2002

Javaserver page, Java servlet and JavaBean technology: Online real estate company

Kevin Tzu-Jung Chen

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

Part of the Digital Communications and Networking Commons

Recommended Citation
Chen, Kevin Tzu-Jung, "Javaserver page, Java servlet and JavaBean technology: Online real estate company" (2002). Theses Digitization Project. 2204.
https://scholarworks.lib.csusb.edu/etd-project/2204

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.
JAVASERVER PAGE, JAVA SERVLET AND JAVABEAN TECHNOLOGY:

ONLINE REAL ESTATE COMPANY

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
Kevin Tzu-Jung Chen
June 2002
JAVASERVER PAGE, JAVA SERVLET AND JAVABEAN TECHNOLOGY:
ONLINE REAL ESTATE COMPANY

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

by
Kevin Tzu-Jung Chen
June 2002
Approved by:

Dr. David Turner, Chair, Computer Science 6/5/2002
Dr. Richard Botting
Dr. Ernesto Gomez
ABSTRACT

In this project, I will simulate an online real estate company using JSP, Java Servlet and JavaBean technology. It is not as complicated as an online company in the real world, but it includes all the basic functions that an online company needs. This online real estate company - SweetHome - is a brokerage agent that provides easy online solutions for house buyers and sellers. Properties information will be stored in a Relational Database for the users to search. The customer can register as a new member, or even only just as a guest on line.

I designed the system to use a 3-tier architecture. In the client tier, users can access various information and services through a web browser. In the Application server tier, I use JSP and Java Servlets. In the business logic tier, I use JavaBeans to connect to the database by JDBC.

JSP (JavaServer Page) is an extension of the Java servlet technology from Sun that provides a simple programming vehicle for displaying dynamic content on a Web page. Java Servlet is a Java application that
runs in a Web server or application server and provides server-side processing, typically to access a database or perform e-commerce processing.

JavaBean is a reusable software component that can be visually manipulated in builder tools. I use JavaBeans in the web application tier to communicate with the MySQL database in the database server tier.

The goal of this project is to use JSP, Java Servlet and JavaBean technologies to design and implement the SweetHome system.
ACKNOWLEDGMENTS

I would like to thank Dr. Richard Botting and Dr. Ernesto Gomez for being my project's committee members. They have provided me with many precious suggestions and instructions.

Also, extreme thanks to my advisor Dr. David Turner. Dr. Turner has given me many ideas, recommendations, and help throughout the project design and implementation. He is not only a good teacher in the computer field but also a good friend who always reminds me what I should do next with patience.

Finally, profound thanks to my parents and my sister; without their support and encouragement, I could not have had this opportunity to pursue higher education in America. They are the most important people in my life.
# TABLE OF CONTENTS

ABSTRACT ...................................................... iii

ACKNOWLEDGMENTS ........................................... v

LIST OF TABLES ................................................ viii

LIST OF FIGURES .............................................. ix

CHAPTER ONE: INTRODUCTION

1.1 Introduction ............................................. 1

1.2 Scope ..................................................... 2

1.3 Definitions .............................................. 3

CHAPTER TWO: SYSTEM ARCHITECTURE

2.1 The 3-tier Architecture ................................. 7

2.2 Client Tier ................................................ 10

2.3 Application Server Tier ................................. 11

2.4 Database Server Tier ................................... 12

2.5 Middleware Services ................................... 12

CHAPTER THREE: JAVASERVER PAGE, JAVA SERVLET AND JAVABEAN

3.1 JavaServer Page versus Active Server Page .......... 15

3.2 JavaServer Page ......................................... 18

3.3 Java Servlet ............................................. 22

3.4 JavaBean ................................................. 24

CHAPTER FOUR: SOFTWARE DESIGN
LIST OF TABLES

Table 1. Definitions ................................................. 3
Table 2. Active Server Page versus JavaServer Page ......................... 17
Table 3. Property Table .................................................. 47
Table 4. Branch Table .................................................... 48
Table 5. Employee Table .................................................. 49
Table 6. Broker Table .................................................... 50
Table 7. BrokerLanguage Table ............................................ 51
Table 8. Customer Table .................................................. 52
Table 9. Appointment Table ............................................... 53
Table 10. Buyer Table .................................................... 54
Table 11. Owner Table .................................................... 55
Table 12. SaleRecord Table ............................................... 56
Table 13. Test Table 1 .................................................... 65
Table 14. Test Table 2 .................................................... 66
Table 15. Test Table 3 .................................................... 67
Table 16. Test Table 4 .................................................... 69
Table 17. Test Table 5 .................................................... 70
Table 18. Test Table 6 .................................................... 71
Table 19. Test Table 7 .................................................... 72
Table 20. Test Table 8 .................................................... 72
Table 21. Source Code Table .............................................. 74
LIST OF FIGURES

Figure 1. Physical Diagram ........................................ 9
Figure 2. Java Database Connectivity Diagram .... 14
Figure 3. Basic JavaServer Page Request Model ... 19
Figure 4. JavaServer Page and Servlet
          Flow Diagram ........................................ 21
Figure 5. Use Case Diagram ................................. 26
Figure 6. Home Page ........................................ 29
Figure 7. Customer Registration Page ................. 30
Figure 8. Property Search Page ......................... 32
Figure 9. Broker Search Page ......................... 33
Figure 10. Office Search Page ....................... 34
Figure 11. Appointment Set Up Page ................. 35
Figure 12. Buyer Guide Page ......................... 37
Figure 13. Owner Guide Page ......................... 38
Figure 14. Deployment Diagram ....................... 39
Figure 15. Class Diagram-1 ......................... 40
Figure 16. Class Diagram-2 ......................... 41
Figure 17. Class Diagram-3 ......................... 42
Figure 18. Data Requirement Diagram ................ 43
Figure 19. Property Data in MySQL ................ 47
Figure 20. Branch Data in MySQL ................ 48
Figure 21. Employee Data in MySQL ............... 49
Figure 22. Broker Data in MySQL .................. 50
Figure 23. BrokerLanguage Data in MySQL .......... 51
Figure 24. Customer Data in MySQL ................ 52
Figure 25. Appointment Data in MySQL ............. 53
Figure 26. Buyer Data in MySQL .................... 54
Figure 27. Owner Data in MySQL ................... 55
Figure 28. SaleRecord Data in MySQL ............... 56
Figure 29. Entity Relationship Diagram ............. 64
CHAPTER ONE

INTRODUCTION

1.1 Introduction

Web based application is becoming more and more popular these days. Many dot com companies such as Amazon, eBay, and other online companies provide their business services through the Internet. Customers can log into a company’s web site to purchase goods and services. Online companies can provide their information and services without opening a real retail store.

In this project, I will simulate an online real estate company. This online real estate company - SweetHome - is a brokerage agent that buys and sells properties, especially houses, in California. SweetHome provides basic functions for those who want to find appropriate properties or brokers through the Internet. Customers can search property in the SweetHome website. They can also choose their brokers depend on what language the customers speak, as well as the brokers’ experience, and sale records etc. After registration, if customers want to buy or sell
their houses, they can make an appointment with their brokers.

The most two popular platforms for web application development are Microsoft .NET and Java J2EE. I choose the Java platform in this project is based on the following reasons:

First, it works on multiple platforms; once you finish the web application, you can run it both on Linux or Windows.

Second, it is an open source; you can download the J2SE or J2EE from Sun’s website for free.

Third, it is object oriented; compared to C++ and Visual Basic, Java is an object oriented language.

Thus, I chose to use Java related technologies to implement this system.

1.2 Scope

SweetHome will provide services and information for customers who want to buy or sell their houses. This online real estate company could support two kinds of scenarios.

First, the customers are property buyers. The application will provide the interfaces so that the
buyers can search the properties on the market, find the appropriate office and broker, register them on the web site, and make an appointment with their broker.

Second, the customers are property sellers. The scenario for the sellers will be the same as the one for buyers except for searching for properties.

The application starts with customers logging into the web site and ends with customers logging off the web site.

1.3 Definitions

Table 1. Definitions

<table>
<thead>
<tr>
<th>Java</th>
<th>An object oriented language developed by Sun Microsystems. Java programs are capable of running on most popular computer platforms without the need for recompilation.</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSP</td>
<td>JavaServer Page, an extension to the Java servlet technology from Sun that provides a simple programming vehicle for displaying dynamic content on a Web page.</td>
</tr>
<tr>
<td>Java Servlet</td>
<td>A Java application that runs in a Web server or application server and provides server-side processing, typically to access a database or perform e-commerce processing.</td>
</tr>
<tr>
<td><strong>JavaBean</strong></td>
<td>A component architecture for the Java programming language, developed initially by Sun, but now available from several other vendors. JavaBeans components are called &quot;beans.&quot;</td>
</tr>
<tr>
<td><strong>JavaScript</strong></td>
<td>A scripting language that is widely supported in Web browsers and other Web tools. It adds interactive functions to HTML pages, which are otherwise static.</td>
</tr>
<tr>
<td><strong>JDBC</strong></td>
<td>Java Database Connectivity, a programming interface that lets Java applications access a database via the SQL language.</td>
</tr>
<tr>
<td><strong>JDK</strong></td>
<td>Java Development Kit, a free Sun Microsystems product which provides the environment required for programming in Java. The JDK is available for a variety of platforms, such as Sun Solaris, Microsoft Windows and Linux.</td>
</tr>
<tr>
<td><strong>EJB</strong></td>
<td>Enterprise JavaBeans, server-side component architecture for writing reusable business logic and portable enterprise applications. EJB is the basis of Sun's J2EE.</td>
</tr>
<tr>
<td><strong>HTML</strong></td>
<td>Hyper Text Markup Language, a document format used on the World-Wide Web.</td>
</tr>
<tr>
<td><strong>HTTP</strong></td>
<td>Hyper Text Transfer Protocol, the protocol that defines how messages are formatted and transmitted on the World Wide Web.</td>
</tr>
<tr>
<td><strong>IEEE</strong></td>
<td>Institute of Electrical and Electronics Engineers. The world's largest technical professional society, based in the USA. Founded in 1884 by a handful of</td>
</tr>
</tbody>
</table>
practitioners of the new electrical engineering discipline, today's Institute has more than 320,000 members who participate in its activities in 147 countries.

<table>
<thead>
<tr>
<th>OS</th>
<th>Operating System. The low-level software which handles the interface to peripheral hardware, schedules tasks, allocates storage, and presents a default interface to the user when no application program is running.</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL</td>
<td>Structured Query Language. A standard language that provides controlled access to databases.</td>
</tr>
<tr>
<td>Browser</td>
<td>A program capable of retrieving HTML documents that includes references to images and Java byte code and rendering it into a user-readable document.</td>
</tr>
<tr>
<td>Server</td>
<td>A program which provides some service to other (client) programs. The connection between client and server is normally by means of message passing, often over a network, and uses some protocol to encode the client's requests and the server's responses.</td>
</tr>
<tr>
<td>Client</td>
<td>A computer system or process that requests a service of another computer system or process (a &quot;server&quot;) using some kind of protocol and accepts the server's responses. A client is part of client-server software architecture.</td>
</tr>
</tbody>
</table>
| API     | Application Programming Interface. The interface by which an application program accesses operating system and other services. An API is defined at source code level and provides a level of abstraction between the application and the kernel (or other privileged
<table>
<thead>
<tr>
<th><strong>utilities</strong></th>
<th>to ensure the portability of the code.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GUI</strong></td>
<td>Graphical User Interface. The graphical representation of physical or pseudo-physical objects (such as buttons, trees, and lists) that allow the user to direct the flow of the program through the use of a mouse or other pointing devices.</td>
</tr>
<tr>
<td><strong>URL</strong></td>
<td>Universal Resource Locator, a standard way of specifying the location of an object, typically a web page, on the Internet.</td>
</tr>
</tbody>
</table>
CHAPTER TWO

SYSTEM ARCHITECTURE

2.1 The 3-tier Architecture

The infrastructure of SweetHome is a 3-tier architecture (See Figure 1), using web technologies including multi-platform browsers, JavaBean, HTTP servers, HTML documents, JSP, Java Servlet, and JDBC access to Relational Databases.

I considered several other architectures available for this project, such as the 2-tier architecture. The 2-tier architecture is the traditional idea of a "client-server" system. Usually, this architecture combines both presentation logic and business logic together in one tier. The server provides persistent storage via a database.

The 3-tier architecture separates out the presentation logic from the business logic (which resides on a middle-tier system). Considering the scalability and flexibility for this business, I decide to set up a 3-tier web deployed architecture. There are many advantages of using a 3-tier architecture with the web.
First, the 2-tier architecture does not have an application server tier. Business-objects that implement the business rules "live" here, and are available to the client-tier. This tier protects the data from direct access by the clients.

Second, as the Internet and Intranet become more and more powerful and widely-used in business and daily life, the World Wide Web is the perfect place to advertise the products.

Third, the cost of setting up a site is reasonable and affordable ($70 to obtain a domain name for first 2 years and $35 per year after 2 years).

Finally, a system deployed through the World Wide Web can avoid the hassles of client-software installation and upgrades. Clients simply need access to a browser, which can be downloaded free of charge from the web. The security, scalability, and platform independence are the main reasons why I rejected the 2-tier architecture.
Figure 1. Physical Diagram
2.2 Client Tier

The web browser is the minimalist client that sends users’ requests and interprets information it receives from a server, and displays it graphically to a user. The client is simply there to interpret the server’s commands and render the contents of a HTML page to the user. Web Browsers are primary interpreters of HTML syntax. The browser executes the HTML commands to properly display text and images on a specific GUI platform. Users navigate from one page to another using the embedded hypertext links.

In this project, I needed to develop a website that contained the main page and a pop-up window for mobile users to login to the Internet.

Operating System

This is a client platform independent system. The client can use any operating systems, such as Windows 95/98/NT/2000/XP workstation, Mac OS, OS/2, UNIX, Linux etc. Netscape Navigator 3.0 or higher or Microsoft’s Internet Explorer 3.x or higher is required to view the HTML documents and web forms on the Web Server.
2.3 Application Server Tier

A World Wide Web server is simply a program that answers requests for documents from World Wide Web clients over the Internet. All World Wide Web servers use a language, or protocol to communicate with web clients called the Hyper Text Transfer Protocol. This is where the http in a web URL comes from. All types of data can be exchanged using this protocol including HTML, graphics, sound and video. Web clients convert open URL commands into HTTP GET requests.

Considering the performance issue, which is basically the waiting time, and also about the budget issue, which has to be pressed as low as possible, this project was designed to run in Jakarta Tomcat 4.0.1 web server.

SweetHome web site is not a very heavy site that provides thousands of services to people. This is different from sites such as yahoo.com, which has yahoo games, yahoo maps, online shopping, chatting, messenger, and so on. Tomcat web server, which is the product of Jakarta project from Apache, is offered for free by downloading directly from the Apache website,
Moreover, Tomcat also supports Servlets and JSP programming, which are the main technology used in this project.

2.4 Database Server Tier

Of course, every system must have a place to store and retrieve massive loads of data. I decided to use MySQL 3.23 as my DBMS server.

MySQL is free software and there are different versions that can run on Windows, MacOS, Solaris, HP-UX, AIX and Linux. It supports APIs for C, C++, Eiffel, Java, Perl, PHP, Python, and Tcl. It provides very fast joins using an optimized one-sweep multi-join. You can connect to MySQL by running a middle-tier separate application server that has a MySQL driver.

2.5 Middleware Services

Middleware starts with the API set on the client side that is used to invoke a service and covers the transmission of the request over the network and the resulting response. In N-tier environments, middleware must provide a platform for running server-side
components, balancing their loads, managing the integrity of transactions, maintaining high-availability, and securing the environment. Here are two general middleware systems:

**Object-Specific Middleware - The Hypertext Transfer Protocol**

The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. It is a generic, stateless, object-oriented protocol that can be used for many tasks, such as name servers and distributed object management systems, through extension of its request methods. A feature of HTTP is the typing and negotiation of data representation, allowing systems to be built independently of the data being transferred. HTTP has been in use by the World-Wide Web global information initiative since 1990.

**Database-Specific Middleware - Java Database Connectivity**

JavaSoft's Java Database Connectivity (JDBC) is an API that lets you access virtually any tabular data source from the Java programming language. It provides cross-DBMS connectivity to a wide range of SQL
databases (See Figure 2). It makes Java codes DBMS-independent. Almost all of main database vendors provide their own JDBC drivers in multi-platform.

Therefore, at most time, the database operation program written in java with JDBC is easy to be migrated to many kinds of platforms without changing the code.

---

**Figure 2. Java Database Connectivity Diagram**
CHAPTER THREE

JAVASERVER PAGE, JAVA SERVLET
AND JAVA BEAN

3.1 JavaServer Page versus
Active Server Page

JavaServer Pages (JSP) and Microsoft Active Server
Pages (ASP) technologies have many similarities. Both
provide a simplified, fast way to create web pages
that display dynamically generated content. But they
also differ significantly in some ways. Here are some
main strengths of JSP as compared to ASP:

Portability

JSP is being developed by Sun Microsystems and is
designed to be both platform and server independent
(See Table 2.).

Therefore, it has "Write Once, Run Anywhere"
capability. In contrast, ASP is purely a Microsoft
based technology deployed primarily on Windows
servers. JSP technology was designed to support
numerous servers, browsers and tools. For example,
Apache web server, which hosts more than 70% of the
websites worldwide, will now fully support the JSP technology. So, you do not have to worry about your OS being Windows or Linux.

Reusability

Most JSP pages rely on reusable, cross-platform components (See Table 2.) to perform the more complex processing required of the application. Developers can instantiate JavaBeans components, set or retrieve bean attributes and perform other functions that are otherwise more difficult and time-consuming to code.

They also can share and exchange components that perform common operations, or make them available to larger customer communities. The component-based approach speeds overall development and lets organizations leverage their existing expertise and development efforts for optimal results.

