11-1-1999

Electronic Commerce on Business Application

Fang Xiao

University of Nebraska at Omaha

Follow this and additional works at: https://digitalcommons.unomaha.edu/studentwork

Recommended Citation

https://digitalcommons.unomaha.edu/studentwork/1513
Electronic Commerce on Business Application:

A Thesis
Presented to the
Department of Computer Science
and the
Faculty of the Graduate College
University of Nebraska
In Partial Fulfillment
of the Requirements for the Degree of
Master of Science

University of Nebraska at Omaha

By
Fang Xiao

November, 1999
THESIS ACCEPTANCE

Acceptance for the faculty of the Graduate College,
University of Nebraska, in partial fulfillment of the
Requirements for the degree of Master of Science,
University of Nebraska at Omaha

Committee

[Signatures and names]

Chairperson

Date  11-11-1999
Acknowledgements

I would like to express my sincere appreciation to my thesis advisor Professor Peter A. NG, for his kind guidance, time and support. I would also like to say thank you to my supervisory committee Dr. Qiuming Zhu, Dr. Bin Chen for their suggestions, help and cooperation.
Abstract

For some time now, Whole size business enterprises have used electronic commerce to conduct their business activities. The On-Line store is one of a number of popular web applications. This thesis focuses on one of the emerging On-Line store technologies which is known as Server-Side Java Application.

In early 1960, private networks were dedicated for the use of electronic data interchange (EDI) and electronic funds transfer (EFT) in banking business. Recently, however, with the increased awareness and popularity of the Internet, electronic commerce has come to encompass individual consumers as well as businesses of all size. For most people, electronic commerce is defined as the buying and selling of products and services over the Internet, but there are many more aspects. We will discuss different definitions of electronic commerce in this paper.

The most popular Web applications are the On-Line Stores. Most of such applications involve functions as searching for product information, ordering products, paying for goods and services, and for providing online customer service. A Web server, running continuously, accepts user’s connection; gets user requests and forward requests to business procedure; retrieves data (answers to user requests) from business procedure and sends it back to the user.
An on-line store system is different from a traditional web publishing. It needs a customer’s interactive communication and frequent data update. The traditional HTML page does not satisfy the customer’s needs because all it does is simply publish static contents. More complex technologies are required in order to produce dynamic contents.

There are a few technologies that can be used for implementing the on-line stores. Common Gateway Interface (CGI), Microsoft Active Server Page (ASP) and Server-side Java application are the most popular technologies. Introducing the Server-Side Java application is one of the latest and most exciting trends in Java programming. The Server-side Java application has many advantages over the other technologies.
# Table of Contents

Chapter 1  Introduction .................................................................1

Chapter 2  Electronic Commerce..................................................2

  2.1 What Is Electronic Commerce? ..............................................2

  2.2 The Importance of Electronic Commerce.............................3

  2.3 Types of Electronic Commerce.............................................3

  2.4 Why Most Internet Based Electronic Commerce Is In

   The Business-To-Consumer World? ..........................................6

  2.5 Current State of Business-To-Consumer E-Commerce.............7

  2.6 The Structure of The On-Line Store.....................................8

Chapter 3  Technologies For On-Line Store System.......................11

  3.1 Traditional Web Application – Common Gateway Interface(CGI)...11

  3.2 Active Server Page...............................................................12

  3.3 Server-Side Java Application.................................................14

Chapter 4  Design of On-Line Bookstore Using Java Servlets:

  A Practical Verification............................................................18

  4.1 The Design of Home Page of Tiny Book Store.......................18

  4.2 Design of Search by Different Requests...............................27

  4.3 Servlets Chaining To Process Requests In Different Levels........28

  4.4 Database Connectivity and Search from Database...................29
List of Figure

Figure 2.6.1 The Structure of An Online Store.................................................................10
Figure 3.2.1 Handle Requests Flow Chart.................................................................13
Figure 3.3.1 HttpService of The Java Web Server Using Servlets............................15
Figure 4.1.1 The Home Page of Tiny Book Store.......................................................22
Figure 4.1.2 One of The Searching by Author Pages..................................................26
Figure 4.4.1 One of The Outputs of Servlet AuthorProces.........................................34
Figure 4.5.1 A Page Show All Information About A Book...........................................39
Figure 4.5.2 A Page Show Items In The Shopping Cart After A User
Added Books To The Shopping Cart.......................................................................40
Figure 4.5.3 A Page Show Items In The Shopping Cart After A User Deleted A
Book From The Shopping Cart .............................................................................42
Figure 4.6.1 A Check Out Page.....................................................................................43
Figure 4.7.1 An Invoice Page.......................................................................................48
Chapter 1

Introduction

As we step in the year 2000, we can no longer look to the past as the guide for the future. A combination of business, social, and technical possibilities drives the rapid progress in electronic commerce.

Electronic commerce can offer a company both short-term and long-term benefits. Not only can it open new markets, enabling a business firm to reach new customers, but it can also make it easier and faster for the firm to do business with their existing customer base. Moving business practices, such as ordering, invoicing, and customer support, to network-based systems can also reduce the paperwork involved in business-to-business transactions. When more of the information is digitized, it allows that business firm to focus on meeting their customer’s needs. Tracking customer feedbacks, and presenting customer solutions for their clientele are just some of the opportunities that can stem from electronic commerce.
2.1 What Is Electronic Commerce?

For some time now, large business enterprises have used electronic commerce to conduct their business-to-business transactions. In early 1960, private networks were dedicated for the use of electronic data interchange (EDI) and electronic funds transfer (EFT) in the banking industry. Recently, however, with the increased awareness and popularity of the Internet, electronic commerce has come to encompass individual consumers as well as businesses of all size.

To many, electronic commerce is defined as the buying and selling of products and service over the Internet. But there are many other aspects. Depending on whom you ask, electronic commerce has different definitions.

From a communications perspective, electronic commerce is the delivery of information, products/services, or payments via telephone lines, computer networks or any other means. From a business process perspective, electronic commerce is the application of technology toward the automation of business transactions and workflow. From a service perspective, electronic commerce is a tool that addresses the desire of firms, consumers, and management to cut service costs while improving the quality of goods and increasing
the speed of service delivery. From an online perspective, electronic commerce provides the capability of buying and selling products and information on the Internet and other online services [16].

These definitions are valid with their perspectives. It is a matter of which lens is used to view the electronic commerce landscape. Broadly speaking, electronic commerce emphasizes the generation and exploitation of new business opportunities by “generating business value” or “doing more with less.”

2.2 The Importance of Electronic Commerce

As we approach the year 2000, we can no longer look to the past as guide to the future. In the face of strong market forces created by electronic commerce and mounting competition, corporations can no longer plod along historical tracks or seek the preservation of the status quo. Companies are discovering that old solutions do not work with new problems. The business parameters have changed, and so have the risks and payoffs.

2.3 Types of Electronic Commerce

From a broad view, there are three distinct general classes of electronic commerce applications: Inter-organizational (business-to-business), intra-organizational (within business), and customer-to-business application. [16]
2.3.1 Inter-Organizational Electronic Commerce

Inter-organizational electronic commerce makes business more convenient on the following aspects:

Firstly, e-commerce can improve supplier management, since electronic applications can reduce the processing costs and cycle times for each purchase order, and it can reduce the number of people who work on purchase orders.

Secondly, e-commerce can improve inventory management, since electronic applications make the business partners can fast exchange information and easily track their documents to ensure that they were received. This not only can eliminate out-of-stock occurrences but also can reduce inventory and improve inventory turn.

Thirdly, e-commerce can improve distribution management, since electronic applications can make the business partners receive the shipping documents as soon as possible and enable better resource management by ensuring that the documents contain more accurate data.

Fourthly, e-commerce can improve channel management, since electronic application can let the business partner post the information to electronic bulletin boards. This can eliminate repeated telephone calls and countless labor hours.

Fifthly, e-commerce can improve payment management, since the business partner are linked by network, so the payments can be sent and received electronically. Electronic payment is more accurate and fast than traditional payment.

2.3.2 Intra-Organizational Electronic Commerce
Intra-organizational electronic commerce facilities the following business applications:

- **Workgroup communications.** Electronic applications enable managers to communicate with employees using electronic mail, videoconferencing and bulletin boards.
- **Electronic publishing.** Electronic applications enable companies to publish information by using tools such as the World Wide Web. On-line publishing can show information immediately and clearly and reduce costs for printing and distributing documentation.
- **Sales promotion.** Electronic applications improve the flow of information between the production and sales forces, and between the firms and customers. This can make companies have greater access to market and competitor information.

Today, Intranets are primarily set up to publish and access vital corporate information. Some of the most common types of information are: Human Resources information, employee communications, product development and project management data, internal catalogs, sales support data, equipment and shipment tracking, and accessing corporate database.

### 2.3.3 Consumer-To-Business Electronic Commerce

In consumer-to-business transactions, customers learn about products through electronic publishing, buy products with electronic cash and other secure payment systems, and even have information goods delivered over the network.
From the consumer’s perspective, there are three transactions:

1. Electronic applications enable consumers to communicate with each other through electronic mail, videoconferencing, and news groups.
2. Electronic applications enable consumers to manage investments and personal finances using online banking tools.
3. Electronic applications enable consumers to find online information about existing and new products/services.

Consumers consistently demand greater convenience and lower prices. Electronic commerce provides consumers with convenient shopping methods, from online catalog ordering to phone banking, both of which eliminate the costs of expensive retail branches. Electronic commerce facilitates factory orders by eliminating many intermediary steps, thereby reducing manufacturers’ inventory and distribution costs, and indirectly providing consumers with lower prices.

2.4 Why Most Internet Based Electronic Commerce Is In The Business-To-Consumer World?

For the Business-to-Business e-commerce, the lack of well-accepted standards is hindering the success in promoting Business-to-Business electronic commerce solutions[5]. VAN (Value-Added Networks) EDI (Electronic Data Interchange) based solutions are only accessible to large organizations due to the cost factor. Corporate
buyers and suppliers, large and small, are looking for Internet based solutions to streamline the procurement procedures and to reduce the cost of establishing trading relationship and the trading transactions. Such demands put forward some fundamental challenge on issue like trust infrastructure on the Internet, standards and inter-operability etc.

2.5 Current State of Business-To-Consumer E-Commerce

Business activity on the Internet is currently limited to publicizing the business opportunity and to catalog based sales, but it will rapidly expand to include the negotiations conducted to settle the price of the goods or commodities being traded.

These negotiations are currently conducted by human intermediaries through various forms of auctions, bidding systems for awarding contracts, and brokerages. The role of the intermediaries can now be performed by Internet trading applications at a fraction of the cost[4]. Trading on the Internet allows a business to reach a large number of potential customers and suppliers in a shorter time and a lower cost than possible by other modes of communication, and to settle business transaction with lower cost overhead in a shorter time.

Auctioned or brokered sales are the norm in business world for negotiating trades of large monetary value. But consumer sales and small-scale purchases have used the fixed price
mode, perhaps because of the high overhead cost of using the auction or brokerage method. The new economics of the Internet will make auctions popular in consumer and small business transactions also.

Auctions are just one form of business negotiations. Other examples are competitive bids for procurement, brokerages/exchanges/cartels, and two party negotiations.

Nowadays, the free-market economy of software agents is in the embryonic stage. But in the near future, the software agents will populate the Internet and provide, trade, and use a rich variety of information goods and services in an open, free-market economy. In a free-market economy of software agents, information is produced, traded, and consumed by vast numbers of autonomous, self-motivated agents. An essential task in such an economy is the retailing or brokering of information: gathering it from the right producers and distributing it to the right consumers[10]. The overall research goal is to characterize and understand the dynamic behavior of information economies: very large open economies of automated information agents that are likely to come into existence in the Internet.

2.6 The Structure of The On-Line Store
Nowadays, there are thousands of on-line stores running on the Internet. A few well-known ones are amazon.com and bookpool.com. Most of such applications involve common functions which are as following:

- Searching for product information
- Ordering products
- Paying for goods and services
- Providing online customer service

So, they have a similar structure as shown in Figure 2.6.1. The system contains the following components:

1. A web server, running continuously, accepts user’s connection; gets user requests and forward requests to business procedure; retrieves data (answers to user requests) from business procedure and send it back to user.

2. Business procedure usually consists of a user identification procedure like credit card check; user request analysis procedure, analyzing requests, calculating, discomposing (if necessary) and dispatching request to different data source component. Also, in some case, business procedure is responsible to compose data obtained from different data sources into user-readable format.

3. Database is the place to hold relevant data. These data not only serve as a data source, describing commercial goods and prices, but can be used for data mining to improve store service quality as well. Database is updated and maintained by both customers’ input and system applications. A customer’s input usually includes order information.
4. System applications can include a search engine, running all the time and searching relevant information from the internet; data mining application deals with mining and analyzing customer and sale data for management use; database maintaining application handles update data when product information changes. Varying from system size to system size, not every system has clear/distinct boundaries between these components. The system can be 3-tired or n-tired, but these systems all operate in similar ways.
Chapter 3

Technologies For On-Line Store System

An on-line store system is different from traditional web publishing. It needs customer’s interactive communications and frequent data update. The traditional HTML page is insufficient because all it does is provide static content publishing. Other technologies are required in order to implement the on-line store.

Currently, there are three mainstream technologies which are widely used: Common Gateway Interface (CGI), Microsoft Active Server Page (ASP) and Server-side Java application. We briefly explore these technologies here.

3.1 Traditional Web Application - Common Gateway Interface (CGI)

The common Gateway Interface, normally referred to as CGI, is one of the first practical techniques for creating dynamic content. With CGI, a web server passes certain requests to an external program. The output of this program is then sent to the client in place of a static file. The advent of CGI makes it possible to implement all sorts of new functionality in web pages, and CGI quickly become a defacto standard, implemented on dozens of web servers. Even though a CGI program can be written in almost any language, the Perl programming language is the predominant choice. However CGI have some distinct disadvantages.
Firstly, CGI program is transient. A request is made each time to a CGI program. It must be loaded and executed by the Web server. When the CGI program is complete, it is removed from memory and the results are returned to the client. All program initialization (such as connecting to a database) must be repeated each time a CGI program is used.

Secondly, when a server receives a request that accesses a CGI program, it must create a new process to run the CGI program and then pass to it, via environment variables and standard input. Every bit of information might be necessary to generate a response. Creating a process for every such request requires time and significant server resources, which limits the number of requests a server can handle concurrently.

Thirdly, a CGI program cannot interact with the web server or take the advantage of the server’s abilities once it begins execution because it is running in a separate process.

3.2 Active Server Page

As shown in figure 3.2.1, the following list walks you, step-by-step, through what happens if the client requests the page xyz.asp.

- The client requests the page xyz.asp from the server. The Web server checks the file extension to see whether a special program (such as the Active server Pages engine)
must be invoked to process the request. If there's a .asp extension, the Web server
determines that it should invoke ASP to process this page.

Figure 3.2.1 Handle Requests Flow Chart.

- If this page has never been requested before or has been changed since the last
  request, it must be parsed and the syntax checked and then be compiled by the Web
  server. Otherwise, the page might be read from a cache of recently processed pages,
  which aids in performance. During the parsing process, the HTML and scripting code
are separated. IIS (Internet Information Server) determines which scripting engine is responsible for which part of the script and delegates the work of syntax checking and compiling to the proper scripting engine (such as VB Script).

