15

Step-by-Step Exercises for Building the VBS

15.1 Introduction

15.2 Exercise 1 - VBS Homepage Design (Weeks 1 and 2)
15.3 Exercise 2 - Form Validation Using Javascript (Weeks 3 and 4)
15.4 Exercise 3 - Search Engines (Weeks 5-7)
15.5 Exercise 3A - Quick Search
15.6 Exercise 3B - Category Search
15.7 Exercise 3C - Advanced Search
15.8 Exercise 4 - Access Control (Weeks 8 and 9)
15.9 Exercise 4A - CartLogin
15.10 Exercise 4B - Create a New Customer Account
15.11 Exercise 4C - Change Password
15.12 Exercise 4D - CartLogout
15.13 Exercise 5 - Virtual Shopping (CartServices) (Weeks 10 and 11)
15.14 Exercise 6 - E-Payment (Week 12)

15.1 INTRODUCTION

In previous chapters, we used the VBS to explain the basic technologies for building an e-commerce system as well as the principles of e-commerce. In this chapter, we provide some step-by-step exercises for building the VBS using Java Servlet as
the main programming tool. These exercises will give you hands-on programming experiences in building a B2C e-commerce system from scratch. The programming techniques applied in the exercises can in fact be adapted in many other e-commerce systems.

Altogether there are six exercises. Each exercise is devoted to building part of the system. After completing all the exercises, you will build the complete VBS, which incorporates most of the B2C e-commerce system functions including access control, shopping cart, and search engine.

It takes about 12 weeks to complete all the exercises. If this book is used for teaching a computing course in e-commerce, the exercises should fit neatly into a 12-week laboratory project for the students. It is recommended that students should form teams of three to four people to build the VBS. The exercises also provide a good reference for e-commerce programmers/developers and Information Technology managers.

In each exercise, we provide you with the objective, the program instruction, and some program hints. The objective defines the program requirements. In some cases, a flow chart is included to explain the main logic of the program. The program instruction guides you in writing the program, and the program hints help you to complete it. Of course, there are many different ways to write the programs. For your reference, the sample answers can be downloaded from the web site of this book. The program listings are labeled as Fig. Ax.y where x and y are the exercise number and program number, respectively. In other words, this chapter is designed to work in conjunction with the Web site of this book. By integrating all the programs, you can build the VBS. It is strongly encouraged that you try to write the programs yourself first. By doing so, you will learn a lot more. (For instructors who adopt this book, the “laboratory exercise version” is available for developing your own laboratory exercises.)

15.1.1 Typical e-shopping scenario

Let us first look at a typical shopping scenario for a VBS customer. To purchase some books, the customer will usually go through the following steps:

1. access the Web site of the VBS;
2. login to the VBS;
3. search for the required books;
4. select the required books and put them into a virtual shopping cart;
5. view/change the content of the shopping cart if required; and
6. check out and settle the payment.
Note that step 2 can also be deferred until check out is required. The aforementioned assumes that the customer has already registered with the VBS (i.e., he has an account in the VBS system already) and so he can login to the system at any time. The VBS should also provide a function for new customers to create an account in the VBS system.

15.1.2 VBS – system overview

Figure 15.1 gives a general overview of the major modules of the VBS system. It consists of four main systems as follows:

1. **Search engine system** – Provides different types of book search functions including quick search, category search, and advanced search.

2. **Access control system** – Provides various user login functions including creation of new user account and validation of login information.

3. **Shopping cart system** – Provides virtual shopping cart functions by using the Java Servlet session tracking technique.

4. **Checkout system** – Provides the checkout functions after shopping (it is referred to as e-payment).

![Figure 15.1 Schematic diagram of VBS system](image-url)
The VBS project consists of six interrelated exercises (12 weeks) as follows:

- Exercise 1 (Weeks 1 and 2): VBS homepage design
- Exercise 2 (Weeks 3 and 4): Form validation with Javascripts
- Exercise 3 (Weeks 5–7): Search engines
- Exercise 4 (Weeks 8 and 9): Access control (CartLogin)
- Exercise 5 (Weeks 10 and 11): Shopping cart system
- Exercise 6 (Week 12): Checkout (e-Payment)

15.2 EXERCISE 1 – VBS HOMEPAGE DESIGN (WEEKS 1 AND 2)

15.2.1 Objectives

In the first two weeks, the major objective is to build the homepage of the VBS by using the nested framed page technique. Furthermore, we will also create a dynamic banner by using both the Servlet technique and the JavaScript technique. The dynamic banner can be rented out for generating advertising revenue for the VBS!

15.2.2 Program instructions

1. Build the VBS homepage by using three nested frames: banner, index, and content, as shown in Figure 15.2. The sample answer is given in Figure A1.1. Create the following HTML files for building the VBS homepage:
   - Welcome.html: The main welcome page/content page
   - Top.html: The banner page
   - Menu.html: The index page

   You are encouraged to design the homepage with as many web publishing techniques as possible. Of course, you can use the web publishing tools. Be innovative!

2. Using the Servlet technique, create a dynamic banner by displaying the four images as shown in Figure 15.3 in a cyclic manner. The sample answer is given in Figure A1.2.

3. Repeat step 2 by using JavaScript. The sample answer is given in Figures A1.3a–A1.3c.

4. Compare the pros and cons of using the two approaches (i.e. steps 2 and 3). Which one would you choose for your VBS system? Why?
Figure 15.2  VBS homepage using nested frame page technique

Figure 15.3  Dynamic banner based on four different images
15.2.3 Program hints

1. Nested frame pages can be created by using the "frameset" and "frame" tags. Details can be found in Chapter 3.

2. For the Servlet approach, the "dynamic" effect can be achieved by setting the response object "resp" of the HTML document as

   ```
   resp.setHeader("Refresh", "2");
   ```

   By doing so, it will ask the web browser to "refresh" the current web page once every 2 s. You should think about what to include in the web page each time and how to do it.

3. For the JavaScript approach, we cannot include the Javascript in the banner page. Why? We need to include the Javascript in a static page, otherwise it will be overwritten every time a new page is loaded. The index page is the best choice. Therefore, it is recommended that the Javascript that is used to invoke the animation in the banner page should be written in the index page.

   Let us give an overview of what should be done. In the "welcome.html" file, a function should be included to start the animation when the web page is loaded. In the "menu.html" file, an array should be used to keep track of which image is to be loaded next time. When the time comes, the image should be written in the banner page accordingly. To implement this using Javascript, please visit http://developer.netscape.com/docs/examples/javascript/formval/overview.html and other Javascript references on the internet.

15.3 EXERCISE 2 – FORM VALIDATION USING JAVASCRIPT (WEEKS 3 AND 4)

15.3.1 Objectives

In Weeks 3 and 4, the objectives are to create an HTML form for creating a new account and to write the corresponding Javascripts for validating the input data.

15.3.2 Program instructions

1. Create an HTML form for creating a new user account. A sample is shown in Figure 15.4. The HTML form contains the following fields:

   - Customer ID (Text input: length 30)
   - Password (Password input: length 10)
2. Write the Javascripts to perform the following validations*:
   - to "trim" all text inputs;
   - to check for the existence of "@" and "." characters in the e-mail field;
   - to check whether the expiry date of the credit card is before or after current day. Return false if it is before (or equal to) current date;

* Bonus: All the form validation Javascript functions being used in the VBS system are "embedded" into the HTML file "JavaScriptCode.html" (Figure A2.4), which is invoked by the Java class "JavaScriptCode" (Figure A2.3). For clarity purpose, all the "global" constants being used in VBS system are collected into the file "MALL_CONST.java" (Figure A2.2).
- to ensure that numbers rather than characters are entered into the number fields (e.g. credit card number); and
- to ensure that the characters entered in the "Password" field are the same as that entered in the "Re-enter Password" field.

The sample HTML document for "Create New Customer Account" form validation using Javascripts is given in Figure A2.1.

15.3.3 Program hints

1. Details on HTML form elements (e.g., text input, password input, checkbox and buttons etc.) as well as an overview of JavaScript can be found in Chapter 3.

2. Please visit http://developer.netscape.com/docs/examples/javascript/formval/overview.html to learn more about JavaScript and how it can be used to perform form validation.

15.4 EXERCISE 3 – SEARCH ENGINES (WEEKS 5–7)

15.4.1 VBS – System flow

In the previous two exercises, we created the “front-door” of the VBS. Using this as a start, we will build the functional modules for the VBS, which include search engines, user login, shopping cart, and checkout.

A general overview of the VBS system is shown in Figure 15.5.