Performance

The JSP page is compiled into a Java Servlet class and remains in server memory after it has been called for the first time, so subsequent calls to the page have faster response time whereas in ASP the page needs to be recompiled for every request.
Custom Tag Libraries

The JSP technology is extensible through the development of customized tag libraries. The web page developers can create custom tag libraries, so page authors can access more functionality using XML-like tags and depend less on scripting. With custom tags, developers can shield page authors from the complexities of page creation logic and extend key functions to a broader range of authors.

Table 2. Active Server Page versus JavaServer Page

<table>
<thead>
<tr>
<th></th>
<th>ASP Technology</th>
<th>JSP Technology</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web Server</td>
<td>Microsoft IIS or Personal Web Server</td>
<td>Any Web server, including Apache, Netscape, and IIS</td>
</tr>
<tr>
<td>Platforms</td>
<td>Microsoft Windows</td>
<td>Most popular platforms, including the Solaris,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Microsoft Windows, Mac OS, Linux, and other UNIX</td>
</tr>
<tr>
<td></td>
<td></td>
<td>platform implementations'</td>
</tr>
<tr>
<td>Reusable, Cross-</td>
<td>No</td>
<td>JavaBeans, Enterprise JavaBeans, custom JSP tags</td>
</tr>
<tr>
<td>Platform Components</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Compiles</td>
<td>Every time</td>
<td>First time</td>
</tr>
<tr>
<td>Customizable Tags</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>
3.2 JavaServer Page

Java Server Pages (JSP) is a server-side scripting language which is based on the Servlet framework and allows combining of HTML text with Java source code in the same document. The suffix traditionally ends with .jsp to indicate to the web server that the file is a JSP file. JSP is a server side technology - you can not do any client side validation with it.

JSP sits on top of a Java servlets model and makes working with HTML easier. It allows developers to mix static HTML content with server-side scripting to produce dynamic output. By default, JSP uses Java as its scripting language; however, the specification allows other languages to be used.

To offer the best of both worlds - a robust web application platform and a simple, easy-to-use language and tool set - JSP provides a number of server-side tags that allow developers to perform most dynamic content operations without ever writing a single line of Java code. So developers who are only familiar with scripting, or even those who are simply HTML designers,
can use JSP tags for generating simple output without having to learn Java. Advanced scripters or Java developers can also use the tags, or they can use the full Java language if they want to perform advanced operations in JSP pages.

**JavaServer Page Request Model**

HTTP requests are processed under the JSP model. In the basic request model, a request is sent directly to a JSP page. Figure 3 illustrates the flow of information in this model. JSP code controls interactions with JavaBeans components for business and data logic processing, and then displays the results in dynamically generated HTML mixed with static HTML code.

![Figure 3. Basic JavaServer Page Request Model](image)

The beans depicted can be JavaBeans or EJB (Enterprise JavaBeans) components. Other, more
complicated request models include calling out to other JSP pages or Java servlets from the requested JSP page.

In this project, I wrote branch.jsp, broker.jsp, brokerprofile.jsp and etc (See Figure 4 and Appendix B) to handle the user's input from the browser and display the dynamic result. For example, in the office search page, when customers select one of the SweetHome's offices, the branch.jsp will handle the request and display the result.

JSP files actually get compiled into Servlets. Some benefits can be acquired by writing JSP instead of just the Servlets. For example, Java and the JSP extensions assist in making the HTML more functional. Servlets on the other hand allow outputting of HTML, but it is a tedious process. The other benefit is easier to make a change in the HTML format.
Figure 4. JavaServer Page and Servlet Flow Diagram
3.3 Java Servlet

Servlet are modules of Java code that run in a server application (hence the name "Servlets" similar to "Applets" on the client side) to answer client requests and extend request/response oriented servers, such as Java-enabled web servers. Even tough servlets are not tied to a specific client-server protocol, but they are most commonly used with HTTP and the word "Servlet" is often used in the meaning of "HTTP Servlet".

In this project, appServlet, makeAppServlet, and forgetidServlet etc (See Figure 4 and Appendix C) are responsible for taking data in an HTML order-entry form and applying the business logic used to query or update SweetHome's database. For Example, in the appointment set up page, makeAppServlet provides functions that check the customers enter both their customer and broker Id, and also check whether appointment date is in the past or date is valid, e.g. February 31 is not a valid date.

Since Servlets are written in the highly portable Java language and follow a standard framework, they
provide a means to create sophisticated server extensions in a server and operating system independent way. Typical uses for HTTP Servlets include:

- Processing and/or storing data submitted by an HTML form.
- Providing dynamic content, e.g. returning the results of a database query to the client.
- Managing state information on top of the stateless HTTP, e.g. for an online shopping cart system which manages shopping carts for many concurrent customers and maps every request to the right customer.

Servlets provide a component-based, platform-independent method for building web-based applications, without the performance limitations of other programs. And unlike proprietary server extension mechanisms, Servlets are server and platform independent. This leaves you free to adopt a "best of breed" strategy for your servers, platforms, and tools.
3.4 JavaBean

A JavaBean is a reusable component. Beans are a platform-neutral architecture for the Java application environment. It's the ideal choice for developing or assembling network-aware solutions for different hardware and operating system environments--within the enterprise or across the Internet. In fact, it's the only component architecture you should consider if you're developing for the Java platform.

The JavaBean component architecture extends "Write Once, Run Anywhere" capability to reusable component development. In fact, the JavaBean architecture takes interoperability a major step forward--your code runs on every OS and also within any application environment. A beans developer secures a future in the emerging network software market without losing customers that use proprietary platforms, because JavaBeans components interoperate with ActiveX. JavaBeans architecture connects via bridges into other component models such as ActiveX. Software components that use JavaBeans APIs are thus portable to containers.
including Internet Explorer, Visual Basic, Microsoft Word, Lotus Notes, and others.

In this project, I write accessBean, PropertyInfo and BrokerInfo (See Figure 15 and Appendix C) as a reusable component for SweetHome system. For example, every time the psQuery servlet or makeappServlet want to connect to the database, they can just reuse accessBean instead of writing a new one.

JavaBeans is a complete component model. It supports the standard component architecture features of properties, events, methods, and persistence. In addition, JavaBeans provides support for introspection (to allow automatic analysis of a JavaBeans component) and customization (to make it easy to configure a JavaBeans component).
CHAPTER FOUR
SOFTWARE DESIGN

4.1 Use Case and Functions

The following figure shows a Use Case diagram that graphically depicts the users and principal functions of SweetHome. The functions and the actors in the diagram are further described in the next.

![Use Case Diagram](image)

**Figure 5. Use Case Diagram**
Login and Logout

All users can login to and logout of the SweetHome system. Customers must register first before login to the system.

Change Password

All users are able to change their passwords. The system administrator, on the other hand, can change all users' passwords. This is for the circumstance when a user forgets his password and needs help from the administrator.

Search Property

Customer is asked to enter the search criteria, including the location, property type, number of bedrooms and bathrooms desired, property age, price range, and property size. The system will return the results that meet the searching criteria.

Register

Customers can register their personal information such as SSN, name, address and telephone number into the SweetHome system.

Search Broker

This function will ask the customers to specify their searching criteria that include the broker's
language, experience, sales record, and the location of the broker’s office. The system will return the results, and the customer can select a broker with whom to set up an appointment in the appointment set up page.

Search Office

This function will list all the branches of SweetHome. Customers can choose one of them and get the branch’s address and telephone number.

Make an Appointment

The customer fills in the broker ID and his customer ID, selects the appointment time, and click the submit button. The system will return either the message “Sorry, the broker is not available at that time” or the message “Congratulations, your appointment has been confirmed.”

Delete Customer Account

The administrator can delete customer accounts if necessary.
4.2 Scenarios and User Interface

The following scenario demonstrates a few ways that the SweetHome site could be used by describing a user’s view of interaction with the systems.

1. A customer connects to the application by pointing the browser to the URL for the application’s home page (See Figure 6).

![Figure 6. Home Page](image)

This allows the customer to browse through the buyer’s guide that will redirect the customer
to some search interface.

2. At any point during the whole interaction, the customers can click the “Customer Registration” to register themselves in the SweetHome web site if they have not registered yet (See Figure 7).

![Customer Registration Page](http://localhost:8090/tevfhpndex.html)

**Figure 7. Customer Registration Page**

The customer fills in their personal information. If the format is wrong, the application will display a warning page and
require the customer to fill the information in the proper format. If the customer has already registered, the application will remind them. After the registration is completed, the customer can use their SSN as ID to browse the whole site.

3. The customer wanting to search the properties in the SweetHome Inc. can click the "Property Search" bar in the left side. The customer is asked to enter the searching criteria, including the location, property type, number of bedroom required, number of bathroom required, property age, price range, and property size (See Figure 8). The application will return the result that meets the searching criteria. The customer can select one of the properties and click the summit button to make an appointment with the listed broker.
4. The customer decides to find a broker in SweetHome, Inc. The broker search function will ask the customer to specify their searching criteria that include the language, experience, sale record, and the broker's office (See Figure 9). The application will return the result, and the customer can select one broker with whom to set up an appointment to discuss their needs.
5. The customer wants to find branch information in SweetHome (See Figure 10). The customer selects one office and clicks summit button to view the office information like the address, office phone number, fax number and map.
Another choice from the office information page file is to view the broker profile in that office. If the customer chooses to view the broker profile, the application will return the brokers list in that office. The customer can select one of them to view the broker's profile and then to set up an appointment.

6. At any point during the whole interaction, the customers can click "Make Appointment" to set
up an appointment with the broker (See Figure 11). First, the customer fills in the broker ID and his customer ID, selects the appointment time and clicks the submit button.

![SweetHome](image)

**Figure 11. Appointment Set Up Page**

The application will return either the message “Sorry, the broker is not available at that time” or the message “Congratulations, your appointment has been confirmed.” If the customer chooses a time slot that is in the
past, the message will show "Error! The time you've chosen was in the past. Please choose another time."

The customer clicks the "Forget ID?" link to remember his ID if the customer forgets the customer ID. The customer inputs their SSN and zip code and clicks the submit button. The application will return the condition of the customer and how to use the customer ID.

Above is the typical real estate service scenario in SweetHome. If the customers are new to the site, they can start from buy property guide and sell property guide to get more information in SweetHome (See Figure 12 and 13).
Property Buyer Guide

Welcome! Choose one of the Following:

- If you already know a broker and you would like to make an appointment with him/her.
- Find a property matching you need.
- If you don't know a broker and you would like to search one by choosing an office near you.
- If you don't know a broker and you would like to search one by broker's qualification.

Figure 12. Buyer Guide Page
Figure 13. Owner Guide Page
Customer can see HTML page through browser in their computers. The customers' requests are transmitted to the web server via HTTP protocol. The Servlet Engine will load and execute the JSP and JavaServlet to response the customers' requests and display the dynamic content. JavaBean will handle the
business logic, and it can connect to the database by JDBC.

Class Relationship

The figure below briefly shows the relationship of each class in SweetHome system.

Figure 15. Class Diagram-1
## Class Specification

The figures below show each class’s detailed attributes and operations. The java source code for each class is in Appendix C.

### accessBean

- database : String
- user : String
- password : String
- sql : String
- resultArray : String[]
- colArray : String[]
- maxResultArrayLen : int
- maxColArrayLen : int
- fillStatus : boolean
- numColumns : int
- errMsg : String

#### accessBean methods
- execQuery()
- execUpdate()
- removePropertyChangeListener()
- getFillStatus()
- setFillStatus()
- getColArray()
- setColArray()
- getResultArray()
- setResultArray()
- setNumColumns()
- getNumColumns()
- getMaxResultArrayLen()
- setMaxResultArrayLen()
- getMaxColArrayLen()
- setMaxColArrayLen()
- getErrMsg()

### appServlet

- doPost()
- doGetServletInfo()

### makeAppServlet

- aB : accessBean
- adapter : PropertyChangeListenerAdapter
- queryStatus : boolean
- flag : boolean
- userId : String
- password : String

#### makeAppServlet methods
- doPost()
- doGetServletInfo()
- getErrorMsg()
- setQueryStatus()
- getQueryStatus()

### crServlet

- aB : accessBean
- adapter : PropertyChangeListenerAdapter
- queryStatus : boolean
- userId : String
- password : String

#### crServlet methods
- doPost()
- doGetServletInfo()
- getErrorMsg()
- setQueryStatus()
- getQueryStatus()

### forgetidServlet

- aB : accessBean
- adapter : PropertyChangeListenerAdapter
- queryStatus : boolean
- userId : String
- password : String

#### forgetidServlet methods
- doPost()
- doGetServletInfo()
- getErrorMsg()
- setQueryStatus()
- getQueryStatus()

---

Figure 16. Class Diagram-2
<table>
<thead>
<tr>
<th>PropertyInfo</th>
<th>psCheck</th>
<th>bsCheck</th>
</tr>
</thead>
<tbody>
<tr>
<td>city: String</td>
<td>doPost()</td>
<td>doPost()</td>
</tr>
<tr>
<td>zip: String</td>
<td>getServletInfo()</td>
<td>getServletInfo()</td>
</tr>
<tr>
<td>propertyId: String</td>
<td></td>
<td></td>
</tr>
<tr>
<td>type: String</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bedroom: String</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bathroom: String</td>
<td></td>
<td></td>
</tr>
<tr>
<td>builyear: String</td>
<td></td>
<td></td>
</tr>
<tr>
<td>price: String</td>
<td></td>
<td></td>
</tr>
<tr>
<td>size: String</td>
<td></td>
<td></td>
</tr>
<tr>
<td>errMsg: String</td>
<td></td>
<td></td>
</tr>
<tr>
<td>setCity()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getCity()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>setZip()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getZip()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>setPropertyId()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getPropertyId()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>setType()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getType()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>setBedroom()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getBedroom()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>setBathroom()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getBathroom()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>setBuilyear()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getBuilyear()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>setPrice()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getPrice()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>setSize()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>setSize()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>setErrMsg()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getErrMsg()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>reset()</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>psQuery</th>
<th>bsQuery</th>
</tr>
</thead>
<tbody>
<tr>
<td>url: String</td>
<td>url: String</td>
</tr>
<tr>
<td>driver: String</td>
<td>driver: String</td>
</tr>
<tr>
<td>aBean: accessBean</td>
<td>aBean: accessBean</td>
</tr>
<tr>
<td>database: String</td>
<td>database: String</td>
</tr>
<tr>
<td>login: String</td>
<td>login: String</td>
</tr>
<tr>
<td>password: String</td>
<td>password: String</td>
</tr>
<tr>
<td>init()</td>
<td>init()</td>
</tr>
<tr>
<td>doPost()</td>
<td>doPost()</td>
</tr>
<tr>
<td>getServletInfo()</td>
<td>getServletInfo()</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>psAgain</th>
<th>BrokerInfo</th>
</tr>
</thead>
<tbody>
<tr>
<td>doPost()</td>
<td>setLanguage()</td>
</tr>
<tr>
<td>getServletInfo()</td>
<td>setLanguage()</td>
</tr>
<tr>
<td></td>
<td>getLanguage()</td>
</tr>
<tr>
<td></td>
<td>setYear()</td>
</tr>
<tr>
<td></td>
<td>setCity()</td>
</tr>
<tr>
<td></td>
<td>getCity()</td>
</tr>
<tr>
<td></td>
<td>setNumOfSales()</td>
</tr>
<tr>
<td></td>
<td>getNumOfSales()</td>
</tr>
<tr>
<td></td>
<td>reset()</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>office</th>
<th>language: String</th>
</tr>
</thead>
<tbody>
<tr>
<td>branchno: String</td>
<td>setLanguage()</td>
</tr>
<tr>
<td>aB: accessBean</td>
<td>getLanguage()</td>
</tr>
<tr>
<td>information()</td>
<td>setYear()</td>
</tr>
<tr>
<td>manager()</td>
<td>setCity()</td>
</tr>
<tr>
<td>aggregate()</td>
<td>getCity()</td>
</tr>
<tr>
<td>brokers()</td>
<td>setNumOfSales()</td>
</tr>
<tr>
<td>brokerprofile()</td>
<td>getNumOfSales()</td>
</tr>
<tr>
<td>brokerexp()</td>
<td>reset()</td>
</tr>
<tr>
<td>sale()</td>
<td></td>
</tr>
<tr>
<td>setbranchno()</td>
<td></td>
</tr>
<tr>
<td>getbranchno()</td>
<td></td>
</tr>
</tbody>
</table>

Figure 17. Class Diagram-3
5.1 Data Requirements

The following figure shows the Data requirements that SweetHome needs.

Figure 18. Data Requirement Diagram
Branch Office

Each branch office is identified by a unique branch number and has an address (number, street, city, post code), telephone number and fax number.

Employee

Each branch has employees. Each employee has an SSN, uniquely across all branch offices. Information held on each employee includes name (first name, middle name, and last name), sex, salary, address (number, street, city, post code), SSN, telephone number, date of birth, job title, languages they can speak, and the starting date they were employed.

Brokers have information on the broker license number, pager number, and the starting date they were employed. Each broker has unique broker ID across branches.

Property

Each property has property ID, property type (ex: condo, town house, and single family house), sale price, square feet, number of rooms, number of baths, built year and address (number, street, city, post code). The years of the property can be calculated
from the built year. Property ID can uniquely identify the property.

Customer

Customers who want to buy or sell the house have to register with SweetHome and make appointments with SweetHome brokers. Each customer is allowed to make appointments with only one broker in SweetHome at a time. Each customer who registers with SweetHome has SSN, phone number (home number and work number), address (number, street, city, post code), name (first name, middle name, and last name) and registered date with the branch. SSN can uniquely identify customers. Customers in SweetHome database are all property buyers and property owners. Customers that currently have properties to sell or that would like to buy the houses are considered as active customers in the database.

Appointment

Property owners provide property information to brokers. Brokers will arrange an appointment with property buyers on available properties. Appointment dates, time, and comments will be recorded with every
appointment. Appointment date and time uniquely identifies appointments between one customer and one broker.

