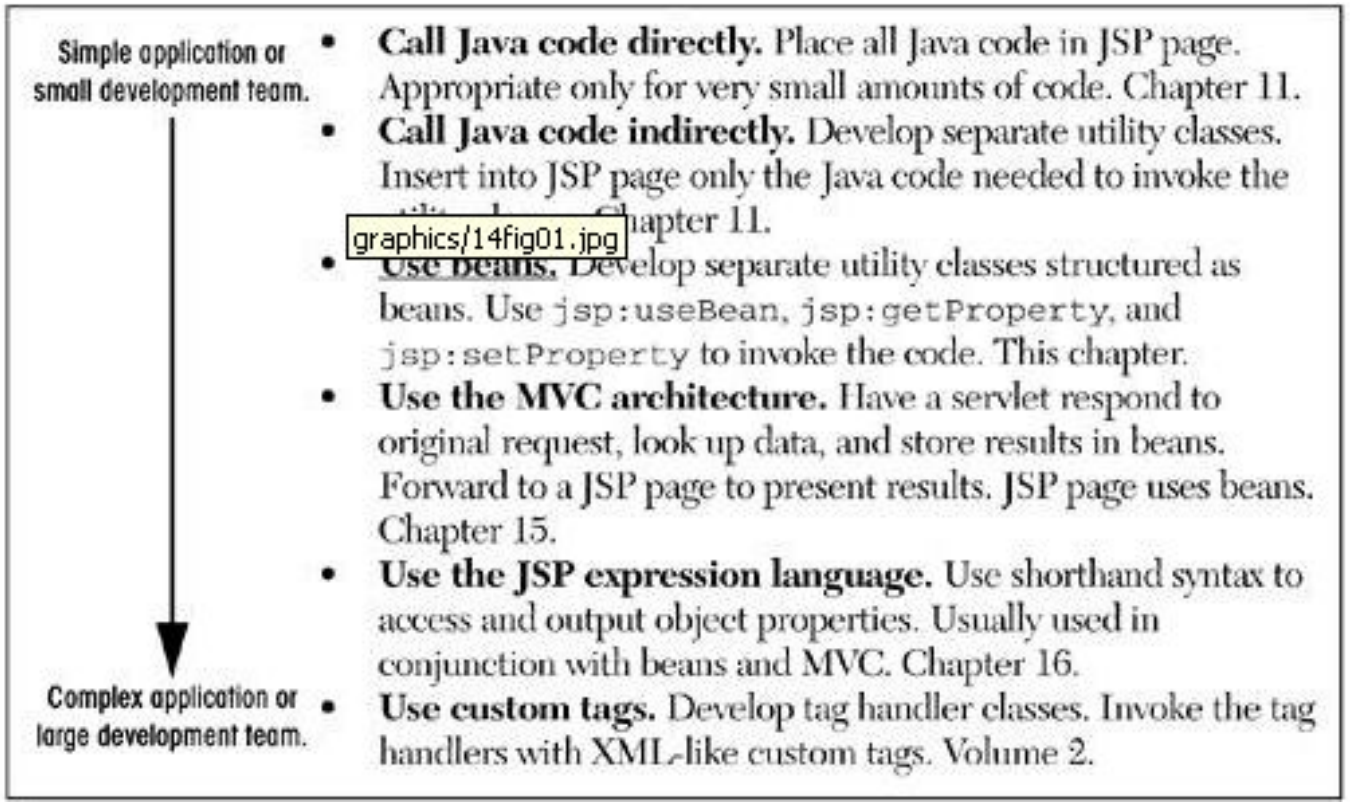
Integrating Servlets & JSP: The MVC Model Architecture

Contents

- Understanding the benefits of MVC
- Using RequestDispatcher to implement MVC
- sendRedirect method of Response & forward method of RequestDispatcher
- Summarizing MVC Code
- JSP/Servlet & JavaBean MVC Examples

Strategies for invoking dynamic code from JSP



Understanding the benefits of MVC

- **Servlets are good at data processing**: reading and checking data, communicating with databases, invoking business logic, and so on.
- **JSP pages are good at presentation**: building HTML to represent the results of requests.

How to combine servlets and JSP pages to best make use of the strengths of each technology?

Understanding the benefits of MVC

The original request is handled by a servlet

- **Model**: The servlet invokes the business-logic and data-access code and creates beans to represent the results. That's the model.
- **View**: The servlet decides which JSP page is appropriate to present those particular results and forwards the request there (the JSP page is the view)
- **Controller**: The servlet decides what business logic code applies and which JSP page should present the results (the servlet is the controller).