As shown in Figure 15.5, the major modules are invoked by clicking the corresponding buttons in the menu page of the index frame. They include:

- Quick search button – for performing the quick search
- Advanced search button – for performing the advanced search
- Category search button – for performing the category search
- Login button – for customer login
- View cart button – for viewing the shopping cart
- Logout button – for logging out the system
- Change password button – for changing the password
- New account button – for creating a new account
Figure 15.5  System flow of VBS

Figure 15.5 also shows the major program flow and the corresponding servlets/HTML file being invoked in each case.

The VBS has a simple Microsoft Access database called mall.mdb. It contains the following tables:

- **Bookstore** – for storing the book information in the VBS
- **Transaction** – for storing the order information
- **Customer** – for storing registered customers' information.
The table fields are shown as follows:

<table>
<thead>
<tr>
<th>Fields</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bookstore table</strong></td>
<td>ISBN code</td>
</tr>
<tr>
<td>Name</td>
<td>Title of the book</td>
</tr>
<tr>
<td>Author</td>
<td>Author name</td>
</tr>
<tr>
<td>Publisher</td>
<td>Publisher name</td>
</tr>
<tr>
<td>Year</td>
<td>Year of publication</td>
</tr>
<tr>
<td>Price</td>
<td>Price of book</td>
</tr>
<tr>
<td>Category</td>
<td>Category of the book</td>
</tr>
<tr>
<td>Desc</td>
<td>Description of the book</td>
</tr>
<tr>
<td>Quantity</td>
<td>Available quantity of the book</td>
</tr>
<tr>
<td><strong>Transaction table</strong></td>
<td>Customer name of the customer</td>
</tr>
<tr>
<td>Isbn</td>
<td>The ISBN code of the book</td>
</tr>
<tr>
<td>DeliveryQty</td>
<td>The quantity of the book delivered to the customer</td>
</tr>
<tr>
<td>SubscribeQty</td>
<td>The quantity of the book ordered by the customer</td>
</tr>
<tr>
<td><strong>Customer table</strong></td>
<td>CustID customer ID</td>
</tr>
<tr>
<td>Password</td>
<td>Password of the customer</td>
</tr>
<tr>
<td>Email</td>
<td>E-mail of the customer</td>
</tr>
<tr>
<td>GetNews</td>
<td>Indicate whether the customer is willing to receive latest news</td>
</tr>
<tr>
<td>Bonus</td>
<td>Bonus points of the customer</td>
</tr>
<tr>
<td>Gender</td>
<td>Gender of the customer</td>
</tr>
<tr>
<td>Fname</td>
<td>First name of the customer</td>
</tr>
<tr>
<td>Lname</td>
<td>Last name of the customer</td>
</tr>
<tr>
<td>CreditNo1</td>
<td>The credit card number of the customer</td>
</tr>
<tr>
<td>Expiry date</td>
<td>Expiry date of the credit card</td>
</tr>
<tr>
<td>Address</td>
<td>The address of the customer</td>
</tr>
</tbody>
</table>

**15.4.2 Objectives**

In this exercise, the major objective is to build a search system that is composed of different search options:

- **Quick search** – based on simple keyword(s)
- **Category search** – based on a predefined book category
- **Advance search** – based on a combination of different book attributes including book name, author’s name, price, etc.
15.4.3 Search engine: Program flow

Irrespective of the search techniques, the main program flow is the same as shown in Figure 15.6, which is given by the Java program "DbQuery.java" (Figure A3.2).

![Flowchart](image)

**Figure 15.6** Generic logic flow for search engine
For each search method, the main steps are as follows:

1. retrieve the search criteria entered by the user
2. build the SQL statement
3. collect the search results
4. format the search results in HTML format and display them

Steps 3 and 4 will be performed by two methods, namely `BookQuery.booksearch` and `DisplayResult.displayTable`, respectively.

As shown in Figure 15.7, the "search engine" package has three search programs, namely, `QuickSearch.java`, `CategorySearch.java`, and `AdvancedSearch.java` for performing quick search, category search, and advanced search, respectively.

In addition, `BookQuery.java` and `DisplayResult.java` contain the common servlet methods that can be shared by the three different search programs. `SE_CONST.java` (Figure A3.1) contains the common constant values (i.e. String values) that are used in the search engine programs.

15.5 **EXERCISE 3A – QUICK SEARCH**

15.5.1 Program instructions

1. Write the Quick Search Servlet (namely `QuickSearch.java`) to perform the simple keyword search. In this case, a user enters the key word in the text-box for the keyword search (see Figure 15.2). The search is then triggered by clicking the "GO" button.

2. The program will search for books in the database that have the specified keyword(s) in the following fields:
3. Write two programs `BookQuery.java` and `DisplayResult.java` to perform the SQL query and format the query results, respectively. Note that these two programs can be shared by the other two search programs in the following exercises. Basically, by giving the search criteria (SQL command), the `booksearch()` method can perform the search and generate the resultset object. The three different search programs can make use of `BookQuery.java` to perform the search by providing the respective search criteria. Once the resultSet object is generated, it can be passed to the `DisplayResult.class` to display the search results. The sample answers for `BookQuery.java`, `DisplayResult.java`, and `QuickSearch.java` are given in Figures A3.3, A3.4, and A3.5, respectively.

15.5.2 Program hints

1. Figure 15.8 shows the proposed program structure of the VBS up to the current stage. The root directory contains the "image" folder, the HTML files for the VBS, the "search engine" folder, and the `MALL_CONST.class`. By now you should have developed the HTML files. Let us explain the other components as follows.

   For ease of programming and management, a `MALL_CONST.class` can be written to define the String values that will be used throughout the VBS package. A sample can be found in Figure A2.2. The "image" folder contains the image files (e.g. book images, common icons, etc.) for the VBS. Similarly, `SE_CONST.class` defines the String values that will be used for the search engine package only (i.e., the three search programs only).

2. Suppose that the keyword entered by the user is "Java." The corresponding SQL statement for the quick search looks like this:

   ```
   Select name, author, publisher, price, ISBN, year from bookstore
   where name like "%Java%" or author like "%Java%" or publisher
   like "%Java%" ... .
   ```

   Now the problem is that we need to turn the SQL statement into a String object and express it in terms of the String constants as defined in `SE_CONST.java`. Suppose that the entered keyword is stored in the variable "keyword." The search criteria (`bookQuery`) for the Quick search program in terms of the
constant values as defined in SE CONST.java looks like the following:

```java
bookQuery = "select" + SE CONST.DB_BOOKNAME+ "," 
+ SE CONST.DB_AUTHOR+ "," + SE CONST.DB_PUBLISHER+ "," 
+ SE CONST.DB_PRICE+ "," + SE CONST.DB_ISBN+ "," 
+ SE CONST.DB_YEAR+ "from" + SE CONST.BOOKSTORE+ "where" 
+ SE CONST.DB_ISBN+ "like '%%";
```

```java
if (!keyword.equals(""))
bookQuery +="and ("+SE CONST.DB_AUTHOR+ "like '%%" +keyword+ "%%" 
or" +SE CONST.DB_BOOKNAME+ "like '%%" +keyword+ "%%" or" 
+SE CONST.DB_PUBLISHER+ "like '%%" +keyword+ "%%" or" 
+SE CONST.DB_ISBN+ "like '%%" +keyword+ "%%" or" 
+SE CONST.DB_CATEGORY+ "like '%%" +keyword+ "%%")";
```

The query contains two parts. The first part specifies the "bookQuery" string (the SQL string), which collects all the related book attributes for the query criteria. The second part appends the "keyword" string to the bookQuery to
complete the whole SQL statement. In doing so, the "like" statement is used to perform the pattern matching.

3. Having defined the search criteria in terms of SQL, we can perform the search by BookQuery.java. There is a booksearch() method in BookQuery.java, which basically performs the following: (1) creates the JDBC connection, (2) executes the SQL statement, and (3) calls the displayTable() method of the DisplayResult.class to display the search result in HTML format.

4. Note that the DisplayResult.class needs to implement the multiple page function as described in Chapter 5.

15.6  EXERCISE 3B - CATEGORY SEARCH

15.6.1  Objective

Category search provides an HTML page (namely category.html) as shown in Figure 15.9 so that a user can search for the books under different categories. The category of each book is indicated by the "category" field in the book database (i.e. mall.mdb).

15.6.2  Program instructions

1. Create the category.html page similar to the one in Figure 15.9. The sample answer is given in Figure A3.6.

2. Write the Servlet CategorySearch.java to perform the category search. Basically, when the hyperlink of a particular category is clicked, the books of that category will be displayed. CategorySearch.java should make use of BookQuery.java and DisplayResult.java to perform the SQL query and to display the search results, respectively. The sample answer for CategorySearch.java is given in Figure A3.7.