SaleRecord

The branch keeps all transactions in the records. Records have information on transaction number, property number; broker’s SSN who made the deal, transaction amount, and the date. The transaction number can uniquely identify records within a branch.
5.2 Database Relational Model

Table 3. Property Table

<table>
<thead>
<tr>
<th>PropertyId</th>
<th>Price</th>
<th>SquareFt</th>
<th>BuiltYear</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>NoRoom</td>
<td>NoBath</td>
<td>Street</td>
<td>City</td>
<td>State</td>
</tr>
</tbody>
</table>

Figure 19. Property Data in MySQL
Table 5. Employee Table

<table>
<thead>
<tr>
<th>SSN</th>
<th>LName</th>
<th>MName</th>
<th>FName</th>
<th>Sex</th>
<th>City</th>
<th>State</th>
<th>Code</th>
<th>StartDate</th>
<th>BranchNo</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>San</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>40219681</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>35019652</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>92570018</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>510920189</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5107501903</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3729651</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>392596096</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>375196268</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>594594064</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>39253402640</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 6. Broker Table.

<table>
<thead>
<tr>
<th>SSN</th>
<th>BrokerId</th>
<th>BLicenseNo</th>
<th>PageNo</th>
<th>DateExpStart</th>
</tr>
</thead>
<tbody>
<tr>
<td>0119991811</td>
<td>8001</td>
<td>EBI7277699</td>
<td>3258402203</td>
<td>1982-03-28</td>
</tr>
<tr>
<td>213936891</td>
<td>8003</td>
<td>BCO2325966</td>
<td>5198620301</td>
<td>1974-11-13</td>
</tr>
<tr>
<td>2122272314</td>
<td>8004</td>
<td>REI6009291</td>
<td>5104602324</td>
<td>1975-05-17</td>
</tr>
<tr>
<td>4838925230</td>
<td>8005</td>
<td>GEI6476946</td>
<td>3102620243</td>
<td>1982-03-16</td>
</tr>
<tr>
<td>8843916265</td>
<td>8006</td>
<td>WAI6089257</td>
<td>5106802597</td>
<td>1980-05-21</td>
</tr>
<tr>
<td>9365161643</td>
<td>8007</td>
<td>NCI2607206</td>
<td>7352702570</td>
<td>1990-05-09</td>
</tr>
<tr>
<td>9770422733</td>
<td>8008</td>
<td>YBI1006311</td>
<td>7101026066</td>
<td>1992-08-15</td>
</tr>
<tr>
<td>5197548190</td>
<td>8009</td>
<td>iCloud240246</td>
<td>5101830251</td>
<td>1982-04-19</td>
</tr>
<tr>
<td>0331891374</td>
<td>8010</td>
<td>ACP0302341</td>
<td>5104602303</td>
<td>1982-04-19</td>
</tr>
<tr>
<td>0395943934</td>
<td>8011</td>
<td>VRM6477154</td>
<td>9256402327</td>
<td>1987-07-03</td>
</tr>
<tr>
<td>439095112</td>
<td>8012</td>
<td>VM0196283</td>
<td>5101830251</td>
<td>1982-04-19</td>
</tr>
<tr>
<td>1600229370</td>
<td>8013</td>
<td>ACP0302341</td>
<td>5104602303</td>
<td>1982-04-19</td>
</tr>
<tr>
<td>1522864090</td>
<td>8014</td>
<td>BCI6036077</td>
<td>5125202269</td>
<td>1988-05-15</td>
</tr>
<tr>
<td>3780766297</td>
<td>8015</td>
<td>TAT6389778</td>
<td>2533402264</td>
<td>1976-12-16</td>
</tr>
<tr>
<td>5489226629</td>
<td>8016</td>
<td>WAI6077415</td>
<td>5199602235</td>
<td>1980-07-17</td>
</tr>
<tr>
<td>1457065598</td>
<td>8017</td>
<td>WCI6049698</td>
<td>3106702580</td>
<td>1978-07-17</td>
</tr>
<tr>
<td>2669272621</td>
<td>8018</td>
<td>XST053169</td>
<td>5215202869</td>
<td>1983-04-05</td>
</tr>
<tr>
<td>99866092</td>
<td>8019</td>
<td>XGI602666</td>
<td>5298120229</td>
<td>1991-07-10</td>
</tr>
<tr>
<td>2162771996</td>
<td>8020</td>
<td>XMT0331434</td>
<td>5102402256</td>
<td>1990-08-20</td>
</tr>
<tr>
<td>3711062820</td>
<td>8021</td>
<td>XLI446016</td>
<td>5277782021</td>
<td>1985-08-16</td>
</tr>
<tr>
<td>1911600000</td>
<td>8022</td>
<td>XMI603853</td>
<td>5354402238</td>
<td>1979-10-27</td>
</tr>
<tr>
<td>2222560906</td>
<td>8023</td>
<td>FMI2723292</td>
<td>3109320200</td>
<td>1980-04-03</td>
</tr>
<tr>
<td>2030801014</td>
<td>8024</td>
<td>WM6468772</td>
<td>5109120203</td>
<td>1982-08-29</td>
</tr>
<tr>
<td>0575543407</td>
<td>8025</td>
<td>WAI6077415</td>
<td>5199602235</td>
<td>1980-05-15</td>
</tr>
<tr>
<td>0555545787</td>
<td>8026</td>
<td>NCI187381</td>
<td>9251802868</td>
<td>1985-06-02</td>
</tr>
<tr>
<td>2288971622</td>
<td>8027</td>
<td>GET0003410</td>
<td>5104602329</td>
<td>1990-03-14</td>
</tr>
<tr>
<td>2020371822</td>
<td>8028</td>
<td>WCI6049698</td>
<td>5199602235</td>
<td>1980-07-17</td>
</tr>
<tr>
<td>1932226269</td>
<td>8032</td>
<td>WCI6049698</td>
<td>5199602235</td>
<td>1980-07-17</td>
</tr>
<tr>
<td>4735225293</td>
<td>8033</td>
<td>XTH009939</td>
<td>5215202869</td>
<td>1983-04-05</td>
</tr>
<tr>
<td>9989910503</td>
<td>8034</td>
<td>XG0341466</td>
<td>5298120229</td>
<td>1991-07-10</td>
</tr>
<tr>
<td>1421986029</td>
<td>8035</td>
<td>SWD0053036</td>
<td>5225702565</td>
<td>1992-07-14</td>
</tr>
<tr>
<td>8698915993</td>
<td>8036</td>
<td>TTV2602603</td>
<td>9258026211</td>
<td>1990-06-27</td>
</tr>
<tr>
<td>4880311215</td>
<td>8037</td>
<td>XDI013834</td>
<td>4348602235</td>
<td>1993-07-26</td>
</tr>
</tbody>
</table>

Figure 22. Broker Data in MySQL
Table 7. BrokerLanguage Table

<table>
<thead>
<tr>
<th>BrokerId</th>
<th>Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>English</td>
</tr>
<tr>
<td>2</td>
<td>English</td>
</tr>
<tr>
<td>3</td>
<td>English</td>
</tr>
<tr>
<td>4</td>
<td>Mandarin</td>
</tr>
<tr>
<td>5</td>
<td>Spanish</td>
</tr>
<tr>
<td>6</td>
<td>English</td>
</tr>
<tr>
<td>7</td>
<td>English</td>
</tr>
<tr>
<td>8</td>
<td>English</td>
</tr>
<tr>
<td>9</td>
<td>English</td>
</tr>
<tr>
<td>10</td>
<td>English</td>
</tr>
<tr>
<td>11</td>
<td>English</td>
</tr>
<tr>
<td>12</td>
<td>English</td>
</tr>
<tr>
<td>13</td>
<td>English</td>
</tr>
<tr>
<td>14</td>
<td>English</td>
</tr>
<tr>
<td>15</td>
<td>Mandarin</td>
</tr>
<tr>
<td>16</td>
<td>English</td>
</tr>
<tr>
<td>17</td>
<td>Spanish</td>
</tr>
<tr>
<td>18</td>
<td>English</td>
</tr>
<tr>
<td>19</td>
<td>Spanish</td>
</tr>
<tr>
<td>20</td>
<td>English</td>
</tr>
<tr>
<td>21</td>
<td>Spanish</td>
</tr>
<tr>
<td>22</td>
<td>English</td>
</tr>
<tr>
<td>23</td>
<td>Spanish</td>
</tr>
<tr>
<td>24</td>
<td>Mandarin</td>
</tr>
<tr>
<td>25</td>
<td>English</td>
</tr>
<tr>
<td>26</td>
<td>English</td>
</tr>
<tr>
<td>27</td>
<td>Spanish</td>
</tr>
<tr>
<td>28</td>
<td>English</td>
</tr>
<tr>
<td>29</td>
<td>English</td>
</tr>
<tr>
<td>30</td>
<td>English</td>
</tr>
<tr>
<td>31</td>
<td>Mandarin</td>
</tr>
<tr>
<td>32</td>
<td>English</td>
</tr>
</tbody>
</table>

Figure 23. BrokerLanguage Data in MySQL
<table>
<thead>
<tr>
<th>SSN</th>
<th>LName</th>
<th>MName</th>
<th>FName</th>
<th>HomeNo</th>
<th>WorkNo</th>
<th>Street</th>
<th>City</th>
<th>State</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>5108600370</td>
<td>Becker</td>
<td>T</td>
<td>Isabel</td>
<td>5108101125</td>
<td>27586</td>
<td>SHEFFIELD PLACE</td>
<td>San Bernardino</td>
<td>CA</td>
<td>94112</td>
</tr>
<tr>
<td>5108100376</td>
<td>Isabel</td>
<td>T</td>
<td>Jack</td>
<td>5108100376</td>
<td>27558</td>
<td>Odando Road</td>
<td>Los Angeles</td>
<td>CA</td>
<td>95115</td>
</tr>
<tr>
<td>5107500383</td>
<td>Kirkland</td>
<td>T</td>
<td>Jack</td>
<td>5107500383</td>
<td>28958</td>
<td>Orlando Road</td>
<td>Los Angeles</td>
<td>CA</td>
<td>95115</td>
</tr>
<tr>
<td>5107100382</td>
<td>Davies</td>
<td>T</td>
<td>Michael</td>
<td>5107100382</td>
<td>2758</td>
<td>Vintaae Street</td>
<td>Hollywood</td>
<td>CA</td>
<td>90201</td>
</tr>
<tr>
<td>5107100389</td>
<td>Ghclndv</td>
<td>T</td>
<td>Michael</td>
<td>5107100389</td>
<td>2758</td>
<td>Jennifer Street</td>
<td>San Bernardino</td>
<td>CA</td>
<td>94112</td>
</tr>
<tr>
<td>5108700374</td>
<td>Nicholson</td>
<td>T</td>
<td>Michael</td>
<td>5108700374</td>
<td>27635</td>
<td>DOBBEL Street</td>
<td>Los Angeles</td>
<td>CA</td>
<td>95127</td>
</tr>
<tr>
<td>5106000402</td>
<td>Squire</td>
<td>T</td>
<td>Randi</td>
<td>5106000402</td>
<td>13235</td>
<td>Manon Avenue</td>
<td>Los Angeles</td>
<td>CA</td>
<td>95122</td>
</tr>
<tr>
<td>4084100422</td>
<td>West</td>
<td>T</td>
<td>Rene</td>
<td>4084100422</td>
<td>2766</td>
<td>BUENA POINT CT  #2766</td>
<td>Ontario</td>
<td>CA</td>
<td>95038</td>
</tr>
<tr>
<td>5103300431</td>
<td>Zoltan</td>
<td>T</td>
<td>Rene</td>
<td>5103300431</td>
<td>27844</td>
<td>DOBBEL Street</td>
<td>Los Angeles</td>
<td>CA</td>
<td>95127</td>
</tr>
<tr>
<td>5108500380</td>
<td>Lapalme</td>
<td>T</td>
<td>Rita</td>
<td>5108500380</td>
<td>29644</td>
<td>DOBBEL Street</td>
<td>Los Angeles</td>
<td>CA</td>
<td>95127</td>
</tr>
<tr>
<td>9252100445</td>
<td>Metsch</td>
<td>T</td>
<td>Tom</td>
<td>9252100445</td>
<td>279</td>
<td>Haqeman Drive</td>
<td>Hollywood</td>
<td>CA</td>
<td>92002</td>
</tr>
<tr>
<td>5104701170</td>
<td>Sable</td>
<td>T</td>
<td>Tom</td>
<td>5104701170</td>
<td>2800</td>
<td>Baylis Road</td>
<td>Riverside</td>
<td>CA</td>
<td>92115</td>
</tr>
<tr>
<td>5101701201</td>
<td>Biqelow</td>
<td>T</td>
<td>Scott</td>
<td>5101701201</td>
<td>2801</td>
<td>Clyder Street</td>
<td>Riverside</td>
<td>CA</td>
<td>92115</td>
</tr>
<tr>
<td>5101700122</td>
<td>Davis</td>
<td>U</td>
<td>Ryan</td>
<td>5101700122</td>
<td>2802</td>
<td>Fox Hollow Street</td>
<td>Los Angeles</td>
<td>CA</td>
<td>95115</td>
</tr>
<tr>
<td>5103001196</td>
<td>Work</td>
<td>U</td>
<td>Ryan</td>
<td>5103001196</td>
<td>2803</td>
<td>Fox Hollow Street</td>
<td>Los Angeles</td>
<td>CA</td>
<td>95115</td>
</tr>
<tr>
<td>5108801133</td>
<td>Clarke</td>
<td>U</td>
<td>Alex</td>
<td>5108801133</td>
<td>2804</td>
<td>MCDUFF Road</td>
<td>Riverside</td>
<td>CA</td>
<td>92115</td>
</tr>
<tr>
<td>5108700382</td>
<td>Dodis</td>
<td>U</td>
<td>Amy</td>
<td>5108700382</td>
<td>2805</td>
<td>MCDUFF Road</td>
<td>Riverside</td>
<td>CA</td>
<td>92115</td>
</tr>
<tr>
<td>5108701136</td>
<td>Thompson</td>
<td>U</td>
<td>Abbey</td>
<td>5108701136</td>
<td>2806</td>
<td>MCDUFF Road</td>
<td>Riverside</td>
<td>CA</td>
<td>92115</td>
</tr>
<tr>
<td>5103001194</td>
<td>Tse</td>
<td>U</td>
<td>Abbey</td>
<td>5103001194</td>
<td>2807</td>
<td>MCDUFF Road</td>
<td>Riverside</td>
<td>CA</td>
<td>92115</td>
</tr>
<tr>
<td>5106801157</td>
<td>Thompson</td>
<td>U</td>
<td>Abbey</td>
<td>5106801157</td>
<td>2808</td>
<td>MCDUFF Road</td>
<td>Riverside</td>
<td>CA</td>
<td>92115</td>
</tr>
<tr>
<td>5107001152</td>
<td>Kline</td>
<td>U</td>
<td>Crystal</td>
<td>5107001152</td>
<td>2809</td>
<td>MCDUFF Road</td>
<td>Riverside</td>
<td>CA</td>
<td>92115</td>
</tr>
</tbody>
</table>

Figure 24. Customer Data in MySQL
Table 9. Appointment Table

<table>
<thead>
<tr>
<th>BrokerId</th>
<th>CustomerSSN</th>
<th>AppDate</th>
<th>AppTime</th>
<th>AppComment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>000000000</td>
<td>2000-04-17</td>
<td>10:00</td>
<td>BUY</td>
</tr>
<tr>
<td>2</td>
<td>614191198</td>
<td>2003-02-22</td>
<td>10:00</td>
<td>BUY</td>
</tr>
<tr>
<td>3</td>
<td>614191198</td>
<td>2003-02-27</td>
<td>16:00</td>
<td>BUY</td>
</tr>
<tr>
<td>4</td>
<td>614191198</td>
<td>2003-04-17</td>
<td>16:00</td>
<td>BUY</td>
</tr>
<tr>
<td>5</td>
<td>6448448444</td>
<td>2002-04-29</td>
<td>10:00</td>
<td>BUY</td>
</tr>
</tbody>
</table>

Figure 25. Appointment Data in MySQL
Table 10. Buyer Table

<table>
<thead>
<tr>
<th>SSN</th>
<th>Active</th>
</tr>
</thead>
</table>

Figure 26. Buyer Data in MySQL
### Table 11. Owner Table

<table>
<thead>
<tr>
<th>SSN</th>
<th>Active</th>
<th>BrokerId</th>
</tr>
</thead>
<tbody>
<tr>
<td>117739363</td>
<td>N</td>
<td>B3052V</td>
</tr>
<tr>
<td>136994680</td>
<td>N</td>
<td>B5068V</td>
</tr>
<tr>
<td>159996962</td>
<td>N</td>
<td>B4064V</td>
</tr>
<tr>
<td>121999317</td>
<td>N</td>
<td>B4061V</td>
</tr>
<tr>
<td>135999692</td>
<td>N</td>
<td>B4008V</td>
</tr>
<tr>
<td>141999731</td>
<td>N</td>
<td>B6105V</td>
</tr>
<tr>
<td>155999453</td>
<td>N</td>
<td>B3004V</td>
</tr>
<tr>
<td>142999453</td>
<td>W</td>
<td>AA7</td>
</tr>
<tr>
<td>131999731</td>
<td>N</td>
<td>B6052V</td>
</tr>
<tr>
<td>144699305</td>
<td>N</td>
<td>B1007V</td>
</tr>
<tr>
<td>155499638</td>
<td>N</td>
<td>B5058V</td>
</tr>
<tr>
<td>145999638</td>
<td>N</td>
<td>B6068V</td>
</tr>
<tr>
<td>157499381</td>
<td>N</td>
<td>B6071V</td>
</tr>
<tr>
<td>152199293</td>
<td>N</td>
<td>B5022V</td>
</tr>
<tr>
<td>154599613</td>
<td>N</td>
<td>B3013V</td>
</tr>
<tr>
<td>155599243</td>
<td>N</td>
<td>B4030V</td>
</tr>
<tr>
<td>150799686</td>
<td>N</td>
<td>B3120V</td>
</tr>
</tbody>
</table>

