.cert.X509Certificate[] certs, String autoType) {
            }
        }
    };

    try {
        // connect to payment gateway server
        String server = defaultServer + "&ReferenceNumber=0123&TransactionAmount=" + ticket*10.00 + "&AccountNumber=" + credit + "&ExpirationMonth=" + expMonth + "&ExpirationYear=" + expYear;

        SSLSocketFactory sf = null;
        SSLContext ctx = SSLContext.getInstance("TLS");
        ctx.init(null, trustAllCerts, new java.security.SecureRandom());
        sf = ctx.getSocketFactory();
        URL u = new URL(server);
        URLConnection uc = u.openConnection();
        HttpsURLConnection connection = (HttpsURLConnection) uc;
        connection.setSSLSocketFactory(sf);
        connection.setDoOutput(true);
        connection.setDoInput(true);
        connection.setRequestMethod("GET");

        InputStreamReader in = new InputStreamReader(connection.getInputStream());
        BufferedReader ind = new BufferedReader(in);
        String c;
        c = ind.readLine();
    }
}}
StringTokenizer strz = new StringTokenizer(c);
String t = strz.nextToken("&");
t = strz.nextToken("&");
t = strz.nextToken("&");
StringTokenizer result = new StringTokenizer(t);
String Result = result.nextToken("=");
Result = result.nextToken();

String s = strz.nextToken("&");
s = strz.nextToken("&");
s = strz.nextToken("&");
s = strz.nextToken("&");
s = strz.nextToken("&");
s = strz.nextToken("&");
s = strz.nextToken();
StringTokenizer amount = new StringTokenizer(s);
String Amount = amount.nextToken("=");
Amount = amount.nextToken();
ind.close();
in.close();
connection.disconnect();

// the fee is charged successfully
if(Result.equals("APPROVED"))
{
    try
    {
        Visitor vi = new Visitor();
        Random r = new Random();
        confirmednum = Math.abs(r.nextLong());

        session.setAttribute("confirmednum", Long.toString(confirmednum));
        vi.addVisitors(fname, lname, address, city, state, zip, phone, email, credit, expdate, day, ticket, confirmednum);
    }
    catch(SQLException ignored)
    {
    }
}
String alink = "/page/receiptV.jsp";
RequestDispatcher dispatcher = req.getRequestDispatcher(alink);
dispatcher.forward(req, res);

session.removeAttribute("fname");
session.removeAttribute("lname");
session.removeAttribute("email");
session.removeAttribute("credit");
session.removeAttribute("expMonth");
session.removeAttribute("expYear");
session.removeAttribute("sticket");
session.removeAttribute("address");
session.removeAttribute("city");
session.removeAttribute("state");
session.removeAttribute("zip");
session.removeAttribute("phone");
session.removeAttribute("day");
session.removeAttribute("confirmednum");
session.removeAttribute("today");
}
else {
    // the credit card information is not valid
    String alink = "/page/vError2.jsp";
    RequestDispatcher dispatcher = req.getRequestDispatcher(alink);
    dispatcher.forward(req, res);
}
}

} catch (Exception e) {
    System.err.println(e);
    e.printStackTrace();
}
}
}

Filename: ProcessE.java
	package project.expo;

import java.io.*;
import java.util.*;
import java.text.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.net.*;
import javax.net.ssl.*;
import java.security.*;

/* ProcessE class handles the data of exhibitor
charge the deposit on the exhibitor's credit card
save the exhibitor's information to database */
public class ProcessE extends HttpServlet {

    private static String defaultServer =
        "https://testefsn.net.concordebiz.com/efsnet.dll?Method=CreditCardAuthorize&StoreID=coyote01&StoreKey=A04B300D4C279E0B0730FC86759E0B030F8675006B2821D4876388CABC4C5E0&ApplicationID=EFSnet+samples+1%2e0";

    public void doGet(HttpServletRequest req, HttpServletResponse res)
throws ServletException, IOException
{
    res.setContentType("text/html");
    PrintWriter out = res.getWriter();
    HttpSession session = req.getSession();
    ResultSet rs = null;
    int booth = 0;

    String password = req.getParameter("password");
    String passwordagain = req.getParameter("passwordagain");
    String username = req.getParameter("username");
    String fname = req.getParameter("firstname");
    String lname = req.getParameter("lastname");
    String company = req.getParameter("company");
    String product = req.getParameter("product");
    String email = req.getParameter("email");
    String credit = req.getParameter("creditnumber");
    String expMonth = req.getParameter("expMonth");
    String expYear = req.getParameter("expYear");
    String sbooth = req.getParameter("booth");
    String address = req.getParameter("address");
    String city = req.getParameter("city");
    String state = req.getParameter("state");
    String zip = req.getParameter("zipcode");
    String phone = req.getParameter("phonenumber");