15.6.3  Program hints

1. The category.html file is a simple HTML document that contains a table of hyperlinks for a user to invoke the CategorySearch program with different category parameters. A typical link is shown here:

   <a href="servlet/vbs.mall.searchengine.CategorySearch?category=%28e-commerce%29">Electronic Commerce</a>

In this case, a Servlet called "CategorySearch" with the input parameter "category = e-commerce" will be invoked when a user clicks this hyperlink.
2. The major difference between Category search and Quick search is the SQL statement. In this case we search only for books with the specified word(s) in the category field. In terms of the constants as defined in $\text{SECONST}.\text{java}$, the SQL statement is

```
bookQuery = "select" +SE_CONST.DATABASE_NAME+ ",",
+SE_CONST.DATABASE_AUTHOR+ ",", +SE_CONST.DATABASE_PUBLISHER+ ",",
+SE_CONST.DATABASE_PRICE+ ",", +SE_CONST.DATABASE_ISBN+ ",",
+SE_CONST.DATABASE_YEAR+ "from" +SE_CONST.BOOKSTORE+"where"
+SE_CONST.DATABASE_ISBN+ "like '__' and" +SE_CONST.DATABASE_CATEGORY+
"like '__' + category+ '__' order by" +SE_CONST.DATABASE_ISBN+ ","
+SE_CONST.ASC;
```

### 15.7 EXERCISE 3C – ADVANCED SEARCH

#### 15.7.1 Objective

Compared with the previous search options, Advanced search provides the greatest flexibility because a user can perform the search based on a combination of the
book attributes. The input form for the Advanced search is shown in Figure 15.10. Basically, the books that match with any of the specified keywords in the form will be displayed.

15.7.2 Program instructions

1. Construct the Advanced Book Search input form (namely Searchengine.html) with the following attributes:
   - Author name: text field (size 50)
   - Book title: text field (size 50)
   - Publisher: text field (size 50)
   - ISBN number: text field (size 20)
   - Display format: select box with three options of (1) 10 items per page, (2) 30 items per page, or (3) 50 items per page
   - Ordering: select box with three options of ordering by (1) price; (2) ISBN, or (3) year of publication
• A radio box for indicating whether the books should be displayed in ascending or descending order in accordance with the ordering criteria
• A checkbox for indicating whether the cover page image should be shown.

The form should have two buttons: "Reset" button for resetting the fields and "Search" button for triggering the search. The sample answer is given in Figure A3.8.

2. Write the corresponding Servlet program (namely AdvanceSearch.java). Again AdvanceSearch.java should make use of BookQuery.java and DisplayResult.java to perform the SQL query and to display the search results, respectively. The sample answer is given in Figure A3.9.

15.7.3 Program hints

1. The SearchEngine.html is a simple HTML form, which makes use of various HTML form elements (e.g., text boxes, radio buttons, checkboxes, list boxes, etc.). (Please refer to Chapter 3 for details.)

2. Note that in building the SQL statement, we need to take into account that some of the fields in the input form may be empty. The most straightforward way is to build the SQL query as follows by using the "if-then" statement during the parsing procedure.

```java
bookQuery = "select " + SE_CONST.DB_BOOKNAME + ",,"
        + SE_CONST.DB_AUTHOR + "," + SE_CONST.DB_PUBLISHER + ",,"
        + SE_CONST.DB_PRICE + "," + SE_CONST.DB_ISBN + ","
        + SE_CONST.DB_YEAR + "from" + SE_CONST.BOOKSTORE + "where"
        + SE_CONST.DB_ISBN + "like '%";
if (!author.equals(""))
    bookQuery += "and" + SE_CONST.DB_AUTHOR + "like '%" + author
        + ";";
if (!name.equals(""))
    bookQuery += "and" + SE_CONST.DB_BOOKNAME + "like '%" + name
        + ";";
...
```

Bonus: Book details

In order to speed up the search process and to display as many books as possible on a single page, the DisplayResult class will display only a concise list of books, as shown in Figure 15.11.
Only when a user clicks the reference link for a particular book, the detail book summary will be displayed. (see Figure 15.12).

In fact, the BookDetail program is another type of search program which will use an “exact” match for matching a particular book. The program listing of this program is given in Figure A3.10 as a bonus.

15.8 **EXERCISE 4 – ACCESS CONTROL (WEEKS 8 AND 9)**

15.8.1 **Objective**

As an extension to the Servlet session tracking technique that we have learned about in Chapter 6, the main objective of this exercise is to construct a comprehensive customer’s shopping cart login and account maintenance system for the VBS.

The main functions of the system include the following modules:

- shopping cart customer login module – CartLogin.java
- create new customer account module – NewCustAccount.java
- change customer account password module – ChCustPwd.java
- shopping cart customer logout module – CartLogout.java
15.8.2 Program structure

Figure 15.13 shows the program structure up to the current stage. The grey boxes indicate the modules completed in the previous exercises. In this exercise, we will complete the modules of the CartLogin package.

As shown in Figure 15.13, the CartLogin system consists of four major program modules, which will be invoked accordingly when a user clicks on the respective buttons in the menu page. These modules are

- CartLogin.class – for logging into the system
- NewCustAccount.class – for creating a new customer account
- ChCustPwd.class – for changing the customer password
- CartLogout.class – for logging out of the system

In addition, there are two Servlet classes USER CONST.class and ShoppingCart.class. For ease of management, the former defines the constant values being used for developing the CartLogin package. The latter is a session object for customer login control. In fact, one of the unique features for the VBS system is that we make use of the ShoppingCart object (Figure A4.2) for both customer login and
virtual shopping purpose. In other words, whenever a customer visits VBS, he will either be automatically assigned with a new shopping cart or "retrieve" the previous engaged shopping cart if his session object (ShoppingCart object) still exists. Moreover, he can log in to the VBS system (and the customer ID will be "logged" into the ShoppingCart object as well), or the system will ask for customer login when he decides to check out.

15.9 **EXERCISE 4A – CartLogin**

15.9.1 **Objective**

The system flow diagram for the CartLogin module is given in Figure 15.14.

As shown in Figure 15.14, the program logic for CartLogin is similar to the one given in Chapter 6 except that the program needs to cater for more cases as indicated
by the "action" values passed to the server through the HttpServletRequest object. The possible commands are:

- **COMMAND_CART_LOGIN** – show the customer login screen
- **COMMAND_CART_LOGIN_CHECK** – trigger the validation of the login information.
- **COMMAND_PAYMENT_LOGIN** – prompt the customer to log in when he wants to check out but he has not logged in before.
- **COMMAND_PAYMENT_LOGIN_CHECK** – trigger the validation of the login information for the case of **COMMAND_PAYMENT_LOGIN**.
Note that in general there are two login scenarios. In the first scenario, the customer requests to log in by clicking the "Login" button. In the second scenario, the system forces the customer to login because he wants to check out without logging into the system before.

The objective of the program is to cater for all the aforementioned situations.

15.9.2 Program instructions

1. For ease of management, it is recommended that the constant values to be used should be placed in a separate class file (see the sample USER CONSTANTS in Figure A4.1).

2. Based on the program given in Chapter 6, write the CartLogin.java program to satisfy the requirements as shown in Figure 15.14. The sample answer is given in Figure A4.3.

15.9.3 Program hints

1. In fact, the CartLogin program for this exercise is very similar to the one given in Chapter 6. We just need to extend the previous program to cater for more situations.

2. Similar to the approach used in Chapter 6, the CartLogin program should consist of the following main modules:
   - doPost() - handle the different action options
   - showWelcome() - display the VBS welcome page or the "front-door"
   - showPayment() - invoke the payment module to be developed later
   - showLoginDialog() - display the login screen
   - checkLogin() - validate the login information

3. Since we need to use many field constants in the CartLogin program (and in the upcoming programs) such as for catering different "action" options, it is recommended that they be defined in a separate class file for ease of management as mentioned earlier. (Please refer to Figure A4.1.)

15.10 EXERCISE 4B – CREATE A NEW CUSTOMER ACCOUNT

15.10.1 Objective

The system flow diagram for the Create New Customer Account module is shown in Figure 15.15.
The main steps are given as follows:

1. Display the new account HTML form when the customer clicks the "Create Account" button;
2. Validate the customer information after submitting the form; in particular, it is necessary to cross-check whether the user has registered already, in which case an error message should be displayed; and
3. Create the customer account by updating the database accordingly.
1. Design a "Create New Customer Account" form similar to the one given in Figure 15.16.

2. Write the Create New Customer Account program, namely NewCustAccount.java to satisfy the above requirements as given in Figure 15.15. Note that you need to incorporate the form created in step 1 into the NewCustAccount.java. The sample answer is given in Figure A4.4. Note that you need to perform form validation by means of JavaScript as well.

