Week 13:

Aim : Write a program to implement MVC architecture.

Description :

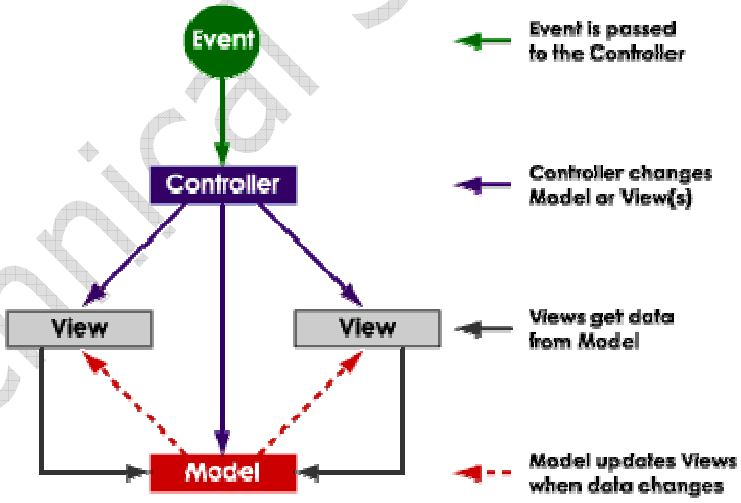
Model-View-Controller Pattern

Model-View-Controller (MVC) is a classic design pattern often used by applications that need the ability to maintain multiple views of the same data. The MVC pattern hinges on a clean separation of objects into one of three categories — models for maintaining data, views for displaying all or a portion of the data, and controllers for handling events that affect the model or view(s).

Because of this separation, multiple views and controllers can interface with the same model. Even new types of views and controllers that never existed before can interface with a model without forcing a change in the model design.

How It Works

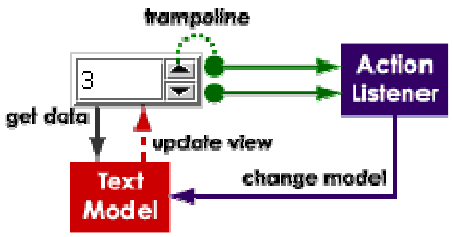
The MVC abstraction can be graphically represented as follows.



Events typically cause a controller to change a model, or view, or both. Whenever a controller changes a model's data or properties, all dependent views are automatically updated. Similarly, whenever a controller changes a view, for example, by revealing areas that were previously hidden, the view gets data from the underlying model to refresh itself.

A Concrete Example

We explain the MVC pattern with the help of a simple **spinner** component which consists of a text field and two arrow buttons that can be used to increment or decrement a numeric value shown in the text field. We currently do not have an element type that can directly represent a spinner component, but it easy is to synthesize a spinner using existing element types.



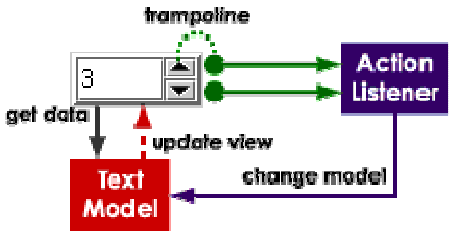
The spinner's data is held in a **model** that is *shared* with the text field. The text field provides a **view** of the spinner's current value. Each button in the spinner is an **event source**, that spawns an **action event** every time it is clicked. The buttons can be hooked up to trampolines that receive action events, and route them to an **action listener** that eventually handles that event. Recall that a trampoline is a predefined action listener that simply delegates action handling to another listener.

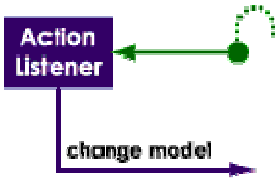
Depending on the source of the event, the ultimate action listener either increments or decrements the value held in the model — The action listener is an example of a **controller**.

The trampolines that initially receive the action events fired by the arrow buttons, are also controllers — However, instead of modifying the spinner's model directly, they delegate the task to a separate controller (action listener).

Multiple Controllers

The MVC pattern allows any number of controllers to modify the same model. While we have so far focused only on the two arrow buttons as likely source of events, there is, in fact, a third event source in this example — Whenever the text field has focus, hitting the **enter** key fires off an action event that may potentially be handled by a different action listener than the one handling action events from the buttons.