    /* Exhibitor user = new Exhibitor(username,password,
    fname,lname,company,product,address,city,state,zip,phone,email,credit,expdate, booth);
    session.setAttribute("user",user); */
    session.setAttribute("username", username);
    session.setAttribute("fname", fname);
    session.setAttribute("lname", lname);
    session.setAttribute("company", company);
    session.setAttribute("product", product);
    session.setAttribute("email", email);
    session.setAttribute("credit", credit);
    session.setAttribute("expMonth", expMonth);
    session.setAttribute("expYear", expYear);
    session.setAttribute("booth", sbooth);
    session.setAttribute("address", address);
    session.setAttribute("city", city);
    session.setAttribute("state", state);
    session.setAttribute("zip", zip);
    session.setAttribute("phone", phone);

    // check if the exhibitor forgets to fill requested information
    if(password.equals("") || passwordagain.equals("") || username.equals("") || fname.equals(""")
        || lname.equals("") || company.equals("") || product.equals("") ||
        credit.equals("") || email.equals("") ||
        expMonth.equals("") || expYear.equals("") || sbooth.equals(""))
    {
        String alink = "\page/eError.jsp";
        RequestDispatcher dispatcher = req.getRequestDispatcher(alink);
        dispatcher.forward(req, res);
make sure the exhibitor’s passwords match
else if(!password.equals(passwordagain))
{
    String alink = "/page/eError1.jsp";
    RequestDispatcher dispatcher = req.getRequestDispatcher(alink);
    dispatcher.forward(req, res);
}
else
{
    if (expMonth.length() == 1)
        expMonth = "0" + expMonth;
    if (expYear.length() >= 2)
        expYear = expYear.substring(expYear.length() - 2);
    String expdate = expMonth + "/" + expYear;
    // if (sbooth != null)
    booth = Integer.parseInt(sbooth);
    try {
        Exhibitor exh = new Exhibitor();
        rs = exh.searchE(username); // make sure the username has already existed in database
        if(rs.next())
        {
            String alink = "/page/eError4.jsp";
            RequestDispatcher dispatcher = req.getRequestDispatcher(alink);
            dispatcher.forward(req, res);
        }
        // make sure the exhibitor orders the booths
        else if (booth <= 0)
        {
            String alink = "/page/eError2.jsp";
            RequestDispatcher dispatcher = req.getRequestDispatcher(alink);
            dispatcher.forward(req, res);
        }
        else {
            java.util.Date Today = new java.util.Date();
            int b = DateFormat.SHORT;
            DateFormat df = DateFormat.getDateInstance(b);
            String today = df.format(Today);
            session.setAttribute("today", today);
            // disable the sever side authentication
            TrustManager[] trustAllCerts = new TrustManager[]
            {
                new X509TrustManager()
                {
                    public java.security.cert.X509Certificate[]
                    getAcceptedIssuers() { return null; }
                }
            }
        }
    }
}
public void checkClientTrusted(
    java.security.cert.X509Certificate[] certs,
    String authType) {}

public void checkServerTrusted(
    java.security.cert.X509Certificate[] certs,
    String authType) {}

try {
    // connect to payment gateway server
    SSLSocketFactory sf = null;
    SSLContext ctx = SSLContext.getInstance("TLS");
    ctx.init(null, trustAllCerts, new java.security.SecureRandom());
    sf = ctx.getSocketFactory();
    String server = defaultServer +
    "&ReferenceNumber=1123&TransactionAmount=500.00" + credit +
    "&ExpirationMonth=" + expMonth + "&ExpirationYear=" + expYear;
    URL u = new URL(server);
    URLConnection uc = u.openConnection();
    HttpsURLConnection connection = (HttpsURLConnection) uc;
    connection.setSSLSocketFactory(sf);
    connection.setDoOutput(true);
    connection.setDoInput(true);
    connection.setRequestMethod("GET");

    InputStreamReader in = new InputStreamReader(connection.getInputStream());
    BufferedReader ind = new BufferedReader(in);
    String c;
    c = ind.readLine();
    StringTokenizer strz = new StringTokenizer(c);
    String t = strz.nextToken("&");
    t = strz.nextToken("&");
    t = strz.nextToken("&");
    String result = result.nextToken("=");
    Result = result.nextToken();
    String s = strz.nextToken("&");
    s = strz.nextToken("&");
    s = strz.nextToken("&");
    s = strz.nextToken("&");
    s = strz.nextToken("&");
    s = strz.nextToken();
    StringTokenizer amount = new StringTokenizer(s);
String Amount = amount.nextToken("=");
Amount = amount.nextToken();
ind.close();
in.close();
connection.disconnect();

// the deposit is charged successfully
if(Result.equals("APPROVED"))
{
    try
    {
        /* Exhibitor ex = new Exhibitor();
        ex.addExhibitors(username, password, fname, lname, company, product, address,
city, state, zip, phone, email, credit, expdate, booth);*/
        Exhibitor ex = new Exhibitor(username, password, fname, lname, company, product, address, city, state, zip, phone, email, credit, expdate, booth);
        ex.addExhibitors();
    }
    catch(SQLException ignored)
    {
    }
    String alink = "/page/receiptE.jsp";
    RequestDispatcher dispatcher = req.getRequestDispatcher(alink);
    dispatcher.forward(req, res);
}
else
{
    // the credit card information is not valid
    String alink = "/page/eError3.jsp";
    RequestDispatcher dispatcher = req.getRequestDispatcher(alink);
    dispatcher.forward(req, res);
}
catch (Exception e)
{
    System.err.println(e);
    e.printStackTrace();
}
exh.killCon();
catch(SQLException sql)
{
    //error = "SQLException:Could not execute the query.";
    //throw new SQLException(error);
}