15.10.3 Program hints

You should include the following functional modules in NewCustAccount.java:

- `doPost()` – handles the action options
- `showNewCustAccountDialog()` – displays the Create New Customer Account HTML form
- `showNewCustAccountSuccess()` – displays the “Account Created Successfully” message
• `showNewCustAccountError()` – displays the Create New Account error message(s)
• `addCustRecord()` – performs customer account database validation

15.11 **EXERCISE 4C – CHANGE PASSWORD**

15.11.1 **Objectives**

The system flow diagram for the Change Password module is given in Figure 15.17.

![System flow diagram for Change Password module]

**Figure 15.17** System flow of Change Password module
The logic is relatively straightforward. Firstly, the system checks for the "action" passed to the server. If no action is specified, the Change Customer Password HTML form will be displayed. This HTML form will also be displayed if the action is not "USER_CONST.COMMAND_CHANGE_PWD_CONFIRMATION".

When the customer submits the request for changing a password as indicated by the action of USER_CONST.COMMAND_CHANGE_PWD_CONFIRMATION, the password update method will be invoked (namely CustPwdUpdate()). After processing, the result (either success or failure) will be returned via the showChangeCustPwdSuccess() and showChangeCustPwdError() methods, respectively.

15.11.2 Program instructions

1. Write the program Change Customer Password (ChCustPwd.java) to satisfy the aforementioned requirements as indicated in Figure 15.17. The change password form is as shown in Figure 15.18.

2. The program should have the following main methods:
   - ShowChangeCustPwdError – for showing the error messages
   - ShowChangeCustPwdDialog – for showing the Change Customer Password form
- ShowChangeCustPwdSuccess – for showing that the customer password has been changed successfully
- CustPwdUpdate – for updating the new customer password into the database

The sample answer is given in Figure A4.5.

15.11.3 Program hints

The program logic follows a similar approach as the Create Customer Account program.

15.12 **EXERCISE 4D – CartLogout**

15.12.1 Objective

The objective of the CartLogout module is to allow the customer to log out of the system whenever required. Technically, it eliminates the ShoppingCart's session object so that a new session can be started (possibly by another customer) from the same machine.

As shown in Figure 15.19, the core program logic is very simple. Firstly, the program should check whether a session object (ShoppingCart) exists or not. If a session object exists, the program will clear the session object; otherwise, it will display an error message showing that the user has not logged in.

15.12.2 Program instructions

1. Write the program CartLogout.java for performing the log out function as indicated in Figure 15.19. The sample answer is given in Figure A4.6.
2. The program should contain the following methods, namely
   - ShowNotLogin – for showing that the customer has not logged in
   - ShowLogout – for showing that the user has logged out successfully

15.12.3 Program hint

The method for closing a session is `session.invalidate`. 
**EXERCISE 5 – VIRTUAL SHOPPING (CartServices) (WEEKS 10 AND 11)**

### 15.13.1 Objective

In Chapter 6, we discussed how to employ the Servlet session tracking technique for supporting customer login using the ShoppingCart object.

In this exercise, we will implement another core function of the ShoppingCart object—“virtual shopping”—and demonstrate how it can be integrated into our VBS system.

### 15.13.2 Program structure

Figure 15.20 shows the program structure up to the current stage. The grey boxes indicate the modules completed in the previous exercises. In the following exercises,
we are going to build the CartServices program within the "Shopping Cart" package (i.e. the white box).

As shown in Figure 15.20, the Shopping Cart system consists of two program modules:

- CART_CONST.class
- CartServices.class

The CART_CONST.class (see Figure A5.1) defines the constant values (e.g. action commands and field values) to be used for virtual shopping. CartServices.class provides various shopping cart services such as viewing the shopping cart content.

In the last exercise, we have introduced the shopping cart object.
The main attributes of a shopping cart to support virtual shopping are:

- Customer identity (ID)
- Number of items being purchased
- Total price of the goods (books) being selected
- List of goods selected including product code, quantity of purchase, item price, and item description.

For a detailed description of the shopping cart object, please refer to Section 6.4.2 of Chapter 6.

15.13.3 CartServices: System flow

Figure 15.21 shows the flow chart of the shopping cart system for virtual shopping. The main tasks are performed by CartService.class.

In principle, the CartService program implements different virtual shopping functions, including:

- COMMAND_ADD – add books into the shopping cart object
- COMMAND_UPDATE – update the information stored in the shopping cart object
- COMMAND_CLEAR – clear the current shopping cart object while still keeping it for later use

15.13.4 Program instruction

1. Write the CartServices Servlet to satisfy the requirements as indicated in the aforementioned flow chart. Work out the main logic flow of the CartServices program.

2. The CartServices program should have the following main modules:
   - doPost() – handles different types of CartServices commands;
   - addItem() – adds books (i.e. the ISBN number(s)) and order quantity into the shopping cart object (cart).
   - updateItem() – updates the order quantity for a particular book already in the “cart” object and removes the book from the “cart” object if new order quantity is zero.
   - clearCart() – clears all the books in the ucart object.
   - viewCart() – displays the current contents of the shopping cart (see Figure 15.22).

The sample answer is given in Figure A5.2.
Start

Get session object
getSession()

Success? No → Error Message

Yes

Get shopping cart

Success? No → Create new shopping cart

Yes

Get Action command

Check Action

COMMAND_ADD
Add items

COMMAND_UPDATE
Update shopping cart

COMMAND_CLEAR
Clear shopping cart

Default
Display cart content

End

Figure 15.21  System flow for the CartServices
In the final exercise, we are going to complete the whole e-shopping procedure by performing the checkout or e-payment function. For simplicity, we assume that payment is made by simply posting the order into the transaction database. In addition, the e-payment system also performs other necessary checkout functions as follows:

- getting the customer’s personal information from the database according to the user identity stored in the ulog object
- sending an e-mail to the customer for confirming the order
- displaying the books purchased

The main system logic is given in Figure 15.23.
460  STEP-BY-STEP EXERCISES FOR BUILDING THE VBS

![System flow for e-payment system](image)

**Figure 15.23** System flow for e-payment system

### 15.14.2 Program structure

Figure 15.24 shows that we have almost completed all the VBS program modules (all the grey boxes are the completed modules). In this exercise, we are going to complete the e-payment package, which consists of the e-payment class.

### 15.14.3 Program instruction

1. Create a transaction database with a table consisting of at least the following fields:
   - **Customer**: The name of the customer
   - **DeliveryQty**: The quantity of books delivered to the customer
   - **SubscribeQty**: The quantity of books ordered by the customer
2. Write the e-payment Servlet program, namely "Epayment.java" to satisfy the requirements as indicated in the aforementioned flow chart.

3. The program should have the following main modules:
   - payment() – this is the entry point of the program
   - formatEmail() – it obtains the customer's personal information from the database; generates the payment notice; and calls the sendEmail() method to send an e-mail to the customer confirming the purchase
• `sendEmail()` – it sends the confirmation e-mail
• `displayList()` – it displays the order summary and invokes the `updateDB` method to update the transaction database accordingly
• `updateDB()` – it updates the transaction database

### 15.14.4 Program hints

The following program* is for sending an e-mail message "emessage" to "mailhost.vbs.com." To use the program, we need to import "sun.net.smtp.SmtpClient". The sample answer is given in Figure A6.1.

```java
private void sendEmail(PrintWriter out, String sender, String receiver, String subject, String emessage) {
    PrintStream out; SmtpClient _email;
    try{
        _email = new SmtpClient("mailhost.vbs.com");
        _email.from(sender);
        _email.to(receiver);
        out = _email.startMessage();
        out.println("From: " + sender);
        out.println("To: " + receiver);
        out.println("Subject: " + subject);
        out.println(emessage);
        out.flush();
        out.close();
        _email.closeServer();
    }catch(Exception e){
        out.println(e.toString());
    }
}
```