Program :

Eshop.jsp:

```
<%@ page session="true" %>
<html>
<head>
 <title>BOOK STORE</title>
</head>
<body bgcolor="#33CCFF">
 <font face="Times New Roman,Times" size="+3">
  BOOK STORE
 </font>
 <hr><p>
 <center>
 <form name="shoppingForm"
   action="ShoppingServlet"
   method="POST">
 <b>BOOK:</b>
 <select name=BOOK>
  <option> 10588 | Hans Bergsten | Java Server Pages | O'Reilly | 2 | 34.95</option>
  <option> 10589 | Deitel | Internet |McGraw Hill | 4 | 44.95</option>
  <option> 10590 | Winston | XML BIBLE |Wiley | 2 | 14.95</option>
 </select>
 <b>Quantity: </b><input type="text" name="qty" SIZE="3" value=1>
 <input type="hidden" name="action" value="ADD">
 <input type="submit" name="Submit" value="Add to Cart">
```

```
 </form>
 </center>
 <p>
 <jsp:include page="Cart.jsp" flush="true" />
</body>
</html>
```

Cart.jsp:

```
<%@ page session="true" import="java.util.*, shopping.BOOK" %>
<%
 Vector buylist = (Vector) session.getValue("shopping.shoppingcart");
 if (buylist != null && (buylist.size() > 0)) {
%>
<center>
<table border="0" cellpadding="0" width="100%" bgcolor="#FFFFFF">
 <tr>
 <td><b>ISBN</b></td>
 <td><b>AUTHOR</b></td>
 <td><b>TITLE</b></td>
 <td><b>PUBLISHER</b></td>
 <td><b>EDITION</b></td>
 <td><b>PRICE</b></td>
 <td><b>QUANTITY</b></td>
 <td></td>
 </tr>
 <%
  for (int index=0; index < buylist.size();index++) {
   BOOK anOrder = (BOOK) buylist.elementAt(index);
 %>
 <tr>
  <td><b><%= anOrder.getisbn() %></b></td>
```

```
<td><b><%= anOrder.getAuthor() %></b></td>
<td><b><%= anOrder.getTitle() %></b></td>
<td><b><%= anOrder.getPublisher() %></b></td>
<td><b><%= anOrder.getEdition() %></b></td>
 <td><b><%= anOrder.getPrice() %></b></td>
 <td><b><%= anOrder.getQuantity() %></b></td>
 <td>
  <form name="deleteForm"
   action="ShoppingServlet"
   method="POST">
  <input type="submit" value="Delete">
  <input type="hidden" name= "delindex" value='<%= index %>'>
  <input type="hidden" name="action" value="DELETE">
 </form>
    </td>
  </tr>
  <% } %>
 </table>
 <p>
 <form name="checkoutForm"
  action="ShoppingServlet"
  method="POST">
  <input type="hidden" name="action" value="CHECKOUT">
  <input type="submit" name="Checkout" value="Checkout">
 </form>
 </center>
<% } %>
```

Checkout.jsp:

```
<%@ page session="true" import="java.util.*, shopping.BOOK" %>
<html>
<head>
```