**Figure 27.** Owner Data in MySQL
<table>
<thead>
<tr>
<th>BranchNo</th>
<th>TransNo</th>
<th>PropertyId</th>
<th>BuyerSSN</th>
<th>OwnerSSN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BrokerId</th>
<th>TransAmount</th>
<th>TransDate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 28. SaleRecord Data in MySQL
5.3 Data Types and Domain

Structured Query Language for Branch Table

CREATE TABLE BRANCH(
    BRANCHNO VARCHAR(4) NOT NULL,
    TELNO VARCHAR(20) UNIQUE NOT NULL,
    FAXNO VARCHAR(20),
    STREET VARCHAR(30) NOT NULL,
    CITY VARCHAR(20) NOT NULL,
    STATE VARCHAR(2) NOT NULL,
    CODE VARCHAR(5) NOT NULL,
    LICENSENO VARCHAR(15),
    CONSTRAINT BRANCHNO_BRANCH_PK PRIMARY KEY(BRANCHNO),
    CONSTRAINT ADDRESS_BRANCH_U UNIQUE(STREET, CITY, STATE, CODE));

Structured Query Language for Employee Table

CREATE TABLE EMPLOYEE(
    SSN VARCHAR(9) NOT NULL,
    LNAME VARCHAR(20) NOT NULL,
    MNAME VARCHAR(20),
    FNAME VARCHAR(20) NOT NULL,
    SEX CHAR NOT NULL,
Structured Query Language for
Broker Table

CREATE TABLE BROKER(
    SSN       VARCHAR(9)    NOT NULL,
    BROKERID  VARCHAR(9)    UNIQUE NOT NULL,
    BLICENSENO VARCHAR(15) UNIQUE NOT NULL,
    PAGERNOM  VARCHAR(20)   UNIQUE NOT NULL,
    DATEEXPSTART  DATE     NOT NULL,
    CONSTRAINT SSN_BROKER_PK PRIMARY KEY(SSN),
);
CONSTRAINT SSN_BROKER_FK FOREIGN KEY(SSN) REFERENCES EMPLOYEE(SSN) ON DELETE CASCADE;

Structured Query Language for Brokerlanguage Table

CREATE TABLE BROKERLANGUAGE(
  BROKERID VARCHAR(9) NOT NULL,
  LANGUAGE VARCHAR(15) NOT NULL,
  CONSTRAINT BROKERID_LANG_PK PRIMARY KEY(BROKERID, LANGUAGE),
  CONSTRAINT BROKERID_LANG_FK FOREIGN KEY(BROKERID) REFERENCES BROKER(BROKERID) ON DELETE CASCADE);

Structured Query Language for Property Table

CREATE TABLE PROPERTY(
  PROPERTYID VARCHAR(10) NOT NULL,
  PRICE INT(8) NOT NULL,
  SQUAREFT INT(5) NOT NULL,
  BUILTYEAR INT(4) NOT NULL,
  TYPE VARCHAR(20) NOT NULL,
  NOROOM INT(2) NOT NULL,
  NOBATH INT(2) NOT NULL,
  STREET VARCHAR(30) NOT NULL,
  CITY VARCHAR(20) NOT NULL,
STATE VARCHAR(2) NOT NULL,
CODE VARCHAR(5) NOT NULL,
CONSTRAINT PROPERTYID_PRO_PK PRIMARY KEY (PROPERTYID),
CONSTRAINT SQUAREFT_PRO_CK CHECK (SQUAREFT>0),
CONSTRAINT BUILTYEAR_PRO_CK CHECK (BUILTYEAR>1900),
CONSTRAINT ADDRESS_PRO_U UNIQUE (STREET, CITY, STATE, CODE));

Structured Query Language for Customer Table

CREATE TABLE CUSTOMER(
SSN VARCHAR(9) NOT NULL,
LNAME VARCHAR(20) NOT NULL,
MNAME VARCHAR(20),
FNAME VARCHAR(20) NOT NULL,
HOMENO VARCHAR(20) NOT NULL,
WORKNO VARCHAR(20),
STREET VARCHAR(30) NOT NULL,
CITY VARCHAR(20) NOT NULL,
STATE VARCHAR(20) NOT NULL,
CODE VARCHAR(5) NOT NULL,
REGDATE DATE NOT NULL,
CONSTRAINT SSN_CUSTOMER_PK PRIMARY KEY (SSN));
Structured Query Language for Buyer Table

CREATE TABLE BUYER(
  SSN VARCHAR(9) NOT NULL,
  ACTIVE CHAR(1) NOT NULL,
  CONSTRAINT SSN_BUYER_PK PRIMARY KEY(SSN),
  CONSTRAINT SSN_BUYER_FK FOREIGN KEY(SSN) REFERENCES CUSTOMER(SSN) ON DELETE CASCADE,
  CONSTRAINT ACTIVE_BUYER_CK CHECK(ACTIVE IN('Y','N')));

Structured Query Language for Owner Table

CREATE TABLE OWNER(
  SSN VARCHAR(9) NOT NULL,
  ACTIVE CHAR(1) NOT NULL,
  BROKERID VARCHAR(9) NOT NULL,
  CONSTRAINT SSN_OWNER_PK PRIMARY KEY(SSN),
  CONSTRAINT SSN_OWNER_FK FOREIGN KEY(SSN) REFERENCES CUSTOMER(SSN) ON DELETE CASCADE,
  CONSTRAINT BID_OWNER_FK FOREIGN KEY(BROKERID) REFERENCES BROKER(BROKERID),
  CONSTRAINT ACTIVE_OWNER_CK CHECK(ACTIVE IN('Y','N')));
Structured Query Language for Appointment Table

CREATE TABLE APPOINTMENT(
  BROKERID VARCHAR(9) NOT NULL,
  CUSTOMERSSN VARCHAR(9) NOT NULL,
  APPDATE DATE NOT NULL,
  APPTIME VARCHAR(5) NOT NULL,
  APPCOMMENT VARCHAR(100) NOT NULL,
  CONSTRAINT APPOINTMENT_PK PRIMARY KEY(BROKERID, APPDATE, APPTIME),
  CONSTRAINT BROKERID_APP_FK FOREIGN KEY(BROKERID) REFERENCES BROKER(BROKERID) ON DELETE CASCADE,
  CONSTRAINT CUSTMERSSN_APP_FK FOREIGN KEY(CUSTOMERSSN) REFERENCES CUSTOMER(SSN) ON DELETE CASCADE);

Structured Query Language for Salerecord Table

CREATE TABLE SALERECORD(
  BRANCHNO VARCHAR(4) NOT NULL,
  TRANSNO VARCHAR(16) NOT NULL,
  PROPERTYID VARCHAR(10) NOT NULL UNIQUE,
  BUYERSSN VARCHAR(9) NOT NULL,
  OWNERSSN VARCHAR(9) NOT NULL,
  BROKERID VARCHAR(9) NOT NULL,
TRANSAMOUNT INT(11) NOT NULL,
TRANS DATE DATE NOT NULL,
CONSTRAINT BNO_TNO_SR_PK PRIMARY KEY (BRANCHNO, TRANSNO),
CONSTRAINT BNO_SR_FK FOREIGN KEY (BRANCHNO) REFERENCES BRANCH (BRANCHNO) ON DELETE CASCADE,
CONSTRAINT BID_SR_FK FOREIGN KEY (BROKERID) REFERENCES BROKER (BROKERID) ON DELETE CASCADE,
CONSTRAINT PID_SR_FK FOREIGN KEY (PROPERTYID) REFERENCES PROPERTY (PROPERTYID) ON DELETE CASCADE,
CONSTRAINT OWNERSSN_SR_FK FOREIGN KEY (OWNERSSN) REFERENCES OWNER (SSN) ON DELETE CASCADE,
CONSTRAINT BUYERSSN_SR_FK FOREIGN KEY (BUYERSSN) REFERENCES BUYER (SSN) ON DELETE CASCADE);
5.4 Entity Relationship Diagram

Figure 29. Entity Relationship Diagram
CHAPTER SIX
SOFTWARE TEST AND SYSTEM MAINTENCE

6.1 Software Test

Table 13. Test Table 1

<table>
<thead>
<tr>
<th>No</th>
<th>Testing Case</th>
<th>Expect Result</th>
<th>Verified</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Click SweetHome</td>
<td>Go to SweetHome home page</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Click Property Search</td>
<td>Go to property search page</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Click Customer Registration</td>
<td>Go to customer registration page</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Click Broker Search</td>
<td>Go to broker search page</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Click Make Appointment</td>
<td>Go to appointment setup page</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Click Office Search</td>
<td>Go to office search page</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Click Buy Property Guide</td>
<td>Go to buyer guide page</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Click Buy Property Guide</td>
<td>Go to owner guide page</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>
Table 14. Test Table 2

<table>
<thead>
<tr>
<th>No</th>
<th>Testing Case</th>
<th>Expect Result</th>
<th>Verified</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>City: Hollywood</td>
<td>Successful Query</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Zip: 92002</td>
<td>Successful Query</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Property ID: 57328</td>
<td>Successful Query</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>
| 4  | City: Riverside  
Type: Condo | Successful Query | ✓        |         |
| 5  | Zip: 94120  
Type: Single family house  
Bedrooms: 4 | Successful Query | ✓        |         |
| 6  | Property id: 67311  
Bathroom: 2  
Year: Before 1970  
Price: $200,000-$299,999 | Successful Query | ✓        |         |
| 7  | City: San Francisco  
Type: Condo  
Bathroom: 1  
Year: Before 1970  
size: 1,000-1,499 | Successful Query | ✓        |         |
| 8  | Zip: 96116  
Type: Condo  
Bedroom: 3  
Bathroom: 2  
Year: Before 1970  
size: 1,000-1,499 | Successful Query | ✓        |         |
Table 15. Test Table 3

<table>
<thead>
<tr>
<th>No</th>
<th>Testing Case</th>
<th>Expect Result</th>
<th>Verified</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Lack SSN</td>
<td>Error Message</td>
<td>√</td>
<td>Input column can’t be empty</td>
</tr>
<tr>
<td>2</td>
<td>Lack First Name</td>
<td>Error Message</td>
<td>√</td>
<td>Input column can’t be empty</td>
</tr>
<tr>
<td>3</td>
<td>Lack Middle Name</td>
<td>Registration Success</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Lack Last Name</td>
<td>Error Message</td>
<td>√</td>
<td>Input column can’t be empty</td>
</tr>
<tr>
<td></td>
<td>Lack Street</td>
<td>Error Message</td>
<td></td>
<td>Input column can’t be empty</td>
</tr>
<tr>
<td>---</td>
<td>-------------</td>
<td>---------------</td>
<td>---</td>
<td>---------------------------</td>
</tr>
<tr>
<td>6</td>
<td>Lack City</td>
<td>Error Message</td>
<td>√</td>
<td>Input column can’t be empty</td>
</tr>
<tr>
<td>7</td>
<td>Lack State</td>
<td>Error Message</td>
<td>√</td>
<td>Input column can’t be empty</td>
</tr>
<tr>
<td>8</td>
<td>Lack Code</td>
<td>Error Message</td>
<td>√</td>
<td>Input column can’t be empty</td>
</tr>
<tr>
<td>9</td>
<td>Lack Home Number</td>
<td>Error Message</td>
<td>√</td>
<td>Input column can’t be empty</td>
</tr>
<tr>
<td>10</td>
<td>Lack Work Number</td>
<td>Registration Success</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 16. Test Table 4

<table>
<thead>
<tr>
<th>No</th>
<th>Testing Case</th>
<th>Expect Result</th>
<th>Verified</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>language: English experience: 0-10</td>
<td>Successful Query</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td></td>
<td>location: San Bernardino</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>sales: 11 and above</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>language: mandarin experience: 11-20</td>
<td>Successful Query</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td></td>
<td>location: Hollywood</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>sales: 11-20</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>language: Spanish experience: 21 and above</td>
<td>Successful Query</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td></td>
<td>location: Ontario</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>sales: 11 and above</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>language: Mandarin experience: 11-20</td>
<td>Successful Query</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td></td>
<td>location: Riverside</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>sales: 0-10</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 17. Test Table 5

<table>
<thead>
<tr>
<th>No</th>
<th>Testing Case</th>
<th>Expect Result</th>
<th>Verified</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>San Bernardino Office</td>
<td>San Bernardino Office Information</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Los Angeles Office</td>
<td>Los Angeles Office Information</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Hollywood Office</td>
<td>Hollywood Office Information</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>San Francisco Office</td>
<td>San Francisco Office Information</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Ontario Office</td>
<td>Ontario Office Information</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Riverside Office</td>
<td>Riverside Office Information</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>
Table 18. Test Table 6

Testing Content: Appointment Set Up Page

(Test Date: May/1)

<table>
<thead>
<tr>
<th>No</th>
<th>Testing Case</th>
<th>Expect Result</th>
<th>Verified</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Lack Broker ID</td>
<td>Error Message</td>
<td>✓</td>
<td>Broker ID is Required</td>
</tr>
<tr>
<td>2</td>
<td>Lack Customer ID</td>
<td>Error Message</td>
<td>✓</td>
<td>Customer ID is Required</td>
</tr>
<tr>
<td>3</td>
<td>Date: Feb/29/2002</td>
<td>Error Message</td>
<td>✓</td>
<td>Invalid Date</td>
</tr>
<tr>
<td>4</td>
<td>Date: Feb/30/2003</td>
<td>Error Message</td>
<td>✓</td>
<td>Invalid Date</td>
</tr>
<tr>
<td>5</td>
<td>Date: Feb/31/2002</td>
<td>Error Message</td>
<td>✓</td>
<td>Invalid Date</td>
</tr>
<tr>
<td>6</td>
<td>Date: Apr/31/2003</td>
<td>Error Message</td>
<td>✓</td>
<td>Invalid Date</td>
</tr>
<tr>
<td>7</td>
<td>Date: Jun/31/2002</td>
<td>Error Message</td>
<td>✓</td>
<td>Invalid Date</td>
</tr>
<tr>
<td>8</td>
<td>Date: Sep/31/2003</td>
<td>Error Message</td>
<td>✓</td>
<td>Invalid Date</td>
</tr>
<tr>
<td>9</td>
<td>Date: Non/31/2002</td>
<td>Error Message</td>
<td>✓</td>
<td>Invalid Date</td>
</tr>
<tr>
<td>10</td>
<td>Date: Jan/10/2002</td>
<td>Error Message</td>
<td>✓</td>
<td>Date is in the past</td>
</tr>
<tr>
<td>11</td>
<td>Date: May/1/2002</td>
<td>Error Message</td>
<td>✓</td>
<td>Can not make appointment today</td>
</tr>
<tr>
<td>12</td>
<td>Date: Jan/10/2003</td>
<td>Successful Message</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>
Table 19. Test Table 7

Testing Content: Buy Property Guide Page

<table>
<thead>
<tr>
<th>No</th>
<th>Testing Case</th>
<th>Expect Result</th>
<th>Verified</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Click first choice</td>
<td>Go to appointment setup page</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Click Second choice</td>
<td>Go to property search page</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Click third choice</td>
<td>Go to office search page</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Click fourth choice</td>
<td>Go to broker search page</td>
<td>√</td>
<td></td>
</tr>
</tbody>
</table>

Table 20. Test Table 8

Testing Content: Sell Property Guide Page

<table>
<thead>
<tr>
<th>No</th>
<th>Testing Case</th>
<th>Expect Result</th>
<th>Verified</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Click first choice</td>
<td>Go to appointment setup page</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Click second choice</td>
<td>Go to office search page</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Click third choice</td>
<td>Go to broker search page</td>
<td>√</td>
<td></td>
</tr>
</tbody>
</table>

72
6.2 System Maintenance

Operating System: MS Windows 98/NT/2000/XP or Red Hat Linux

Browser: Netscape Navigator 3.0 or higher and Microsoft’s Internet Explorer 3.x or higher.

Memory: 128 MB recommended
Free disk space: 500 MB recommended

Development Tools: Java Development Kit (j2sdk1.4.0)
(Download from http://java.sun.com/j2se/1.4/download.html)

Web Server: Jakarta-tomcat-4.0.1
(Download from http://jakarta.apache.org/builds/jakarta-tomcat-4.0/release/v4.0.1/src/)

Database: MySQL (mysql-3.23.47-win.zip)

JDBC: mm.mysql.jdbc-1.2c.tar
(Download from http://www.mysql.com/downloads/api-jdbc.html)

Mysqlgui: mysqlgui-win32-static-1.7.5-2

SweetHome's HTML File: See Appendix A
SweetHome's JSP File: See Appendix B
SweetHome's JavaServlet and JavaBean File: See Appendix C

Table 21. Source Code Table

<table>
<thead>
<tr>
<th>HTML File</th>
<th>JSP File</th>
</tr>
</thead>
<tbody>
<tr>
<td>index.html</td>
<td>branch.jsp</td>
</tr>
<tr>
<td>left.html</td>
<td>broker.jsp</td>
</tr>
<tr>
<td>top.html</td>
<td>profile.jsp</td>
</tr>
<tr>
<td>welcome.html</td>
<td>brokerprofile.jsp</td>
</tr>
<tr>
<td>ps.html</td>
<td></td>
</tr>
<tr>
<td>broker.html</td>
<td></td>
</tr>
<tr>
<td>office.html</td>
<td></td>
</tr>
<tr>
<td>cr.html</td>
<td></td>
</tr>
<tr>
<td>app.html</td>
<td></td>
</tr>
<tr>
<td>buy.html</td>
<td></td>
</tr>
<tr>
<td>sell.html</td>
<td></td>
</tr>
<tr>
<td>forgetid.html</td>
<td></td>
</tr>
<tr>
<td>JavaServlet File</td>
<td>JavaBean File</td>
</tr>
<tr>
<td>appServlet.java</td>
<td>accessBean.java</td>
</tr>
<tr>
<td>makeAppServlet.java</td>
<td>PropertyInfo.java</td>
</tr>
<tr>
<td>crservlet.java</td>
<td>BrokerInfo.java</td>
</tr>
<tr>
<td>forgetidServlét.java</td>
<td>office.java</td>
</tr>
<tr>
<td>psCheck.java</td>
<td></td>
</tr>
<tr>
<td>psQuery.java</td>
<td></td>
</tr>
<tr>
<td>psAgain.java</td>
<td></td>
</tr>
<tr>
<td>bsCheck.java</td>
<td></td>
</tr>
<tr>
<td>bsQuery.java</td>
<td></td>
</tr>
</tbody>
</table>
CHAPTER SEVEN

FUTURE DIRECTIONS

The future directions for the SweetHome website are as follows:

1. Use J2EE to develop higher security and more reliability on EJB reusable component.

2. Add more functions to the SweetHome website, such as adding a little calendar on the Appointment set up page or sending a notification e-mail when customers make an appointment successfully, and so on.