- Now the code is executed by the scripting engine using resources from IIS, which is hosting the scripting engines. All objects that the language engine cannot handle are requested by the IIS, which is also responsible for handling inputs and outputs for the external ActiveX objects that are created and used inside the script. If it is not able to supply the object, an error is generated.

- Script output and static HTML code in the ASP file are merged.

- The final HTML is sent back to the user in an HTTP response

The important point about ASP is that the Script runs entirely in a server to protect the intellectual property by shielding it from browser which are different from the account for when writing client-side code. The entire server-side code is processed and only plain HTML – or whatever kind of content you chose to generate – is sent to the client.

**3.3 Server-Side Java Application**

The rise of server-side Java applications is one of the latest and most exciting trends in Java programming. A Servlet can be thought of as a server-side applet. Servlets are loaded and executed by a Web server in the same manner that applets are loaded and executed by a Web browser.
As shown in Figure 3.3.1, a servlet accepts requests from a client (via the Web server), performs some task, and returns the results.

The client (mostly likely a Web browser) makes a request via HTTP. The Web server receives the request and forwards it to the servlet. If the servlet has not yet been loaded, the Web server will load it into the Java virtual machine and execute it. The servlet will receive the HTTP request, perform some type of process and return a response back to the Web server. The Web server will forward the response to the client.

Figure 3.3.1: HttpService of The Java Web Server Using Servlets

Servlets have distinct advantages over CGI program.

- Servlets are persistent. Servlets are loaded only once by the Web server and can maintain services (such as a database connection) between requests.
- Servlets are fast, since servlets only need to be loaded once.
Servlets are platform independent. Servlets are written in Java, and conform to a well-defined and widely accepted API. They are highly portable across operating systems and across server implementations. A servlet can be developed on a Windows NT machine running the Java Web Server and later can be deployed effortlessly on a high-end Unix server running Apache.

Servlet portability is not the stumbling block it so often is with applets. There are two reasons. Firstly, servlet portability is not mandatory. Unlike applets, which have to be tested on all possible client platforms, servlets have to work only in the server machines that you are using for development and deployment. Unless you are in the business of selling, your servlets, you don’t have to worry about complete portability. Secondly, servlets avoid the most error-prone and inconsistently implemented portion of the Java language: the Abstract Windowing toolkit (AWT) that forms the basis of Java graphical user interfaces.

Servlets are extendable. Servlets that are written in Java brings all of the other benefits of Java to servlet. Java is a robust, object-oriented programming language, which can be extended easily to suit our needs.

Servlets are secure. The only way to invoke a servlet from the outside world is through a Web server. This brings a high level of security, especially if the Web server is protected behind a firewall.

Servlets can be used with a variety of clients.
Servlets are tightly integrated with the server. This integration allows a servlet to cooperate with the server. For example, a servlet can use the server to translate file paths, to perform logging and to check authorization.

Servlets are quite flexible. An HTTP servlet can be used to generate a complete web page; it can be added to a static page using a <SERVLET> tag in what's known as a server-side include; and it can be used in cooperation with any number of other servlets to filter content in something called a servlet chain.
Chapter 4

Design of On-line Bookstore Using Java Servlets: A Practical Verification

Server-side Java application is one of the latest technologies used to implement an on-line store system. Servlet is a set of classes that come up from Java language. It is designed to be used for server-side Java applications. There are many techniques people can use when they utilize servlet, such as Server-Side Include (SSI), Servlet Chaining, Applet and Servlet communication, etc. We tried to use as many such techniques as necessary in our implementation to demonstrate the advantages of servlets.

The demonstration system module is same as the one we mentioned above Figure 3.3.1. We use this module to show how such system works.

The system, called Tiny Book Store, emulates an on-line bookstore selling computer books. It has the full functionality of other on-line stores. The only difference between this system and the other on-line systems is the database size. We don’t want to spend much developing time for database maintaining part, because this part is similar to ordinary database maintaining and it is an independent subsystem. The Tiny Book Store mainly consists of three modules: user interface, business procedure, and data access.
4.1 The Design of Home Page of Tiny Book Store

The home page – welcome page – is designed using the Server-Side Include technique. That is, embedding a servlet inside HTML file with a specific HTML tag <SERVLET>. In our case, this servlet when invoked can run a query on database, via data access objects, to retrieve most recent top 5 best-seller books and then returns the retrieved data to the place within the page where the <SERVLET> tag was. As long as the data in database update timely, it is guaranteed that customer will get the most recent data.

The embedded servlet is invoked by Java Web Server before the page is sent back to client. Java Web Server processes any HTML file having .shtml extension and invokes the embedded servlet marked by <SERVLET> tags.

Server-Side Includes are useful when a page is primarily static but contains a few distinct dynamic portions. One of the servlets produces the “Five Hot Books” which is shown in the following example.

Example 4.1.1: A Part of Code of welcome.shtml

......

<p>
<h3><strong><font color="ff6347">Welcome to the</font></strong></h3>

TinyBookStore.com</i></font></strong></h3>
The servlet EchoServletTag searches database according to some business rule and gets data from data access objects to produce the Five Hot Books. The following example is the sample code.

Example 4.1.2: A Part of Code of The EchoServletTag.java

```java
public void service(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException {

    res.setContentType("text/html");
    PrintWriter out = res.getWriter();

    try {
        Statement stmt = con.createStatement();
```
ResultSet rs = stmt.executeQuery("select * from book where title like '%Programming%'");

int count = 1;
out.println("<dl>");
while (rs.next() && count < 6) {
    out.println("<dt><b>" + count + ". </b>"");
    String isbn = rs.getObject("ISBN").toString();
    out.println("<a href=/servlet/DetailContent/" + isbn + ">");
    out.println(rs.getObject("Title").toString() + "</a></b>"));
    out.println("<dd><nobr><font color=#990033>Usually ships in 24 hours</font></nobr>");
    out.println("<br>" + rs.getObject("Author").toString());
    out.println("/ " + rs.getObject("Publisher").toString() + "/ " + rs.getObject("PublicationDate").toString());
    out.println("<br> Our Price: " + rs.getObject("Price").toString());
    out.println("<br><br>");
count++;

}

out.println("</dl>");

}

catch (SQLException e) {
    e.printStackTrace();
}

......

}

The servlet EchoServletTag, by the business rule that searching a specific table, sends a request query through data access object, stmt, and gets search result through another data access object, rs. It then passes data to the page. The content between <servlet> and </servlet> tags in the HTML of home page is replaced by the search result at run time.

The center part of the sample result page in Figure 4.1.1 is an example.
Figure 4.1.1: The Home Page of Tiny Book Store

From this practice, we know Server-Side Include not only can produce dynamic portion of the page but also prevent from other people spy into your code. Since Java servlet runs entirely on a server, thus protecting your intellectual property and shielding you from the browser. This means, what the client sees can be totally different from what the server does. From the client side, it is difficult to figure out how the server works.

The left-hand menu items are implemented by using a characteristic of servlet – extra path information or virtual path. The extra path information can be used as a kind of parameter passed to a servlet. In general, this extra path information is used to indicate a
file on the server that should use for something. We use this feature here in an abnormal way to manage customer requests and process these requests in a single place.

Consider a traditional method. When a menu item is selected by a customer, it redirects the customer to another page where it hyperlinks to. If we have several ten items, we need to pre-create several ten pages. This is the disadvantage of static HTML pages. However, with servlet which can dynamically generate page content, we can direct all the customer requests to a central servlet and process the requests there according to some condition (parameters) and then generate the page which customer need to see. In this way, we only need one place to hold the code, to save server’s space. And this place is easy to maintain.

The following is how it was doing for the Tiny Book Store.

The Author item uses the following hyperlink:

```html
<A href="/servlet/ReqCollect/author">Author</A>
```

The ReqCollect is the name of a servlet. We pass the “author” in the hyperlink as a parameter to ReqCollect. When the Author item is selected, ReqCollect is invoked by Java Web Server. ReqCollect generates a html page to let a user input author search condition. The ReqCollect code is listed below.
Example 4.1.3: A part of Source Code of ReqCollect.java

```java
public class ReqCollect extends HttpServlet {
    
    ......
    
    protected final String AUTHOR = "author";
    
    ......

    public void service(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {
        
        PrintWriter out = res.getWriter();

        String path = req.getPathInfo();
        if (path == null) {
            // error handling
        }
        else {
            // Create standard header
            CreateHeader(out);

            // Create left side menu
            CreateLeftMenu(out);
```
path = path.substring(1);
if (path.equalsIgnoreCase(CATEGORY)) {
    // Create category request collect

    ......
}
else if (path.equalsIgnoreCase(AUTHOR)) {
    // Create author request collect
    out.println("<font color=#CC6600 face=verdana,arial,helvetica size=+1><b>Enter Author</b></font>");
    out.println("<br> You can enter author's full name, first name or last name.<p>");
    out.println("<form method="POST" action="http://localhost:8080/servlet/AuthorProcess" id=form1 name=form1>");
    out.println("<table border=0>");
    out.println("<tr> <td valign=middle align=left><tt><b>Author:</b></tt></td> ");
    out.println("<td valign=middle align=left><input type="text" name="author" size=40 value=""/></td>");
    out.println("</tr>");
}
out.println("<tr> <td colspan=2> <font size=-1>\n");
out.println("<center> <input type="radio" name="author_mode" value="exact" checked> Exact Name\n");
out.println("<input type="radio" name="author_mode" value="last">Last, First Name (or Initial)\n");
out.println("</center>\n");
out.println("</font>\n");
out.println("</td> </tr>\n");
out.println("<tr> <td colspan=2> <p>\n");
out.println("<br> \n");
out.println("<tr> <td colspan=2> <p>\n");
out.println("</table>\n");
out.println("<center>\n");
out.println("<input type="submit" value="Search Now" id=submit1 name=submit1>\n");
out.println("<input type="reset" value="Clear Form" id=reset1 name=reset1>\n");
out.println("</center> </form>\n");
}
......

// Create standard footer
4.2 Design of Search by Different Requests
According to different requests, the different pages should be produced. There are many common parts in the produced pages. So, we don’t need to code the common parts for each page. We can design the common parts into standard utility classes or static functions. You can simply call the standard classes or functions when you needed them. An example looks like this:

```java
CreateHeader(out);
CreateLeftMenu(out);
CreateFooter(out);
```

The three lines above invoke the standard functions CreateHeader(), CreateFooter() and CreateLeftMenu(), which are developed in this thesis writing. The CreateHeader() function produces the top part of the pages. The CreateLeftMenu() function produces the left part of the pages. The CreateFooter() function produces the bottom part of the pages. You can find the source code in several servlets in Appendix.

### 4.3 Servlet Chaining To Process Requests In Different Levels

A request can be handled by a sequence of servlets. This system is designed to use the first level servlets to handle the search requests. If needed, the first level servlets can cooperates with the second level servlets by passing their processing result to the second level servlets and let them to proceed the request until the final result is obtained. The final result is returned by the last servlet. This is called *servlet chaining*. The request
from the client browser is sent to the first servlet in the chain. The response from the last
servlet in the chain is returned to the browser. In between, the output from each servlet is
passed as an input to the next servlet. Each servlet in the chain has the option to change
or extend the content. We can see the example in the ReqCollect code:

Example 4.3.1 : A Part of Code of The ReqCollect.java

......

if (path.equalsIgnoreCase(AUTHOR)) {

    ......

    out.println("<form method="POST"
    action="http://localhost:8080/servlet/AuthorProcess" id=form1
    name=form1>");

    ......

}

......

The ReqCollect servlet uses the getPathInfo() method to get the extra path information.
This method returns the extra path information associated with request. According to the
different extra path information the first level servlet invokes the different servlet of the
second level. In this case, ReqCollect invokes AuthorProcess servlet. The final output
has been listed in Figure 4.1.2.
4.4 Database Connectivity and Search from Database

The biggest advantage for servlets with regard to database connectivity is that the servlet life cycle allows servlets to maintain open database connections. An existing connection can trim several seconds from a response time, compared to a CGI program that has to reestablish its connection for every invocation.

Another advantage of the servlets over CGI and many other technologies is that JDBC is database-independent. JDBC is a SQL-level API – one that allows you to execute SQL statements and retrieve the results, if any. The API itself is a set of interfaces and classes designed to perform action against any database.

An individual database system is accessed via a specific JDBC driver that implements the java.sql.Drivers interface. Drivers exist for nearly all-popular RDBMS systems. The database connection used for Tiny Book Store uses a JDBC-ODBC bridge driver coming along with the JDK 1.2 to allow access to standard ODBC data source, such as the Microsoft Access database.

The first step in using a JDBC driver to get a database connection involves loading the specific driver class into the application’s Java Virtual Machine. This makes the driver available later, when we need it for opening the connection. An easy way to load the driver class is to use the Class.forName() method:
When the driver is loaded into memory, it registers itself with the `java.sql.DriverManager` class as an available database driver.

The next step is to ask the `DriverManager` class to open a connection to a given database, where the database is specified by a specified formatted URL. The method used to open the connection is `DriverManager.getConnection()`. It returns a class that implements the `java.sql.Connection` interface:

```java
Connection con = DriverManager.getConnection("jdbc:odbc:somedb", "user", "passwd");
```

A JDBC URL identifies an individual database in a driver-specific manner. Different drivers may need different information in the URL to specify the host database.

During the call to `getConnection()`, the `DriverManager` object asks each registered driver if it recognizes the URL. If a driver says yes, the driver manager uses that driver to create the `Connection` object. Here is an example.

Example 4.4.1: A Part of Code of AuthorProcess.java
After we got the database connection we need to have some way to execute queries. The simplest way to execute a query is to use the java.sql.Statement class. Statement objects are never instantiated directly; instead, a program calls the createStatement() method of Connection to obtain a new Statement object:

```java
Statement stmt = con.createStatement();
```
A query that returns data can be executed using the executeQuery() method of Statement. This method executes the statement and returns a java.sql.ResultSet that encapsulates the retrieved data:

Example 4.4.2: A Part of Code of AuthorProcess.java

......
String sqlStmt = "";
if (mode.equalsIgnoreCase(EXACT)) {
    sqlStmt = "select * from BOOK where author='" + name + "'";
}
else {
    sqlStmt = "select * from BOOK where author like '%" + name + "%'";
}

ResultSet rs = null;

Statement stmt = con.createStatement();
rs = stmt.executeQuery(sqlStmt);

......
A ResultSet object can be thought as a representation of the query result returned one row at a time. The next() method of ResultSet is used to move from row to row. The ResultSet interface also boasts a multitude of methods designed for retrieving data from the current row. The getString() and getObject() methods can be used for retrieving column values.

Example 4.4.3: A Part of Code of AuthorProcess.java

......

while (rs.next()) {
    count++;
    out.println("<dt><b>" + count + ":" + "");
    String isbn = rs.getObject("ISBN").toString();
    out.println("<a href="/servlet/DetailContent/" + isbn + ">");
    out.println(rs.getObject("Title").toString() + "</a>"});
    out.println("<dd>" + rs.getObject("Author").toString());
One of the outputs of servlet AuthorProcess is show in Figure 4.2.1 and Figure 4.4.1.