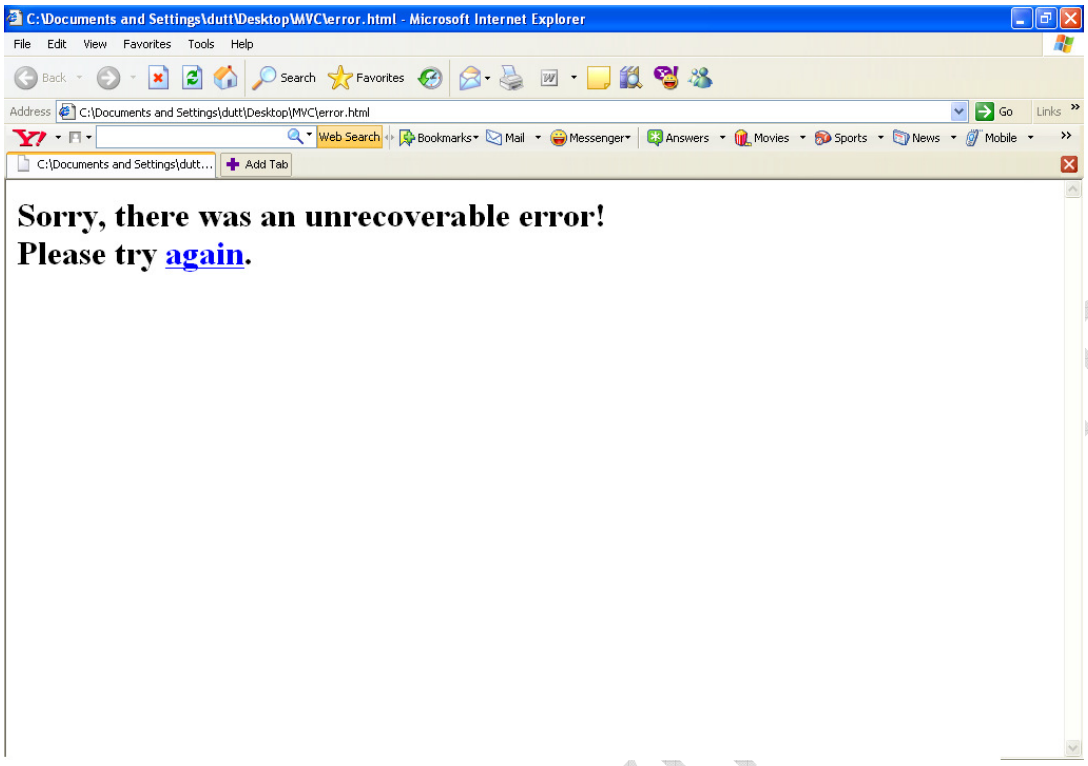
```
<title>BOOK STORE</title>
</head>
<body bgcolor="#33CCFF">
 <font face="Times New Roman,Times" size=+3>
  BOOK STORE CHECK OUT
 </font>
 <hr><p>
 <center>
 <table border="0" cellpadding="0" width="100%" bgcolor="#FFFFFF">
 <tr>
 <td><b>ISBN</b></td>
 <td><b>AUTHOR</b></td>
 <td><b>TITLE</b></td>
 <td><b>PUBLISHER</b></td>
 <td><b>EDITION</b></td>
 <td><b>PRICE</b></td>
 <td><b>QUANTITY</b></td>
 <td></td>
 </tr>
 <%
  Vector buylist = (Vector) session.getValue("shopping.shoppingcart");
  String amount = (String) request.getAttribute("amount");
  for (int i=0; i < buylist.size();i++) {
   BOOK anOrder = (BOOK) buylist.elementAt(i);
 %>
 <tr>
 <td><b><%= anOrder.getisbn() %></b></td>
 <td><b><%= anOrder.getAuthor() %></b></td>
 <td><b><%= anOrder.getTitle() %></b></td>
 <td><b><%= anOrder.getPublisher() %></b></td>
 <td><b><%= anOrder.getEdition() %></b></td>
```

```
<td><b><%= anOrder.getPrice() %></b></td>
<td><b><%= anOrder.getQuantity() %></b></td>
</tr>
<%
 }
 session.invalidate();
%>
<tr>
<td>    </td>
<td>    </td>
<td><b>TOTAL</b></td>
<td><b>$<%= amount %></b></td>
<td>    </td>
</tr>
</table>
<p>
<a href="Eshop.jsp">Shop some more!</a>
</center>
</body>
</html>
```

Error.html:

```
<html>
<body>
<h1>
  Sorry, there was an unrecoverable error! <br>
  Please try <a href="/examples/jsp/shopping/EShop.jsp">again</a>.
</h1>
</body>
</html>
```

OUTPUT:



Web.xml:

```
<web-app>
<servlet>
<servlet-name>ShoppingServlet</servlet-name>
<servlet-class>ShoppingServlet</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>ShoppingServlet</servlet-name>
<url-pattern>/ShoppingServlet</url-pattern>
</servlet-mapping>
</web-app>
```