* This program is written by Gary Li; listed with his permission.
Index

Absolute URL, 70–71
Abstract Syntax One (ANS.1) notation, 222
Abstract user interface (UI) objects, 407–411
Acceptability, of payment, 286
Access control system, 429, 445–447
Acknowledgment, 35
Acquirer, 288
Action, 412–414
ACTION attribute, 79
Activated condition, 412
Active Server Page (ASP) code, 96, 100–101
ActiveX controls, 56
Address, 32
ADSL. See Asymmetric Digital Subscriber Line
Advanced search engine, 126–127, 151–155
Advertising, 389–426
banners, 392–396, 430
e-mail and, 392
effectiveness of, 399
pricing models and, 399–400
quality assurance, 414–423
usability testing, 414–419
web presence and visibility, 423–424
Agent-based auction supporting sites, 385
Agent Transfer Protocol (ATP), 254
Agents, mobile, 251–263
Aggregation, 363
Agile wallet, 311
Aglet Software Development Kit (ASDK), 254
Aglets, 254–262
AH. See Authentication Header
Alert message, 90, 91
Alert protocol, 247–248
ALIGN attribute, 68, 74–75
Allow header, 44
Amazon.com, 8–9, 321, 325–328, 338
American National Standards Institution (ANSI), 367
American Online (AOL), 393
Anchor tag, 70
Andree seen, Marc, 39
anewShop.com, 338–339
Anonymity, of payment, 286, 298
Anonymous session tracking, 176
ANS.1. See Abstract Syntax One
ANSI. See American National Standards Institution
AOL. See American Online
APDUs. See Application Protocol Data Units
API. See Application programming interface
Apple script, 100
Applets, 101–102
Application gateway, 238–239
Application layer, 36–38
Application programming interfaces (APIs), 20, 103–105
JDBC. See Java Database Connectivity
session tracking techniques, 177–195
Application Protocol Data Units (APDUs), 308
Architecture, of web, 39–40
ARIBA, 357, 359, 364
ASC X12 standard, 367
ASCII format, 42
ASDK. See Aglet Software Development Kit
ASP. See Active Server Page
Association rules, 278–279
Asymmetric Digital Subscriber Line (ADSL), 31
Asymmetric key encryption, 207
ATP. See Agent Transfer Protocol
Attributes, 68, 74–75, 79–80
Auction Universe, 386
Auctions, 384–387
Audience, 402, 405–406
Authentication, 50, 175, 187, 204–205, 216–225, 288
Authentication Header (AH), 230, 232–233
Authorization, 44, 352
Autonomous systems, 34–35
Availability, 204

B2B. See Business-to-business
B2C. See Business-to-consumer
Babel, tower of, 274
Backend systems, 40, 96, 125, 344
Bank of America, 375
Banking, 301–303, 375–377
Banner advertising, 392–396, 430
Banner Exchange, 395
Barnes and Noble, 337
BaseT standards, 31
Basic Encoding Rules (BER), 222
Basic Rate Interface (BRI), 31
Bastion host, 240–241
Beenz, 310
BER. See Basic Encoding Rules
Best-effort service, 29
Bezos, Jeff, 8
BGCOLOR attribute, 74–75
BGP. See Border Gateway Protocol
BizTalk (MS), 359, 369
Black box testing approach, 419
Blind signature, 289–299
Block cipher, 214
Bonus points system, 310
Bookstore, virtual. See Virtual bookstore
BORDER attribute, 74
Border Gateway Protocol (BGP), 35
Border properties, 84
Borland JBuilder, 109
BPR. See Business Process Reengineering
Brand name products, 335
BRI. See Basic Rate Interface
Bridge, 97
Brokers, 338–341
Browsers, 60, 87, 176, 423
Burn rate, 7
Business models, 352–367
B2B. See Business-to-business
B2C. See Business-to-consumer
electronic commerce and, 6, 17–21, 243
emergent models, 342–344
internet advertising. See Advertising
internet publishing. See Publishing
See also specific types, processes
Business Object, 404
Business-oriented (B2B) e-commerce, 315, 349–372
Business platforms, 273
Business process, defined, 343
Business Process Reengineering (BPR), 343
Business-to-business (B2B) systems, 3, 126, 159, 315, 349–372
auctions and, 386–387
INDEX

features of, 351–352
model, 8
Trading Process Network, 10
XML and, 368–370
See also specific companies, processes
Business-to-consumer (B2C) systems, 3
Amazon.com, 8–9
applications, 126
e-commerce, 8–9, 161
model, 6–8
Priceline, 10–11
web site, 176
See also specific companies, processes
Button, 413
Buyer-oriented systems, 15, 353–357
C code, 100
C++ code, 100, 271
C2C. See Consumer-to-consumer systems
CA. See Certification authority
Cable TV, 398
Cache control, 50
Caesar cipher, 207
Calvin Klein, 400
Capital costs, 321
Capture codes, 297
Cardholder, 288
Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA), 31
Carrier Sense Multiple Access with Collision Detection (CSMA/CD), 30–31
CartLogin, 447–449
CartServices, 455–459
Cascading style sheets (CSS), 82–86, 271
Cash-and-transfer, 301–303
Catalogs, 15, 341, 357
CBC. See Cipher block chaining
CDF. See Channel Definition Format
CDMA. See Code Division Multiple Access
CDNow, 329
CDPD. See Cellular Digital Packet Data
Cellular Digital Packet Data (CDPD), 32, 267
Central processing unit (CPU), 307
CERN. See European Particle Physics Laboratory
Certificate chain, 222–223
Certificate revocation list (CRL), 225
Certification authority (CA), 220–225, 289
CGI. See Common gateway interface
Challenge and response protocol, 224–225
Change cipher protocol, 247–248
Channel Definition Format (CDF), 273
Chat rooms, 396, 400
Checkbox, 80–81
CHECKED attribute, 80
Checkout system, 429
Checksum, 226
Chemdex.com, 360
Cipher block chaining (CBC), 214, 233
Circuit level gateway, 239–240
Cisco systems, 335, 357
CLASS attribute, 86
Click-through rate, 400
Clickshare, 310
Client-server programming, 95
Client-side image map. See Image map
Client-side programming, 55–94
CNET store, 318
COBRA, 177
Codd, E.F., 127
Code Division Multiple Access (CDMA), 32, 267
Code model, 342
Coin signing, 298
Collaborative filtering, 331
Collaborative supply chain management, 365–367
Commerceone.com, 363
Common Gateway Interface (CGI), 96, 98–100
Compartmentalized organization, 17
Compatibility, 96
Competitive pricing, 320
Compilation, of servlets, 109-110
Compiled languages, 100
Complexity-based measuring, 415
Components-based models, 342
Compression methods, 42, 51, 71, 247
Conceptual model, 342, 407
Concordia, 254
Concrete user interface objects, 62, 413
Condition, 412
Confidentiality, 204-205, 247, 288
Consistency, 58, 417
Consortiums, 364
Consumer-oriented e-commerce, 315-348
Consumer-to-consumer (C2C) systems, 9-10, 384-387
Container tags, 64
Content creation, 63
Content integrity, 51
Content, of web sites, 281
Content-type header, 44
Convenience, 320
Cookies, 48-49
    servlet fundamentals and, 110-115
    session tracking techniques and, 175-176
Core server-side programming tool, 105
CORRA programming techniques, 103
Cost-effectiveness, 96
Covisint.com, 364
CPM model, 399
CPU. See Central processing unit
Credit card payments, 287-298, 301
CRL. See Certificate revocation list
Cross-selling, 391
Cryptography. See Encryption
CSMA/CA. See Carrier Sense Multiple Access with Collision Avoidance
CSMA/CD. See Carrier Sense Multiple Access with Collision Detection
CSS. See Cascading Style Sheets
Customer account creation, 449-452
Customer service, 321, 357
Customization, 13, 15, 16, 320, 344
Cybershopping, 13
Data Encryption Standard (DES), 207-208
temporary, 213
Data mining, 277-281
Data networks, 5
Data validation, 57
Database connectivity, 125-158, 160
Database systems, 96-97, 126-127, 359
DCM. See Demand Chain Management
Decision trees, 279-280
Dedicated file (DF), 308
DELETE statement, 106, 129, 133, 142
Dell, 320, 335, 357
Demand Chain Management (DCM), 13, 351
Demand-driven model, 13
Demilitarized zone (DMZ), 242
Department stores, 317
Deposit-and-clear, 301
DER. See Distinguished Encoding Rules
DES. See Data Encryption Standard
DF. See Dedicated file
Dial-up modem, 30
Diffie-Hellman protocol, 209-210, 231, 246
Digital catalogue, 15
Digital certificates, 220, 223, 224
    authentication and, 220-221
    chain, 222-223
    hierarchy of, 223
    revocation of, 225
SET protocol and, 289
X.509 framework, 221-222, 225
Digital envelope, 292-293
Digital goods, vs physical goods, 15
DIGITAL phenomenon, 3-4
Digital signature, 297
Digital signature standard (DSS), 210, 217-219
Direct fund transfer, 303
Direct mailing, 318
Direct-selling, 15, 335–336
Directory services, 424
Disintermediation, 381
Disney, 384
Distance vector routing protocol, 34
Distinguished Encoding Rules (DER), 222
Distribution portals, 365
Divisibility, of payment, 286
DMZ. See Demilitarized zone
DNS. See Domain Name System
Document mark up, 273
Document object, 88
Document Object Model (DOM), 272, 276
Document Style and Semantics Language (DSSSL), 271
Document Type Definition (DTD), 268–269
DOM. See Document Object Model
Domain Name System (DNS), 36–37
Domino Go web Server, 104
Door-to-door sales, 318
Dot-decimal format, 32
Dotcoms, 7
Download time, 57, 419–420
DS. See Dual signature
DSS. See Digital Signature Standard
DSSSL. See Document Style and Semantics Language
DTD. See Document Type Definition
Dual home bastion host, 240
Dual signature (DS) method, 289–292
Dutch auction, 384
Dynamic pricing, 15
Dynamic Web Enterprises, 368
Dynamic web pages, 46–48, 98
Earnings per share (EPS), 7
eBay, 9–10, 386
E-business. See E-commerce
E-cash, 298–301
Ecashtchnologies.com, 298
ECB. See Electronic Code Book
ECC. See Elliptic Curve Cryptography
E-checks, 301–303
E-commerce
  advanced technologies, 21, 251–282
  advertising. See Advertising
  automation and, 5
  business models and, 6, 17–21, 243
  changes brought by, 13–14
  defined, 2–3
  development of, 14–18
  different perspectives on, 4–6
  electronic business model and, 17–18
  funding for, 16
  implementation of, 14–18
  key drivers of, 5
  myths about, 14–18
  physical commerce and, 2–3
  process-oriented approach, 344–345
  publishing and. See Publishing scenario
  retailing and, 11–13, 317–319
  security, 20
  servicing scenario. See Servicing scenario
  supply chain management scenario. See Supply chain management scenario
  three-layer model, 4
  traditional business model and, 17–18
types of, 6–8
  typical architecture of, 19
  web page building and, 16
  web programmers and, 16–17
See also specific applications, companies, processes
EDL. See Electronic Data Interchange
EDIFACT system, 367
Education, 378–379
EF. See Elementary file
Efficiency, 59, 96
Egghead.com, 17
Ehitax.com, 364
Electronic Code Book (ECB), 214
Electronic commerce. See E-commerce
Electronic data interchange, 2
Electronic Data Interchange (EDI), 351, 367
Electronic information, 5
Electronic payment. See Payment methods
Element Handler, 272
Elementary file (EF), 308
ElGamal algorithm, 213, 219
E-mail, 392
E-mails, 334–335
Embedded style sheets, 84–85
Emergent business model, 342–344
Empirical testing, 415, 417
Employment, 381
Encapsulating security payload (ESP) service, 230, 233–234
Encryption, 2, 21, 203–228
DES, 207–208, 213
Diffie-Hellman protocol, 209–210, 231, 246
dual signature, 291
Elliptic Curve Cryptography (ECC), 213–214
hybrid, 213
IDEA code, 208
key distribution problem, 209
private key encryption, 207–209
process of, 205
RSA methods, 210–213, 217–220, 291, 298
substitution, 207
transposition, 207
two types of, 207
End-to-end testing, 419
English auction, 9, 384
Enterprise Resource Planning (ERP) systems, 351, 370
Entertainment, 384
Entity header, 44
EPROM. See Erasable programmable read-only memory
EPS. See Earnings per share
Erasable programmable read-only memory (EPROM), 308
ERP. See Enterprise Resource Planning
E-R relationship, 127
Error handling testing, 419
Error recovery, 59–60
E-services, 373–388
categories of, 374–375
ESP. See Encapsulating security payload
ESPN, 399
Ethernet connections, 30–31, 57
Eurobanner, 394–396
European Particle Physics Laboratory (CERN), 38–39
Event handlers, 88
E-wallet, 311
Exception handling, 419
Exit button, 414
Expires header, 44, 50
eXtensible Markup Language (XML), 252, 267–277
applications of, 273–275
architecture for, 275–277
displaying documents, 271
Document Type Definition, 269
HTML and, 267–268
processing of, 271–273
programming interfaces, 271–273
recommended web sites for, 282
style sheets, 271
syntax of, 267–271
eXtensible Style Language (XSL), 271
Extent driven model, 255–256
External style sheets, 83–84
Feedback, 58, 417
File input field, 81
File transfer protocol (FTP), 36, 40
FIN. See Flow of Interaction Net
Financial advice, 384
Financial Services Technology Consortium (FSTC), 301
Firewalls, 236–242
Firing rules, 412
Flow of interaction, 61, 411
Flow of Interaction Net (FIN), 404, 407
Flowers, selling of, 328
FOBchemicals.com, 363
Fonts, 68, 84
INDEX