Filename: Editfile.java
	package project.expo;

import java.io.*;
import java.util.*;
import java.text.*;
import java.text.DateFormatSymbols;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.http.HttpSession;

public class Editfile extends HttpServlet
{
    public void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException
    {
        res.setContentType("text/html");
        PrintWriter out = res.getWriter();
        String error;
        String firstname = "";
        String lastname = "";
        String company = "";
        String product = "";
        String address = "";
        String city = "";
        String state = "";
        String zipcode = "";
        String phonenumber = "";
        String email = "";
        String creditnumber = "";
        String expdate = "";
        int booth;
    }
String pass = null;
ResultSet rs = null;
Exhibitor ex = new Exhibitor();
HttpSession session = req.getSession();

String username = req.getParameter("username");
String password = req.getParameter("password");

if(username.equals(""))
{
    String alink = "/page/loginError.jsp";
    RequestDispatcher dispatcher = req.getRequestDispatcher(alink);
    dispatcher.forward(req, res);
}
else if(password.equals(""))
{
    String alink = "/page/loginError1.jsp";
    RequestDispatcher dispatcher = req.getRequestDispatcher(alink);
    dispatcher.forward(req, res);
}
else
{
    try
    {
        rs = ex.searchE(username);
    }
    catch(SQLException sql)
    {
        //error = "SQLException:Could not execute the query.";
        //throw new SQLException(error);
    }
    try
    {
        if(rs.next())
        {
            String alink = "/page/loginError2.jsp";
            RequestDispatcher dispatcher = req.getRequestDispatcher(alink);
            dispatcher.forward(req, res);
            ex.killCon();
        }
    }
    catch(SQLException sqle)
    {
        //error = "SQLException:Could not get the password from database.";
        throw new SQLException(error);
    }
}
if(!password.equals(pass))
{
    String alink = "/page/loginError3.jsp";
    RequestDispatcher dispatcher = req.getRequestDispatcher(alink);
    dispatcher.forward(req, res);
}
else
{
    firstname = rs.getString("firstname");
    lastname = rs.getString("lastname");
    company = rs.getString("company");
    product = rs.getString("product");
    address = rs.getString("address");
    city = rs.getString("city");
    state = rs.getString("state");
    zipcode = rs.getString("zipcode");
    phonenum = rs.getString("phonenumber");
    email = rs.getString("email");
    creditnumber = rs.getString("creditnumber");
    expdate = rs.getString("expdate");
    booth = rs.getInt("booth");
    Exhibitor user = new Exhibitor(username, pass, firstname, lastname, company, product, address, city, state, zipcode, phonenum, email, creditnumber, expdate, booth);
    session.setAttribute("user", user);
    String alink = "/page/edit.jsp";
    RequestDispatcher dispatcher = req.getRequestDispatcher(alink);
    dispatcher.forward(req, res);
}
ex.killCon();
}
catch(SQLException sql)
{
    //error = "SQLException:Could not execute the query.";
    //throw new SQLException(error);
}
}
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.http.HttpSession;

public class Update extends HttpServlet {
    public void doGet(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
        res.setContentType("text/html");
        PrintWriter out = res.getWriter();
        boolean newpass = true;
        String error = "";
        ResultSet rs = null;
        Exhibitor ex = new Exhibitor();
        Exhibitor user = new Exhibitor();
        HttpSession session = req.getSession();

        user = (Exhibitor)session.getAttribute("user");
        String username = user.getUsername();
        String company = user.getCompany();
        String creditnumber = user.getCreditnumber();
        String expdate = user.getExpdate();

        String password = req.getParameter("password");
        String passwordagain = req.getParameter("passwordagain");
        String fname = req.getParameter("firstname");
        String lname = req.getParameter("lastname");
        String product = req.getParameter("product");
        String address = req.getParameter("address");
        String city = req.getParameter("city");
        String state = req.getParameter("state");
        String zipcode = req.getParameter("zipcode");
        String phone = req.getParameter("phonenumber");
        String email = req.getParameter("email");
        int booth = Integer.parseInt(req.getParameter("booth"));

        if (!password.equals(passwordagain))
            {  
                String alink = "/page/updateError1.jsp";
                RequestDispatcher dispatcher = req.getRequestDispatcher(alink);
                dispatcher.forward(req, res);
            }

        else if (booth <= 0)
            {  
                String alink = "/page/updateError2.jsp";
                RequestDispatcher dispatcher = req.getRequestDispatcher(alink);
                dispatcher.forward(req, res);
            }

        else
{ if(password.equals(""))
{ newpass = false;
 password = user.getPassword();
}

try
{ ex.edit(username, newpass, password, fname, lname, product, address, city, state, zipcode, phone, email, booth);

 user = new Exhibitor(username,password,
fname,lname,company,product,address,city,state,zipcode,phone,email,creditnumber,expdate,
booth);
 session.setAttribute("user", user);
 String alink = "/page/update1.jsp";
 RequestDispatcher dispatcher = req.getRequestDispatcher(alink);
 dispatcher.forward(req, res);
} catch(SQLException sql)
{ }
}