Figure 4.4.1: One of The Outputs of Servlet AuthorProcess
4.5 Session Tracking

Session Tracking is one of the most important techniques for any on-line store systems. This is because that HTTP protocol is a stateless protocol. A HTTP server sees only a series of requests and it, by itself, has no way to know exactly who is making the request. This means that when a customer puts a book in his shopping cart and then goes to other pages to find more books, next time when he puts another book in his shopping cart, there is no guarantee that he puts the book in the same cart. It is very possible that he uses another customer’s shopping cart at this time.
To solve this problem, session-tracking is introduced. Traditional techniques used by CGI include hidden form fields, URL rewriting, user authorization and persistent cookies. For server-side Java applications, a built-in class supplied servlet can be used easily to handle such tasks.

When a user first accesses the site, web server assigns a new HttpSession object and a unique session ID to the user. The session ID identifies the user and is used to match the user with the HttpSession object in subsequent requests. Usually the session ID is saved on the client in a cookie or sent as part of a rewritten URL. The Tiny Book Store uses this technique.

The Tiny Book Store is a classic shopping cart application. A client can put items in his virtual cart, accumulating them until he checks out several page requests later. How to make sure a user’s shopping items will be put in his shopping cart or delete from his shopping cart? The Java Servlet API provide built in session tracking to handle this problem. Every user of a site is associated with a javax.servlet.http.HttpSession object that servlets can use to store or retrieve information about that user. We can save any set of arbitrary Java objects in a session object. For example, a user’s session object provides a convenient location for a servlet to store the user’s shopping cart contents. An example shows below.

Example 4.5.1: A Part of Code of AddToShoppingCart.java
1 // Get current session object, create one if necessary
2 HttpSession session = req.getSession(true);
3 m_cart = (ShoppingCart)session.getValue("cart");
4 if (m_cart == null) { // need new cart,
5     // create a shopping cart
6     m_cart = new ShoppingCart();
7     // add to current session
8     session.putValue("cart", m_cart);
9 }
10
11 try {
12     Statement stmt = m_con.createStatement();
13     ResultSet rs = stmt.executeQuery("select * from book where ISBN='" + isbn +"'");
14     if (rs.next()) {
15         book = new
16         BookDetails(rs.getObject("ISBN").toString(),
17         rs.getObject("Title").toString(),
18         rs.getObject("Author").toString(),
19         rs.getDouble("Price"),
20         rs.getObject("PublicationDate").toString());
21     }
22 }
The servlet uses its request object’s `getSession()` method to retrieve the current `HttpSession` object.

```java
public HttpSession HttpServletRequest.getSession(boolean create)
```

This method returns the current session associated with the user making the request. If the user has no current valid session, this method creates one if `create` is true or return null if `create` is false. To ensure the session is properly maintained, this method must be called at least once before any output is written to the response. Please look at the line 2 in the example 4.5.1.

If you want to add data to an `HttpSession` object you can use the `putValue()` method:

```java
public void HttpSession.putValue(String name, Object value)
```
This method binds the specified object value under the specified name. Any existing
binding with the same name is replaced. This specifies at line 8 of the example 4.5.1.

If you want to retrieve an object from a session you can use getValue() method:

\[
\text{public Object HttpSession.getValue(String name)}
\]

This method specified at line 3 of example 4.5.1, returns the object bound under the
specified name or null if there is no binding.

From line 4 to line 9, the servlet checks whether the user already has a shopping cart. If
the user didn’t have a shopping cart, the servlet assigns a shopping cart to this user and
add the shopping cart to the current session to keep tracking. The ShoppingCart.java class
is developed in this thesis writing, code is provided in the appendix.

From line 10 to line 21, the servlet knows the user already has a shopping cart. m_con is
a data member of AddtoShoppingCart class. Lines 12-13 have been discussed in the
previous section. Lines 16-20 get some information of the book, which the user selected.
The information will be the data members of the “book” object that is an instance object
of BookDetails.java class. BookDetails.java class is developed in this thesis writing. the
source code is provided in the Appendix.

Line 25 puts the book which is selected by the user into the shopping cart.
Please see a set of pages are shown below.

Figure 4.5.1: A Page Show All Information About A Book

TinyBookStore.com

The C++ Programming Language
by Bjarne Stroustrup

Price: 31.47
Availability: Usually ships within 24 hours.

ISBN 0201889544
Addison-Wesley Pub Co. 1997

Reviews

In this brand-new third edition of The C++ Programming Language, author Bjarne Stroustrup, the creator of C++, presents the full specification for the C++ language and standard library, a spec that will soon become the joint ISO/ANSI C++ standard.
Figure 4.5.2: A Page Show Items In The Shopping Cart after A User Added Books To The Shopping Cart
4.6 Check Out

This part is designed only using the HTML. It collects all information from a user. The information will be written into the database.

An important line in this HTML file is shown below.
<form method=POST
action="http://localhost:8080/servlet/CustomerInfoProcess">

This line will invoke the servlet CustomerInfoProcess that will process all data collected from the user. This servlet will discuss in the next section. The check out page shows in Figure 4.6.1.

Figure 4.6.1: A Check Out Page

![Customer Information Form](image)

4.7 Customer Information Processing

The customer information processing is handled by a servlet CustomerInfoProces. Its source code is provided in the appendix.
All information, collected from the user, should be written into the database. Therefore we have to open a connection to a given database at first. Example 4.6.1 is given for this purpose.

Example 4.7.1: A Part of Code of CustomerInfoProcess.java

```java
public void init() throws ServletException {
    try {
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        m_con = riverManager.getConnection("jdbc:odbc:fxbookstore", null, null);
    }
    catch (ClassNotFoundException e) {
        e.printStackTrace();
    }
    catch (SQLException e) {
        e.printStackTrace();
    }
}
```

There is a POST request in the check out page. The servlet CustomerInfoProcess will handle the POST request by `doPost` function which is demonstrated in the following example.
Example 4.7.2: A Part of Source Code of CustomerInfoProcess.java

```java
public void doPost(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException {
    res.setContentType("text/html");
    PrintWriter out = res.getWriter();

    // get information
    String[] values = req.getParameterValues("firstname");
    if (values != null) {
        firstname = values[0];
    }

    values = req.getParameterValues("middleinit");
    if (values != null) {
        middleinit = values[0];
    }
    ....

    In the example, the HttpServletRequest object res will get each data which was input by
    the user by the getParameterValues() function.

    If the user is a returned user we don’t need open a new account again. If the returned user
    use the same credit card as previous we also don’t need to insert the credit card
```
information into the CREDITCARD table. Otherwise, we should insert all related information into related table. This is shown in the following example.

Example 4.7.3: A Part of Source Code of CustomerInfoProcess.java

```java
Statement stmt = m_con.createStatement();
String sql = "select * from CUSTOMERACCOUNT where EmailAddress='" + email + ";";
ResultSet rs = stmt.executeQuery(sql);
if (!rs.next()) {
    sql = "insert into CUSTOMERACCOUNT (emailaddress, password, firstname, middleinit, lastname) values (" + email + ";" + password + ";" + firstname + ";" + middleinit + ";" + lastname + ");";
    stmt.execute (sql);
}
sql = "select * from CREDITCARD where creditcardnumber='" + creditcard + ";";
rs = stmt.executeQuery (sql);
if (!rs.next()) {
```
sql = "insert into CREDITCARD (CreditcardNumber, CreditCardtype,
   HolderName, ExpirationDate) values(" + "'" + creditcard + "'','" +
cardtype + "','" + cardholder + "','" + expiredate + ");";

stmt.executeUpdate(sql);

All information related the books, selected by the user, can be obtained from the
shopping cart, as shown in the following example.

Example 4.7.4. A Part of Source Code of CustomerInfoProcess.java

BookDetails book = null;
ShoppingCartItem bookAndQuantity = null;

Enumeration items =
((ShoppingCart)m_session.getValue("cart")).getItems();

int aQuantity = 0;
String aISBN = null;

while (items.hasMoreElements()) {
   // insert into OrderItems
   bookAndQuantity = (ShoppingCartItem)items.nextElement();
   book = bookAndQuantity.getItem();
aQuantity = bookAndQuantity.getQuantity();

sql = "insert into ORDERITEMS (shoppingCartID, ISBN, quantity) values (" +" + m_cartID + "," + aISBN + "," + aQuantity + ");";

stmt.execute (sql);

After all information inserted into related tables the servlet CustomerInfoProcess will produce invoice page. The example page shows in Figure 4.7.1.
Your order reads as follows

E-Mail Address: fiaoxo@mitec.net

Fang Xiao
12345 Western Plaza #23
Omaha
NE
68154

Tel: (402)123-4567

Items:  
Title: Core Java 2, Qty: 1, Price: 25.79, Subtotal: 25.79

Title: Java Servlet Programming, Qty: 1, Price: 19.77, Subtotal: 19.77

Total: 45.56
Chapter 5
Future Work

In this chapter, we discuss two major problems for our future work. They are the Connection Pooling and Object-Oriented for better encapsulation.

5.1 Connection Pooling

After we have used JDBC for a short time, it will become evident that the major performance bottleneck often comes right at the beginning, when you are opening a database connection. This is because that opening a connection might involve a series of low level communications between network protocols and database connection protocols, such as TCP/IP, ODBC, etc.

Servlet life cycle allows for extremely fast database access. Servlet is loaded into Web server only once. Once it is loaded, it can hold a connection to database all its life cycle. Every time it receives a request to read from or write data to database, it can just use the connection to do this.

Here comes a problem, however. Suppose that a customer comes to the on-line store. During the period of selecting some items from the store, he leaves his desktop and do something else, leaving this connection to database idling. This will waste the precious resource on the database. From database port of view, the connection to the database will assign a space to hold corresponding status and to cache data. For example, Microsoft SQL-Server™ assigns about 36K bytes for each of its connection. Resource on the
database is limited and idling connection causes such resource unable to be reused for other users.

To solve this problem, we can use the technique called connection pooling. Essentially, a connection pool is an object holding connections to backend database and managing the number and status of connections. For example, if a connection in the pool idles over a certain time of period, the pool can terminate this connection; if the requests are not the same size of the connection pool, the pool can dynamically shrink or expand the size of itself. A connection pool can run on a separate thread and is initialized when the server starts or when a request to database is arrived.

To implement a connection pool is pretty straight-forward. Because of time constraint, however, we did not do it in this paper and leave it as a possible future improvement.

5.2 More Object-Oriented - Better Encapsulation

Another improvement might be done in the future is to create a HTML-related utility class. We have seen in this thesis that there are some duplicated code processing HTML output, such as "\texttt{out.println("\texttt{"<tb><tr>......</tr></tb>"})}". All these HTML-related processes, right now, are used in a raw way. That is they are not encapsulated. A better way to do this is to create a utility class or classes to deal with all HTML-related processes and hide all HTML tags and format inside the class or classes. For example, suppose that we create a Table class, then we can use it in a way like:

```
......
Table table = new Table(......);
```
In this way, we do not need to use HTML tags in our program and concentrate on the programming logic. The implementation is enriched with more object-oriented property and therefore it is easy to maintain.
Chapter 6

Conclusion

The increased awareness and popularity of the Internet make all kinds of electronic commerce grow vigorously. The most popular Web applications are the on-line stores. An on-line store system usually contains four components: Web server, business procedure, Database, system application. An on-line store system is different from web publishing. Currently, there are three mainstream technologies which are widely used to develop the on-line store systems: CGI, Microsoft ASP, and Server-side Java application.

Introducing the server-side Java application is one of the latest and most exciting trends in Java programming. We have discussed this technology in detail in this thesis. The server-side Java applications have many distinct advantages over CGI and other technologies. We present plenty of examples and figures to analysis the characteristics of the server-side Java application.

As a glance into the future, electronic commerce will continue to grow, driven by purchases of home computers and other Web-enabled devices as well as new business opportunities. Electronic commerce is changing whole business procedures. The future of electronic commerce will be exciting and full of opportunities.
APPENDIX A

Bibliography


APPENDIX B

The Source Code of An On-Line Store System – Tiny Book Store
1. welcome.shtml

<html>
<head>
<title>Search Tinybookstore.com</title>
</head>
<body>

<table border=0 width="100%" cellspacing=0 cellpadding=0 bgcolor=#eeeecc>
<tr>
<td rowspan=2 align=left valign=top width="15%" cellspacing="0" cellpadding="0" align=left valign=center>
<img border=0 hspace=0 id=IMG1 src="flower.gif" />
</td>
<td rowspan=2 align=left valign=center><strong><h2><i>TinyBookStore.com</i></h2></strong>
</td>
</tr>
</table>

<table border=0 width="100%" cellspacing=0 cellpadding=0>
<tr>
<td valign=top width=155 rowspan=3 bgcolor=#eeeecc>
<p>
</p>
<table width="100%" cellspacing=0 cellpadding=5 border=0>
<tr> <TD> </TD> <TD colspan="2"> <FONT FACE=verdana,arial,helvetica SIZE="-1"> &nbsp; </FONT> </TD> </TR>
<tr> <TD> </TD> <TD colspan="2"> Search Books </TD> </TR>
<tr> <TD> </TD> <TD colspan="2"> <a href="welcome.shtml"> Home </a> </TD>
</TR>
<tr> <TD> </TD> <TD colspan="2"> <a href="servlet/ReqCollect/category"> Category </a> </TD>
</TR>
<tr> <TD> </TD> <TD colspan="2"> <a href="servlet/ReqCollect/author"> Author </a> </TD>
</TR>
<tr> <TD> </TD> <TD colspan="2"> <a href="servlet/ReqCollect/title"> Title </a> </TD>
</TR>
</table>
</td>
</tr>
</table>

</body>
</html>
<TR> <TD><TD> <TD width=1>&nbsp;</TD> </TD><TD><font face=verdana,arial,helvetica size="-1"> <A href="servlet/ReqCollect/isbn">ISBN</A></TD> </TR> <TR> <TD><TD> <TD width=1>&nbsp;</TD> </TD><TD><font face=verdana,arial,helvetica size="-1"> <A href="servlet/ReqCollect/publisher_date">Publisher, Date</A></TD> </TR> <TD></table></p> </td> </TR> <TD rowspan=4 width=5>&nbsp;&nbsp;</TD> <TD valign=top align=left> <br clear=all> <p> <h3><strong><font color="#ff6347">Welcome to the Tiny Book Store</font></strong></h3> <strong>Five Hot Books</strong></center></center></h3></p> </TD> </TR> </TABLE> <TD width="50%" valign=top align=left> </TD> <TD width="50%" valign=top align=right> <font size=-1><A href="#top">Top of Page</A></font></TD> </TR>
2. AddToShoppingCart.java

    /**
     * AddToShoppingCart is the servlet that process the action of adding
     * item to shoppingcart.
     *
     * Author: Fang Xiao
     * Purpose: Thesis project
     * Copyright (C) Fang Xiao 1999 - 2001
     */

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

    public class AddToShoppingCart extends HttpServlet {

        protected Connection m_con = null;
        protected ShoppingCart m_cart;
        protected BookDetails book;

        public void init() throws ServletException {
            try {
                Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
                m_con = DriverManager.getConnection("jdbc:odbc:fxbookstore",
                null, null);
            } catch (ClassNotFoundException e) {
                e.printStackTrace();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }

        public void doPost(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
            res.setContentType("text/html");
            PrintWriter out = res.getWriter();
            String[] params = req.getParameterValues("isbn");
        }
    }
String isbn = ""; // get isbn from extra path information
if (params != null) {
    isbn = params[0];
}