Ford Co., 335
Form, 79–82
  Javascript and, 89–92
  object, 88
  posting, 177
  validation, 86, 89–92, 432–434
Forums, 341
4C payment methods, 286–287
Fragmentation, 247
Frames, 75–79
Franchise stores, 317
From header, 44
Front end, 96
FSTC. See Financial Services Technology
  Consortium
FTP. See File transfer protocol
Funding, 16
Fuzzy audience, 402
General Electric, 357
Generalized e-stores, 317–318
GenericServlet, 105
GEOShops, 318
GET command, 41, 47, 51, 106
GIF. See Graphics Interchange Format
Gift products, 328
Gigabit Ethernet standard, 31
Global certification system, 224
Global economy, 3, 5
Global reach, 321, 391
Global System for Mobile
  Communication (GSM), 267
Goodman-Kruskall Tau, 279
Graph, directed, 279
Graphical User Interface (GUI), 96, 109
Graphics Interchange Format (GIF), 42, 71
Group-oriented pricing, 15
GSM. See Global System for Mobile
  Communication
GUI. See Graphical User Interface
Handshaking procedures, 244–247, 309
HEAD request method, 51
Headers, types of, 43–44
HFF. See Hidden Form Fields
Hidden Form Fields (HFFs), 81–82, 155, 161, 177
  session tracking techniques and,
    162–170
Hierarchical organization, 15
Hierarchical trust system, 223–224
Hit rate, 399
HMAC algorithm, 216, 233
Hop-by-hop basis, 51
Horizontal rule, 64, 68
Hotel Registration system, 407–410, 414
Hotjobs, 281
HTML. See Hypertext markup language
HTTP. See Hypertext transfer protocol
HTTPS. See Secure hypertext transfer
  protocol
Human resources, 370
Hybrid encryption, 213
Hyperlinks, 41, 71, 177, 275, 276, 421
Hypertext markup language (HTML), 20
  basic structure of, 64–66
  links, 69
  overview of, 63–64
  syntax of, 268
  tags, 252
XML and, 267–268
Hypertext transfer protocol (HTTP), 19, 27, 36, 40
  authentication scheme, 175
  entity header, 44
  overview of, 41–42
  request methods, 42–44, 51
  server response, 44–46
  servlet model, 105–106
  session tracking and, 175
  stateless protocol, 48, 98
  status codes, 45
  user identification, 175
  version 1.0, 41, 44–46
  version 1.1, 41, 49–51
I/O. See Input/output interface
IBM DB2, 368
IBM LMX converter, 369
IBM VisualAge, 109
IDEA. See International Data Encryption Algorithm
IETF. See Internet Engineering Task Force
If-modified-since header, 44
IKE. See Internet Key Exchange
Image map, 72–73
Images, 71–72
Inductive learning, 278
Information Age, 5
Information selling, 383–384
Informix, 134, 359
Inline style, 85–86
Input/output interface (I/O), 308
Input/output tests, 419–420
INSERT statement, 129, 130–131, 142
Inspection, 415
Integrated organization, 17
Integrated Services Digital Network (ISDN), 31
Integration, 367–371
Integrity, 204–205, 247, 288
Interactive entertainment, 384
Interactive servlet programs, 110–115
Interactivity, 61, 400, 411
Intercompany activities, 353, 367–368
Interface widgets, 62
International Data Encryption Algorithm (IDEA), 208
International Telecommunication Union (ITU) X.509 standard, 221
Internet, 2, 27–54
access methods, 31
application layer, 36–38
layered model, 29
link layer, 29–32
network architecture, 28–29
network layer, 32–35
next generation, 38
overview of, 28–38
routers, 28
security, 20
standards, 32
transit layer, 35–36
See also specific sites, systems
Internet advertising. See Advertising
Internet Engineering Task Force (IETF), 32, 244
Internet Key Exchange (IKE), 231
Internet Protocol (IP), 32, 229
classes of, 33
DNS, 37
IPv4, 38, 232
IPv6, 38
Internet Protocol Security (IPSec), 38
application of, 235–236
gateway, 233
header, 232
protocol, 230, 230–231
security association, 231
Internet security, 229–250
Internet Security Association and Key Management Protocol (ISAKMP), 231
Internet Service Provider (ISP), 28
Internetsecure, 310
Intracompany activities, 353, 370–371
Intradomain routing, 34
Inventory, 370
Investing, 377–378
Investment information, 383
IP. See Internet Protocol
IPowerB2B.com, 361
IPSec. See IP Security Protocol
ISAKMP. See Internet Security Association and Key Management Protocol
ISDN. See Integrated Services Digital Network
ISDN BRI. See Basic Rate Interface
Islands of trust, 224
ISP. See Internet Service Provider
Issuer, 288
ITU. See International Telecommunication Union