Filename: Logout.java
package project.expo;
import java.io.*;
import java.util.*;
import java.text.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.http.HttpSession;

public class Logout extends HttpServlet
{
 public void doGet(HttpServletRequest req, HttpServletResponse res)
 throws ServletException, IOException
{
 res.setContentType("text/html");
 PrintWriter out = res.getWriter();
 HttpSession session = req.getSession();

 Exhibitor user = (Exhibitor)session.getAttribute("user");
 String username = user.getUsername();
String alink = "/page/logout.jsp";
RequestDispatcher dispatcher = req.getRequestDispatcher(alink);
dispatcher.forward(req, res);

session.removeAttribute("user");

Filename: AverageValueObject.java
public class AverageValueObject
{
    private int averageResponseTime;
    private int averageThroughput;

    public AverageValueObject(int averageResponseTime, int averageThroughput)
    {
        this.averageResponseTime = averageResponseTime;
        this.averageThroughput = averageThroughput;
    }

    public int getAverageResponseTime() { return averageResponseTime; }
    public int getAverageThroughput() { return averageThroughput; }
}

Filename: ResponseObject.java
// ResponseObject class holds the responseTime object
public class ResponseObject
{
    private int responseTime;

    public ResponseObject(int responseTime)
    {
        this.responseTime = responseTime;
    }

    public int getResponseTime() { return responseTime; }
}

Filename: SampleCreator.java
import java.net.*;
import javax.net.*;
import javax.net.ssl.*;
import java.security.*;
import java.io.*;
import java.util.*;

/* SampleCreator class produces the threads */
public class SampleCreator extends Thread
{
    // number of pages per minute
    private int pagesPerMin;
    // total running pages
    private int totalPages;
    private String link;
    // total running minutes
    private int mins;
    private int lastResponse = -1;
    private GregorianCalendar calendar;

    // disable the server side authentication
    private static TrustManager[] trustAllCerts = new TrustManager[]
    {
        new X509TrustManager() {
            public java.security.cert.X509Certificate[] getAcceptedIssuers() { return null; }

            public void checkClientTrusted(java.security.cert.X509Certificate[] certs, String authType) {}

            public void checkServerTrusted(java.security.cert.X509Certificate[] certs, String authType) {}
        }
    };

    public SampleCreator(String link, int pagesPerMin, int mins)
    {
        this.pagesPerMin = pagesPerMin;
        this.mins = mins;
        this.link = link;
        this.totalPages = pagesPerMin * mins;
    }

    public int getResponseTime()
    {
        return lastResponse;
    }

    public void setResponseTime()
    {
        lastResponse = -1;
    }

    // produce the threads that the testing procedure needs
    public void run()
    {

int delayTime = (60 * 1000) / pagesPerMin;
while(true)
{
    try
    {
        totalPages--;
        (new RequestCreator()).start();
    }
    catch (Exception e) {
        e.printStackTrace();
    }
    if (totalPages == 0) break;
    try
    {
        Thread.sleep(delayTime);
    }
    catch (Exception e) {
        e.printStackTrace();
    }
}

/* RequestCreator class connects to server and counts the
response time */
private class RequestCreator extends Thread
{
    private int responseTime;

    public void run()
    {
        // connect to http protocol
        if (!link.substring(0, 5).equals("https"))
        {
            calendar = new GregorianCalendar();
            long startTime = calendar.getTimeInMillis();
            try
            {
                URL u = new URL(link);
                URLConnection uc = u.openConnection();
                HttpURLConnection connection = (HttpURLConnection) uc;
                connection.setRequestMethod("GET");
                connection.getContent();
                calendar = new GregorianCalendar();
                responseTime = (int) (calendar.getTimeInMillis() - startTime);
                connection.disconnect();
            }
        }
        catch (Exception e) {
            responseTime = -999;
        }
    }
}
else
{
    // connect to https protocol
    try
    {
        SSLSocketFactory sf = null;
        SSLContext ctx = SSLContext.getInstance("TLS");
        ctx.init(null, trustAllCerts,
                   new java.security.SecureRandom());
        sf = ctx.getSocketFactory();
        calendar = new GregorianCalendar();
        long startTime = calendar.getTimeInMillis();
        URL u = new URL(link);

        URLConnection uc = u.openConnection();
        HttpsURLConnection connection = (HttpsURLConnection) uc;
        connection.setSSLSocketFactory(sf);
        connection.setRequestMethod("GET");
        connection.getContent();

        calendar = new GregorianCalendar();
        responseTime = (int) (calendar.getTimeInMillis() - startTime);
        connection.disconnect();
    }
    catch (Exception e) {
        responseTime = -999;
    }
}

while (true)
{
    // make sure no another thread changed the lastResponse
    if (lastResponse == -1) {
        lastResponse = responseTime;
        break;
    }
    try {
        /* if another thread changed the lastReponse then wait
         5 ms to do it again */
        Thread.sleep(5);
    }
    catch (Exception e) {
        e.printStackTrace();
    }
}
Filename: NormalPage.java