// Get current session object, create one if necessary
HttpSession session = req.getSession(true);
m_cart = (ShoppingCart)session.getValue("cart");

if (m_cart == null) { // need new cart,
    // create a shopping cart
    m_cart = new ShoppingCart();
    // add to current session
    session.putValue("cart", m_cart);
}

try {
    Statement stmt = m_con.createStatement();
    ResultSet rs = stmt.executeQuery("select * from book where ISBN=' " + isbn + "'");
    if (rs.next()) {
        book = new BookDetails(rs.getObject("ISBN").toString(),
                                rs.getObject("Title").toString(),
                                rs.getObject("Author").toString(),
                                rs.getDouble("Price"),
                                rs.getObject("PublicationDate").toString());
    }
} catch (SQLException e) {
    e.printStackTrace();
}
m_cart.add(book.getISBN(), book);

showCartContents(out);

public void showCartContents(PrintWriter out) {
    out.println("<html><head><title>TinyBookStore.com Shopping Cart</title></head>");
    out.println("<body bgcolor="#FFFFFF" link="#003399" alink="#FF9933" vlink="#996633" text="#000000">");
    out.println("<a name="top"></a>!--Top of Page--></a>");
```java
Enumeration allitems = m_cart.getItems();
ShoppingCartItem item = null;
BookDetails book = null;
double total = 0.0;

while (allitems.hasMoreElements()) {
    item = (ShoppingCartItem)allItems.nextElement();
total += item.getItem().getPrice() * item.getQuantity();
    out.println("<tr bgcolor="#cccc99">
        <td align="left" valign="top" colspan="2" face=verdana,arial,helvetica><b>Shopping Cart Items</b></td>
        <td align="left" valign="top" face=verdana,arial,helvetica><font face=verdana,arial,helvetica><b>Price</b></font></td>
    </tr>"
    + item.getTitle() + item.getAuthor() + "Quantity: " + item.getQuantity() + "<br>
    + item.getISBN() + "form method="POST" action="/servlet/DeleteFromShoppingCart/item getItem().getISBN()">
```

out.println("<p> <input type=hidden name=isbn value=" +
item.getItemId().getISBN());

out.println("<p> <input type="submit" name="ditem"
value="delete">");

out.println("</td>");

out.println("<td bgcolor="#FFFFFF" font size=2
face=verdana,arial,helvetica color=#000000>");
out.println("<NOBR><b>Price: <font color=#990000>$" +
item.getItemId().getPrice() + ":</font></b></NOBR><br>");
out.println("<NOBR><b>Subtotal: <font color=#990000>$" +
item.getItemId().getPrice() * item.getQuantity() + ":</font></b></NOBR><br>");
out.println("</font>");
out.println("</td>");
}

out.println("<tr>");
out.println("<td colspan=3 align=right>");
out.println("<font face=verdana,arial,helvetica size="2">
<b>Total: <font color=#990000>" + total + "</b></font>");
out.println("</font></font>");
out.println("</td></tr>");
out.println("</table>");
out.println("<p><a href="http://localhost:8080/checkout.html">";
out.print(<img src="d:\JavaWebServer2.0\servlets\image\proceed-to-checkout.gif">"/
"a"));
out.println("</body> </html>");
out.flush();
out.close();
}
3. AuthorProcess.java

/**
 * <p> AuthorProcess
 * 
 * This is the servlet that process the Author-search request and generate search result.
 * 
 * Author: Fang Xiao
 * Purpose: Thesis project
 * Copyright © Fang Xiao 1999-2001
 */

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

public class AuthorProcess extends HttpServlet {
    protected final String EXACT = "exact";
    protected final String OTHERS = "last";

    protected Connection con = null;

    public void init() throws ServletException {
        try {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            con = DriverManager.getConnection("jdbc:odbc:fxbookstore", null, null);
        }
        catch (ClassNotFoundException e) {
            // need do something here
        }
        catch (SQLException e) {
            // need do something here
        }
    }

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

String[] values;
String name = "";

values = req.getParameterValues("author");
if (values != null) {
    name = values[0];
}

String mode = "";
values = req.getParameterValues("author_mode");
if (values != null) {
    mode = values[0];
}

String sqlStmt = "";
if (mode.equalsIgnoreCase(EXACT)) {
    sqlStmt = "select * from BOOK where author=\"" + name + "\";"
} else {
    sqlStmt = "select * from BOOK where author like \"%\" + name + "\";"
}

ResultSet rs = null;
int count = 0;

// create header here
CreateHeader(out);

// Create left side menu
CreateLeftMenu(out);

// "Your Book Search Results"
try {
    if (con != null) {
        Statement stmt = con.createStatement();
        rs = stmt.executeQuery(sqlStmt);
        out.println("<dl>\"" + count + ".  \\
        String isbn = rs.getObject("ISBN").toString();
out.println("<a href="/servlet/DetailContent/" + isbn + ">");
out.println(rs.getObject("Title").toString() + 
"</a></b>");
out.println("<nobr><font color=#990033>Usually ships in 24 hours</font></nobr>");
out.println("<dd>" + 
rs.getObject("Author").toString();
out.println("/ " + 
rs.getObject("Publisher").toString() + " / " + rs.getObject("PublicationDate").toString();
out.println("<br> Our Price: " + 
rs.getObject("Price").toString();
out.println("<br><br>");
}
if (count == 0) {
out.println("<dt> No matched data found";
}
out.println("</dl>"); // end Glossary List
out.println("<table border=0 width="100%" cellspacing=0
cellpadding=0 bgcolor="EEEECC">")
out.println("<tr>");
out.println("<td align=left valign=top width=90>");
out.println("<img border=0 hspace=0 id=IMG1 src="d:\JavaWebServer2.0\public_html\flower.gif"></td>");
out.println("<td width=30></td>");
out.println("<td align=left valign=center width=300>");
out.println("<strong><h2>Tiny Book Store.com</h2></strong>");
out.println("<td align=right valign=center>");
out.println("</td>");
out.println("<td width=20></td>");
out.println("</tr>");
out.println("</table>");
}

protected void CreateFooter(PrintWriter out) {
// Create StandardFooter
out.println("<table width=100%>");
out.println("<tr> <td width=50% valign=top align=left> </td>");
out.println("<td width=50% valign=top align=right> <font size=-1><a href="#top">Top of Page</a></font> </td>");
out.println("</tr>");
out.println("</table>");
out.println("<center>");
out.println("</center>");
out.println("<p>");
protected void CreateLeftMenu(PrintWriter out) {
    System.out.println("LeftMenu");
    out.println("<table border=0 width="100\%" cellspacing=0 cellpadding=0>\n");
    out.println("<tr> <td valign=top width=155 rowspan=3 bgcolor=#eeeecc>\n");
    out.println("<p>\n");
    out.println("<table width="100\%" cellspacing=0 cellpadding=5 border=0>\n");
    out.println("<tr> <td> </td> <td colspan="2">\n");
    out.println("<font face=verdana,arial,helvetica SIZE="-1">\n");
    out.println("<a href="http://localhost:8080/welcome.shtml">Home</a>\n");
    out.println("<tr> <td> </td> <td colspan="2">\n");
    out.println("<font face=verdana,arial,helvetica size="-1">\n");
    out.println("<a href="http://localhost:8080/servlet/ReqCollect/category">Category</a>\n");
    out.println("</font></tr>\n");
    out.println("</td> <td> </td> <td width=2>\n");
    out.println("</table>\n");
    out.println("</p>\n");
    out.println("</table>\n");
    out.println("</td> <td colspan=2>\n");
    out.println("</tr>\n");
    out.println("</font></strong>\n");
    out.println("</td> <td> </td> <td width=2>\n");
    out.println("</table>\n");
    out.println("</td> <td> </td> <td width=2>\n");
    out.println("</table>\n");
    out.println("</td> <td colspan=2>\n");
    out.println("</tr>\n");
    out.println("</font>\n");
    out.println("</center>\n");
    out.println("<p>\n");
    out.println("<center>\n");
    out.println("<b font size=-l>\n");
    out.println("<A href=Y7copyright.html">Copyright and disclaimer</A>\n");
    out.println("© 1999-2001, Fang Xiao\n");
    out.println("</font>\n");
    out.println("</center>\n");
    out.println("</p>\n");
    out.println("</center>\n");
    out.println("</font>\n");
    out.println("</html>\n");
<table>
<thead>
<tr>
<th>Author</th>
<th>Title</th>
<th>ISBN</th>
<th>Publisher, Date</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```
4. BookTeatails.java

/**
 * This is the class describing a book item in shopping cart
 * 
 * Author: Fang Xiao
 * Purpose: Thesis project
 * Copyright (C) Fang Xiao 1999 - 2001
 */

public class BookDetails {
    private String m_isbn = null;
    private String m_title = null;
    private String m_author = null;
    private double m_price = 0.0;
    private String m_year = null;

    public BookDetails(String isbn, String title, String author, double price, String year) {
        this.m_isbn = isbn;
        this.m_title = title;
        this.m_author = author;
        this.m_price = price;
        this.m_year = year;
    }

    public String getISBN() {
        return m_isbn;
    }

    public String getTitle() {
        return m_title;
    }

    public String getAuthor() {
        return m_author;
    }

    public double getPrice() {
        return m_price;
    }

    public String getYear() {
    }
return m_year;
}
5. CategoryProcess.java

/**
 * CategoryProcess
 *
 * This is the servlet that process the Category-search request and generate
 * search result.
 *
 * Author: Fang Xiao
 * Purpose: Thesis project
 * Copyright (C) Fang Xiao 1999-2001
 */

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

public class CategoryProcess extends HttpServlet {
    //protected final String EXACT = "exact";
    //protected final String OTHERS = "last";

    protected Connection con = null;

    public void init() throws ServletException {
        try {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            con = DriverManager.getConnection("jdbc:odbc:fxbookstore",
            null, null);
        }
        catch (ClassNotFoundException e) {
            // need do something here
        }
        catch (SQLException e) {
            // need do something here
        }
    }
}
public void service(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException {
    res.setContentType("text/html");
    PrintWriter out = res.getWriter();

    String[] values;
    String name = "";
    String sqlStmt = "";

    String path = req.getPathInfo();
    if (path == null) {  // should not happen
        out.println("Check the URL to make sure it is correct");
        out.flush();
        out.close();
    } else {
        path = path.substring(1);
        if (path.equalsIgnoreCase("c01"))
            sqlStmt = "select * from BOOK where categoryid='c01'";
        else if (path.equalsIgnoreCase("c02"))
            sqlStmt = "select * from BOOK where categoryid='c02'";
        else if (path.equalsIgnoreCase("c03"))
            sqlStmt = "select * from BOOK where categoryid='c03'";
    }

    ResultSet rs = null;
    int count = 0;

    // create header here
    CreateHeader(out);

    // Create left side menu
    CreateLeftMenu(out);

    // "Your Book Search Results
    try {

if (con != null) {
    Statement stmt = con.createStatement();
    rs = stmt.executeQuery(sqlStmt);
    
    out.println("<dl>"); // start Glossary List
    while (rs.next()) {
        count++;
        out.println("<dt><b>" + count + ". "");
        String isbn = rs.getObject("ISBN").toString();
        out.println("<a href="/servlet/DetailContent/"") + isbn + ">");
        out.println(rs.getObject("Title").toString() + "</a></b>");
        out.println("<brx/font color=#990033>Usually
ships in 24 hours</font>");
        out.println("<dd>" +
        rs.getObject("Author").toString();
        out.println("/ " +
        rs.getObject("Publisher").toString() + " / " + rs.getObject("PublicationDate").toString();
        out.println("<br> Our Price: " +
        rs.getObject("Price").toString();
        
        // <font color=#990000>You Save: $11.00
        (20%)</font>
        }
        out.println("<br><br>");
    }
    if (count == 0) {
        out.println("<dt> No matched data found">
    }
    out.println("</dl>"); // end Glossary List
}

// create footer here
out.println("</td> </tr>");
out.println("</table>");

    // Create standard footer
CreateFooter(out);
out.flush();
out.close();
}
protected void CreateHeader(PrintWriter out) {
    out.println("<html>");
    out.println("<head>");
    out.println("<title> TinyBookStore.com </title>");
    out.println("</head>");

    out.println("<body>");
    out.println("<table border=0 width="100%" cellspacing=0 cellpadding=0bgcolor="EEEECC"/>");
    out.println("<tr>");
    out.println("<td align=left valign=top width=90">
    <IMG border=0 hspace=0 id=IMG1 src="d:\JavaWebServer2.0\public_html\flower.gif"></td>
    <td width=30></td>");
    out.println("<td align=left valign=center width=300>
    <strong><h2><i>TinyBookStore.com</i></h2></strong></td>");
    out.println("<td align=right valign=center>");
    out.println("</td>");
    out.println("<td width=20></td>");
    out.println("</tr>");
    out.println("</table>" );
}

protected void CreateFooter(PrintWriter out) {
    // Create StandardFooter
    out.println("<table width=100%/>");
    out.println("<tr><td width=50% valign=top align=left></td>");
    out.println("<td width=50% valign=top align=right>");
    out.println("<font size=-1><a href="#top">Top of Page</a></font>");
    out.println("</td>");
    out.println("</tr>");
    out.println("</table>" );

    out.println("<center>");
    out.println("<font size=-1>");
    out.println("<a href="http://localhost:8080/welcome.shtml">Tinybookstore.com Home</a> &nbsp; &nbsp; " );
    out.println("<a href="http://localhost:8080/servlet/ReqCollect/category">Category</a> &nbsp; &nbsp; ");
out.println("<a href="/"http://localhost:8080/welcome.shtml">Home</a></td></tr>");
out.println("<tr><td><td width=2>&nbsp;</td><td><font face=verdana,arial,helvetica size=1">")
out.println("<a href="/"http://localhost:8080/servlet/ReqCollect/category">Category</a></td></tr>");
out.println("<tr><td><td width=2>&nbsp;</td><td><font face=verdana,arial,helvetica size=1">")
out.println("<a href="/"http://localhost:8080/servlet/ReqCollect/author">Author</a></td></tr>");
out.println("<tr><td><td width=2>&nbsp;</td><td><font face=verdana,arial,helvetica size=1">")
out.println("<a href="/"http://localhost:8080/servlet/ReqCollect/title">Title</a></td></tr>");
out.println("<tr><td><td width=2>&nbsp;</td><td><font face=verdana,arial,helvetica size=1">")
out.println("<tr><td><td width=2>&nbsp;</td><td><font face=verdana,arial,helvetica size=1">")
out.println("<a href="/"http://localhost:8080/servlet/ReqCollect/publisher_date">Publisher, Date</a></td></tr>");
out.println("<table></p>");
out.println("</td>");
out.println("<td rowspan=4 width=5>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&n
import java.sql.*;
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class CustomerInfoProcess extends HttpServlet {
    protected Connection m_con = null;
    protected HttpSession m_session = null;
    protected static int m_cartID = 1;
    protected String firstname = "";
    protected String middleinit = "";
    protected String lastname = "";
    protected String email = "";
    protected String password = "";
    protected String creditcard = "";
    protected String cardtype = "";
    protected String cardholder = "";
    protected String expiredate = "";
    protected String shipping = "";
    protected String receiver = "";
    protected String address = "";
    protected String city = "";
    protected String state = "";
    protected String zipcode = "";
    protected String phone = "";

    public void init() throws ServletException {
        try {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
m_con = DriverManager.getConnection("jdbc:odbc:fxbookstore", null, null);