Implementing MVC with RequestDispatcher

- **Define beans to represent the data**: Your first step is to define beans to represent the results that will be presented to the user.
- **Use a servlet to handle requests**: In most cases, the servlet reads request parameters.
- **Populate the beans**: The servlet invokes business logic, or data-access code to obtain the results. The results are placed in the beans that were defined in step 1

Implementing MVC with RequestDispatcher

- **Store the bean in the request, session, or servlet context**: The servlet calls setAttribute on the request, session, or servlet context objects to store a reference to the beans that represent the results of the request.
- **Forward the request to a JSP page**: The servlet determines which JSP page is appropriate to the situation and uses the forward method of RequestDispatcher to transfer control to that page.
- **Extract the data from the beans**: The JSP page accesses beans with jsp:useBean and a scope. The page then uses jsp:getProperty to output the bean properties.

sendRedirect & RequestDispatcher

- sendRedirect requires the client to reconnect to the new resource, whereas the forward method of RequestDispatcher is handled completely on the server.
- sendRedirect results in a different final URL, whereas with forward, the URL of the original servlet is maintained.
- sendRedirect does not automatically preserve all of the request data; forward does.

Notes

If your JSP pages only make sense in the context of servlet-generated data, place the pages under the WEB-INF directory. That way, servlets can forward requests to the pages, but clients cannot access them directly.

Applying MVC Example: Random Number

```
package coreservlets;

public class NumberBean {

    private double num = 0;

    public NumberBean(double number) {

        setNumber(number);

    }

    public double getNumber() {

        return(num);

    }

    public void setNumber(double number) {

        num = number;

    }

}
```

Applying MVC Example: Random Number

```
package coreservlets;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
/** Servlet that generates a random number, stores it in a bean, and forwards to JSP
page to display it. */
public class RandomNumberServlet extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException  {
```

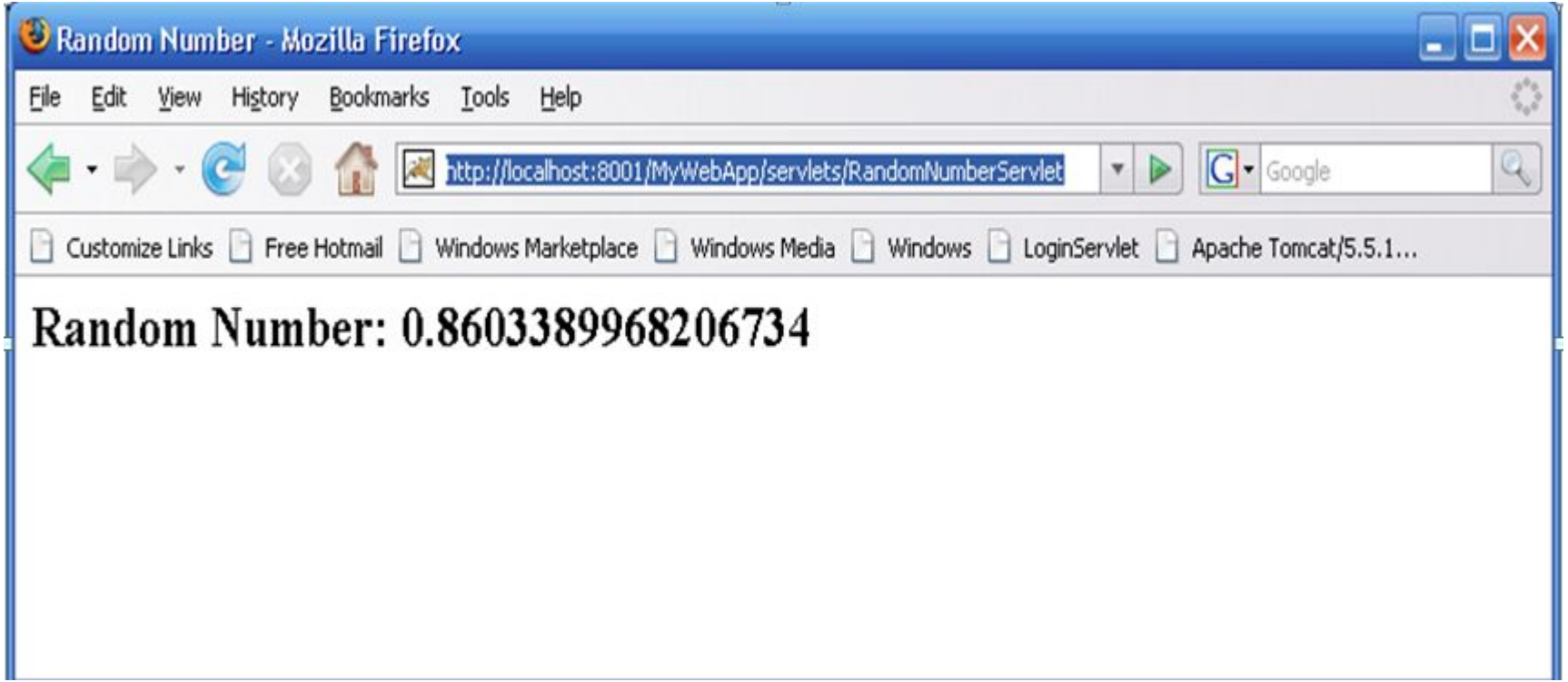
Applying MVC Example: Random Number

```
        NumberBean bean = new NumberBean(Math.random());
        request.setAttribute("randomNum", bean);
        String address = "/WEB-INF/mvc-sharing/RandomNum.jsp";
        RequestDispatcher dispatcher = request.getRequestDispatcher(address);
        dispatcher.forward(request, response);
    }
}
```