3. Use a more powerful database such as Oracle 9i.

4. Purchase firewall software to protect the system.

To apply SweetHome in the real world, it needs a well organized team and enough expense. This team should include real estate experts, web designers, database administrators, and professional programmers. It also needs to improve performance and security by purchasing a domain name, advanced firewall software, and reliable server hardware.
APPENDIX A

SOURCE CODE: HTML PART
Index.html

<html>
<head><title>Sweet Home Inc.</title></head>

<frameset rows="101,*" framespacing="0" border="0" frameborder="0">
  <frame name="top" scrolling="no" noresize target="contents" src="top.html">
  <frameset cols="195,*">
    <frame name="contents" target="main" src="left.html">
    <frame name="main" src="welcome.html">
  </frameset>
</frameset>

<noframes>
  <body>
  <p>This page uses frames.</p>
</body>
</noframes>
</html>

left.html

<html>
<head><title>Left</title></head>
<body>

<p><a href="welcome.html"><img border="0" src="images/sweethome.gif" width="120" height="25"></a></p>
<p><a href="ps.html"><img border="0" src="images/property.gif" width="120" height="25"></a></p>
<p><a href="cr.html"><img border="0" src="images/customer.gif" width="120" height="25"></a></p>
<p><a href="broker.html"><img border="0" src="images/broker.gif" width="120" height="25"></a></p>
<p><a href="app.html"><img border="0" src="images/appt.gif" width="120" height="25"></a></p>
<p><a href="office.html"><img border="0" src="images/office.gif" width="120" height="25"></a></p>
<p><a href="buy.html"><img border="0" src="images/buy.gif" width="120" height="25"></a></p>
<p><a href="sell.html"><img border="0" src="images/sell.gif" width="120" height="25"></a></p>
</body>
</html>

top.html

<html>
<head><title>TOP</title></head>
<body>
<base target="contents">
</body>
</html>
<body>

<p><img src="images/houses_animation.gif" align="left"></p>
<p><img src="images/sweethome.gif" width="500" height="62" align="center"></p>
<p><img src="images/logo_5.gif" align="center"></p>
</body>
</html>

welcome.html

<html>
<head><title>Sweethome.com Homepage</title></head>
<body>
<p><img src="images/dream-bubble.jpg" width="300" height="205" align="Left">
<br></p>
<p>Where is my dream home?</p>
<p>Sweethome.com helps me to get my dream home.</p>
<table cellpadding="5">
<caption>Dream home come true!</caption>
<tr><td><img src="images/house1.jpg"></td></tr>
<tr><td><img src="images/house2.jpg"></td></tr>
<tr><td><img src="images/house4.jpg"></td></tr>
<tr><td><img src="images/house3.jpg"></td></tr>
<tr><td><img src="images/house5.jpg"></td></tr>
<tr><td><img src="images/house6.jpg"></td></tr>
</table>
<p><img src="images/list_link.gif" width="600" height="68" border="0" align="Middle"></p>
</body></html>
<html>
<title>CR</title>
<body>
<H2>CUSTOMER REGISTRATION</H2>
<form method="post" action="/servlet/crServlet">
<table align=left>
<tr><td>SSN:</td>
<td>
<input type="text" name="SSN1" size="3" -
<input type="text" name="SSN2" size="2" -
<input type="text" name="SSN3" size="4" />
</tr></td>
<tr><td>First Name:</td><td>
<input type="text" size=40 name=FIRST /></td></tr>
<tr><td>Middle Name:</td><td>
<input type="text" size=20 name=MID>(optional)</td></tr>
<tr><td>Last Name:</td><td>
<input type="text" size=40 name=LAST /></td></tr>
<tr><td>Street:</td><td>
<input type="text" size=60 name=STREET /></td></tr>
<tr><td>City:</td><td>
<input type="text" size=60 name=CITY /></td></tr>
<tr><td>State:</td><td>
<select name=STATE>
<option value="" selected> </option>
<option value=AK >Alaska
<option value=AL >Alabama
<option value=AR >Arkansas
<option value=AZ >Arizona
<option value=CA >California
<option value=CO >Colorado
<option value=CT >Connecticut
<option value=DE >Delaware
<option value=FL >Florida
<option value=GA >Georgia
<option value=HI >Hawaii
<option value=IA >Iowa
<option value=ID >Idaho
<option value=IL >Illinois
<option value=IN >Indiana
<option value=KS >Kansas
<option value=KY >Kentucky
<option value=LA >Louisiana
<option value=MA >Massachusetts
<option value=MD >Maryland
<option value=ME >Maine
<option value=MJ >Michigan
<option value=MN >Minnesota
<option value=MO >Missouri
<option value=MS >Mississippi
<option value=MT >Montana
<option value=NC >North Carolina
<option value=ND >North Dakota
<option value=NE >Nebraska
<option value=NH >New Hampshire
</select>
</td></tr>
</table>
</form>
</body>
</html>
<option value=NJ>New Jersey</option>
<option value=NM>New Mexico</option>
<option value=NV>Nevada</option>
<option value=NY>New York</option>
<option value=OH>Ohio</option>
<option value=OK>Oklahoma</option>
<option value=OR>Oregon</option>
<option value=PA>Pennsylvania</option>
<option value=RI>Rhode Island</option>
<option value=SC>South Carolina</option>
<option value=SD>South Dakota</option>
<option value=TN>Tennessee</option>
<option value=TX>Texas</option>
<option value=UT>Utah</option>
<option value=VA>Virginia</option>
<option value=VT>Vermont</option>
<option value=WA>Washington</option>
<option value=WI>Wisconsin</option>
<option value=WV>West Virginia</option>
<option value=WA>Washington</option>
<option value=WY>Wyoming</option>

</select>
</td></tr>
<tr><td>Zip Code:</td><td><input type=text size=20 name=ZIP></td></tr>
<tr><td>Home Number:</td><td><input type=text name=HOME1 size=3></td>
<input type=text name=HOME2 size=3></td>
<input type=text name=HOME3 size=4></tr>
<tr><td>Work Number:</td><td><input type=text name=WORK1 size=3></td>
<input type=text name=WORK2 size=3></td>
<input type=text name=WORK3 size=4> (optional)</tr>
<tr align=center><td colspan=2>
<input type=submit name=SEND value=Submit>
<input type=reset value=Reset></td></tr>
</table>
</form>
</body>
</html>

ps.html

<html>
<head><title>Property Search</title></head>
<body>
<form method="post" action="/servlet/psCheck">

</body>
</html>
<h2>Property Search</h2>

<table>
<thead>
<tr>
<th>Option 1</th>
<th>Option 2</th>
<th>Option 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>City</td>
<td>Zip Code</td>
<td>Property ID</td>
</tr>
</tbody>
</table>

**Location for Search**

**Additional Information**

- **Property Type:**
  - None
  - Single family house
  - Town house
  - Condo

- **Number of Bedroom(s):**
  - None
  - 1
  - 2
  - 3
  - 4 and above

- **Number of Bathroom(s):**
  - None
<table>
  <tr>
    <th align=left>Property built year:</th>
    <td>
      <select size="1.5" name="builtyear">
        <option selected>None</option>
        <option>1995 - Current</option>
        <option>1990 - 1994</option>
        <option>1985 - 1989</option>
        <option>1980 - 1984</option>
        <option>1970 - 1979</option>
        <option>Before 1970</option>
      </select>
    </td>
  </tr>
  <tr>
    <th align=left>Property price (US dollars):</th>
    <td>
      <select size="1.5" name="price">
        <option selected>None</option>
        <option>Below $200,000</option>
        <option>$200,000 - $299,999</option>
        <option>$300,000 - $399,999</option>
        <option>$400,000 - $499,999</option>
        <option>$500,000 - $599,999</option>
        <option>$600,000 - $699,999</option>
        <option>$700,000 - $799,999</option>
        <option>$800,000 and above</option>
      </select>
    </td>
  </tr>
  <tr>
    <th align=left>Property size (square feet):</th>
    <td>
      <select size="1.5" name="size">
        <option selected>None</option>
        <option>Below 1,000</option>
        <option>1,000 - 1,499</option>
        <option>1,500 - 1,999</option>
        <option>2,000 - 2,499</option>
        <option>2,500 - 2,999</option>
        <option>3,000 and above</option>
      </select>
    </td>
  </tr>
</table>

<input type="submit" value="Submit"/>
<input type="reset" value="Reset"/>
office.html

<html>
<head><title>Office Search</title></head>
<body>
<form method="POST" action="branch.jsp">
<table>
<tr>
<td>[ <a href="welcome.html">Home</a> ]</td>
<td>[ <a href="mailto:webmaster@www.sweethome.com">Contact us</a> ]</td>
</tr>
</table>
<form method="POST" action="branch.jsp">
<p><input type="radio" value="Bl" checked name="Rl"> San Bernardino Office</p>
<p><input type="radio" name="Rl" value="B2"> Los Angeles Office</p>
<p><input type="radio" name="Rl" value="B3"> Hollywood Office</p>
<p><input type="radio" name="Rl" value="B4"> San Francisco Office</p>
<p><input type="radio" name="Rl" value="B5"> Ontario Office</p>
<p><input type="radio" name="Rl" value="B6"> Riverside Office</p>
<p><input type="submit" value="Submit"> <input type="reset" value="Reset"></p>
</form>
</body>
</html>

app.html

<html>
<head><title>MAKE APPOINTMENT</title></head>
<body>
<table>
<tr><td><img src="images/handshake.jpg" width=75 height=100></td>
<td valign=bottom><H2>SET UP APPOINTMENT</H2></td></tr>
<form method="POST" action="/servlet/makeAppServlet">
<table>
<tr><td width=100>Broker Id:</td><td><input type="text" name="BID" size="20"></td></tr>
<tr><td width=100>Customer id:</td><td><input type="text" name="CID" size="20"></td></tr>
<tr><td>Date:</td><td>mm <SELECT NAME="MONTH"> 
OPTION VALUE="1">Jan</OPTION>
OPTION VALUE="2">Feb</OPTION>
</table>
</form>
</body>
</html>
buy.html

<html>
<head><title>Property Buyer Guide</title></head>
<body>

<p><img src="images/handshake.jpg" align="left"><h2>Property Buyer Guide</h2></p>
<p>hello</p>
<p>If you already know a broker and you would like to make an appointment with him/her</p>
<p>Find a property matching you need</p>
<p>If you don't know a broker, and you would like to search one by choosing an office near you</p>
<p>If you don't know a broker, and you would like to search one by broker's qualification</p>
</body>
</html>

sell.html

<html>
<head><title>Property Owner Guide</title></head>
<body>
</body>
</html>
Welcome! Choose one of the Following:

- **If you already know a broker and you would like to make an appointment with him/her**
- **If you don't know any broker, and you would like to search one by choosing an office near you**
- **If you don't know any broker and you would like to search one by broker's qualifications**

### forgetid.html

```html
<html>
<head><title>Forget ID</title></head>
<body>
<br>
<form method="post" action="./servlet/forgetidServlet">
Answer the following questions to retrieve your ID:
<p>
<table>
<tr><td>SSN: </td><td><input type="text" name="SSN1" size="3"> - 
<input type="text" name="SSN2" size="2"> - 
<input type="text" name="SSN3" size="4"></td></tr>
<tr><td>Zip Code: </td><td><input type="text" name="ZIP" size="5"></td></tr>
</table>
<br>
<input type="submit" value="Submit">  
<input type="reset" value="Reset"></form>
</body>
</html>
```
APPENDIX B

SOURCE CODE: JAVASERVER PAGE

PART
branch.jsp

<jsp:useBean id="officesearch" scope="session" class="sweethome.office"/>
<html>
<head><title>Office Search</title></head>

<%String branch = request.getParameter("R1");
officesearch.setbranchno(branch);
%>
<body>
<h2>Office Search</h2>
<table>
<tr>
    <td>[ <a href="welcome.html">Home</a> ]</td>
    <td>[ <a href="mailto:webmaster@www.sweethome.com">Contact us</a> ]</td>
</tr>
</table>
<br size=5>
<p><b>%=officesearch.information()[3]%> Office</b></p>
<table>
<tr>
    <td><b>Address:</b></td>
    <td><%=officesearch.information()[2]%>
        <%=officesearch.information()[3]%>,<%=officesearch.information()[4]%>,<%=officesearch.information()[5]%></td>
</tr>
<tr>
    <td><b>Phone:</b></td>
    <td>(<%=officesearch.information()[0].substring(0,3)%>)<%=officesearch.information()[0].substring(3,10)%></td>
</tr>
<tr>
    <td><b>Fax:</b></td>
    <td>(<%=officesearch.information()[1].substring(0,3)%>)<%=officesearch.information()[1].substring(3,10)%></td>
</tr>
<tr>
    <td><b>Map:</b></td>
    <td><img src="images/<%=branch%>.gif"/></td>
</tr>
</table>
<b>Manager:</b><table>
<%java.util.Vector managers = officesearch.manager();
    int i =0;
    while(i<managers.size()){%>
<tr>
    <td><%=managers.elementAt(i)%>,
    <%=managers.elementAt(++i)%>
</tr>
</table>
Profile.jsp

<jsp:useBean id="officesearch" scope="session" class="sweethome.office"/>
<html>
<head><title>Office Profile</title></head>

<body>
<h2>Office Search</h2>
<table>
<tr>
<td>[<a href="welcome.html">Home</a> ]</td>
<td>[<a href="mailto:webmaster@www.sweethome.com">Contact us</a> ]</td>
</tr>
</table>
<hr size=5>
<p>&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;Office</p>
<p>&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;Sales Profile:</p>
<p>&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;#<%=officesearch.aggregate()[3]%> properties sold this year.</p>
<p>&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;Highest Property Sale Price: $<%=officesearch.aggregate()[1]%></p>
<p>&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;Lowest Property Sale Price: $<%=officesearch.aggregate()[2]%>
<p>&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;Average Sale Price: $<%=officesearch.aggregate()[0]%>
<p>&lt;a href="office.html"&gt;View other office&lt;/a&gt;&lt;/p&gt;
<p>&lt;a href="welcome.html"&gt;Go to sweetHome&lt;/a&gt;&lt;/p&gt;
</body>
</html>

broker.jsp

<jsp:useBean id="officesearch" scope="session" class="sweethome.office"/>
<html>
<head><title>Broker</title></head>

<body>
<h2>Office Search</h2>
<table>
<tr>
<td>[ <a href="welcome.html">Home</a> ]</td>
<td>[ <a href="mailto:webmaster@www.sweethome.com">Contact us</a> ]</td>
</tr>
</table>
<hr size=5>
<p>%String branch=officesearch.getbranchno();%><b>Office</b></p>
<form method="POST" action="brokerprofile.jsp">
<select size="10" name="BROKERID">
<option value=""> </option>
%java.util.Vector brokers = officesearch.brokers();%>
<option value=""></option>
%int i = 3; while(i<brokers.size()){ %>
<option value=""></option>
% }>
</select>
<p>To see the broker's profile, click- <input type="submit" value="See Profiles"></p>
</form>
</body>
</html>

brokerprofile.jsp

<jsp:useBean id="brokersearch" scope="session" class="sweethome.office"/>
<html>
%String brokerid = request.getParameter("BROKERID");%>
%brokersearch.setbrokerid(brokerid);%>
<head><title>Broker Profile</title></head>
<body>
<h2>Broker Profile</h2>
<table>
<tr>
<td>[ <a href="welcome.html">Home</a> ]</td>
<td>[ <a href="mailto:webmaster@www.sweethome.com">Contact us</a> ]</td>
</tr>
</table>
<hr size=5>
<form method="POST" action="/servlet/appServlet">
<b>Broker ID: </b><%=brokerid%><input type="hidden" name="BROKERID" value=""></form>
</body>
</html>
This broker sold <%=brokersearch.brokerprofile()[3]%> properties this year.
The average sale price is: $<%=brokersearch.brokerprofile()[0]%>
The highest sale price is: $<%=brokersearch.brokerprofile()[1]%>
The lowest sale price is: $<%=brokersearch.brokerprofile()[2]%>
No sale record in year 2000.