} catch (ClassNotFoundException e) {
    e.printStackTrace();
}
} catch (SQLException e) {
    e.printStackTrace();
}

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

    // get information
    String[] values = req.getParameterValues("firstname");
    if (values != null) {
        firstname = values[0];
    }

    values = req.getParameterValues("middleinit");
    if (values != null) {
        middleinit = values[0];
    }

    values = req.getParameterValues("lastname");
    if (values != null) {
        lastname = values[0];
    }

    values = req.getParameterValues("email");
    if (values != null) {
        email = values[0];
    }

    values = req.getParameterValues("password");
    if (values != null) {
        password = values[0];
    }

    values = req.getParameterValues("creditcard");
if (values != null) {
    creditcard = values[0];
}

values = req.getParameterValues("cardtype");
if (values != null) {
    cardtype = values[0];
}

values = req.getParameterValues("cardholder");
if (values != null) {
    cardholder = values[0];
}

values = req.getParameterValues("expiredate");
if (values != null) {
    expiredate = values[0];
}

values = req.getParameterValues("shipping");
if (values != null) {
    shipping = values[0];
}

values = req.getParameterValues("receiver");
if (values != null) {
    receiver = values[0];
}

values = req.getParameterValues("address");
if (values != null) {
    address = values[0];
}

values = req.getParameterValues("city");
if (values != null) {
    city = values[0];
}

values = req.getParameterValues("state");
if (values != null) {
    state = values[0];
}
values = req.getParameterValues("zipcode");
if (values != null) {
    zipcode = values[0];
}

values = req.getParameterValues("phone");
if (values != null) {
    phone = values[0];
}

// get session object
m_session = req.getSession();

try {
    // update database
    Statement stmt = m_con.createStatement();
    String sql = "select * from CUSTOMERACCOUNT where
    EmailAddress=" + email + ";"
    ResultSet rs = stmt.executeQuery(sql);
    if (!rs.next()) {
        System.out.println("insert customeraccount");
        sql = "insert into CUSTOMERACCOUNT (emailaddress,
    password, firstname, middleinit, lastname) values (" +
    ",", + email + ",", + password + ",", + firstname +
    ",", + middleinit + ",", + lastname + ");";
        System.out.println(sql);
        stmt.execute (sql);
    }
    sql = "select * from CREDITCARD where creditcardnumber=" +
    creditcard + ";"
    rs = stmt.executeQuery (sql);
    if (!rs.next()) {
        System.out.println("insert creditcard");
        sql = "insert into CREDITCARD (CreditcardNumber,
    CreditCardtype, HolderName, ExpirationDate) values(" +
    ",", + creditcard + ",", + cardtype + ",", + cardholder + ",", + expiredate + ");";
        System.out.println(sql);
        stmt.execute (sql);
    }
}
sql = "insert into SHOPPINGCART (ShoppingCartID, EmailAddress, CreditCardNumber, ShippingType, ReceiverName, ReceiverAddress, ReceiverCity, ReceiverState, ReceiverZipcode, ReceiverPhone) values (" + "" + m_cartID + "," + email + "," + creditcard + "," + shipping + "," + receiver + "," + address + "," + city + "," + state + "," + zipcode + "," + phone + ");";
System.out.println(sql);
stmt.executeUpdate(sql);

BookDetails book = null;
ShoppingCartItem bookAndQuantity = null;
Enumeration items =
((ShoppingCart)m_session.getValue("cart")).getItems();

int aQuantity = 0;
String aISBN = null;

while (items.hasMoreElements()) {
    // insert into OrderItems
    bookAndQuantity =
(ShoppingCartItem)items.nextElement();
    book = bookAndQuantity.getItem();

    aQuantity = bookAndQuantity.getQuantity();

System.out.println("insert OrderItems");
sql = "insert into ORDERITEMS (shoppingCartID, ISBN, quantity) values (" + "+" + m_cartID + "," + aISBN + "," + aQuantity + ");";
System.out.println(sql);
stmt.executeUpdate(sql);

}
catch (SQLException e) {
    e.printStackTrace();
}

printOrder(out);

public void printOrder(PrintWriter out) {
    out.println("<html> <head> <title> Order List </title> </head>");
    out.println("<body>");
    out.println("<h1> Your order reads as follows </h1>");
    out.println("<table>");
    out.println("<tr>");
    out.println("<td>E-Mail Address: </td>");
    out.println("<td>" + email + "</td></tr>");
    out.println("<tr>");
    out.println("<td>ship to:</td>" + receiver + " <br>");
    out.println(address + "<br>");
    out.println(city + "<br>");
    out.println(state + "<br>");
    out.println(zipcode + "<br></td></tr>");
    out.println("<tr>");
    out.println("<td>Tel: </td>");
    out.println("<td>" + phone + "</td></tr>");
    out.println("<tr>");
    out.println("<td>Items: </td>");
    out.println("<td>");

    BookDetails book = null;
    ShoppingCartItem bookAndQuantity = null;
    Enumeration items =
    ((ShoppingCart)m_session.getValue("cart")).getItems();

    double total = 0.0;

    while (items.hasMoreElements()) {
        bookAndQuantity = (ShoppingCartItem)items.nextElement();

        total += bookAndQuantity.getItem().getPrice() *
        bookAndQuantity.getQuantity();

        book = bookAndQuantity.getItem();
out.println("<td>Title:" + book.getTitle() + ";Qty:" + 
bookAndQuantity.getQuantity() + "<br>");
    out.println("Price:" + book.getPrice() + ";Subtotal:" + 
bookAndQuantity.getQuantity() * book.getPrice() + ";<br>");
}
out.println("Total:" + total);
out.println("</td></tr>");
out.println("</table>");
out.flush();
out.close();
}
public class DeleteFromShoppingCart extends HttpServlet {

    protected ShoppingCart m_cart;
    protected BookDetails book;

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

        String[] params = req.getParameterValues("isbn");
        String isbn = ""; // get isbn hidden field
        if (params != null) {
            isbn = params[0];
        }
        System.out.printlnf("isbn = " + isbn);

        // Get current session object
        HttpSession session = req.getSession();
        m_cart = (ShoppingCart)session.getValue("cart");

        m_cart.remove(isbn);
    }
}
showCartContents(out);
}

public void showCartContents(PrintWriter out) {
    out.println("<html><head><title>Amazon.com Shopping Cart</title></head>");
    out.println("<body bgcolor="#FFFF00" link="#003399"
    alink="#FF9933" vlink="#996633" text="#000000">\n"
    out.println("<a name="top"><!-- Top of Page--></a>\n"
    out.println("<p><a href="http://localhost:8080/checkout.html"><img
    src="d:\JavaWebServer2.0\servlets\image\proceed-to-checkout.gif"></a>\n"
    out.println("<table width="100%" cellpadding="0" cellspacing="0"
    border="0">\n"
    out.println("<tr bgcolor="#cccc99">\n"
    out.println("<td bgcolor="#cccc99">\n"
    out.println("<td align="left" valign="top" colspan="2">
    "Shopping Cart Items</font></b></td>\n"
    out.println("<td align="left" valign="top">
    "Price</font></b></td>\n"
    out.println("</tr>\n"
    Enumeration allItems = m_cart.getItems();
    ShoppingCartItem item = null;
    BookDetails book = null;
    double total = 0.0;
    while (allItems.hasMoreElements()) {
        item = (ShoppingCartItem) allItems.nextElement();
        total += item.getItem().getPrice() * item.getQuantity();
        out.println("<tr bgcolor="#FFFF00">\n"
        out.println("<td><em><b>" + item.getItem().getTitle() + "</b></em>
        " + item.getItem().getAuthor());
        out.println("</td><em><b>" + item.getItem().getPrice() + "
        out.println("</em><b>" + item.getItem().getTitle() + "</b></em>
        out.println("</td>\n"
        out.println("</tr>\n"
out.println("<br>Quantity: " + item.getQuantity());
out.println("<br>");
out.println("</td>");

out.println("<td align="left">");
out.println("<form method="POST"
action="/servlet/DeleteFromShoppingCart">
out.println("<p><input type=hidden name=isbn value=" +
item.getItem().getISBN());
out.println("<p><input type="submit" name="ditem" value="delete">" );
out.println("</p>");
out.println("</td>");

out.println("<td bgcolor="#FFFFFF" f"o n t s i z e=2
face=verdana,arial,helvetica color=#000000>");
out.println("<NOBR><b>Price: <font color=#990000>$" +
item.getItem().getPrice() + "</font></b></NOBR><br>");
out.println("<NOBR><b>Subtotal: <font color=#990000>$" +
item.getItem().getPrice() * item.getQuantity() + "</font></b></NOBR><br>");
out.println("</font>");
out.println("</td>");

}

out.println("</tr>");
out.println("<td colspan=3 align=right>" );
out.println("<font face=verdana,arial,helvetica size="2"">Total:
<font color=#990000>" + total + "</font></b>" );
out.println("</font>");
out.println("</td></tr>");
out.println("</table>");
out.println("<p><a href="http://localhost:8080/checkout.html">
<img src="d:\JavaWebServer2.0\servlets\image\proceed-to-checkout.gif"></a>");
out.println("</body> </html>");

out.flush();
out.close();

}
8. **DetailContent.java**

```java
/**
 * <p>DetailContent is the servlet responsible for generating
detailed information page for a book. The book is identified
by its ISBN

* Author: Fang Xiao
* Purpose: Thesis project
* Copyright (C) Fang Xiao 1999-2001
*/

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

public class DetailContent extends HttpServlet {
    protected Connection con = null;

    public void init() throws ServletException {
        try {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            con = DriverManager.getConnection("jdbc:odbc:fxbookstore",
            null, null);
        } catch (ClassNotFoundException e) {
        } catch (SQLException e) {
        }
    }

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

        String isbn = req.getPathInfo();
        isbn = isbn.substring(1);  // skip over '/'
    }
```
try {
    String sql = "select * from book where ISBN='" + isbn + "'";
    Statement stmt = con.createStatement();
    ResultSet rs = stmt.executeQuery(sql);

    // create header here
    CreateHeader(out);

    // create left menu here
    CreateLeftMenu(out);

    out.println("<tr><td rowspan=4 width=5>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&n..."
out.println(" <td align=right>");
out.println(" <table border=0 cellspacing=0
cellpadding=0>");

out.println(" <tr><td>");
out.println(" <form method="POST" action="http://localhost:8080/servlet/AddToShoppingCart">
out.println(" <input type=hidden name=isbn value=" + strISBN + ""/>
out.println(" <input type=submit value="Add to cart"/>
out.println(" </td></tr>");

out.println(" </table></td>");
out.println(" </tr>");
out.println(" <tr><td colspan=3 align=left> ISBN ");
out.println(" strISBN +" </td></tr>");
out.println(" <tr><td colspan=3 align=left>");
out.println(" rs.getObject("Publisher").toString()
+rs.getObject("PublicationDate").toString() +" </td></tr>");
out.println(" </table>");
out.println(" <br>");
out.println(" <br>");

// output review
String review = rs.getObject("BookReview").toString();
if (review != null) {
    review = "D:\JavaWebServer2.0\servlets\review\" + review;
    out.println("<b>Reviews</b> <br>");
    try {
        BufferedReader br = new BufferedReader(new FileReader(review));
        String thisLine = null;
        while ((thisLine = br.readLine()) != null) {
            out.println("<p>");
            out.println(thisLine);
            out.println(" </p>");
        }
    } catch (IOException e) {
        // need do something here
        e.printStackTrace();
    }
}
out.println("<br>");
String comment = rs.getObject("BookComment").toString();
if (comment != null) {
    comment = "D:\JavaWebServer2.0\servlets\comment\" +
    out.println("<br>Reader Comments</b> <br>");
    try {
        BufferedReader br = new BufferedReader(new FileReader(comment));
        String thisLine = null;
        while ((thisLine = br.readLine()) != null) {
            out.println("<p>");
            out.println(thisLine);
            out.println("</p>");
        }
    }
    catch (IOException e) {
        // need do something here
        e.printStackTrace();
    }
    out.println("<br>");
}
}

protected void CreateHeader(PrintWriter out) {
    out.println("<html>");
out.println("<head>");
out.println("<title> TinyBookStore.com </title>");
out.println("</head>");

out.println("<body>");
out.println("<table border=0 width="100\%" cellspacing=0 cellpadding=0gcolor="EEEECC">");
out.println("<tr>");
out.println("<td align=left valign=top width=90 >");
out.println("<IMG border=0 hspace=0 id=IMG1 src="d:\JavaWebServer2.0\public_html\flower.gif"></td>");
out.println("<td width=30"><td>");
out.println("<td align=left valign=center width=300>");
out.println("<strong><h2><i>TinyBookStore.com</i></h2></strong></td>");
out.println("<td align=right valign=center>");
out.println("<td width=20></td>");
out.println("<\table>");

protected void CreateFooter(PrintWriter out) {
    // Create StandardFooter
    out.println("<table width=100%>");
    out.println("<tr> <td width=50% valign=top align=left> </td>");
    out.println("<td width=50% valign=top align=right> </td>");
    out.println("<font size=-1><a href="#top">Top of Page</a></font>");
    out.println("<\table>");
    out.println("<\center>");
    out.println("<font size=-1>");
    out.println("<a href="http://localhost:8080/welcome.shtml">Tinybookstore.com Home</a>
    &nbsp;&nbsp;");
    out.println("<a href="http://localhost:8080/servlet/ReqCollect/category">Category</a>
    &nbsp;&nbsp;");
    out.println("<a href="http://localhost:8080/servlet/ReqCollect/author">Author</a> &nbsp;&nbsp;");
    out.println("<a href="http://localhost:8080/servlet/ReqCollect/title">Title</a> &nbsp;&nbsp;");
protected void CreateLeftMenu(PrintWriter out) {
    out.println("<table border=0 width="100%" cellspacing=0 cellpadding=0>");
    out.println("<tr> <td valign=top width=155 rowspan=3 bgcolor=#eeeecc>");
    out.println("<p>");
    out.println("<table width="100%" cellspacing=0 cellpadding=5 border=0>");
    out.println("<tr> <td colSpan=2">Search Books</td> </tr>");
    out.println("<font face=verdana,arial,helvetica size=-1">Home</font>");
}

out.println("<a href="/"&nbsp;href="http://localhost:8080/servlet/ReqCollect/isbn">ISBN</a> &nbsp;t&nbsp;publish?&nbsp;");
out.println("<a href="/"&nbsp;href="http://localhost:8080/servlet/ReqCollect/publisher_date">Publisher Date</a>");
97

out.println("<a href="http://localhost:8080/servlet/ReqCollect/category">Category</font></ax/td>
<tr>
<td></td>
<td width=2>&nbsp;</td>
<td><font face=verdana,arial,helvetica size="-1">"");
out.println("<a href="http://localhost:8080/servlet/ReqCollect/author">Author</font></ax/td>
<tr>
<td></td>
<td width=2>&nbsp;</td>
<td><font face=verdana,arial,helvetica size="-1">"");
out.println("<a href="http://localhost:8080/servlet/ReqCollect/title">Title</font></ax/td>
<tr>
<td></td>
<td width=1>&nbsp;</td>
<td><font face=verdana,arial,helvetica size="-1">"");
<tr>
<td></td>
<td width=1>&nbsp;</td>
<td><font face=verdana,arial,helvetica size="-1">"");
out.println("<a href="http://localhost:8080/servlet/ReqCollect/publisher_date">Publisher, Date</a></font></td>
<tr>
</table>
</td>
<td rowspan=4 width=5>&nbsp;&nbsp;<br/td>
<td valign=top align=left>
<br clear=all>
</table>
"
public class EchoServletTag extends HttpServlet {

    protected Connection con = null;

    public void init() throws ServletException {
        try {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            con = DriverManager.getConnection("jdbc:odbc:fxbookstore", null, null);
        }
        catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
        catch (SQLException e) {
            e.printStackTrace();
        }
    }