Applying MVC Example: Random Number

```
…
<jsp:useBean id="randomNum" type="coreservlets.NumberBean"
             scope="request" />
<H2>Random Number:
<jsp:getProperty name="randomNum" property="number" />
…
```

Applying MVC Example: Random Number



Applying MVC Example: Shopping Cart

Applying MVC Example: Shopping Cart

Select thing to buy

ID	Title	Price	Action
1005	A Teaspoon of Java	55.55	Add To Card
1004	A Cup of Java	44.44	Add To Card
1003	More Java for more dummies	33.33	Add To Card
1001	Java for dummies	11.11	Add To Card
1002	More Java for dummies	22.22	Add To Card

View Shopping Cart

Applying MVC Example: Shopping Cart

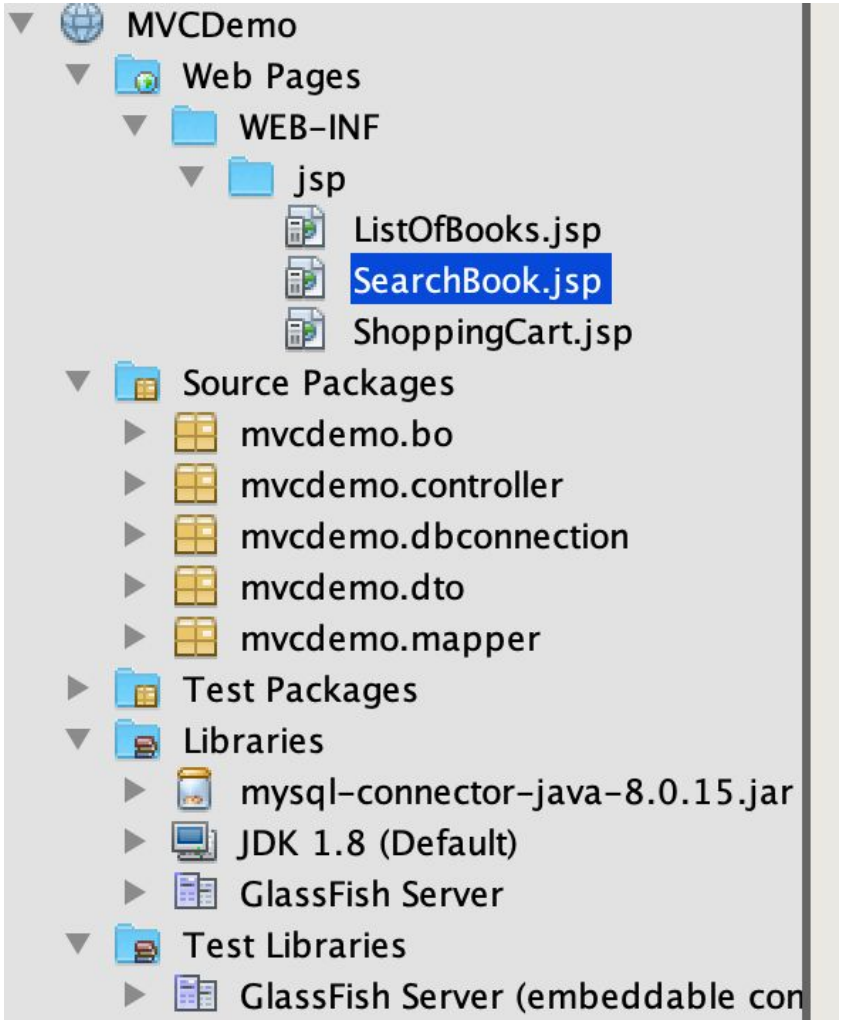
Your Shopping Cart

Book Title	Quantity	Price
A Teaspoon of Java	3	55.55
A Cup of Java	1	44.44

Total: 211.09

confirm

Back to search


MVCDemo
Web Pages
WEB-INF
jsp
ListOfBooks.jsp
SearchBook.jsp