If you want to make an appointment with this broker, click: <input type="submit" value="Make an appointment">

View other office
Go to SweetHome
APPENDIX C

SOURCE CODE: SERVLET AND JAVABEAN PART
import java.beans.*;
import java.sql.*;
import java.io.*;

public class accessBean
{
    public accessBean(String db, String login, String pwd, String query)
    {
        database = db;
        user = login;
        password = pwd;
        sql = query;
        colArray = new String[getMaxColArrayLen()];
        numColumns = 0;
        fillStatus = false;
        changes = new PropertyChangeSupport(this);
    }

    public void execQuery()
    {
        String url = "jdbc:mysql://localhost:3306/" + database;
        String driver = "org.gjt.mm.mysql.Driver";
        Connection con = null;
        java.sql.Statement stmt = null;
        ResultSet rs = null;
        ResultSetMetaData rsmd = null;
        String result;
        int index = 0;
        int numCols = 0;
        String colValue = null;
        try
        {
            Class.forName(driver);
            con = DriverManager.getConnection(url, user, password);
            stmt = con.createStatement();
            rs = stmt.executeQuery(sql);
            rsmd = rs.getMetaData();
            numCols = rsmd.getColumnCount();
            setNumColumns(numCols);

            for(int i = 1; i <= numCols; i++)
            {
                colValue = new String(""");
                colValue = rsmd.getColumnLabel(i);
                setColArray(index, colValue);
                index++;
            }
            index = 0;
            while(rs.next())
            {
                for(int i=1; i<=numCols;i++)
                {
                    result = rs.getString(i);
                    setResultArray(index, result);
                    index++;
                }
            }
        }
    }
}
public void execUpdate()
{
    String url = "jdbc:mysql://localhost:3306/" + database;
    String driver = "org.gjt.mm.mysql.Driver";
    Connection con = null;
    java.sql.Statement stmt = null;
    int index = 0;
    int numCols = 0;
    String colValue = null;
    try
    {
        Class.forName(driver);
        con = DriverManager.getConnection(url, user, password);
        stmt = con.createStatement();
        stmt.executeUpdate(sql);
        setFillStatus(true);
        stmt.close();
        con.close();
    }
    catch(Exception e)
    {
        errMessage = e.getMessage();
        return;
    }
}

public synchronized boolean getFillStatus()
{
    return fillStatus;
}

public void setFillStatus(boolean newValue)
{
    boolean oldValue = fillStatus;
    fillStatus = newValue;
    Boolean oldObj = new Boolean(oldValue);
    Boolean newObj = new Boolean(newValue);
}
changes.firePropertyChange("fillStatus", oldObj, newObj);
}

public String getColArray(int index)
{
    return this.colArray[index];
}

public void setColArray(int index, String value)
{
    this.colArray[index] = value;
}

public String getResultArray(int index)
{
    return this.resultArray[index];
}

public void setResultArray(int index, String value)
{
    this.resultArray[index] = value;
}

public void setNumColumns(int n)
{
    numColumns = n;
}

public int getNumColumns()
{
    return numColumns;
}

public int getMaxResultArrayLen()
{
    return maxResultArrayLen;
}

public void setMaxResultArrayLen(int i)
{
    maxResultArrayLen = i;
}

public int getMaxColArrayLen()
{
    return maxColArrayLen;
}

public void setMaxColArrayLen(int i)
{
    maxColArrayLen = i;
}

public String getErrorMsg()
{
    return errorMessage;
}

// bean properties
private String database;
private String user;
private String password;
private String sql;
// bean properties associated with query results.
private String[] resultArray = null;
private String[] colArray = null;
private int maxResultArrayLen = 1000;
private int maxColArrayLen = 30;
private boolean fillStatus = false;
private int numColumns;
private String errMessage = null;

// used for firing events
private PropertyChangeSupport changes = null;

PropertyInfo.java

/* This class is used to stored information entered by users. Its life cycle is
 * transient as long as the session for that user exists.*/

public class PropertyInfo
{
    private String city = "";
    private String zip = "";
    private String propertyId = "";
    private String type = "";
    private String bedroom = "";
    private String bathroom = "";
    private String builtyear = "";
    private String price = "";
    private String size = "";
    private String errMsg = "";

    public void setCity(String city)
    {
        this.city = city;
    } 

    public String getCity()
    {
        return city;
    }

    public void setZip(String zip)
    {
        this.zip = zip;
    }

    public String getZip()
    {
        return zip;
    }

    public void setPropertyId(String id)
    {
        propertyId = id;
    }

    public String getPropertyId()
}


```java

public void setType(String type)
{
    this.type = type;
}

public String getType()
{
    return type;
}

public void setBedroom(String bedroom)
{
    this.bedroom = bedroom;
}

public String getBedroom()
{
    return bedroom;
}

public void setBathroom(String bathroom)
{
    this.bathroom = bathroom;
}

public String getBathroom()
{
    return bathroom;
}

public void setBuiltyear(String builtyear)
{
    this.builtyear = builtyear;
}

public String getBuiltyear()
{
    return builtyear;
}

public void setPrice(String price)
{
    this.price = price;
}

public String getPrice()
{
    return price;
}

public void setSize(String size)
{
    this.size = size;
}

public String getSize()
{
    return size;
}

public void setErrMsg(String msg)
{
```java
BrokerInfo.java

public class BrokerInfo {
    private String language = "";
    private String year = "";
    private String city = "";
    private String numofsales = "";

    public void setLanguage(String language) {
        this.language = language;
    }

    public String getLanguage() {
        return language;
    }

    public void setYear(String year) {
        this.year = year;
    }

    public String getYear() {
        return year;
    }

    public void setCity(String city) {
        this.city = city;
    }

    public String getCity() {
        return city;
    }

    // Other methods...
}
```
public void setNumofSales(String numofsales)
{
    this.numofsales = numofsales;
}

global String getNumofSales()
{
    return numofsales;
}

public void reset()
{
    language = "";
    year = "";
    city = "";
    numofsales = "";
}

office.java

package sweethome;
import java.io.*;
import java.lang.*;
import java.util.*;
public class office{
    private String branchno;
    private String brokerid;
    private String userid = "root";
    private String passwd = "123456";
    public accessBean aB = null;
    public office()
    {
    }
    public String[] information()
    {
        String sql = "SELECT TELNO,FAXNO,STREET,CITY,STATE,CODE FROM BRANCH WHERE BRANCHNO =""+branchno+"";"
        aB = new accessBean("sweethome", userid, passwd, sql);
        aB.execSQL();
        int numElements = aB.getMaxResultArrayLen();
        int currentElement = 0;
        String[] rs = new String[numElements];
        while(currentElement < numElements &&
            aB.getResultArray(currentElement) != null)
        {
            rs[currentElement] = aB.getResultArray(currentElement).trim();
            currentElement++;
        }
        return rs;
    }

    public Vector manager()
    {
        String sql = "select lname,fname,workno from branch,duration,employee where"
employee.branchno = branch.branchno and branch.branchno ="'+branchno+'";

aB = new accessBean("sweethome", userid, passwd, sql);
    aB.executeQuery();
    int numElements = aB.getMaxResultArrayLen();
    int currentElement = 0;
    Vector rs = new Vector();
    while(currentElement < numElements &&
        aB.getResultArray(currentElement) != null)
    {
      rs.addElement(aB.getResultArray(currentElement).trim());
      currentElement++;
    }
    return rs;
}

public int[] aggregate(){
    String sql="SELECT AVG(TRANSAMOUNT), MAX(TRANSAMOUNT),
                MIN(TRANSAMOUNT), COUNT(TRANSNO)
            FROM SALERECORD
            WHERE BRANCHNO ="'+bfanchno+'" and year(transdate)=2002 GROUP BY
            BRANCHNO";
    aB = new accessBean("sweethome", userid, passwd, sql);
    aB.executeQuery();
    int[] rs=new int[4];
    for(int i=0;i<4;i++)
    {
      rs[i] = Float.valueOf(aB.getResultArray(i).trim()).intValue();
    }
    return rs;
}

public Vector brokers(){
    String sql = "select brokerid,lname,fname from employee,broker where
    broker.ssn=employee.ssn and branchno="'+branchno+'";"
    aB = new accessBean("sweethome", userid, passwd, sql);
    aB.executeQuery();
    int numElements = aB.getMaxResultArrayLen();
    int currentElement = 0;
    Vector rs = new Vector();
    while(currentElement < numElements &&
        aB.getResultArray(currentElement) != null)
    {
      rs.addElement(aB.getResultArray(currentElement).trim());
      currentElement++;
    }
    return rs;
}

public boolean sale(){
    String sql="SELECT BROKERID FROM SALERECORD WHERE BROKERID
            ="'+brokerid+'" AND YEAR(TRANSDATE)=2002";
    aB = new accessBean("sweethome", userid, passwd, sql);
    aB.executeQuery();
    if(aB.getResultArray(0)!=null)
    return true;
    else
    return false;
public int[] brokerprofile()
{
    String sql = "SELECT AVG(TRANSAMOUNT), MAX(TRANSAMOUNT),
    MIN(TRANSAMOUNT), COUNT(TRANSNO) FROM SALERECORD
    WHERE BROKERID =""+brokerid+"" and year(transdate)=2002 GROUP BY
    BROKERID";
    aB = new accessBean("sweethome", userid, passwd, sql);
    aB.execSQL();
    int[] rs = new int[4];
    for(int i =0;i<4;i++){
      rs[i] = Float.valueOf(aB.getResultArray(i).trim()).intValue();
    }
    return rs;
}

public int brokerexp()
{
    String sql = "SELECT year(DATETIMEEXPSTART) FROM BROKER WHERE
    BROKERID =""+brokerid+""; 
    aB = new accessBean("sweethome", userid, passwd, sql);
    aB.execSQL();
    int rs = 2002- Float.valueOf(aB.getResultArray(0).trim()).intValue();
    return rs;
}

public void setbranchno(String bno) {
    branchno = bno;
}

public String getbranchno() {
    return branchno;
}

public void setbrokerid(String bid) {
    brokerid = bid;
}

public String getbrokerid() {
    return brokerid;
}

psCheck.java

/* This servlet checks user's submit information from ps.html. If information is valid,
* the request will be forwarded to psQuery servlet, if information is not valid,
* the request will be forwarded to psAgain servlet. */

import javax.servlet.*;
import java.io.*;

public class psCheck extends HttpServlet
{
    public void doPost(HttpServletRequest req, HttpServletResponse res)
    throws IOException, ServletException
    {
        // Get user's session and property info
        HttpSession session = req.getSession(true);
        PropertyInfo pInfo = (PropertyInfo) session.getValue(session.getId());
        // If the user has no property info object, create a new one
        if (pInfo == null)
        {
            pInfo = new PropertyInfo();
            session.putValue(session.getId(), pInfo);
        }
        // Gather property search information
        String city = req.getParameter("city");
        String zip = req.getParameter("zip");
        String id = req.getParameter("id");
        String house = req.getParameter("house");
        String bedroom = req.getParameter("bedroom");
        String bathroom = req.getParameter("bathroom");
        String builtYear = req.getParameter("builtYear");
        String price = req.getParameter("price");
        String size = req.getParameter("size");
        city = city.trim();
        zip = zip.trim();
        id = id.trim();
        // Error checking
        if (city.equals("") && zip.equals("") && id.equals("")
        {
            pInfo.setErrMsg("Error: You have to specify a location for search or" + " a valid property id.");
            RequestDispatcher dispatcher =
                getServletContext().getRequestDispatcher("/servlet/psAgain");
            if (dispatcher == null)
            {
                // No dispatcher means the html file can not be delivered
                res.sendError(res.SC_NO_CONTENT);
            }
            dispatcher.forward(req, res);
        }
        else
        {
            // Information OK. Set information in and send to psOutput servlet
            pInfo.setCity(city);
            pInfo.setZip(zip);
            pInfo.setPropertyId(id);
            pInfo.setType(house);
            pInfo.setBedroom(bedroom);
            pInfo.setBathroom(bathroom);
            pInfo.setBuiltYear(builtYear);
            pInfo.setPrice(price);
            pInfo.setSize(size);
            RequestDispatcher dispatcher =
                getServletContext().getRequestDispatcher("/servlet/psQuery");
        }
    }
}
if (dispatcher == null)
{
    // No dispatcher means the html file can not be delivered
    res.sendError(res.SC_NO_CONTENT);
}
dispatcher.forward(req, res);
}

public String getServletInfo()
{
    return "A servlet that checks user's input information and forwards request"
+ "to appropriate servlets";
}

psQuery.java

/* This servlet displays property information based on property information user entered
* in ps.html. */

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

class psQuery extends HttpServlet
{
    private String url;
    private String driver;
    private accessBean aBean;
    private String database;
    private String login;
    private String password;

    public void init()
    {
        database = "sweethome";
        login = "root";
        password = "123456";
    }

    public void doPost(HttpServletRequest req, HttpServletResponse res)
        throws IOException, ServletException
    {
        String type = "";
        String city = "";
        String zip = "";
        String id = "";
        String bedroom = "";
        String bathroom = "";
        String builtyear = "";
        String price = "";
        String size = "";

        // Get user's session and DB info
        HttpSession session = req.getSession(true);
        PropertyInfo plInfo = (PropertyInfo)session.getValue(session.getAttribute());
if(pInfo == null)
{
    // If the user has no property information object,
    // forward to an error
    RequestDispatcher dispatcher =
        getServletContext().getRequestDispatcher("/servlet/psErrSessionPage");
    if (dispatcher == null)
    {
        // No dispatcher means the html file can not be delivered
        res.sendError(res.SC_NO_CONTENT);
    }
    dispatcher.forward(req, res);
}
else
{
    // Get user input property information
    type = pInfo.getType();
    city = pInfo.getCity();
    zip = pInfo.getZip();
    id = pInfo.getPropertyId();
    bedroom = pInfo.getBedroom();
    bathroom = pInfo.getBathroom();
    builtyear = pInfo.getZibiltyear();
    price = pInfo.getPrice();
    size = pInfo.getSize();
}

PrintWriter out = res.getWriter();
res.setContentType("text/html");
out.println("<html>");
out.println("<head><title>Property Search Result</title></head>");
out.println("<body>");
out.println("<form method="post" action="/appServlet">");
out.println("<h2>Property Search Results</h2>");
out.println("<table>");
out.println("<tr>");
out.println("<td>[<a href="/kevin-p/welcome.html">Home</a>]</td>");
out.println("<td>[<a href=mailto:webmaster@sweethome.com">Contact us</a> + ]</td>");
out.println("</tr></table>");
out.println("<hr size=5>");
out.println("<br>");
out.println("<table width=80%>");
String where = "where ";
if(!city.equals(""))
    where += "lower(p.city) = lower(" + city + ") AND ";
if(!zip.equals(""))
    where += "p.code = " + zip + " AND ";
if(!id.equals(""))
    where += "lower(p.propertyid) = lower(" + id + ") AND ";
if(type.equalsIgnoreCase("Single family house"))
where += "p.type = 'Single family house' AND ";
else if(type.equals("Town house"))
    where += "p.type = 'Town house' AND ";
else if(type.equals("Condo"))
    where += "p.type = 'Condo' AND ";

if(bedroom.equals("1"))
    where += "p.noroom = 1 AND ";
else if(bedroom.equals("2"))
    where += "p.noroom = 2 AND ";
else if(bedroom.equals("3"))
    where += "p.noroom = 3 AND ";
else if(bedroom.equalsIgnoreCase("4 and above"))
    where += "p.noroom >= 4 AND ";

if(bathroom.equals("1"))
    where += "p.nobath = 1 AND ";
else if(bathroom.equals("2"))
    where += "p.nobath = 2 AND ";
else if(bathroom.equalsIgnoreCase("3 and above"))
    where += "p.nobath >= 3 AND ";

if(builtyear.equalsIgnoreCase("1995 - Current"))
    where += "p.builtyear >= 1995 AND ";
else if(builtyear.equalsIgnoreCase("1990 - 1994"))
    where += "p.builtyear >= 1990 AND p.builtyear <= 1994 AND ";
else if(builtyear.equalsIgnoreCase("1985 - 1989"))
    where += "p.builtyear >= 1985 AND p.builtyear <= 1989 AND ";
else if(builtyear.equalsIgnoreCase("1980 - 1984"))
    where += "p.builtyear >= 1980 AND p.builtyear <= 1984 AND ";
else if(builtyear.equalsIgnoreCase("1970 - 1979"))
    where += "p.builtyear >= 1970 AND p.builtyear <= 1979 AND ";
else if(builtyear.equalsIgnoreCase("Before 1970"))
    where += "p.builtyear <= 1970 AND ";

if(price.equalsIgnoreCase("Below $200,000"))
    where += "p.price < 200000 AND ";
else if(price.equalsIgnoreCase("$200,000 - $299,999"))
    where += "p.price >= 200000 AND p.price <= 299999 AND ";
else if(price.equalsIgnoreCase("$300,000 - $399,999"))
    where += "p.price >= 300000 AND p.price <= 399999 AND ";
else if(price.equalsIgnoreCase("$400,000 - $499,999"))
    where += "p.price >= 400000 AND p.price <= 499999 AND ";
else if(price.equalsIgnoreCase("$500,000 - $599,999"))
    where += "p.price >= 500000 AND p.price <= 599999 AND ";
else if(price.equalsIgnoreCase("$600,000 - $699,999"))
    where += "p.price >= 600000 AND p.price <= 699999 AND ";
else if(price.equalsIgnoreCase("$700,000 - $799,999"))
    where += "p.price >= 700000 AND p.price <= 799999 AND ";
else if(price.equalsIgnoreCase("$800,000 and above"))
    where += "p.price >= 800000 AND ";
```java
if(size.equalsIgnoreCase("Below 1,000"))
    where += "p.squareft <= 1000 AND ";
else if(size.equalsIgnoreCase("1,000 - 1,499"))
    where += "p.squareft >= 1000 AND p.squareft <= 1499 AND ";
else if(size.equalsIgnoreCase("1,500 - 1,999"))
    where += "p.squareft >= 1500 AND p.squareft <= 1999 AND ";
else if(size.equalsIgnoreCase("2,000 - 2,499"))
    where += "p.squareft >= 2000 AND p.squareft <= 2499 AND ";
else if(size.equalsIgnoreCase("2,500 - 2,999"))
    where += "p.squareft >= 2500 AND p.squareft <= 2999 AND ";
else if(size.equalsIgnoreCase("3,000 and above"))
    where += "p.squareft >= 3000 AND ";