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

        try {
            Statement stmt = con.createStatement();
            ResultSet rs = stmt.executeQuery("select * from book where title like "%Programming%"");
            int count = 1;
            out.println("<dl>");
            while (rs.next() && count < 6) {
                out.println("<dt><b>" + count + ". ");
            }
        }
    }
}
String isbn = rs.getObject("ISBN").toString();
out.println("<a href=/servlet/DetailContent/"+isbn+"/>");
out.println(rs.getObject("Title").toString() + "/a"></a>");
out.println("<dd><nobr><font color=#990033>Usually ships in 24 hours</font></nobr>");
out.println("<br>" + rs.getObject("Author").toString());
out.println("/" + rs.getObject("Publisher").toString() + "/" +
rs.getObject("PublicationDate").toString());
out.println("<br> Our Price: " +
rs.getObject("Price").toString());
out.println("<br><br>");
count++;}
out.println("</dl>");
}
catch (SQLException e) {
e.printStackTrace();
}
out.println("</td</tr>");
out.println("</table>");
out.flush();
out.close();
}
10. ISBNProcess.java

/**
 * <p> ISBNProcess
 * 
 * This is the servlet that process the ISBN-search request and generate
 * search result.
 * 
 * Author: Fang Xiao
 * Purpose: Thesis project
 * Copyright (C) Fang Xiao 1999-2001
 */

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

public class ISBNProcess extends HttpServlet {

    protected Connection con = null;

    public void init() throws ServletException {
        try {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            con = DriverManager.getConnection("jdbc:odbc:fxbookstore",
                    null, null);
        } catch (ClassNotFoundException e) {
        } catch (SQLException e) {
        }
    }

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

        String[] values;
String sISBN = "";

values = req.getParameterValues("isbn");
if (values != null) {
    sISBN = values[0];
}

String sqlStmt = "select * from BOOK where ISBN= " + sISBN + "";

ResultSet rs = null;
int count = 0;

// create header here
CreateHeader(out);

// Create left side menu
CreateLeftMenu(out);

// "Your Book Search Results"
try {
    if (con != null) {
        Statement stmt = con.createStatement();
        rs = stmt.executeQuery(sqlStmt);

        out.println("<dl>"); // start Glossary List
        while (rs.next()) {
            count++;
            out.println("<dtb>" + count + ");
            String isbn = rs.getObject("ISBN").toString();
            out.println("<a href="/servlet/DetailContent/" + isbn + ">");
            out.println(rs.getObject("Title").toString() + "
" + "<br/>
"); // usually ships in 24 hours
            out.println("<nobr><font color="#990033">Usually
" + "</a></b>");
            out.println(rs.getObject("Author").toString() + ");
            out.println("<dd>" + rs.getobject("Publisher").toString() + " / " + rs.getObject("PublicationDate").toString());
        }
    }
}
out.println("<br> Our Price: " +
rs.getObject("Price").toString());
out.println("<br><br>
}
if (count == 0) {
    out.println("<dt> No matched data found");
}
out.println("</dl>"); // end Glossary List
}
}
catch (SQLException e) {
}
// create footer here
out.println("</td> </tr>");
out.println("</table>");

// Create standard footer
CreateFooter(out);
out.flush();
out.close();

protected void CreateHeader(PrintWriter out) {
    out.println("<html>");
    out.println("<head>");
    out.println("<title> TinyBookStore.com </title>");
    out.println("</head>");
    out.println("<body>");
    out.println("<table border=0 width="100%" cellspacing=0 cellpadding=0gcolor="EEEECC">")
    out.println("<tr>");
    out.println("<td align=left valign=top width=90 >")
    out.println("<img border=0 hspace=0 id=IMG src="d:\JavaWebServer2.0\public_html\flower.gif"></td>");
    out.println("<td width=30></td>");
    out.println("<td align=left valign=center width=300>");
    out.println("<strong><h2><i>TinyBookStore.com</i></h2></strong></td>");
    out.println("<td align=right valign=center>");
}
protected void CreateFooter(PrintWriter out) {
    // Create StandardFooter
    out.println("<table width=100\%>");
    out.println("<tr><td width=50\% valign=top align=left><a href="#top">Top of Page</a></td>");
    out.println("</table>");
    out.println("<center>");
    out.println("<font size=-1>");
    out.println("<a href="/copyright.html">Copyright and disclaimer</a>© 1999-2001, Fang Xiao";
    out.println("</font>";
    out.println("</center>";
    out.println("</p>";
    out.println("</font>";
    out.println("</center>";
    out.println("</p>";
    out.println("</center>";
    out.println("<br><font size=-1>";
    out.println("<A href="/copyright.html">Copyright and disclaimer</A> &copy; 1999-2001, Fang Xiao";
    out.println("</font>";
    out.println("</center>";
    out.println("</body>";
    out.println("</html>";
protected void CreateLeftMenu(PrintWriter out) {
    System.out.println("LeftMenu");
    out.println("<table border=0 width="100\%" cellspacing=0
cellpadding=0">
        <tr> <td valign=top width=155 rowspan=3
bgcolor=#eeeecc>
        <p>
        <table width="100\%" cellspacing=0 cellpadding=5
border=0>
            <tr> <td> </td> <td colspan="2">
            <font face=verdana,arial,helvetica SIZE="-1">&nbsp;</font>
            </td> </tr>
            <tr> <td> </td> <td colspan="2">
            <strongxfont face=verdana,arial,helvetica size="-1">Search Books</strong>
            </td> </tr>
            <tr> <td> </td> <td width=2>&nbsp;</td> <td>
            <font face=verdana,arial,helvetica size="-1">
            <a href="http://localhost:8080/welcome.shtm1">Home</font>
            </td> </tr>
            <tr> <td> </td> <td width=2>&nbsp;</td> <td>
            <font face=verdana,arial,helvetica size="-1">
            <a href="http://localhost:8080/servlet/ReqCollect/category">Category</a>
            </td> </tr>
            <tr> <td> </td> <td width=2>&nbsp;</td> <td>
            <font face=verdana,arial,helvetica size="-1">
            <a href="http://localhost:8080/servlet/ReqCollect/author">Author</a>
            </td> </tr>
            <tr> <td> </td> <td width=2>&nbsp;</td> <td>
            <font face=verdana,arial,helvetica size="-1">
            <a href="http://localhost:8080/servlet/ReqCollect/title">Title</a>
            </td> </tr>
            <tr> <td> </td> <td width=2>&nbsp;</td> <td>
            <font face=verdana,arial,helvetica size="-1">
            </td> </tr>
        </table>
    </td> </tr>
</table>");
    out.println("</font>");
}
<table>
<thead>
<tr>
<th>Publisher, Date</th>
</tr>
</thead>
</table>

```java
out.println("<tr> <td> </td> <td width=1>&nbsp;</td> <td> <font face=verdana,arial,helvetica size="-1">";
out.println("<a href="http://localhost:8080/servlet/ReqCollect/publisher_date">Publisher, Date</a></font></td> </tr>");
out.println("</table></p>";
out.println("</td>");
out.println("<td rowspan=4 width=5>&nbsp;&nbsp;<br/></td> ");
out.println("<td valign=top align=left>";
out.println("<br clear=all>";
```

```java
}
```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.sql.*;

public class PublisherDateProcess extends HttpServlet {

    protected Connection con = null;

    public void init() throws ServletException {
        try {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            con = DriverManager.getConnection("jdbc:odbc:fxbookstore", null, null);
        } catch (ClassNotFoundException e) {
        } catch (SQLException e) {
        }
    }

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

        String[] values;
        String name1 = "";
    }
values = req.getParameterValues("publisher");
if (values != null) {
    name1 = values[0];
}

String name2 = "";
values = req.getParameterValues("publication_date");
if (values != null) {
    name2 = values[0];
}

String sqlStmt = "";
sqlStmt = "select * from BOOK where publisher=' " + name1 + "' and publication_date=' " + name2 + "'";

ResultSet rs = null;
int count = 0;

// create header here
CreateHeader(out);

// Create left side menu
CreateLeftMenu(out);

// "Your Book Search Results"
try {
    if (con != null) {
        Statement stmt = con.createStatement();
        rs = stmt.executeQuery(sqlStmt);

        out.println("<dl>"); // start Glossary List
        while (rs.next()) {
            count++;
            out.println("<dt><b>" + count + ".</b> ");
            String isbn = rs.getObject("ISBN").toString();
            out.println("<a href="/servlet/DetailContent/" + isbn + ">");
            out.println(rs.getObject("Title").toString() + "</a></b>"));
        }
    }
}
out.println("<nobr><font color=#990033>Usually ships in 24 hours</font></nobr>");
out.println("<dd> +
rs.getObject("Author").toString();
out.println("/ " +
rs.getObject("Publisher").toString() + " / " + rs.getObject("PublicationDate").toString();
out.println("<br> Our Price: " +
rs.getObject("Price").toString();
out.println("<br><br>");
}
if (count == 0) {
out.println("<dt> No matched data found</dt>");
} out.println("</dl>"); // end Glossary List

} catch (SQLException e) {
// create footer here
out.println("</td> </tr>");
out.println("</table>");

    // Create standard footer
CreateFooter(out);
out.flush();
out.close();
}

protected void CreateHeader(PrintWriter out) {
out.println("<html>");
out.println("<head>");
out.println("<title> TinyBookStore.com </title>");
out.println("</head>");

out.println("<body>");
out.println("<table border=0 width="100%" cellspacing=0 cellpadding=0
gcolor="EEEECC"/>");
out.println("<tr>");
out.println("<td align=left valign=top width=90 >");
out.println("<IMG border=0 hspace=0 id=IMG1 src="d:\JavaWebServer2.0\public_html\flower.gif"></td> ");
out.println("<td width=30></td>");
out.println("<td align=left valign=center width=300> ");
out.println("<strong><h2>i<TinyBookStore.com</i></h2></strong></td>");
out.println("<td align=right valign=center> ");
out.println("</td>");
out.println("<td width=20></td>");
out.println("</tr>");
out.println("</td>");
out.println("</tr>");
out.println("</table>");

protected void CreateFooter(PrintWriter out) {
    // Create StandardFooter
    out.println("<table width=100%>");
    out.println("<tr><td width=50% valign=top align=left> </td>");
    out.println("<td width=50% valign=top align=right>");
    out.println("<font size=-1><a href="#top">Top of Page</a></font>");
    out.println("</td>");
    out.println("</tr>");
    out.println("</table>");

    out.println("<center>");
    out.println("<font size=-1>");
    out.println("<a href="http://localhost:8080/welcome.shtml">Tinybookstore.com Home</a>
    &nbsp;&nbsp;");
    out.println("<a href="http://localhost:8080/servlet/ReqCollect/category">Category</a>
    &nbsp;&nbsp;");
    out.println("<a href="http://localhost:8080/servlet/ReqCollect/author">Author</a>
    &nbsp;&nbsp;");
    out.println("<a href="http://localhost:8080/servlet/ReqCollect/title">Title</a>
    &nbsp;&nbsp;");
    &nbsp;&nbsp;");
    out.println("<a href="http://localhost:8080/servlet/ReqCollect/publisher_date">Publisher & #44 Date</a>");
    out.println("<p>");
    out.println("</font>");
    out.println("</center>");
    out.println("<p>");
    out.println("</center>");
protected void CreateLeftMenu(PrintWriter out) {
    System.out.println("LeftMenu");
    out.println("<table border=0 width="100\%" cellspacing=0 cellpadding=0>");
    out.println("<tr> <td valign=top width=155 rowspan=3 bgcolor=#eeeecc>");
    out.println("<p>");
    out.println("<table width="100\%" cellspacing=0 cellpadding=5 border=0">");
    out.println("<tr> <td> </td> <td colspan="2"> <font face=verdana,arial,helvetica SIZE="-1">&nbsp;</font> </td> </tr>");
    out.println("<tr> <td> </td> <td colspan="2"> <strong><font face=verdana,arial,helvetica size=\-V\-l">Search Books</font></strong> </td> </tr>");
    out.println("<tr> <td> </td> <td width=2>&nbsp;</td> <td> <font face=verdana,arial,helvetica size="-1"> <a href="http://localhost:8080/welcome.shtmI">Home</a> </font> </td> </tr>");
    out.println("<tr> <td> </td> <td width=2>&nbsp;</td> <td> <font face=verdana,arial,helvetica size="-1"> <a href="http://localhost:8080/servlet/ReqCollect/category">Category</a> </font> </td> </tr>");
    out.println("<tr> <td> </td> <td width=2>&nbsp;</td> <td> <font face=verdana,arial,helvetica size="-1"> <a href="http://localhost:8080/servlet/ReqCollect/author">Author</a> </font> </td> </tr>");
}

<table>
<thead>
<tr>
<th>Title</th>
<th>ISBN</th>
<th>Publisher, Date</th>
</tr>
</thead>
</table>

```java
out.println("<a href="http://localhost:8080/servlet/ReqCollect/title">Title</a></td></tr>");
out.println("<tr> <td width=1>&nbsp;</td> <td width=1>&nbsp;</td> <td> <font face=verdana,arial,helvetica size="-1">"\>ISBN</a></td></tr>");
out.println("<a href="http://localhost:8080/servlet/ReqCollect/publisher_date">Publisher, Date</a></td></tr>");
```

```java
out.println("</table></p>");
```
12. ReqCollect.java

/**
 * This is a servlet responsible for generating search request collection page. It read extra path information from URL, and then, according to the extra path information, it generates corresponding page format
 *
 * Author: Fang Xiao
 * Purpose: Thesis project
 * Copyright (C) Fang Xiao 1999-2001
 */

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

public class ReqCollect extends HttpServlet {

    protected final String CATEGORY = "category";
    protected final String AUTHOR = "author";
    protected final String TITLE = "title";
    protected final String ISBN = "isbn";
    protected final String PUBLISHER_DATE = "publisher_date";

    public void service(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException {

        res.setContentType("text/html");
        PrintWriter out = res.getWriter();

        String path = req.getPathInfo();
        if (path == null) {  // should not happen
            out.println("Check the URL to make sure it is correct");
            out.flush();
            out.close();
        }
        else {
            // Create standard header
            CreateHeader(out);