ShoppingCart.jsp
Source Packages
mvcdemo.bo
mvcdemo.controller
mvcdemo.dbconnection
mvcdemo.dto
mvcdemo.mapper
Test Packages
Libraries
mysql-connector-java-8.0.15.jar
JDK 1.8 (Default)
GlassFish Server
Test Libraries
GlassFish Server (embeddable con

Shopping Cart Example - Database

- Login MySQL Server: ***mysql -u myuser -p***
- Setup DB:
- mysql> *create database if not exists* ***ebookshop****;*
- mysql> *use ebookshop;*
- mysql> *drop tables if exists books*
- mysql>
  create table ***books*** *(****id*** *int,* ***title*** *varchar(50),* ***author*** *varchar(50),*
  price *float,* ***qty*** *int, primary key (id));*

Shopping Cart Example - Database

insert into books values (1001, 'Java for dummies', 'Tan Ah Teck', 11.11, 11);

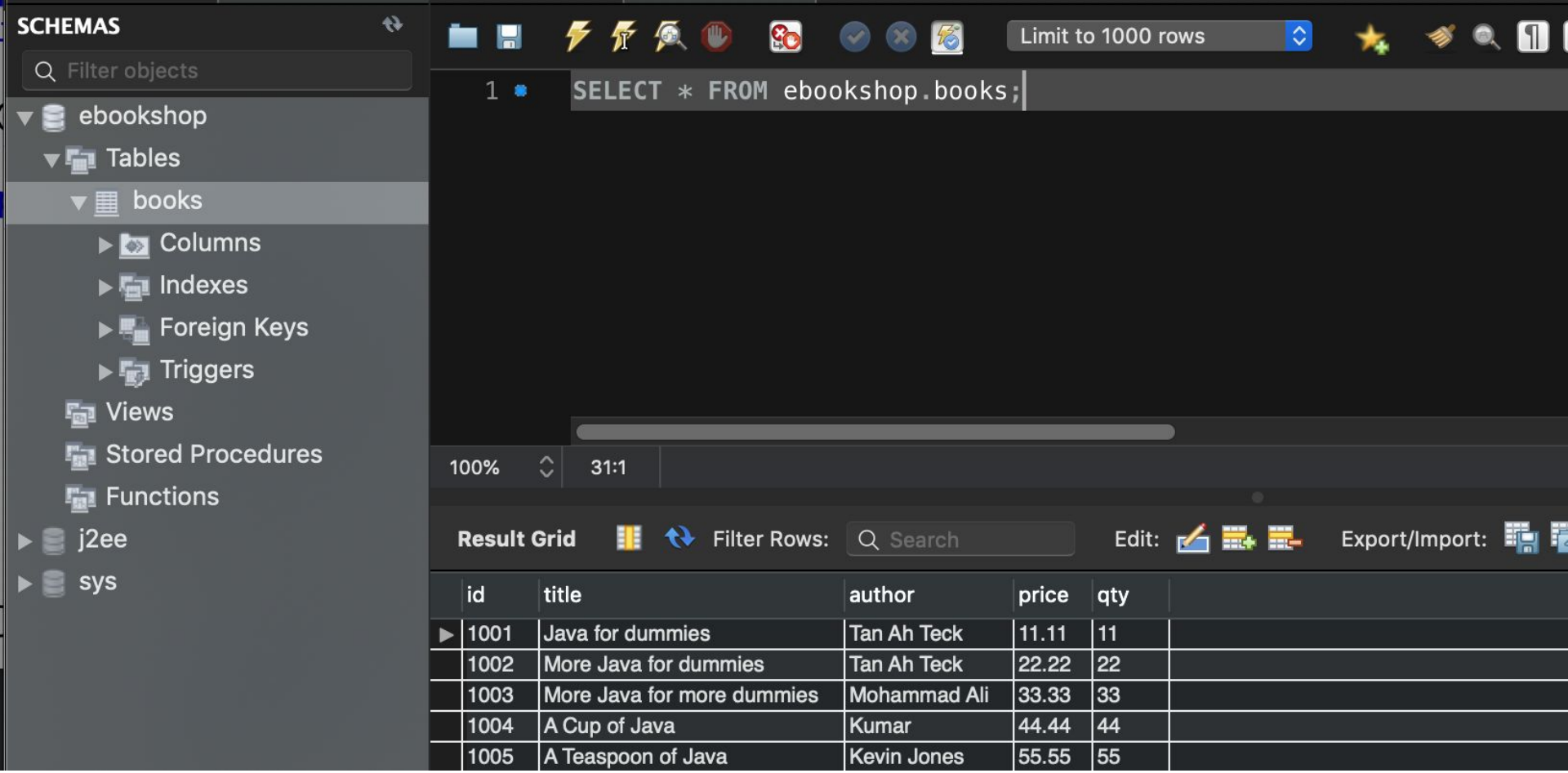
insert into books values (1002, 'More Java for dummies', 'Tan Ah Teck', 22.22, 22);