    + " FROM property p, propertyowned po, owner o " + where
    + " p.propertyid=po.propertyid AND po.currentowned='Y' "
    + " AND po.ssn=o.ssn ";
aBean = new accessBean(database, login, password, sql);
aBean.execQuery();

if(!aBean.getFillStatus())
{
    out.println("<p>No matched properties have been found</p><br>&nbsp;");
    out.println("<a href="/kevin-p/ps.html">Go to Property Search</a></br>";
    out.println("<a href="/kevin-p/welcome.html">Go to Sweethome</a>");
}
else
{
    int numCol = aBean.getNumColumns();
    String propertyId;
    String street;
    String state;
    String code;
    String noRoom;
    String noBath;
    String squareFt;
    String builtYear;
    String brokerId;
    String result;
    int index=0;

    if(aBean.getResultArray(index)!=null)
    {
        while((propertyId=aBean.getResultArray(index++))!=null)
        {
            type = aBean.getResultArray(index++);
            street = aBean.getResultArray(index++);
            city = aBean.getResultArray(index++);
            state = aBean.getResultArray(index++);
            code = aBean.getResultArray(index++);
            noRoom = aBean.getResultArray(index++);
```
noBath = aBean.getResultArray(index++);  
squareFt = aBean.getResultArray(index++);  
builtYear = aBean.getResultArray(index++);  
price = aBean.getResultArray(index++);  
brokerId = aBean.getResultArray(index++);  
out.println("<tr><td><input type=radio  
name="BROKERID" value=""  
+ brokerId + ">");  
out.println("<td align=left>");  
out.println("Property ID:");  
out.println("<th align=left>");  
out.println(propertyId);  
out.println("<td><tr><td><th align=left>");  
out.println("Address:");  
out.println("</th><td align=left>");  
out.println(street + " , " + city + " , " + state + " " + code);  
out.println("</td><tr><td><th align=left>");  
out.println("Type:");  
out.println("<th align=left>");  
out.println(type);  
out.println("<br></tr><tr><td><th align=left>");  
out.println("No. of bedroom(s):");  
out.println("<th align=left>");  
out.println(noRoom);  
out.println("<td><tr><td><th align=left>");  
out.println("No. of bathroom(s):");  
out.println("<th align=left>");  
out.println(noBath);  
out.println("<td><tr><td><th align=left>");  
out.println("Size:");  
out.println("<th align=left>");  
out.println(squareFt + " square feet");  
out.println("<td><tr><td><th align=left>");  
out.println("Built year:");  
out.println("<th align=left>");  
out.println(builtYear);  
out.println("<td><tr><td><th align=left>");  
out.println("Price:");  
out.println("<th align=left>");  
out.println("$" + price + " USD");  
out.println("<td><tr><td><th align=left>");  
out.println("Broker ID:");  
out.println("<th align=left>");  
out.println(brokerId);  
out.println("<td><tr><tr><tr></tr>");  
  
}  
out.println("</table>\n"));  
else  
{  
  out.println("<p>No matched records found</p>\n"));
If you are not registered with us previously, please go to Customer Registration before you make an appointment with our broker(s).

If you are interested in one of the listed properties, select the property and click on Make an appointment button to make an appointment with our broker.

Make an appointment

getServletInfo()

getServletInfo()

psAgain.java

This servlet displays error messages and re-prompt users for property information.

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

public class psAgain extends HttpServlet
{
    public void doPost(HttpServletRequest req, HttpServletResponse res)
    throws IOException, ServletException
    {
        res.setContentType("text/html");
        PrintWriter out = res.getWriter();
        out.println("<html>");
        out.println("<head><title>Property Search</title></head>");
        out.println("<body>");
        // Get user's session and property info
        HttpSession session = req.getSession(true);
        PropertyInfo plnfo = (PropertyInfo)session.getValue(session.getId());
        out.println("<h2>Re-Enter Property Information</h2>");
        out.println("<table>");
        out.println("<tr>");
        out.println("<td>[ <a href="/welcome.html">Home</a> ]</td>");
        out.println("<td>[ <a href="mailto:webmaster@sweethome.com">Contact us</a> ]</td>");
        out.println("</tr></table>");
        out.println("<hr size=5>");
        out.println("<br>");
    }
}
// If the user has no property info object, displays error message
if (pInfo == null)
{
    out.println("<p>Error on session. Please re-enter property information</p>");
}
else
{
    out.println("<p>" + pInfo.getErrMsg() + "</p>");
}
out.println("<form method="post" action="/psCheck">");
out.println("<h2>Location for Search</h2>");
out.println("<table>");
out.println("<tr>");
out.println("<th align=left>(Option 1)</th>");
out.println("<th align=left>Enter a city:</th>");
out.println("<td><input type="text" name="city" size="20"></td>");
out.println("</tr>");
out.println("<th align=left>(Option 2)</th>");
out.println("<th align=left>Enter a zip code:</th>");
out.println("<td><input type="text" name="zip" size="20"></td>");
out.println("</tr>");
out.println("<th align=left>(Option 3)</th>");
out.println("<th align=left>Property id (if known):</th>");
out.println("<td><input type="text" name="id" size="20"></td>"');
out.println("</tr>");
out.println("</table>");
out.println("<br>");
out.println("<h2>Additional Information</h2>");
out.println("<table border=1>"');
out.println("<tr><th align=left>Property type:</th>");
out.println("<td><select size=1.5 name="house"><option selected>None</option>"');
String type = pInfo.getType();
if(type.equals("None"))
    out.println("<option selected>None</option>");
else
    out.println("<option>None</option>"');
if(type.equals("Single family house"))
    out.println("<option selected>Single family house</option>");
else
    out.println("<option>Single family house</option>");
if(type.equals("Town house"))
    out.println("<option selected>Town house</option>");
else
    out.println("<option>Town house</option>");
if(type.equals("Condo"))

110
out.println("<option selected>Condo</option>");
else
out.println("<option>Condo</option>");

out.println("</select></td>");
out.println("</tr></tr>");
out.println("<th align=left>Number of bedroom(s):</th>");
out.println("<td><select size="1.5" name="bedroom">")
String bedroom = pInfo.getBedroom();
if(bedroom.equals("None"))
    out.println("<option selected>None</option>");
else
    out.println("<option>None</option>");
if(bedroom.equals("1"))
    out.println("<option selected>1</option>");
else
    out.println("<option>1</option>");
if(bedroom.equals("2"))
    out.println("<option selected>2</option>");
else
    out.println("<option>2</option>");
if(bedroom.equals("3"))
    out.println("<option selected>3</option>");
else
    out.println("<option>3</option>");
if(bedroom.equals("4 and above"))
    out.println("<option selected>4 and above</option>");
else
    out.println("<option>4 and above</option>");
out.println("</select></td>");
out.println("</tr></tr>");
out.println("<th align=left>Number of bathroom(s):</th>");
out.println("<td><select size="1.5" name="bathroom">")
String bathroom = pInfo.getBathroom();
if(bathroom.equals("None"))
    out.println("<option selected>None</option>");
else
    out.println("<option>None</option>");
if(bathroom.equals("1"))
    out.println("<option selected>1</option>");
else
    out.println("<option>1</option>");
if (bathroom.equals("2")
    out.println("<option selected>2</option>");
else
    out.println("<option>2</option>");
if (bathroom.equals("3 and above"))
    out.println("<option selected>3 and above</option>");
else
    out.println("<option>3 and above</option>");
out.println("<select></select>");
out.println("<tr><td>");
out.println("<th align=left>built year of the property (years):</th>");
out.println("<td><select size="1.5" name="builtyear">
    String builtyear = plinfo.getBuiltYear();
if (builtyear.equals("None"))
    out.println("<option selected>None</option>");
else
    out.println("<option>None</option>");
if (builtyear.equals("1995 - Current"))
    out.println("<option selected>1995 - Current</option>");
else
    out.println("<option>1995 - Current</option>");
if (builtyear.equals("1990 - 1994"))
    out.println("<option selected>1990 - 1994</option>");
else
    out.println("<option>1990 - 1994</option>");
if (builtyear.equals("1985 - 1989"))
    out.println("<option selected>1985 - 1989</option>");
else
    out.println("<option>1985 - 1989</option>");
if (builtyear.equals("1980 - 1984"))
    out.println("<option selected>1980 - 1984</option>");
else
    out.println("<option>1980 - 1984</option>");
if (builtyear.equals("1970 - 1979"))
    out.println("<option selected>1970 - 1979</option>");
else
    out.println("<option>1970 - 1979</option>");
if (builtyear.equals("Before 1970"))
    out.println("<option selected>Before 1970</option>");
else
    out.println("<option>Before 1970</option>");
out.println("<select></select>");
String price = pInfo.getPrice();

if(price.equals("None"))
    out.println("<option selected>None</option>");
else
    out.println("<option>None</option>");

if(price.equals("Below $200,000"))
    out.println("<option selected>Below $200,000</option>");
else
    out.println("<option>Below $200,000</option>");

if(price.equals("$200,000 - $299,999"))
    out.println("<option selected>$200,000 - $299,999</option>");
else
    out.println("<option>$200,000 - $299,999</option>");

if(price.equals("$300,000 - $399,999"))
    out.println("<option selected>$300,000 - $399,999</option>");
else
    out.println("<option>$300,000 - $399,999</option>");

if(price.equals("$400,000 - $499,999"))
    out.println("<option selected>$400,000 - $499,999</option>");
else
    out.println("<option>$400,000 - $499,999</option>");

if(price.equals("$500,000 - $599,999"))
    out.println("<option selected>$500,000 - $599,999</option>");
else
    out.println("<option>$500,000 - $599,999</option>");

if(price.equals("$600,000 - $699,999"))
    out.println("<option selected>$600,000 - $699,999</option>");
else
    out.println("<option>$600,000 - $699,999</option>");

if(price.equals("$700,000 - $799,999"))
    out.println("<option selected>$700,000 - $799,999</option>");
else
    out.println("<option>$700,000 - $799,999</option>");

if(price.equals("$800,000 and above"))
    out.println("<option selected>$800,000 and above</option>");
else
    out.println("<option>$800,000 and above</option>");
out.println("</select></td>");
out.println("</tr></tr>");
out.println("<th align=left>Property size (square feet):</th>");
out.println("<td><select size="1.5" name="size"> String size = pInfo.getSize();
if(size.equals("None"))
    out.println("<option selected>None</option>");
else
    out.println("<option>None</option>");
if(size.equals("Below 1,000"))
    out.println("<option selected>Below 1,000</option>");
else
    out.println("<option>Below 1,000</option>");
if(size.equals("1,000 - 1,499"))
    out.println("<option selected>1,000 - 1,499</option>");
else
    out.println("<option>1,000 - 1,499</option>");
if(size.equals("1,500 - 1,999"))
    out.println("<option selected>1,500 - 1,999</option>");
else
    out.println("<option>1,500 - 1,999</option>");
if(size.equals("2,000 - 2,499"))
    out.println("<option selected>2,000 - 2,499</option>");
else
    out.println("<option>2,000 - 2,499</option>");
if(size.equals("2,500 - 2,999"))
    out.println("<option selected>2,500 - 2,999</option>");
else
    out.println("<option>2,500 - 2,999</option>");
if(size.equals("3,000 and above"))
    out.println("<option selected>3,000 and above</option>");
else
    out.println("<option>3,000 and above</option>");
out.println("</select></td>");
out.println("</tr>");
out.println("</table>");
out.println("<br>&nbsp;");
out.println("</table>");
out.println("<input type="submit" value="Submit">");
out.println("</td></td>");
out.println("<input type="reset" value="Reset">");
public String getServletInfo()
{
    return "A servlet that re-prompt user for property information";
}
}

bsCheck.java

/* This servlet checks user's submit information from broker.html. If information is valid, */
/* the request will be forwarded to bsQuery servlet. */

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

public class bsCheck extends HttpServlet
{
    public void doPost(HttpServletRequest req, HttpServletResponse res)
        throws IOException, ServletException
    {
        // Get user's session and property info
        HttpSession session = req.getSession(true);
        BrokerInfo bInfo = (BrokerInfo)session.getValue(session.getId());

        // If the user has no broker info object, create a new one
        if(bInfo == null)
        {
            bInfo = new BrokerInfo();
            session.putValue(session.getId(), bInfo);
        }

        // Gather property search information
        String language = req.getParameter("language");
        String year = req.getParameter("year");
        String city = req.getParameter("city");
        String numofsales = req.getParameter("numofsales");

        // Error checking
        // Information OK. Set information in and send to psOutput servlet
        bInfo.setLanguage(language);
        bInfo.setYear(year);
        bInfo.setCity(city);
        bInfo.setNumofSales(numofsales);

        RequestDispatcher dispatcher = getServletContext().getRequestDispatcher("/servlet/bsQuery");

        if (dispatcher == null)
        {
            // No dispatcher means the html file can not be delivered
            res.sendError(res.SC_NO_CONTENT);
        }
        dispatcher.forward(req, res);
    }
}

public String getServletInfo()
{ return "A servlet that checks user's input information and forwards request" + "to appropriate servlets";
}

bsQuery.java

/* The list of the broker search result. */

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

public class bsQuery extends HttpServlet {
    private String url;
    private String driver;
    private accessBean aBean;
    private String database;
    private String login;
    private String password;

    public void init()
    { database = "sweethome";
      login = "root";
      password = "123456";
    }

    public void doPost(HttpServletRequest req, HttpServletResponse res)
    throws IOException, ServletException
    {
        String language = "";
        String year = "";
        String city = "";
        String numofsales = "";

        // Get user's session and DB info
        HttpSession session = req.getSession(true);
        BrokerInfo bInfo = (BrokerInfo)session.getValue(session.getId());
        PrintWriter out = res.getWriter();
        res.setContentType("text/html");
        out.println("<html>");
        out.println(<headxtitle>Broker Search Result</titlex/head>);
        out.println(<body>);
        out.println(<form method="post" action="/servlet/appServlet">);
        out.println(<h2>Broker Search Results</h2>);
        out.println(<table>);
        out.println(<tr>[
            <a href='../welcome.htmr>Home</a>[
        </td>[
    out.println(<tr>[
        <a href="mailto:webmaster@sweethome.com">Contact us</a>
    </td>[
        <hr size=5>[
        <br>&nbsp;
    </table>[

    out.println("<html>[
    out.println("<headxtitle>Broker Search Result</titlex/head>[
    out.println(<body>[
    out.println(<form method="post" action="/servlet/appServlet">[
    out.println(<h2>Broker Search Results</h2>[
    out.println(<table>[
    out.println(<tr>[
        <a href='../welcome.htmr>Home</a>[
    </td>[
    out.println(<form method="post" action="/servlet/appServlet">[
    out.println(<h2>Broker Search Results</h2>[
    out.println(<table>[
    out.println(<tr>[
        <a href="mailto:webmaster@sweethome.com">Contact us</a>
    </td>[
        <hr size=5>[
        <br>&nbsp;
    </table>[

    return "A servlet that checks user's input information and forwards request" + "to appropriate servlets";
}
}
out.println("<table width=60\%>");
language1 = bInfo.getLanguage();
year = bInfo.getYear();
city = bInfo.getCity();
umofsales = bInfo.getNumofSales();
String where = "where";

//get broker language
where += " b.brokerid in (select brokerid from brokerlanguage where language= " + language1 + ")";

//get year of experience
if(year.equals("0-10"))
where += " and b.brokerid in (select brokerid from broker b, employee e" + " where b.ssn=e.ssn AND ((to_days(now())-to_days(yearexpstart))/365)<11" + " AND ((to_days(now())-to_days(yearexpstart))/365)>=0")";
else if(year.equals("11-20"))
where += " and b.brokerid in (select brokerid from broker b, employee e" + " where b.ssn=e.ssn AND ((to_days(now())-to_days(yearexpstart))/365)<21" + " AND ((to_days(now())-to_days(yearexpstart))/365)>=11")";
else if(year.equals("21 and above"))
where += " and b.brokerid in (select brokerid from broker b, employee e" + " where b.ssn=e.ssn AND ((to_days(now())-to_days(yearexpstart))/365)>21")";

//get location of office
where += " and b.brokerid in (select brokerid from broker b, employee e, branch r" + " where b.ssn=e.ssn and e.branchno=r.branchno and r.city= " + city + ") ";

//get # of sales
if(numofsales.equals("0-10"))
where += " and b.brokerid in (select brokerid from salerecord group by brokerid" + " having count(*)>=0 AND count(*)<=10")";
else if(numofsales.equals("11 and above"))
where += " and b.brokerid in (select brokerid from salerecord group by brokerid" + " having count(*)>=11")";

//Query for broker information
String sql = "SELECT b.brokerid, e.lname, e.fname "+ " FROM broker b, employee e "+ Where + " AND b.ssn = e.ssn";
aBean = new accessBean(database, login, password, sql);
aBean.execQuery();
String brokerId;
String Iname;
String fname;
int index=0;
while((brokerId=aBean.getResultArray(index++))!=null)
{Iname = aBean.getResultArray(index++);
fname = aBean.getResultArray(index++);
}
```java
public String getServletInfo()
{
    return "A servlet that perform property search on the sweethome database";
}

public class crServlet extends HttpServlet
{
    private accessBean aB = null;
    public PropertyChangeAdapter adapter = new PropertyChangeAdapter();
    public boolean queryStatus = false;
    //private boolean flag = true;
    private String userid = "root";
    private String passwd = "123456";

    public void doPost(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException
    {
        res.setContentType("text/html");
        ServletOutputStream out = res.getOutputStream();
        out.println("<html>");
        out.println("<head><title>Customer Registration Page</title></head>");
        out.println("<body>");
    }
}
```