JADE. See Java-based Agent Development Environment
Java Applets, 56
Java-based Agent Development Environment (JADE), 262
Java Beans, 134
Java Cryptography Architecture (JCA), 103, 177
Java Database Connectivity (JDBC) advanced search engine, 126, 151-155
API, 126, 136-137, 177
cryptography, 103, 107
diagram of, 135
drivers, 134-136
infrastructure, 134-135
ODBC bridge, 97, 135, 137-138
perspectives, 134-137
programming, 103-106, 137-144
servlets and, 126, 151-155
Java Remote Method Invocation (RMI), 103, 177, 254
Java Security API, 177
Java Servlet Developments Kit (JSDK), 104
Java servlets, 20, 96, 98, 101-104, 160, 429
session tracking and, 98, 429
Java Virtual Machine (JVM), 103
Java Web Server, 104
JavaScript, 56, 86-92, 432-434
JC Penny, 318
JCA. See Java Cryptography Architecture
JDBC. See Java Database Connectivity
JIT. See Just-in-Time
Jobs, on internet, 381
Joint Photographic Experts Group
(JPEG), 42, 71
JPEG. See Joint Photographic Experts Group
JPNET, 368
JSDK. See Java Servlet Developments Kit
Just-in-Time (JIT) systems, 13, 350
JVM. See Java Virtual Machine
Key distribution problem, 209
Keyed MD5, 233
Keyword searches, 60, 423
Knowledge-based systems, 342
Language interface, 100
LANs. See Local area networks
Last-modified header, 44
Lastminute.com, 318, 325, 331-333
Law, 21
Layered architecture, 29, 134, 275
Layout, of page, 62
Legacy interference tests, 419-420
Linking, 69-71, 271, 344
anchor tag pair, 69
HTML document, 69
hyperlinks, 41, 71, 177, 275, 421
link layer, 29-32
routing protocol, 34
Lists, formatting, 68-69
Live Software, 104
Load-balancing, 202
Local area networks (LANs), 30
Location object, 88
Lockbox, 303
Login, 159, 185, 187, 429
Logistics, 370
Lossless compression, 71
MAC. See Message authentication code
MAGICS. See Mobile-Agent-based Internet Commerce System
MAGNET system, 253
Maintenance, repair and operating (MRO) goods, 353
Malls, 317
Many-to-many marketing, 391
Marketing, 15, 16, 390-392
Marketplace, vs marketspace, 15
Mass customization, 321, 344
Mass marketing, 15, 390
Master file (MF), 308
Matchmaking services, 379-383
Mathematical Markup Language
(MATHML), 273
MATHML. See Mathematical Markup Language
MD. See Message digest algorithms
MEB. See Mobile Electronic Business
Order information (OI), 289
Organization, 15, 17
Organization object model, 404
ORM. See Operating Resource Management
Overhead cost, of payment, 286

P3P. See Platform for Privacy References Syntax Specification
Packet filtering router, 237–238, 338
Page layout properties, 84
Paint Shop Pro, 71
Paragraph formatting, 67–68
Parsers, 276
PART system (Boeing), 360
PASS model, 346
Password methods, 80, 175, 219, 452–454
Payment systems, 285–314
4C methods, 286–287
anonymous, 298
authorization, 295–297
capture, 297–298
characteristics of, 286
gateway, 289, 295
instructions, 289
overhead cost, 286
Payword protocol, 305–306, 310
Peopleware, 16
Perceptual design, 62–63
Performance, 59
Perl script, 100, 103, 271
Permission marketing, 396, 398
Persistent connections, 50
Persistent cookies, 202
Personalization, 12, 341, 391, 396
Perspectives, 407
Petri nets, 407
Petsmart.com, 337
PICS. See Platform for Internet Context Selection
Pipelining, 50
Plaintext, 205
Platform for Internet Context Selection (PICS), 274
Platform for Privacy References Syntax Specification (P3P), 274
Plugins, 56, 369
PO. See Purchase order
Pop-up window, 413, 416
Port numbers, 35
Portals, 396
POST commands, 43, 47, 105–106
Practical Extraction and Report Language. See Perl
Priceline.com, 10–11, 337, 339–340
Pricing models, 323, 399–400
Prime numbers, 209, 211, 212
Private key encryption, 207–209
Private key-exchange key, 291
Private signature key, 294
Procter & Gamble, 400
Procurement, 16, 353, 356, 365
Product catalog, 357
Product configuration, 357
Product, customized, 13, 15–16, 320, 344
Product searching, 251
Production planning, 16, 370
Profit margins, 322
Program Resolver, 38
Programming, 16–17, 55–94
Proton system, 309, 311
Proxy authentication, 50
Proxy server, 41, 238–239
Public key encryption, 207, 210–214, 220
Public key-exchange key, 291
Publishing, 12, 310, 389–426
goals and criteria, 400–401
quality assurance, 414–423
usability testing, 414–419
web presence and visibility, 423–424
Purchase order (PO), 368
Purchase protocols, 293–295
Purchase requisition, 355
Purchasing circles, 331, 396
PUT command, 106
Quality assurance, 414–423

Radio button, 81
RAM. See Random-access memory
Random-access memory (RAM), 308
Range request, 50
RDBMS systems, 134
RDF. See Resource Description Framework
Read-only memory (ROM), 308
Record protocol, 247
Referer header, 44
Relational database systems, 126–134
Relative URL, 70
Replay attack, 219, 234–235, 294
Request for Comments
RFC 768, 35
RFC 950, 32
RFC 1034, 36
RFC 1035, 36
RFC 1700, 35
RFC 1825–1829, 230
RFC 1925, 42, 43
RFC 1928, 239
RFC 2104, 216
RFC 2109, 48
RFC 2401, 230, 232
RFC 2406, 234
RFC 2408, 231
RFC 2409, 231
Request HTTP/1.0, 44
Request methods, 43–44, 51, 114
Request signature, 305
Requests for Quotations (RFQs)
10, 354, 364
Resolver program, 38
Resource Description Framework (RDF), 274
Resources Reservation protocol (RSVP), 38
Response methods, 45–46, 116
Response time, 419, 420
Resume-writing, 384
Retailing, 11–12, 317–319
basic features of, 331
benefits of, 319–321
brokers, 338–341
direct selling, 335–336
e-mails, 334–335
features of, 341
future of, 341
generalized e-stores, 333–334
key success factors for, 322–324
models of, 324–341
specialization by function, 331–333
supplementary distribution channel, 336–338
Reverse aggregation, 363
Reverse auction, 384
RFC. See Request for Comments
RFQs. See Requests for Quotations
Rich Text Format (RTF), 271
RIP. See Routing Information Protocol
RMI. See Java Remote Method Invocation
Robot program, 42
ROM. See Read-only memory
Round-robin, 202
Routing, 32–35, 38
Routing Information Protocol (RIP), 34, 38
ROWSSPAN attribute, 75
RSA encryption, 210–213, 217–220, 291, 298
RSVP. See Resources Reservation Protocol
RTF. See Rich Text Format
SA Database (SAD), 232
SAD. See SA Database
Sample API for XML (SAX), 272–273
SAP R/3 system, 371
Save button, 414
SAX. See Sample API for XML
Scalability, 220, 223
Scaleable Vector Graphics (SVG), 274
Schemas, 275
SCM. See Supply-Chain Management
Screened firewall system, 242
INDEX