ShoppingServlet.java:

```
import java.util.*;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import shopping.BOOK;
```

```
public class ShoppingServlet extends HttpServlet {
  public void init(ServletConfig conf) throws ServletException  {
    super.init(conf);
  }
  public void doPost (HttpServletRequest req, HttpServletResponse res)
      throws ServletException, IOException {
    HttpSession session = req.getSession(false);

    if (session == null) {
      res.sendRedirect("/error.html");
    }
    Vector buylist=(Vector)session.getValue("shopping.shoppingcart");
    String action = req.getParameter("action");
    if (!action.equals("CHECKOUT")) {
      if (action.equals("DELETE")) {
        String del = req.getParameter("delindex");
        int d = (new Integer(del)).intValue();
        buylist.removeElementAt(d);
      } else if (action.equals("ADD")) {
        boolean match=false;
        BOOK aCD = getBOOK(req);
        if (buylist==null) {
          buylist = new Vector(); //first order
          buylist.addElement(aCD);
        } else { // not first buy
          for (int i=0; i< buylist.size(); i++) {
            BOOK cd = (BOOK) buylist.elementAt(i);
            if (cd.getTitle().equals(aCD.getTitle())) {
              cd.setQuantity(cd.getQuantity()+aCD.getQuantity());
              buylist.setElementAt(cd,i);
              match = true;
```

```
            } //end of if name matches
          } // end of for
          if (!match)
            buylist.addElement(aCD);
        }
      }
      session.putValue("shopping.shoppingcart", buylist);
      String url="/Eshop.jsp";
//      ServletContext sc = getServletContext();
      RequestDispatcher rd = req.getRequestDispatcher(url);
      rd.forward(req, res);
    }
          else if (action.equals("CHECKOUT"))  {
      float total =0;
      for (int i=0; i< buylist.size();i++) {
        BOOK anOrder = (BOOK) buylist.elementAt(i);
        float price= anOrder.getPrice();
        int qty = anOrder.getQuantity();
        total += (price * qty);
      }
      total += 0.005;
      String amount = new Float(total).toString();
      int n = amount.indexOf('.');
      amount = amount.substring(0,n+3);
      req.setAttribute("amount",amount);
      String url="/Checkout.jsp";
      ServletContext sc = getServletContext();
      RequestDispatcher rd = sc.getRequestDispatcher(url);
      rd.forward(req,res);
    }
  }
```

```
  private BOOK getBOOK(HttpServletRequest req) {

    String myBOOK = req.getParameter("BOOK");
    String qty = req.getParameter("qty");
    StringTokenizer t = new StringTokenizer(myBOOK,"|");
    String isbn= t.nextToken();
    String  author= t.nextToken();
    String title = t.nextToken();
    String publisher = t.nextToken();
    String Edition = t.nextToken();
    String price = t.nextToken();
    price = price.replace('$',' ').trim();
    BOOK book = new BOOK();
    book.setisbn(isbn);
    book.setAuthor(author);
    book.setTitle(title);
    book.setPublisher(publisher);
    book.setEdition(Edition);
    book.setPrice((new Float(price)).floatValue());
    book.setQuantity((new Integer(qty)).intValue());
    return book;
  }
}
```

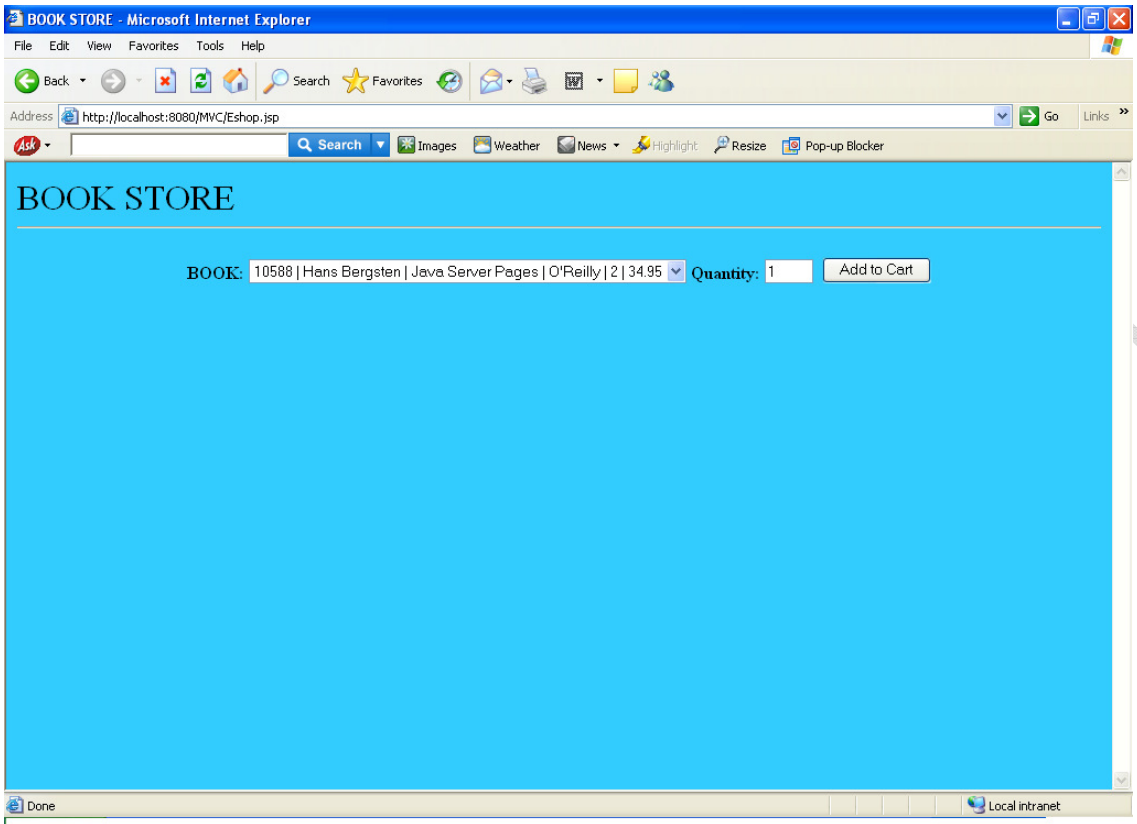
book.java:

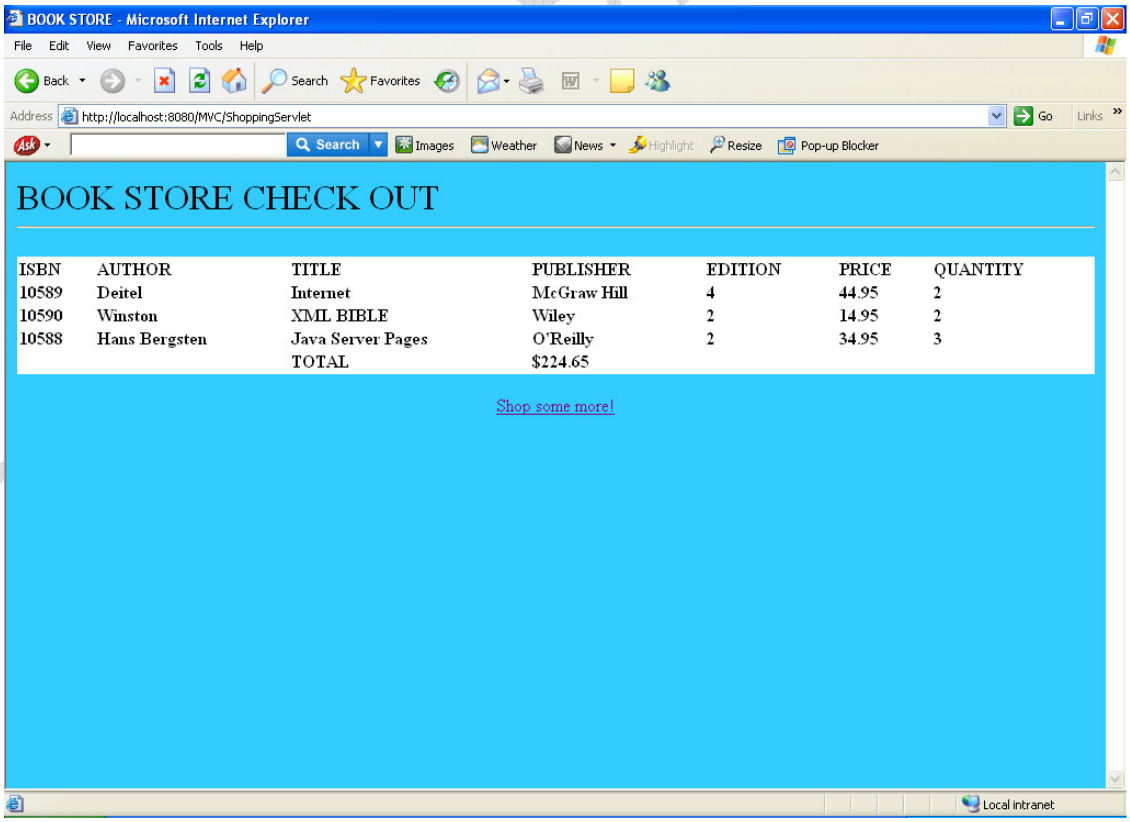
```
package shopping;
public class BOOK {
  String isbn;
  String author;
  String title;
  String publisher;
```