            // Create left side menu
        }
    }
}
CreateLeftMenu(out);

path = path.substring(1);
if (path.equalsIgnoreCase(CATEGORY)) {
    // Create category request collect

    out.println("<p>");
    out.println("<h3><strong>Programming</strong></h3>");
    "http://localhost:8080/servlet/CategoryProcess/c02">C&C++</a> &
    "http://localhost:8080/servlet/CategoryProcess/c03">Java</a> &
    "<a href="http://localhost:8080/servlet/CategoryProcess/c04">Delphi</a> &
    "<a href="http://localhost:8080/servlet/CategoryProcess/c05">MFC, Perl</a> &
    "software Design, UML, Visual, Basic, More...";

    out.println("<h3><strong>Web Development</strong></h3>");
    out.println("ASP, Commerce, Cyberculture, HTML, JavaScript, &
    "Security, Web Design, XML, more...";

    out.println("<h3><strong>Graphics & Software</strong></h3>");
    out.println("Adobe, CAD, Desktop, Publishing, Graphics, &
    "Microsoft, More...";

    out.println("<h3><strong>Networking & OS</strong></h3>");
    out.println("Linux, Macs, Oracle, PCs, SAP R/3, SQL, &
    "TCP/IP, Unix, Windows& DOS, Windows NT, &
    "More...";

}
else if (path.equalsIgnoreCase(AUTHOR)) {
    // Create author request collect
Enter Author: You can enter author's full name, first name or last name.

```
<form method="POST" action="http://localhost:8080/servlet/AuthorProcess" id=form name=form>
  <table border=0>
    <tr> <td valign=middle align=left> Author: </td> <td valign=middle align=left><input type="text" name="author" size=40 value="" /></td> </tr>
    <tr> <td colspan=2> <center> <input type="radio" name="author_mode" value="exact" checked> Exact Name <input type="radio" name="author_mode" value="last"> Last, First Name (or Initial) </center> </td> </tr>
    <tr> <td colspan=2> <p> </p> </td> </tr>
  </table>
  <center> <input type="submit" value="Search Now" id=submit name=submit > <input type="reset" value="Clear Form" id=reset name=reset > </center>
</form>
```

else if (path.equalsIgnoreCase(TITLE)) {

Enter Title: You can enter the exact title or part of title.

```
<form method="POST" action="http://localhost:8080/servlet/TitleProcess" id=form name=form>
  <table border=0>
    <tr> <td valign=middle align=left> Title: </td> <td valign=middle align=left><input type="text" name="title" size=40 value="" /></td> </tr>
  </table>
  <center> <input type="submit" value="Search Now" id=submit name=submit > <input type="reset" value="Clear Form" id=reset name=reset > </center>
</form>
```
out.println("<td align=middle align=left><input type="text" name="title" size=40 value=""/></td>");
out.println("</tr>");
out.println("<tr>");
out.println("<td colspan=2>");
out.println("<font size=-1>");
out.println("<center>");
out.println("<input type="radio" name="title-mode" value="exact" checked> Exact Title">");
out.println("<input type="radio" name="title-mode" value="titlewords"> Title Words >");
out.println("</center> </font> </td> </tr>");
out.println("</table>");
out.println("<center>");
out.println("<input type="submit" value="Search Now" id=submit1 name=submit1>");
out.println("<input type="reset" value="Clear Form" id=reset1 name=reset1>");
out.println("</center>");
out.println("</form>");

out.println("<hr noshade size=1>");

out.println("<B>Examples:</B>");
out.println("<UL>");
out.println("<LI>Entering <b>&quot;Java Servlet Programming&quot;</b> in the title field and choosing <em>Exact Title</em> finds the book without also finding the many books">
out.println("and choosing <em>Exact Title</em> finds the book without also finding the many books">
out.println("<LI>Entering <b>&quot;Java&quot;</b> in the Title field and choosing <em>Title Words</em> finds all books whose title are related to <i>Java</i>
out.println("</UL>");
...
```java
out.println("<br>");
out.println("<form method="POST" action="http://localhost:8080/servlet/PublisherDateProcess" id=form1 name=form1>");
out.println("<table border=0>");
out.println("<tr>");
out.println("<td valign=middle align=left>Publisher:<b>"</b><input type="text" name="publisher" size=40 value=""/>");
out.println("</td>");
out.println("<tr>");
out.println("<td valign=middle align=left>Publication Date:<b>"</b><input type="text" name="publication_date" size=40 value=""/>");
out.println("</td>");
out.println("</tr>");
out.println("</table>");
out.println("<center>");
out.println("<input type="submit" value="Search Now" id=submit1 name=submit1>");
out.println("<input type="reset" value="Clear Form" id=reset1 name=reset1>");
out.println("</center>");
out.println("</form>");
out.println("<hr noshade size=1>");
```

protected void CreateFooter(PrintWriter out) {
    // Create StandardFooter
    out.println("<table width=100\%>");
    out.println("<tr> <td width=50\% valign=top align=left> </td> <td width=50\% valign=top align=right> <font size=-1><a href="#top">Top of Page</a></font> </td> </tr> ");
    out.println("<table> ");
    out.println("<center> ");
    out.println("<a href="http://localhost:8080/welcome.shtml">Tinybookstore.com Home</a> 
    &nbsp;&nbsp;");
    out.println("<a href="http://localhost:8080/servlet/ReqCollect/category">Category</a> 
    &nbsp;&nbsp;");
    out.println("<a href="http://localhost:8080/servlet/ReqCollect/author">Author</a> 
    &nbsp;&nbsp;");
    out.println("<a href="http://localhost:8080/servlet/ReqCollect/title">Title</a> 
    &nbsp;&nbsp;");
    &nbsp;&nbsp;");
    out.println("<a href="http://localhost:8080/servlet/ReqCollect/publisher_date">Publisher Date</a> ");
    out.println("<p> ");
protected void CreateLeftMenu(PrintWriter out) {
    out.println("<table border=0 width='100%' cellspacing=0 cellpadding=0>");
    out.println("<tr> <td valign=top width=155 rowspan=3 bgcolor=#eeeecc>");
    out.println("<p>");
    out.println("<table width=100% cellspacing=0 cellpadding=5 border=0>");
    out.println("<tr> <td> </td> <td colspan="2">&nbsp;</td> </tr>");
    out.println("<tr> <td> </td> <td colspan="2"><strong>Search Books</strong> </td> </tr>");
    out.println("<tr> <td> </td> <td width=2>&nbsp;</td> <td> <a href="http://localhost:8080/welcome.shtml">Home</a></td> </tr>");
    out.println("<tr> <td> </td> <td width=2>&nbsp;</td> <td> <a href="http://localhost:8080/servlet/ReqCollect/category">Category</a></td> </tr>");
    out.println("<tr> <td> </td> <td width=2>&nbsp;</td> <td> <a href="http://localhost:8080/servlet/ReqCollect/author">Author</a></td> </tr>");
}

&copy; 1999-2001, Fang Xiao
<table>
<thead>
<tr>
<th>Title</th>
<th>ISBN</th>
<th>Publisher, Date</th>
</tr>
</thead>
</table>

```java
out.println("<tr>
<td><a href="http://localhost:8080/servlet/ReqCollect/title">Title</a></td>
<td width=2>&nbsp;</td>
<td><font face=verdana,arial,helvetica size="-1">"</font></td>
</tr>

<td width=1>&nbsp;</td>
<td><font face=verdana,arial,helvetica size="-1">"</font></td>
</tr>

out.println("<a href="http://localhost:8080/servlet/ReqCollect/publisher_date">Publisher, Date</a></td>
<td><font face=verdana,arial,helvetica size="-1">"</font></td>
</tr>

out.println("</table></p>");
```
```
13. ShoppingCart.java

/**
 * This is the definition for ShoppingCart
 *
 * Author: Fang Xiao
 * Purpose: Thesis project
 * Copyright (C) Fang Xiao 1999 - 2001
 */
import java.util.*;

public class ShoppingCart {
    Hashtable items = null;
    int numberOfItems = 0;

    public ShoppingCart() {
        items = new Hashtable();
    }

    /**
     * add BookDetails to cart
     * @param bookID isbn
     * @param book book need to be added to cart
     * @return void
     */
    public void add(String bookId, BookDetails book) {
        if(items.containsKey(bookId)) { // checking if item already exists
            ShoppingCartItem scItem = (ShoppingCartItem)items.get(bookId);
            scItem.incrementQuantity();
        } else {
            ShoppingCartItem newItem = new ShoppingCartItem(book);
            items.put(bookId, newItem);
            numberOfItems++;
        }
    }

    /**
     * @param bookId isbn
     */
    public void remove(String bookId) {
        if(items.containsKey(bookId)) {
            items.remove(bookId); // removing item from cart
            numberOfItems--;
        }
    }
}
items.remove(bookId);
    numberOfItems--;

    }
}

public Enumeration getItems() {
    return items.elements();
}

protected void finalize() throws Throwable {
    items.clear();
}

public int getNumberOfItems() {
    return numberOfItems;
}

public void clear() {
    items.clear();
    numberOfItems = 0;
}
public class ShoppingCartItem {
    BookDetails item;
    int quantity;

    public ShoppingCartItem(BookDetails anItem) {
        this.item = anItem;
        this.quantity = 1;
    }

    public void decrementQuantity() {
        if (quantity > 0)
            quantity--;
    }

    public void incrementQuantity() {
        this.quantity++;
    }

    public BookDetails getItem() {
        return this.item;
    }

    public int getQuantity() {
        return this.quantity;
    }
}
package servlets;

import javax.servlet.*;

/**
 * This is a server side include servlet that will
 * format the TinyBookStore HTML footer.
 */

public class StandardFooter extends GenericServlet {

    /**
     * Perform the servlet service
     *
     * @param req The request from the client
     * @param res The response from the servlet
     */
    public void service (ServletRequest req, ServletResponse res) throws ServletException, java.io.IOException {
        java.io.PrintWriter out = new java.io.PrintWriter(res.getOutputStream());

        // Format the standard footer
        out.println("<table width=100%>");
        out.println("<tr> <td width=50% valign=top align=left> </td> <td width=50% valign=top align=right> </td> </tr> ");
        out.println("<font size=-1><a href="/welcome.shtml">Tinybookstore.com Home</a> &nbsp; &nbsp; <a href="/category.html">Category</a>&nbsp; &nbsp; &nbsp; <a href="/author.html">Author</a> &nbsp; &nbsp; &nbsp; <a href="/title.html">Title</a> &nbsp; &nbsp; &nbsp; Top of Page</a></font>");
        out.println("</tr>");
        out.println("</table>");
        out.println("<center>");
        out.println("<a href="/welcome.shtml">Top of Page</a> &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &n
out.println("<a href="/isbn.html">ISBN</a> &nbsp;
};
out.println("<a href="/publisher.html">Publisher</a>
};
out.println("<p>
};
out.println("<center>
};
out.println("<br><font size=-1>
};
out.println("<A href="/copyright.html">Copyright and disclaimer</A> © 1999-2001, Fang Xiao"
};
out.println("</font>
};
out.println("</center>
};
out.println("</body>
};
out.println("</html>
};
out.flush
};
out.close
};
16. StandardHeader.java

package servlets;

import javax.servlet.*;

/**
 * This is a server side include servlet that will format the standard TinyBookStore HTML header. The Title of the page will be set to the value of the title property.
 */
public class StandardHeader extends GenericServlet
{

    /*<p> Performs the servlet service
     */
    @param req The request from the client
    @param res The response from the servlet */
    public void service(ServletRequest req, ServletResponse res)
        throws ServletException, java.io.IOException {
        // Create a PrintWriter to write the response
        java.io.PrintWriter out = new java.io.PrintWriter(res.getOutputStream());
        // Get the title of the page. Set to empty string if no title parameter was given
        String titles[] = req.getParameterValues("title");
        String title = ";";
        if (titles != null) {
            if (titles.length > 0) {
                title = titles[0];
            }
        }
        out.println("<html>");
        out.println("<head>");
        out.println("<title>" + title + "</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<table border=0 width="100%" cellspacing=0 cellpadding=0 bgcolor="EEEECC"/>");
```java
out.println("<tr>");
out.println("<td align=left valign=top width=90 >");
out.println("<IMG border=0 hspace=0 id=IMG1 src="d:\JavaWebServer2.0\public_html\flower.gif"></td>");
out.println("<td width=30 ></td>");
out.println("<td align=left valign=center width=300 >");

out.println("<strong><h2><i>TinyBookStore.com</i></h2></strong></td>");
out.println("<td align=right valign=center>");
out.println("</td>");
out.println("<td width=20 ></td>");
out.println("<tr>");
out.println("</table>");