```
crServlet.java

import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;
import java.beans.*;
import java.io.*;
import java.util.*;
import java.text.*;
public class crServlet extends HttpServlet
{
    private accessBean aB = null;
    public PropertyChangeAdapter adapter = new PropertyChangeAdapter();
    public boolean queryStatus = false;
    //private boolean flag = true;
    private String userid = "root";
    private String passwd = "123456";

    public void doPost(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException
    {
        res.setContentType("text/html");
        ServletOutputStream out = res.getOutputStream();
        out.println("<html>");
        out.println("<head><title>Customer Registration Page</title></head>");
        out.println("<body>");
    }
```
boolean flag = true;
String ssn = req.getParameter("SSN1") + req.getParameter("SSN2") + req.getParameter("SSN3");
String fName = req.getParameter("FIRST");
String mName = req.getParameter("MID");
String lName = req.getParameter("LAST");
String street = req.getParameter("STREET");
String city = req.getParameter("CITY");
String state = req.getParameter("STATE");
String zip = req.getParameter("ZIP");
String home = req.getParameter("HOME1") + req.getParameter("HOME2") + req.getParameter("HOME3");
String work = req.getParameter("WORK1") + req.getParameter("WORK2") + req.getParameter("WORK3");
String dateFormat = "yyyy-MM-dd";
SimpleDateFormat formatter = new SimpleDateFormat(dateFormat);
java.util.Date dt = new java.util.Date(System.currentTimeMillis());
String age = formatter.format(dt);

if(ssn.length() == 0)
{
    out.println("<br>Error ! SSN is required");
    flag = false;
}
else if(ssn.length() != 9)
{
    out.println("<br>Error ! SSN is invalid");
    flag = false;
}
if(fName.length() == 0)
{
    out.println("<br>Error ! First Name is required");
    flag = false;
}
if(lName.length() == 0)
{
    out.println("<br>Error ! Last Name is required");
    flag = false;
}
if(street.length() == 0)
{
    out.println("<br>Error ! Street is required");
    flag = false;
}
if(city.length() == 0)
{
    out.println("<br>Error ! City is required");
    flag = false;
}
if(state.length() == 0)
{
    out.println("<br>Error ! State is required");
    flag = false;
}
else if(state.length() != 2)
{
    out.println("<br>Error ! State is invalid");
    flag = false;
}
if(zip.length() == 0)
{ out.println("<br>Error ! Zip Code is required");
    flag = false;
}
else if(zip.length() != 5)
{ out.println("<br>Error ! Zip Code is invalid");
    flag = false;
}
if(home.length() == 0)
{ out.println("<br>Error ! Home Phone is required");
    flag = false;
}
else if(home.length() != 10)
{ out.println("<br>Error ! Home Phone is invalid");
    flag = false;
}
if(work.length() != 0 && work.length() != 10)
{ out.println("<br>Error ! Work Phone is invalid");
    flag = false;
}
if(flag)
{ String sql = "SELECT ssn FROM customer WHERE ssn = "+ ssn + ";
    aB = new accessBean("sweethome", userid, passwd, sql);
    aB.addPropertyChangeListener(adapter);
    aB.execQuery();
    if(aB.getResultArray() != null)
    { out.println("<p><b>ERROR !!! SSN has been used !!!</b></p>");
        out.println("<br><br>");
        flag = false;
    }
}
if(flag)
{ String sql = "INSERT INTO customer set SSN=" + ssn + ",LName=" + lName + ",MName=" + mName + ",FName=" + fName + ",HomeNo=" + home + ",WorkNo=" + work + ",Street=" + street + ",City=" + city + ",State=" + state + ",Code=" + zip + ",Regdate=" + age + "");
    aB = new accessBean("sweethome", userid, passwd, sql);
    aB.addPropertyChangeListener(adapter);
    aB.executeUpdate();
    if(!getQueryStatus())
    { out.println("<p><b>Error ! " + getErrorMsg() + "</b></p>");
        flag = false;
    }
}
if(flag)
{ out.println("<p><b>Congratulations</b></p>");
    out.println("<p><b>Your Registration has confirmed !</b></p>");
    out.println("<p><b>Please use your SSN as your ID to browse this website !</b></p>");
public String getServletInfo()
{
    return "A Servlet that performs Sweet Home Inc. database access";
}

// Get the error message

public synchronized String getErrorMsg()
{
    return aB.getErrorMsgO;
}

public synchronized void setQueryStatus(boolean b)
{
    queryStatus = b;
}

public synchronized boolean getQueryStatus()
{
    return queryStatus;
}

class PropertyChangeAdapter implements PropertyChangeListener
{
    public void propertyChange(PropertyChangeEvent e)
    {
        String status = new String(e.getNewValue().toString());
        if(status.equals("true"))
        {
            if(aB.getErrorMsg() != null)
                setQueryStatus(false);
            else
                setQueryStatus(true);
        }
    }
}

appServlet.java

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

public class appServlet extends HttpServlet
{
    public void doPost(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException
    {
        res.setContentType("text/html");
        ServletOutputStream out = res.getOutputStream();
        out.println("<html>");


out.println("<OPTION VALUE='25'>25</OPTION>">);
out.println("<OPTION VALUE='26'>26</OPTION>">);
out.println("<OPTION VALUE='27'>27</OPTION>");
out.println("<OPTION VALUE='28'>28</OPTION>");
out.println("<OPTION VALUE='29'>29</OPTION>");
out.println("<OPTION VALUE='30'>30</OPTION>");
out.println("<OPTION VALUE='31'>31</OPTION>");
out.println("<SELECT name='YEAR'>");
out.println("<OPTION VALUE='2002'>2002</OPTION>");
out.println("<OPTION VALUE='2003'>2003</OPTION>");
out.println("</SELECT>");
out.println("<td><input type='radio' value='10:00' name='TIME' CHECKED>I0:00 AM</input>");
out.println("<input type='radio' value='14:00' name='TIME'>14:00 PM</input>");
out.println("<td><input type='radio' value='BUY' name='PURPOSE' value='BUY' CHECKED>Buy</input>");
out.println("<input type='radio' value='SELL' name='PURPOSE'>Sell</input>");
out.println("<input type='radio' value='OTHER' name='PURPOSE'>Other</input>");
out.println("</td></tr>");
out.println("<p><input type='submit' value='Submit' />
<input type='reset' value='Reset' />
</form>");
out.println("<a href='../../kevin-p/forgetid.html'>Forget your ID?</a> + "
"View broker information<a>");
out.println("</body>");
out.println("</html>");
}

public String getServletInfo()
{
return "Make Appointment page with Broker ID ready";
}
})

makeAppServlet.java

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

public class makeAppServlet extends HttpServlet
{
private accessBean aB = null;
public PropertyChangeListener adapter = new PropertyChangeListener();
public boolean queryStatus = false;
private boolean flag = true;
}

123
private String userid = "root";
private String passwd = "123456";
String dateFormat = "yyyy-MM-dd";
SimpleDateFormat formatter = new SimpleDateFormat(dateFormat);
java.util.Date dt = new java.util.Date(System.currentTimeMillis());
String curr = formatter.format(dt);
String currD = curr.substring(8,10);
String currM = curr.substring(7);
String currY = curr.substring(0,4);

public void doPost(HttpServletRequest req, HttpServletResponse res)
throws ServletException, IOException {
    res.setContentType("text/html");
    ServletOutputStream out = res.getOutputStream();
    out.println("<html>");
    out.println("<head><title>Make Appointment</title></head>");
    out.println("<body>");
    String bid = req.getParameter("BID");
    String cid = req.getParameter("CID");
    String month = req.getParameter("MONTH");
    String date = req.getParameter("DATE");
    String year = req.getParameter("YEAR");
    String time = req.getParameter("TIME");
    String purpose = req.getParameter("PURPOSE");
    String appDate = year + "+" + month + "+" + date;
    if(date.length() == 1) date = "0" + date;
    if(month.length() == 1) month = "0" + month;
    if(bid.length() == 0)
    { out.println("<br><b>Error! Broker ID is REQUIRED</b>");
        flag = false;
    }
    else if(bid.length() != 5)
    { out.println("<br><b>Error! Broker ID is INVALID</b>");
        flag = false;
    }
    if(cid.length() == 0)
    { out.println("<br><b>Error! Customer ID is REQUIRED</b>");
        flag = false;
    }
    else if(cid.length() != 9)
    { out.println("<br><b>Error! Customer ID is INVALID</b>");
        flag = false;
    }
    // Date Checking
    if(date.equals("31"))
    { if(month.equals("02") || month.equals("04") || month.equals("06") ||
            month.equals("09") || month.equals("11"))
        { out.println("<br><b>Error! Date is invalid...</b>");
            flag = false;
        } }
} else if(date.equals("30")
{" if(month.equals("02"))
{" out.println("<p><b>Error! Date is invalid...</b>"};
   flag = false;
   }
}
else if(date.equals("29") && month.equals("02") && !year.equals("2000"))
{" out.println("<p><b>Error! Date is invalid...</b>");
   flag = false;
}
if(flag)
{" if(year.equals(currY) && month.equals(currM) && date.equals(currD))
{" out.println("<p><b>Sorry, you cannot make an appointment for today</b>.");
   out.println("<p><b>Please set it up at least 1 day in advance</b>.");
   flag = false;
   }
}
else if((year.compareTo(currY) < 0) || (year.compareTo(currY) == 0 && month.compareTo(currM) < 0) || (year.compareTo(currY) == 0 && month.compareTo(currM) == 0 && date.compareTo(currD) < 0))
{" out.println("<p><b>Error! Time you've chosen was in the past</b>.");
   out.println("<p><b>Please choose another time</b>.");
   flag = false;
   }
}
if(flag)
{" String sql = "SELECT brokerid FROM broker WHERE brokerid = " + bid + ";"
   aB = new accessBean("sweethome", userid, passwd, sql);
   aB.addPropertyChangeListener(adapter);
   aB.execQuery();
   if(aB.getResultArray(0) == null)
{" out.println("<p><b>We're sorry</b>.");
   out.println("<p><b>The Broker ID doesn't exist</b>.");
   out.println("<br>");
   flag = false;
   }
   if(!getQueryStatus())
{" out.println("ERROR - " + getErrorMsgQ);
   }
   sql = "SELECT ssn FROM customer WHERE ssn = " + cid + ";"
   aB = new accessBean("sweethome", userid, passwd, sql);
   aB.addPropertyChangeListener(adapter);
   aB.execQuery();
   if(aB.getResultArray(0) == null)
{" out.println("<p><b>We're sorry</b>.");
   
   125
out.println("<px>"+yourCustomerID+" doesn't exist...</b>");
out.println("<br>"+"");
flag = false;
}
if(!getQueryStatus())
{ out.println("ERROR - " + errorMsg());
}
}

if(flag)
{ String sql = "SELECT appdate, apptime FROM appointment WHERE brokerid = " + bid + " AND customerssn = " + cid + "";
ab = new accessBean("sweethome", userid, passwd, sql);
ab.addPropertyChangeListener(adapter);
ab.execSQL();

if(ab.getResultArray(0) != null &&
ab.getResultArray(0).equals(year + "," + month + "," + date) &&
ab.getResultArray(1).equals(time))
{ out.println("<p><b>We're sorry, Broker is not available at that time!</b></p>");
out.println("<p>Please choose some other time...</b></p>");
flag = false;
}
}

if(flag)
{ String sql = "INSERT INTO appointment VALUES(" + bid +"," + cid +"," + appDate +"," + time +"," + purpose + ")";
ab = new accessBean("sweethome", userid, passwd, sql);
ab.addPropertyChangeListener(adapter);
ab.executeUpdate();

if(!getQueryStatus())
{ out.println("<p><b>Error ! " + errorMsg() + "</b></p>");
flag = false;
}
}

if(flag)
{ out.println("<p><b>Congratulations</b></p>"+yourAppointmenthasbeenconfirmed!"+yourAppointmenthasbeenconfirmed!");
}

out.println("<p><br><br><hr>");
if(!flag)
{ out.println("<p><br><a href="/kevin-p/app.html">GO BACK !</a><br>"+yourAppointmenthasbeenconfirmed!");
}
flag = true;
out.println("</body></html>");
public String getServletInfo()
{
    return "A Servlet that performs Sweet Home Inc. Appointment page";
}

// Get the error message

public synchronized String getErrorMsg()
{
    return aB.getErrorMsg();
}

public synchronized void setQueryStatus(boolean b)
{
    queryStatus = b;
}

public synchronized boolean getQueryStatus()
{
    return queryStatus;
}

class PropertyChangeAdapter implements PropertyChangeListener
{
    public void propertyChange(PropertyChangeEvent e)
    {
        String status = new String(e.getNewValue().toString());

        if(status.equals("true"))
            if(aB.getErrorMsg() != null)
                setQueryStatus(false);
        else
            setQueryStatus(true);
    }
}

forgtidServlet.java

import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;
import java.beans.*;
import java.io.*;
import java.util.*;
public class forgtidServlet extends HttpServlet
{
    private accessBean aB = null;
    public PropertyChangeAdapter adapter = new PropertyChangeAdapter();
    public boolean queryStatus = false;
    private String userid = "root";
    private String passwd = "123456";

    public void doPost(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException
    {
        res.setContentType("text/html");
        ServletOutputStream out = res.getOutputStream();
out.println("<html>");
out.println("<head><title>Customer Registration Page</title></head>");
out.println("<body>

String ssn = req.getParameter("SSN1") + req.getParameter("SSN2") +
req.getParameter("SSN3");
String zip = req.getParameter("ZIP");
boolean flag = true;

if(ssn.length() == 0)
{  out.println("<br>Error ! SSN is required");
   flag = false;
}
else if(ssn.length() != 9)
{  out.println("<br>Error ! SSN is invalid");
   flag = false;
}
if(zip.length() == 0)
{  out.println("<br>Error ! Zip Code is required");
   flag = false;
}
else if(zip.length() != 5)
{  out.println("<br>Error ! Zip Code is invalid");
   flag = false;
}
if(flag)
{  String sql = "SELECT ssn, code, fname FROM customer WHERE ssn = " + ssn + ""
   aB = new accessBean("sweethome", userid, passwd, sql);
   aB.addPropertyChangeListener(adapter);
   aB.executeQuery();
   if(aB.getResultArray(0) == null)
   {  out.println("<p>We're sorry ....");
      out.println("<p>Your SSN is not registered</b>");
      out.println("<p>Please go to the &lt;a href='kevin-p/cr.html'
Customer Registration
page.&lt;/a&gt;&lt;br&gt;");
   flag = false;
   }
   else
   {   if(aB.getResultArray(1).equals(zip))
       {  out.println("<p><b>Welcome, &lt;b&gt;" + aB.getResultArray(2) + ", " +
   out.println("<p>You are a registered customer !");
   out.println("<p>Please use your &lt;b&gt;SSN&lt;/b&gt; as your &lt;b&gt;user id&lt;/b&gt;
and");
   out.println("<p>your &lt;b&gt;zip code&lt;/b&gt; as your &lt;b&gt;password&lt;/b&gt;");
   }
   else
   {  out.println("<p>Sorry, you're a registered customer,");
      out.println("<p>but your zipcode doesn't match !");
      out.println("<p>Please contact our customer service !");
      out.println("<p><br><br>&lt;a href=../../sweethome/forgetid.html&gt;Try Again&lt;/a&gt;");
   }
   }
}
else
{  out.println("<p>Customer Registration Page

String ssn = req.getParameter("SSN1") + req.getParameter("SSN2") +
req.getParameter("SSN3");
String zip = req.getParameter("ZIP");
boolean flag = true;

if(ssn.length() == 0)
{  out.println("<br>Error ! SSN is required");
   flag = false;
}
else if(ssn.length() != 9)
{  out.println("<br>Error ! SSN is invalid");
   flag = false;
}
if(zip.length() == 0)
{  out.println("<br>Error ! Zip Code is required");
   flag = false;
}
else if(zip.length() != 5)
{  out.println("<br>Error ! Zip Code is invalid");
   flag = false;
}
if(flag)
{  String sql = "SELECT ssn, code, fname FROM customer WHERE ssn = " + ssn + ""
   aB = new accessBean("sweethome", userid, passwd, sql);
   aB.addPropertyChangeListener(adapter);
   aB.executeQuery();
   if(aB.getResultArray(0) == null)
   {  out.println("<p>We're sorry ....");
      out.println("<p>Your SSN is not registered</b>");
      out.println("<p>Please go to the &lt;a href='kevin-p/cr.html'
Customer Registration
page.&lt;/a&gt;&lt;br&gt;");
   flag = false;
   }
   else
   {   if(aB.getResultArray(1).equals(zip))
       {  out.println("<p><b>Welcome, &lt;b&gt;" + aB.getResultArray(2) + ", " +
   out.println("<p>You are a registered customer !");
   out.println("<p>Please use your &lt;b&gt;SSN&lt;/b&gt; as your &lt;b&gt;user id&lt;/b&gt;
and");
   out.println("<p>your &lt;b&gt;zip code&lt;/b&gt; as your &lt;b&gt;password&lt;/b&gt;");
   }
   else
   {  out.println("<p>Sorry, you're a registered customer,");
      out.println("<p>but your zipcode doesn't match !");
      out.println("<p>Please contact our customer service !");
      out.println("<p><br><br>&lt;a href=../../sweethome/forgetid.html&gt;Try Again&lt;/a&gt;");
   }
   }
}
}
out.println("<p><br><br><hr>");
if(!flag)
{
  out.println("<p><br><a href='../sweethome/forgetid.html'>GO BACK !</a><br>");
}
out.println("</body></html>");

public String getServletInfo()
{
  return "When a customer forget their id or password";
}

// Get the error message
public synchronized String getErrorMsg()
{
  return aB.getErrorMsg();
}

public synchronized void setQueryStatus(boolean b)
{
  queryStatus = b;
}

public synchronized boolean getQueryStatus()
{
  return queryStatus;
}

class PropertyChangeAdapter implements PropertyChangeListener
{
  public void propertyChange(PropertyChangeEvent e)
  {
    String status = new String(e.getNewValue().toString());
    if(status.equals("true"))
    {
      if(aB.getErrorMsg() != null)
        setQueryStatus(false);
      else
        setQueryStatus(true);
    }
  }
}
}
REFERENCES