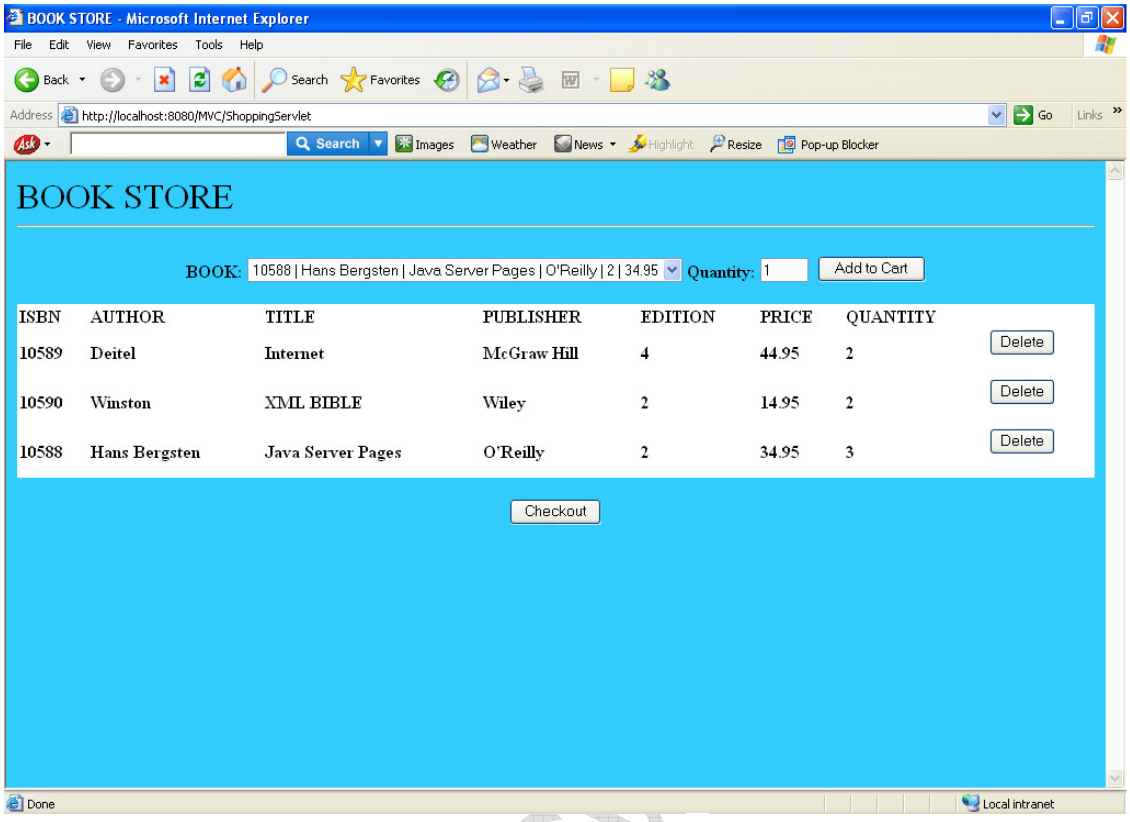
```
String Edition;
float price;
int quantity;
public BOOK() {
  isbn="";
  author="";
  title="";
  publisher="";
  Edition="";
  price=0;
  quantity=0;
}
public void setPublisher(String title1) {
  publisher=title1;
}
public String getPublisher() {
  return publisher;
}
public void setEdition(String group1) {
  Edition=group1;
}
public String getEdition() {
  return Edition;
}
 public void setisbn(String title) {
  isbn=title;
}
public String getisbn() {
```

```
    return isbn;
  }
  public void setAuthor(String group) {
    author=group;
  }
  public String getAuthor() {
    return author;
  }
  public void setTitle(String cty) {
    title=cty;
  }
  public String getTitle() {
    return title;
  }
  public void setPrice(float p) {
    price=p;
  }
  public float getPrice() {
    return price;
  }
  public void setQuantity(int q) {
    quantity=q;
  }
  public int getQuantity() {
    return quantity;
  }
}
```

OUTPUT:


BOOK STORE - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/MVC/Eshop.jsp
Go Links
Search Images Weather News Highlight Resize Pop-up Blocker
BOOK STORE
BOOK: 10588 | Hans Bergsten | Java Server Pages | O'Reilly | 2 | 34.95 Quantity: 1 Add to Cart
Done
Local intranet


BOOK STORE - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/MVC/ShoppingServlet
Go Links
Search Images Weather News Highlight Resize Pop-up Blocker
BOOK STORE CHECK OUT
ISBN AUTHOR TITLE PUBLISHER EDITION PRICE QUANTITY
10589 Deitel Internet McGraw Hill 4 44.95 2
10590 Winston XML BIBLE Wiley 2 14.95 2
10588 Hans Bergsten Java Server Pages O'Reilly 2 34.95 3
TOTAL $224.65
Shop some more!
Local intranet



RESULT: Thus the cart page is dynamited with HTTP sessions and useBean concept.