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

// Testing class: NormalPage class
public class NormalPage
{
    private int counter = 1;
    protected int pagesPerMin;
    protected String link;
    protected int mins;
    private int responseTime;
    protected SampleCreator sample;

    public NormalPage(String link, int pagesPerMin, int mins)
    {
        this.link = link;
        this.pagesPerMin = pagesPerMin;
        this.mins = mins;
    }

    public int getResponseTime()
    {
        return sample.getResponseTimeQ;
    }

    public void setResponseTime()
    {
        sample.setResponseTimeQ;
    }

    // initial the NormalPage
    // call the SampleCreator to produce the threads to connect server
    public void initQ()
    {
        sample = new SampleCreator(link, pagesPerMin, mins);
        sample.startQ;
    }
}

Filename: LoginPage.java

import java.net.*;
import javax.net.ssl.*;
import java.security.*;
import java.io.*;
import java.util.*;

// Testing class: LoginPage class
public class LoginPage extends NormalPage
{
    private String link;
    public LoginPage(String userName, String password, int pagesPerMin, int mins)
    {
        super("https://ors.ias.csusb.edu:8443/expo/editfile.html" + "?username=" + userName + "+password=" + password, pagesPerMin, mins);
    }
}

Filename: VRegPage.java

import java.net.*;
import javax.net.ssl.*;
import java.security.*;
import java.io.*;
import java.util.*;

// Testing class: Visitor Registration Page class
public class VRegPage extends Thread
{
    String link = "";
    String vlink = "";

    int pagesPerMin = 0;
    int mins = 0;
    int responseTime = -1;
    Vector result;
    SampleCreator sample;

    public VRegPage(String link, int pagesPerMin, int mins)
    {
        this.link = link;
        this.pagesPerMin = pagesPerMin;
        this.mins = mins;
        result = new Vector();
    }

    public void run()
    {
        /* count how many times should the SampleCreator connect to server */
        int totalPages = pagesPerMin*mins;
        int delay = (60*1000) / pagesPerMin;
        String fname = "fname";
        String lname = "lname";
    }
}
String address = "address";
String city = "city";
String state = "CA";
String zip = "99999";
String phone = "9091234567";
String email = "";
String creditnumber = "4111111111111111";
String expMonth = "09";
String expYear = "08";
String day = "05%2F02%2F04";
String ticket = "1";

for (int i = 0; i < totalPages; i++)
{
    // email is primary key, should be unique
    // create the long-random to avoid the same email
    email = "tester@expo.com" + Math.abs((new Random().nextLong()));

    vlink = link + "?firstname=\" + fname +
            "&lastname=\" + lname +
            "&address=\" + address +
            "&city=\" + city +
            "&state=\" + state +
            "&zipcode=\" + zip +
            "&phonenumber=\" + phone +
            "&email=\" + email +
          "&creditnumber=4111111111111111&expMonth=09&expYear=08&day=05%2F02%2F04&ticket=\" + ticket;

    // the vlink can only use once
    sample = new SampleCreator(vlink, 1, 1);
    // doesn't know which SampleCreator got the response first
    // so push SampleCreator into a vector
    result.addElement(sample);
    sample.start();
    vlink = "";
    try
    {
        Thread.sleep(delay);
    }
    catch (Exception e) {
        e.printStackTrace();
    }
}

public int getResponseTime()
{
    for (int i = 0; i < result.size(); i++)
    {
        // when the SampleCreator got the response time remove the
        // SampleCreator from vector and return the response time
        // if there is no SampleCreator in the vector return -1
    }
}

126
if (!((responseTime = ((SampleCreator) result.get(i)).getResponseTime()) == -1))
{
    result.remove(i);
    return responseTime;
}
return -1;

Filename: ERegPage.java
import java.net.*;
import javax.net.ssl.*;
import java.security.*;
import java.io.*;
import java.util.*;

// Testing class: Exhibitor Registration Page class
public class ERegPage extends Thread
{
    String link = "";
    String vlink = "";
    int pagesPerMin = 0;
    int mins = 0;
    int responseTime = -1;
    Vector result;
    SampleCreator sample;

    public ERegPage(String link, int pagesPerMin, int mins)
    {
        this.link = link;
        this.pagesPerMin = pagesPerMin;
        this.mins = mins;
        result = new Vector();
    }

    public void run()
    {
        /* count how many times should the SampleCreator connect to server */
        int totalPages = pagesPerMin * mins;
        int delay = (60*1000) / pagesPerMin;
        String username = "";
        String password = "1111";
        String passwordagain = "1111";
        String fname = "fname";
        String lname = "lname";
        String company = "company";
        String product = "product";
String address = "address";
String city = "city";
String state = "CA";
String zip = "99999";
String phone = "9091234567";
String email = "exhibitor@expo.com";
// String creditnumber = "4111111111111111";
// String expMonth = "09";
// String expYear = "08";
String booth = "1";

for (int i = 0; i < totalPages; i++)
{
    // email is primary key, should be unique
    // create the long-random to avoid the same email
    username = "username" + Math.abs((new Random().nextLong()));