out.flush();
out.close();
```
17. TitleProcess.java

/**
 * TitleProcess
 *
 * This is the servlet that process the Title-search request and generate
 * search result.
 *
 * Author: Fang Xiao
 * Purpose: Thesis project
 * Copyright (C) Fang Xiao 1999-2001
 */

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

public class TitleProcess extends HttpServlet {
    protected final String EXACT = "exact";
    protected final String OTHERS = "titleWords";

    protected Connection con = null;

    public void init() throws ServletException {
        try {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            con = DriverManager.getConnection("jdbc:odbc:fxbookstore",
            null, null);
        } catch (ClassNotFoundException e) {
        } catch (SQLException e) {
        }
    }

    public void doPost(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException {
        res.setContentType("text/html");
        PrintWriter out = res.getWriter();
String[] values;
String name = "";

values = req.getParameterValues("title");
if (values != null) {
    name = values[0];
}

String mode = "";
values = req.getParameterValues("title_mode");
if (values != null) {
    mode = values[0];
}

String sqlStmt = "";
if (mode.equalsIgnoreCase(EXACT)) {
    sqlStmt = "select * from BOOK where title='" + name + ";";
} else {
    sqlStmt = "select * from BOOK where title like '%" + name + ";";
}

ResultSet rs = null;
int count = 0;

// create header here
CreateHeader(out);

// Create left side menu
CreateLeftMenu(out);

// "Your Book Search Results"
try {
    if (con != null) {
        Statement stmt = con.createStatement();
        rs = stmt.executeQuery(sqlStmt);

        out.println("<dl>"); // start Glossary List
        while (rs.next()) {
            count++;
            out.println("<dtb>" + count + "");
            String isbn = rs.getObject("ISBN").toString();
        }
    }

    out.println("</dl>"); // end Glossary List
}

// rest of the code...
out.println("<a href="/servlet/DetailContent/" + isbn
+ ">")
"</a></b>");
out.println(rs.getObject("Title").toString() +
"<nobr><font color=#990033>Usually
ships in 24 hours</font></nobr>");
out.println("<br>Our Price: " +
rs.getObject("Author").toString());
out.println("/ " +
rs.getObject("Publisher").toString() + " / " + rs.getObject("PublicationDate").toString());
out.println("<br>Our Price: "+
rs.getObject("Price").toString());
}
out.println("<br></br>");
if (count == 0) {
out.println("<dt>No matched data found";
}
out.println("</dl>"; // end Glossary List
}
}
// create footer here
out.println("</td> </tr>");
out.println("</table>");
// Create standard footer
CreateFooter(out);
out.flush();
out.close();
}

protected void CreateHeader(PrintWriter out) {
out.println("<html>");
out.println("<head>");
out.println("<title>TinyBookStore.com</title>");
out.println("</head>");
out.println("<body>");
out.println("<html>");
out.println("<head>");
out.println("<title>TinyBookStore.com</title>");
out.println("</head>");
out.println("<body>");
1 3 1

out.println("<table border=0 width="100%" cellspacing=0 cellpadding=0gcolor="EEEECC">"); out.println("<tr>"); out.println("<td align=left valign=top width=90 >"); out.println("<IMG border=0 hspace=0 id=IMG1 src="d:\JavaWebServer2.0\public_html\flower.gif"></td>" ); out.println("<td width=30 ></td>"); out.println("<td align=left valign=center width=300 >"); out.println("<strong><h2><i>TinyBookStore.com</ix/h2></strong></td>"); out.println("<td align=right valign=center>"); out.println("<td width=20 ></td>"); out.println("</tr>"); out.println("</table>");

protected void CreateFooter(PrintWriter out) {
// Create StandardFooter
out.println("<table width= 100%>"); out.println("<tr> <td width=50% valign=top align=left> </td> "); out.println("<tr> <td width=50% valign=top align=right> "); out.println("<font size=-1><a href="#top">Top of Page</a></font>"); out.println("</tr>"); out.println("</table>");

out.println("<center>"); out.println("<font size=-1>"); out.println("<a href="http://localhost:8080/welcome.shtml">Tinybookstore.com Home</a>

out.println("<a href="http://localhost:8080/servlet/ReqCollect/category">Category</a> 

out.println("<a href="http://localhost:8080/servlet/ReqCollect/author">Author</a> 

out.println("<a href="http://localhost:8080/servlet/ReqCollect/title">Title</a> 


out.println("<a href="http://localhost:8080/servlet/ReqCollect/publisher_date">Publisher&#44 Date</a> 

out.println("<p>");
protected void CreateLeftMenu(PrintWriter out) {
    System.out.println("LeftMenu");
    out.println("<table border=0 width="100\%" cellspacing=0
   cellpadding=0>\n   <tr> <td valign=top width=155 rowspan=3
   bgcolor=#eeeecc>\n   <p> \n   "Search Books</p> <a href="http://localhost:8080/welcome.shtml">Home</a> <a href="http://localhost:8080/servlet/ReqCollect/category">Category</a>\n   </tr> \n   </table>\n   <table width="100\%" cellspacing=0 cellpadding=5
   border=0>\n   <tr> <td width=2>&nbsp;</td> <td> <font face=verdana,arial,helvetica size="-1">Home</font>\n   </td> <td width=2>&nbsp;</td> <td> <font face=verdana,arial,helvetica size="-1">Category</font>\n   </td> \n   </tr> \n   </table>\n   </font>\n   </body>\n   </html>\n});
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
<html>
<head> <title>Customer Information</title> </head>
<body>
<h1> <center> <b>Customer Information</b> </center> </h1>
<table border=0>
<tr>
<td align=right>First name:</td>
<td colspan=2 align=left><input type=text name=firstname size=40></td>
</tr>
<tr>
<td align=right>Middle initial:</td>
<td colspan=2 align=left><input type=text name=middleinit size=40></td>
</tr>
<tr>
<td align=right>Last name:</td>
<td colspan=2 align=left><input type=text name=lastname size=40></td>
</tr>
<tr>
<td align=right>Email address:</td>
<td colspan=2 align=left><input type=text name=email size=40></td>
</tr>
<tr>
<td align=right>Account password:</td>
<td colspan=2 align=left><input type=password name=password size=40></td>
</tr>
</table>
<br>
<table border=0>
<tr>
<td align=right>Credit card number:</td>
<td colspan=2 align=left><input type=text name=creditcard size=40></td>
</tr>
<tr valign=top>
</tr>
</table>
</form>
</body>
</html>
<td align=right>Credit card type:</td>
<td align=left>
<input type=radio name=cardtype value="visa">Visa<br>
<input type=radio name=cardtype value="mastercard">MasterCard<br>
</td>

<td align=left>
<input type=radio name=cardtype value="bluebird">Bluebird<br>
<input type=radio name=cardtype value="fisher">Fisher<br>
</td>

</tr>

<tr>
<td align=right>Credit card holder:</td>
<td colspan=2 align=left>
<input type=text name=cardholder size=40></td>
</tr>

<tr>
<td align=right>Credit card expire date (mm/dd/yy):</td>
<td colspan=2 align=left>
<input type=text name=expiredate size=40></td>
</tr>

<tr>
<td align=right>Select shipping method:</td>
<td align=left>
<input type=radio name=shipping value="ups">UPS<br>
<input type=radio name=shipping value="express">Express<br>
</td>

<td align=left>
<input type=radio name=shipping value="normal">Normal<br>
<input type=radio name=shipping value="abnormal">Abnormal<br>
</td>

</tr>

<tr>
<td align=right>Receiver Name:</td>
<td colspan=2 align=left>
<input type=text name=receiver size=40></td>
</tr>

<tr>
<td align=right>Address:</td>
<td colspan=2 align=left>
<input type=text name=address size=40></td>
</tr>

<tr>
<td align=right>City:</td>
<td colspan=2 align=left>
<input type=text name=city size=40></td>
</tr>
<table>
<thead>
<tr>
<th>State</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Zipcode</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Phone</td>
<td></td>
</tr>
</tbody>
</table>

<input type=submit value="Process" name=action>
APPENDIX C

Screen Display of the Test
Welcome to the TinyBookStore.com

Five Hot Books

1. **The C++ Programming Language**
   Usually ships in 24 hours
   Bjarne Stroustrup / Addison-Wesley Pub Co. / 1997
   Our Price: $31.47

2. **Programming Visual C++**
   Usually ships in 24 hours
The C++ Programming Language
by Bjarne Stroustrup

Price: 31.47
Availability: Usually ships within 24 hours.

ISBN 0201889544
Addison-Wesley Pub Co. 1997

Reviews
Shopping Cart Items

The C++ Programming Language
Bjarne Stroustrup
Quantity: 1

Price: $31.47
Subtotal: $31.47
Total: $31.47

Customer Information

First name: Fang
Middle initial: A
Last name: Xiao
Email address: fxiao@mitec.net
Account password: pooooc
Credit card number: 1234567891234567
Credit card type: Visa
Credit card holder: Fang Xiao
Credit card expire date (mm/dd/yy): 01/01/02
Select shipping method: UPS

Done
Your order reads as follows

E-Mail Address: fxiao
Fang Xiao
10821 Western Plaza

ship to: Omaha
NE
68154
Tel: (402) 496-7570

Items:
Title: The C++ Programming Language
Qty: 1
Price: 31.47, Subtotal: 31.47
1. **Effective C++** Usually ships in 24 hours
   Scott Meyers / Addison-Wesley Pub Co. / 1997
   Our Price: 37.95

Review:

This is the review of Effective C++.

Price: 37.95
Availability: Usually ships within 24 hours.
1. **Applying UML and Pattern** Usually ships in 24 hours

Craig Larman / Prentice Hall / 1997

Our Price: 45.0

ISBN 0137488807

Prentice Hall 1997

Price: 45.0

Availability: Usually ships within 24 hours.

Reviews

This is the review of *Applying UML and Pattern*.

Reader Comments
Shopping Cart Items

**Applying UML and Pattern**

- Craig Larmen
  - Quantity: 1
  - Price: $45.0
  - Subtotal: $45.0

- Effective C++
  - Scott Meyers
  - Quantity: 1
  - Price: $37.95
  - Subtotal: $37.95

Total: $82.95
**TinyBookStore.com**

### Programming Windows with MFC

**by Jeff Prosise**

- **Price:** $47.99
- **Availability:** Usually ships within 24 hours.

**ISBN:** 1572316950

**Publisher/Year:** Microsoft Press 1999

### Reviews

This is the review of Programming Windows with MFC.

### Reader Comments

---

**Shopping Cart Items**

<table>
<thead>
<tr>
<th>Item</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Programming Windows with MFC</strong></td>
<td><strong>Price:</strong> $47.99</td>
</tr>
<tr>
<td>Jeff Prosise</td>
<td></td>
</tr>
<tr>
<td><strong>Applying UML and Pattern</strong></td>
<td><strong>Price:</strong> $45.0</td>
</tr>
<tr>
<td>Craig Larmen</td>
<td></td>
</tr>
<tr>
<td><strong>Effective C++</strong></td>
<td><strong>Price:</strong> $37.95</td>
</tr>
<tr>
<td>Scott Meyers</td>
<td></td>
</tr>
</tbody>
</table>

**Total:** $130.94

---
TinyBookStore.com

Programming

Algorithms, C/C++, Delphi, Java, MFC, Perl, software Design, UML, Visual, Basic, More...

Web Development

ASP, Commercem, Cyberculture, HTML, JavaScript, Security, Web Design, XML, more...

Graphics & Software

Adobe, CAD, Desktop, Publishing, Graphics, Microsoft, More...
TinyBookStore.com

1. **Core Java 2** Usually ships in 24 hours  
   Cay S Horstmann / Prentice Hall / 1999  
   Our Price: 25.79

2. **Java Servlet Programming** Usually ships in 24 hours  
   Jason Hunter / O'Reilly / 1993  
   Our Price: 19.77

3. **Just Java 2** Usually ships in 24 hours  
   Perter van der Linden / Prentice Hall / 1999  
   Our Price: 31.49
Search Books

Core Java 2
by Cay S Horstmann

Price: 25.79
Availability: Usually ships within 24 hours.

Reviews
This is the review of Core Java 2.

Reader Comments

Shopping Cart Items

Core Java 2
Cay S Horstmann
Quantity: 1

Price: $25.79
Subtotal: $25.79

Total: 25.79
TinyBookStore.com

Java Servlet Programming
by Jason Hunter

Price: 19.77
Availability: Usually ships within 24 hours.

ISBN 156592391x
O'Reilly 1998

Reviews

In this brand-new third edition of Java Servlet Programming, author Bjarne Stroustrup, the creator of Java, presents the full specification for the Java language and standard library, a spec that will soon become the joint ISO/ANSI Java standard.

TinyBookStore.com Shopping Cart

Shopping Cart Items

Core Java 2
Cay S Horstmann
Quantity: 1
Price: $25.79
Subtotal: $25.79

Java Servlet Programming
Jason Hunter
Quantity: 1
Price: $19.77
Subtotal: $19.77

Total: 45.56
Customer Information

First name: Fang
Middle initial: A
Last name: Xiao
Email address: fiao@mitec.net
Account password: jxxxxx

Credit card number: 1234567891234567
Credit card type: Vis a MasterCard Bluebird
Credit card holder: Fang Xiao
Credit card expire date (mm/dd/yy): 01/01/02
Select shipping method: UPS Normal Express Abnormal

Your order reads as follows

E-Mail Address: fiao@mitec.net
Fang Xiao
12345 Western Plaza
Omaha, NE 68154
Tel: (402) 123-4567

Items:
Title: Core Java 2, Qty: 1
Price: 25.79, Subtotal: 25.79
Title: Java Servlet Programming, Qty: 1
Price: 19.77, Subtotal: 19.77
Total: 45.56
TinyBookStore.com

1. Learning Perl
   Usually ships in 24 hours
   by Randal L. Schwartz
   O'Reilly 1997
   Our Price: 19.97

Price: 19.97
Availability: Usually ships within 24 hours.

This is the review of learning Perl.

Reader Comments
TinyBookStore.com

Search Books

Enter ISBN

ISBN: 1565921496

[Search Now] [Clear Form]
Programming Perl
by Larry Wall

Price: 23.97
Availability: Usually ships within 24 hours.

ISBN 1565921496
O'Reilly 1997

Reviews
This is the review of Programming Perl

Reader Comments
<table>
<thead>
<tr>
<th>Product</th>
<th>Price</th>
<th>Subtotal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning Perl</td>
<td>$19.97</td>
<td>$19.97</td>
</tr>
<tr>
<td>Programming Perl</td>
<td>$23.97</td>
<td>$23.97</td>
</tr>
</tbody>
</table>

Total: $43.94
TinyBookStore.com

Enter Publisher and Publication Date

Publisher: [Addison-Wesley Pub Co.]
Publication Date: [1997]

[Search Now] [Clear Form]

Tinybookstore.com Home | Category | Author | Title | ISBN | Publisher, Date

Copyright and disclaimer © 1999-2001, Fang Xiao
### Search Books

1. **Algorithms in C**
   - by Robert Sedgewick
   - Addison-Wesley Pub Co. / 1997
   - Our Price: 44.95

2. **The C++ Programming Language**
   - by Bjarne Stroustrup
   - Addison-Wesley Pub Co. / 1997
   - Our Price: 51.47

3. **Effective C++**
   - by Scott Meyers
   - Addison-Wesley Pub Co. / 1997
   - Our Price: 37.95

4. **UML Distilled**
   - by Martin Fowler
   - Addison-Wesley Pub Co. / 1997
   - Our Price: 29.95

#### Algorithms in C

**by Robert Sedgewick**

- **ISBN**: 0201314525
- **Publisher, Date**: Addison-Wesley Pub Co. / 1997

**Price**: 44.95

**Availability**: Usually ships within 24 hours.

**Reviews**

This is the review of *Algorithms in C*.

**Reader Comments**
### Shopping Cart Items

<table>
<thead>
<tr>
<th>Item Description</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Algorithms in C</em></td>
<td>Price: $44.95</td>
</tr>
<tr>
<td>Robert Sedgewick</td>
<td>Subtotal: $44.95</td>
</tr>
<tr>
<td>Quantity: 1</td>
<td>Total: $44.95</td>
</tr>
</tbody>
</table>

#### Local Intranet

```
http://localhost:8080/servlet/AddToShoppingCart
```
<table>
<thead>
<tr>
<th>Title</th>
<th>Publisher</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Beginning Visual Basic 6 Database Programming</td>
<td>Wrox Press Inc.</td>
<td>$31.99</td>
</tr>
<tr>
<td>2. Beginning Visual Basic 6</td>
<td>Wrox Press Inc.</td>
<td>$31.99</td>
</tr>
</tbody>
</table>
1. **The C++ Programming Language** Usually ships in 24 hours  
   Bjane Stroustrup / Addison-Wesley Pub Co. / 1997  
   Our Price: $31.47

2. **Effective C++** Usually ships in 24 hours  
   Scott Meyers / Addison-Wesley Pub Co. / 1997  
   Our Price: $37.95

3. **Programming Visual C++** Usually ships in 24 hours  
   David J. Kranginski / Microsoft Press / 1998  
   Our Price: $39.99

---

**Programming Visual C++**  
by David J. Kranginski  

**Price:** $39.99  
**Availability:** Usually ships within 24 hours.

ISBN 1572318570  
Microsoft Press 1998

**Reviews**

This is the review of Programming Visual C++.

**Reader Comments**
TinyBookStore.com

Enter Title
You can enter the exact title or part of title.

Title: [UML Distilled]

Examples:
- Entering "Java Servlet Programming" in the title field and choosing Exact Title finds the book without also finding the many books
- Entering "Java" in the Title field and choosing Title Words finds all books whose title are related to Java
TinyBookStore.com

Search Books

1. UML Distilled
   Usually ships in 24 hours
   Martin Fowler / Addison-Wesley Pub Co. / 1997
   Our Price: 29.95

Price: 29.95
Availability: Usually ships within 24 hours

Reviews

This is the review of The UML Distilled.

Reader Comments
<table>
<thead>
<tr>
<th>Book Title</th>
<th>Author(s)</th>
<th>Quantity</th>
<th>Price</th>
<th>Subtotal</th>
</tr>
</thead>
<tbody>
<tr>
<td>UML Distilled</td>
<td>Martin Fowler</td>
<td>1</td>
<td>$29.95</td>
<td>$29.95</td>
</tr>
</tbody>
</table>

Total: $69.94