475

Scrip, 303–304
Scripted languages, 86, 100, 421
Search engines, 152–155, 323, 341, 423, 429, 438–445
SearchEngine.java, 152–155
Secret keys, 291
Secure Electronic Transaction (SET), 285–298
authentication system of, 289
CA and, 289
credit card payments, 287–289
four phases of, 293
RSA keys and, 291
Secure Hash Algorithm (SHA), 216
Secure hypertext transfer protocol (HTTPS), 40
Secure socket layer (SSL), 213, 230, 242–248, 420
alert protocol, 247–248
change cipher protocol, 247–248
handshake protocol, 244–247
record protocol, 247
Security, 96, 421
associations, 231–232
authentication, 204–205
cryptographic, 204–205
e-commerce and, 20, 204
encryption and. See Encryption
integrity, 204–205
internet and, 20
IPSec and, 231
network layer, 229
of payment, 286
testing, 419
transport layer, 229
Security First Network Bank (SFNB), 375
Security Policy Database (SPD), 232
Select menu, 82
SELECT statement, 129–130
Selection menu, 169
Selectors, 232
Sell-side e-commerce, 357–360
Semantics, 268
“Send and Pray” service, 29
Server logs, 420
Server-side applications (SSA), 96, 106, 160
CGI and, 96, 98–100
client-server programming, 95–96
database connectivity, 125–158
servlet fundamentals, 95–124
session tracking and, 159–202
testing, 421
See also Servlets
Servicing, 12
Servlet APIs
overview of, 103–105
Servlet book query
advanced, 144–151
JDBC example for, 137–144
Servlet-enabled web servers, 104
ServletRequest, 105
Servlets, 107–109
APIs, 20, 103–105, 126, 151–155
applets and, 102
application of, 102
basics, 104
common methods for, 179–180
compilation of, 109–110
execution of, 109–110
interactivity, 110–115
Java and, 101–103
methods used to create, 106
request methods, 103, 114
response methods, 116
server-side programming, 95–124
session-tracking. See Session tracking
SimpleCart, 185
Session key, 213
Session tracking, 20, 106, 159–177
advantages and disadvantages, 176
API and, 177–195
authentication, 185
comparison of methods, 176–177
cookies and, 175–177
database connectivity and, 125–158
example, 180–185
hidden form field and, 162–170
Session tracking (cont.)
  HTTP user identification and, 175
  server-side programming and, 159–202
  servlets, common methods for, 179–180
  shopping cart example, 185–195
  shopping in VBS, 160–161
  types of, 98
  URL rewriting and, 162–170
SET. See Secure Electronic Transaction
SFNB. See Security First Network Bank
SGML. See Standard Generalized Markup Language
SHA. See Secure Hash Algorithm
Shopping, 126, 161
Shopping cart, 185, 341, 429
  basic functions of, 195–196
  contents of, 195
  implementation issues, 161
  object, 161
  servlet session tracking API, 185–195
  session tracking, 185–195
  VBS example, 195–201, 447–449
Signature, blind, 289–299
SimpleCart program, 187–189
Site hosting, 63
Site owner, 421
SIZE attribute, 68
Sliding window mechanism, 35, 234, 235
Smart cards, 306–309
SMIL. See Synchronized Multimedia Integration Language
SOCKS protocol, 239
Software engineering, 341–342
Software structure model, 342
SPD. See Security Policy Database
Specialized stores, 317–318
Spiders, 423
SQL. See Structural Query Language
SSA. See Server-side applications
SSL. See Secure socket layer
Staffing, 351
Standard Generalized Markup Language (SGML), 64
Standards, BaseT, 31
Start tag, 273
State tracking. See Session tracking
Static web page, 99
Status codes, 45
Stickiness, 341
Stock evaluation, 383
Stock trading, 377–378
Store layout, 322–323
Stream cipher, 214
Stress load testing, 419
Supplementary distribution channel, 336–338
Supply-Chain Management (SCM), 13, 351, 365–367
Supply-driven model, 13
SVG. See Scaleable Vector Graphics
Sybase, 134
Symmetric key encryption, 207–209
Synchronized Multimedia Integration Language (SMIL), 274
Syntax, 268
System feedback, 58
System operation, 414
System testing, 419–421
Tabican system, 253
Tables, 73–75
Tags
  attributes and, 89
  formatting and, 68
Tahiti server, 254–255
Targeted marketing, 321, 344
Taxation, 21, 352
TCL code, 100
TCP. See Transmission Control Protocol
Telemarketing, 318, 390
TELNET, 36, 41, 338
Testing, system, 418-421
Text formatting, 66-69
Textarea, 82
Textbox, 80
Thematic web sites, 396, 398
Three-i requirements, 307
Three-tier model, 4, 5, 19, 96-98
Timing markers, 415
Token position, 412
ToysRus.com, 337
TPN. See Trading Process Network
TRACE command, 106
Trading Process Network (TPN), 10
Traditional model, 17-18, 321-322
Transferability, of payment, 286
Transmission Control Protocol (TCP),
29, 35-36
Transport layer, 35-36
Transport mode, 234
Transposition, encryption and, 207
Travel services, 379-381
Tree structure, 37, 279
Trust system, hierarchical, 223-224
Tunnelling, 233, 236, 338
UDP. See User Datagram Protocol
UI. See User interface
Uniform resource locator (URL), 40-41
absolute, 70-71
hyperlinks, 41
rewriting, 162-165
routing, 202
session tracking and, 162-170
Unique visitors, 399
Universal Resource Identifier (URI), 276
Unix shell scripting, 100
Unwired Planet, 263
UPDATE statement, 129, 132, 142
URL. See Uniform resource locator
Useability
advertising and, 414-419
automated tests, 417
factors in, 57-59, 415, 417
publishing and, 414-419
user interface and, 414-419
User-agent header, 44
User Datagram Protocol (UDP), 35
User interface (UI), 57
abstract user interface object, 407-411
flow of interactions, 411-414
functional and system testing, 419-421
quality assurance, 414-423
usability testing, 414-419
web design and publishing, 407-414
widgets, 62
User login program, 185
User preregistration, 176
Username, 175, 219
VALIGN attribute, 75
Value-added networks (VANs), 2, 321,
367
Value America, 333-334
Value chain concept, 6
VANs. See Value-added networks
VBS. See Virtual bookstore
Vending machines, 318
Verification path/certificate chain,
222-223
Virtual bookstore (VBS), 105-109
access control, 445-447
advanced search, 126, 151-152,
442-445
book ordering, 127
CartLogin, 447-449
CartServices, 455-459
category search, 441-442
core server-side programming tool, 105
customer account creation, 449-452
exercises for building, 427-462
form validation, 432-434
homepage design, 430-432
Virtual bookstore (cont.)
Javascript and, 432-434
JDBC and, 105, 137-144, 151-155
mobile agents, 256-261
ordering system, 102, 127-128
password changes, 452-454
payments in, 310-311, 459-462
relational database and, 127-128
search engines, 126, 144-152, 434-445
servlets and, 105-107, 137
session tracking, 163
shopping cart, 163, 195-201, 447-449
shopping in, 126, 160-161, 428-429, 455-459
simple, 163
system overview, 429-430
virtual shopping, 455-459
See also specific functions, software
Virtual marketplace, 126, 352, 360-365
Virtual Private Network (VPN), 235-236, 311
Virtual reality, 341
Visa Cash, 309, 311
Visibility, 322-323
Visits, defined, 399
Visual Basic, 100
Volumes, of trade, 351-353
Voyager, 254
VPN. See Virtual Private Network

W3C Micropayment Working Group, 311
WAE. See Wireless Application Environment
Waiting condition, 412
Wall Street Journal, 399
Wallet software, 300
Walmart, 318, 334
WAP. See Wireless Application Protocol
Warehousing facilities, 350
WDP. See Wireless Datagram Protocol
Web. See World Wide Web
Web pages. See Web site design
Web Server Farm, 202

Web site design, 421
audience-centered design, 61
categories of systems, 402-403
CGI programming, 99
client-side programming, 61
client-side programming and, 61-62
content creation and, 63
dynamic, 46-48
electronic commerce and, 16
generation of, 46-48
information requirements, 61
logical design of, 61-62
methodologies for, 401-407
overview of, 403-407
perceptual design and, 62-63
site hosting and, 63
web page design and, 61-62
Webcrawlers, 423
Weights, 412
Well-formed constraints, 270
Widgets, 62, 413
WIDTH attribute, 68, 74-75
Window widget, 413
Wireless Application Environment (WAE), 265-266
Wireless Application Protocol (WAP), 251-252, 263-267
Wireless Datagram Protocol (WDP), 266, 267
Wireless links, 31
Wireless local area network, 31
Wireless Markup Language (WML), 264
Wireless session protocol (WSP), 263, 266
Wireless Telephony Applications (WTA), 264
Wireless Transaction Protocol (WTP), 266
Wireless Transport Layer Security (WTLS), 266
WML. See Wireless Markup Language
World Wide Web, 2, 27-54. See also Internet; Web
World Wide Web (WWW)

- advertising. See Advertising
- architecture, 39–40
- brief history of, 38–39
- data mining and, 277–281
- programming, 15–17, 55–94
- publishing. See Publishing
- usability, 414
- user interface. See User interface

See also specific applications, sites, functions

WSP. See Wireless session protocol

WTA. See Wireless Telephony Applications

WTLS. See Wireless Transport Layer Security

WTP. See Wireless Transaction Protocol

W.W.Grainger, 361

WWW. See World Wide Web

X.500 naming system, 222
X.509 standard (ITU), 221, 225, 289
Xlink, 268, 271
XML. See eXtensible Markup Language
XOR function, 217
Xpointer, 271
XSL. See eXtensible Style Language

Yahoo!Store, 318, 335

Zenus Web Server, 104