    vlink = link + "?username=" + username +
            "&password=" + password +
            "&passwordagain=" + passwordagain +
            "&firstname=" + fname +
            "&lastname=" + lname +
            "&company=" + company +
            "&product=" + product +
            "&address=" + address +
            "&city=" + city +
            "&state=" + state +
            "&zipcode=" + zip +
            "&phonenumber=" + phone +
            "&email=" + email +
            "&creditnumber=4111111111111111&expMonth=09&expYear=08&booth=" +
    booth;
    // the vlink can only use once
    sample = new SampleCreator(vlink, 1, 1);
    // doesn't know which SampleCreator got the response first
    // so push SampleCreator into a vector
    result.addElement(sample);
    sample.start();
    vlink = "";
    try
    {
        Thread.sleep(delay);
    }
    catch (Exception e) {
        e.printStackTrace();
    }
}

public int getResponseTime()
{
    for (int i = 0; i < result.size(); i++)
    {
when the SampleCreator got the response time remove the 
SampleCreator from vector and return the response time 
if there is no SampleCreator in the vector return -1 
if ( !(responseTime = ((SampleCreator) result.get(i)).getResponseTime()) == -1)) 
{ 
result.remove(i); 
return responseTime; 
}
return -1;
}

Filename: DiagramDraw.java

import java.awt.*;
import javax.swing.*;
import java.util.*;

class DiagramDraw extends JPanel
{
    private Vector data = new Vector();
    private Vector averageData = new Vector();
    private int worstResponse;
    private int highestThroughput;
    private int averageResponseTime = 0;
    private int averageThroughput = 0;
    private int yCoordinateUnit;
    private int yCoordinateUnit1;
    private ResponseObject ro;
    private ResponseObject ro1;
    private AverageValueObject ao;

    public void paintComponent(Graphics g)
    {
        g.setColor(Color.BLACK);
        int failures = 0;
        int successCase = 0;
        int responseSum = 0;
        int location1 = 66;
        int location2 = 66;
        int location3 = 66;
        int height = 400;
        worstResponse = 1000;
        highestThroughput = 800;
        int responseTime;
        int throughput;
        super.paintComponent(g);
        g.drawRect(65, 10, 500, 250);
//Draw Y-Coordinates
g.setFont(new Font("Serif", Font.BOLD, 14));
g.drawString("ms", 45, 15);
g.drawString("ms", 45, 260);
g.setFont(new Font("Serif", Font.PLAIN, 14));
g.drawString("0", 25, 260);

//Check responseTime data from a vector
for (int i = 0; i < data.size(); i++)
{
    ro = (ResponseObject) data.get(i);
    if ((responseTime = ro.getResponseTime()) != -999)
    {
        if ((responseTime = ro.getResponseTime()) > worstResponse)
        {
            worstResponse = ro.getResponseTime();
        }
        successCase++;
        responseSum += responseTime;
    }
    else
    {
        failures++;
    }
}

// Set the Y-Coordinate unit of responseTime and averageResponseTime
worstResponse = (int) (worstResponse * 1.2);
g.drawString(Integer.toString(worstResponse), 5, 15);
yCoordinateUnit = (int) (worstResponse / 250);

// Calculate the averageResponseTime and Throughput
if (data.size() > 0 && successCase > 0)
{
    averageResponseTime = responseSum / successCase;
    averageThroughput = 60000 / averageResponseTime;
    averageData.addElement(new AverageValueObject(averageResponseTime, averageThroughput));
    if (averageThroughput > highestThroughput)
    {
        highestThroughput = averageThroughput;
    }
    yCoordinateUnit1 = (int) highestThroughput / 250;
}

// Display number of sample
g.setFont(new Font("Serif", Font.BOLD, 14));
g.drawString("No. of Sample: ", 95, 280);
g.setFont(new Font("Serif", Font.PLAIN, 14));
g.drawString(Integer.toString(data.size()), 195, 280);
g.setFont(new Font("Serif", Font.BOLD, 14));
g.drawString("samples", 235, 280);
// Display last sample response time
setColor(new Color(33, 181, 56));
setFont(new Font("Serif", Font.BOLD, 14));
drawString("Last Sample : ", 95, 300);
setFont(new Font("Serif", Font.PLAIN, 14));

drawString("Last Sample: ", 95, 300);

debug
if (data.size() > 0) {
    ro = (ResponseObject) data.get(data.size()-1);
    if (ro.getResponseTime() != -999)
    {
        drawString(Integer.toString(ro.getResponseTime()), 195, 300);
    }
    else
    {
        drawString("0", 195, 300);
    }
}
setFont(new Font("Serif", Font.BOLD, 14));
drawString("ms", 235, 300);

debug
// Display average response time
setColor(Color.BLUE);
setFont(new Font("Serif", Font.BOLD, 14));
drawString("Average: ", 335, 280);
setFont(new Font("Serif", Font.PLAIN, 14));
drawString(Integer.toString(averageResponseTime), 425, 280);
setFont(new Font("Serif", Font.BOLD, 14));
drawString("ms", 470, 280);

debug
// Display average throughput
setColor(Color.RED);
setFont(new Font("Serif", Font.BOLD, 14));
drawString("Throughput: ", 335, 300);
setFont(new Font("Serif", Font.PLAIN, 14));
drawString(Integer.toString(averageThroughput), 425, 300);
setFont(new Font("Serif", Font.BOLD, 14));
drawString("pages/min", 470, 300);

for (int i = 0; i < data.size(); i++)
{
    ro = (ResponseObject) data.get(i);

    // Draw Response Time Sample
    if (ro.getResponseTime() != -999)
    {
        setColor(new Color(33, 181, 56));
        fillOval(location1, (int) (260-ro.getResponseTime()/yCoordinateUnit), 2, 2);

        location1++;
    }
}

// Draw Failure Rate
g.setColor(Color.ORANGE);
g.fillOval(location2, (int) (260 - (failures / data.size()) * 100), 2, 2);

location2++;

for (int i = 0; i < averageData.size(); i++)
{
   ao = (AverageValueObject) averageData.get(i);

   //Draw Average Response Time
   g.setColor(Color.BLUE);
g.fillOval(location3, (int) (260 - ao.getAverageResponseTime() / yCoordinateUnit), 2, 2);

   //Draw Average Throughput
   g.setColor(Color.RED);
g.fillOval(location3, (int) (260 - ao.getAverageThroughput() / yCoordinateUnit1), 2, 2);

   location3++;
}

public void draw(ResponseObject object)
{
   data.addElement(object);
   repaint();
}

Filename: Interface.java

import java.awt.*;
import javax.swing.*;

public class Interface extends JFrame
{
   private DiagramDraw diagram;

   public Interface()
   {
      super("This is the test Interface");
diagram = new DiagramDraw();
diagram.setBackground(Color.WHITE);

      Container container = getContentPane();
      container.add(diagram, BorderLayout.CENTER);
      setSize(600, 350);
      setVisible(true);
}
public void drawGraphic(ResponseObject ro) {
    diagram.draw(ro);
}

Filename: Tester.java

import java.awt.*;
import javax.swing.*;
import java.net.*;
import java.io.*;
import java.util.*;

public class Tester {
    /* Total test time (min) */
    private int totalTime;

    /* Number of requests per min of normal Page */
    private int nPagesPerMin;

    /* Number of requests per min of login Page */
    private int lPagesPerMin;

    /* Number of requests per min of visitor registration Page */
    private int vRPagesPerMin;

    /* Number of requests per min of exhibitor registration Page */
    private int eRPagesPerMin;

    /* Total pages of whole test */
    private int totalPages;

    /* Average response time of whole test */
    private double averageAll;

    /* Average response time of normal page */
    private double averageNormalPage;

    /* Average response time of login page */
    private double averageLogin;

    /* Average response time of visitor registration page */
    private double averageVisitorRegistration;

    /* Average response time of exhibitor registration page */
    private double averageExhibitorRegistration;
/* Failure times */
private int failure;

/* Success times */
private int success;

/* NormalPage success times */
private int nSuccess;

/* VRegPage success times */
private int vSuccess;

/* ERegPage success times */
private int eSuccess;

/* LoginPage success times */
private int iSuccess;

/* Graphic Interface */
private Interface output;

/* Default Constructor */
public Tester()
{
    averageAll = 0;
    averageNormalPage = 0;
    averageLogin = 0;
    averageVisitorRegistration = 0;
    averageExhibitorRegistration = 0;
    failure = 0;
    success = 0;
    nSuccess = 0;
    vSuccess = 0;
    eSuccess = 0;
    iSuccess = 0;
}

/* Read Properties file and then initial some private variables */
public void readProperties() throws Exception
{
    FileInputStream fs = new FileInputStream(new File("testing.properties"));
    Properties properties = new Properties();
    properties.load(fs);
    totalTime = Integer.parseInt(properties.getProperty("TotalTime"));
    nPagesPerMin = Integer.parseInt(properties.getProperty("NormalPage"));
    iPagesPerMin = Integer.parseInt(properties.getProperty("LoginPage"));
    vRPagesPerMin = Integer.parseInt(properties.getProperty("VRegPage"));
    eRPagesPerMin = Integer.parseInt(properties.getProperty("ERegPage"));
    fs.close();
}

public void initInterface()
{
output = new Interface();
output.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}

public static void main(String[] args) throws Exception
{
    System.out.println("-----Test Start-----");
    Tester tester = new Tester();
tester.readProperties();

    PrintWriter pw;
    PrintWriter pw1;
    int totalResponseTime = 0;
    int totalNormal = 0;
    int totalLogin = 0;
    int totalVisitorRegistration = 0;
    int totalExhibitorRegistration = 0;
    int iteration = 0;

    tester.totalPages = (tester.nPagesPerMin + tester.lPagesPerMin + tester.vRPagesPerMin +
tester.eRPagesPerMin) * tester.totalTime;

    int pages = tester.totalPages;
    int responseTime;

    /* Initialize Page Set */
    VRegPage visitorPage =
        new VRegPage("https://ors.ias.csusb.edu:8443/expo/processV.html",
tester.vRPagesPerMin, tester.totalTime);
    ERegPage exhibitorPage =
        new ERegPage("https://ors.ias.csusb.edu:8443/expo/processE.html",
tester.eRPagesPerMin, tester.totalTime);
    NormalPage normalPage =
tester.nPagesPerMin, tester.totalTime);
    LoginPage loginPage =
        new LoginPage("username", "1111", tester.lPagesPerMin, tester.totalTime);

    pw = new PrintWriter(new FileOutputStream(new File("history.txt")));
    pw1 = new PrintWriter(new FileOutputStream(new File("result.txt")));

    visitorPage.start();
    exhibitorPage.start();
    normalPage.init();
    loginPage.init();

    /* Read response time of each request */
    while (pages > 0)
    {
        /* Check normal page response Time */
        if (!((responseTime = normalPage.getResponseTime()) == -1))
        {
            /* Code... */
        }
    }
--pages;
iteration++;
System.out.println("NormalPage: " + responseTime + " || Iteration: " + iteration);
normalPage.setResponseTime();
if (responseTime == -999) { tester.failure++;
else
{
    totalResponseTime += responseTime;
    totalNormal += responseTime;
    tester.nSuccess++;
    tester.success++;
}
pw.println("NormalPage: " + responseTime);
pw.flush();
}
/* Check login page response Time */
else if (!((responseTime = loginPage.getResponseTime()) == -1))
{
    --pages;
    iteration++;
    System.out.println("LoginPage: " + responseTime + " || Iteration: " + iteration);
    loginPage.setResponseTime();
    if (responseTime == -999) { tester.failure++;
else
{
    totalResponseTime += responseTime;
    totalLogin += responseTime;
    tester.lSuccess++;
    tester.success++;
}
pw.println("LoginPage: " + responseTime);
pw.flush();
}
/* Check visitor registration page response Time */
else if (!((responseTime = visitorPage.getResponseTime()) == -1))
{
    --pages;
    iteration++;
    System.out.println("VisitorRegPage: " + responseTime + " || Iteration: " + iteration);
    if (responseTime == -999) { tester.failure++;
else
{
    totalResponseTime += responseTime;
    totalVisitorRegistration += responseTime;
    tester.vSuccess++;
    tester.success++;
}
pw.println("VisitorPage: " + responseTime);
pw.flush();
}
/* Check exhibitor registration page response Time */
else if (!((responseTime = exhibitorPage.getResponseTime()) == -1))
{
System.out.println("ExhibitorRegPage: "+ responseTime + " || Iteration: "+ iteration);
if (responseTime == -999) { tester.failure++; }
else {
totalResponseTime += responseTime;
totalExhibitorRegistration += responseTime;
tester.eSuccess++;
tester.success++;
}
pw.println("ExhibitorPage: "+ responseTime);
pw.flush();
}
else {
try {
    Thread.sleep(5);
} catch (Exception e) {
    e.printStackTrace();
}
}
}
pw.close();

/* Output the result to a file */
double averageResponseTime;
if (tester.success == 0){
    averageResponseTime = 0;
} else {
    averageResponseTime = (double) totalResponseTime / tester.success;
}
pw1.println("Average Response Time: "+ averageResponseTime + " ms");
if (tester.nSuccess == 0){
pw1.println("Average NormalPage Response Time: "+ 0.0 + " ms");
} else {
pw1.println("Average NormalPage Response Time: "+
    (double) (totalNormal / tester.nSuccess) + " ms");
}
if (tester.lSuccess == 0){
pw1.println("Average LoginPage Response Time: "+ 0.0 + " ms");
} else {
pw1.println("Average LoginPage Response Time: "+
    (double) (totalLogin / tester.lSuccess) + " ms");
}
if (tester.vSuccess == 0)
    pw1.println("Average Visitor Registration Page Response Time: \" + 0.0 + \" ms\";
} else {
    pw1.println("Average Visitor Registration Page Response Time: \" +
        (double) (totalVisitorRegistration / tester.vSuccess) + \" ms\";
}

if (tester.eSuccess == 0)
    pw1.println("Average Exhibitor Registration Page Response Time: \" +
} else {
    pw1.println("Average Exhibitor Registration Page Response Time: \" +
        (double) (totalExhibitorRegistration / tester.eSuccess) + \" ms\";
}

if (averageResponseTime == 0) {
    pw1.println("Throughput: \" + 0.0 + \" pages per min\";
} else {
    pw1.println("Throughput: \" + (60000 / averageResponseTime) + \" pages per min\";
}

pw1.println("Failure Rate: \" + (double) tester.failure / tester.totalPages * 100 + \" %\";
pw1.println("Success cases: \" + tester.success + \" pages\";

pw1.println("LoginPage Success cases: \" + tester.nSuccess + \" pages\";

pw1.println("VRegPage Success cases: \" + tester.vSuccess + \" pages\";

pw1.println("ERegPage Success cases: \" + tester.eSuccess + \" pages\";

pw1.flush();

System.out.println("-----Test End-----");

}
REFERENCES


    http://www.postgresql.com/docs

    http://wp.netscape.com/eng/ssl3