insert into books values (1003, 'More Java for more dummies', 'Mohammad Ali', 33.33, 33);

insert into books values (1004, 'A Cup of Java', 'Kumar', 44.44, 44);

insert into books values (1005, 'A Teaspoon of Java', 'Kevin Jones', 55.55, 55);

```
mysql> select * from books;
+------+---------------------------+--------------+-------+------+
| id   | title                     | author       | price | qty  |
+------+---------------------------+--------------+-------+------+
| 1001 | Java for dummies          | Tan Ah Teck  | 11.11 |   11 |
| 1002 | More Java for dummies     | Tan Ah Teck  | 22.22 |   22 |
| 1003 | More Java for more dummies| Mohammad Ali | 33.33 |   33 |
| 1004 | A Cup of Java             | Kumar        | 44.44 |   44 |
| 1005 | A Teaspoon of Java        | Kevin Jones  | 55.55 |   55 |
+------+---------------------------+--------------+-------+------+
5 rows in set (0.00 sec)
```


SCHEMAS
Filter objects
ebookshop
Tables
books
Columns
Indexes
Foreign Keys
Triggers
Views
Stored Procedures
Functions
j2ee
sys
Limit to 1000 rows
1
SELECT * FROM ebookshop.books;
100%
31:1
Result Grid
Filter Rows:
Search
Edit:
Export/Import:
id
title
author
price
qty
1001
Java for dummies
Tan Ah Teck
11.11
11
1002
More Java for dummies
Tan Ah Teck
22.22
22
1003
More Java for more dummies
Mohammad Ali
33.33
33
1004
A Cup of Java
Kumar
44.44
44
1005
A Teaspoon of Java
Kevin Jones
55.55
55

SearchBook.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<html>
   <head>
      <title>Shopping Cart Example</title>
      <meta charset="UTF-8">
      <meta name="viewport"
            content="width=device-width, initial-scale=1.0">
   </head>
   <body>
      <h2>Shopping cart example</h2>
      <form method="get" action="./SearchBookServlet">
         Enter an author: <input type="text" name="author" />
         <input type="submit" value="Search" />
      </form>
   </body>
</html>
```

ListOfBooks.jsp

```
<%@page import="mvcdemo.dto.BookDTO"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib uri = "http://java.sun.com/jsp/jstl/core" prefix = "c" %>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>List products</title>
    </head>
    <body><h1>Select thing to buy</h1>
        <table border="1">
            <thead>
                <tr><th>ID</th><th>Title</th><th>Price</th><th>Action</th></tr>
            </thead>
```

```
            <tbody>
                <c:forEach items="${books}" var="book" >
                    <tr>
                        <td>${book.id}</td><td>${book.title}</td>
                        <td>${book.price}</td>
                        <td>
                            <a href="AddToCartServlet?id=${book.id}">
                            <input type="submit" value="Add To Card" /></a>
                        </td>
                    </tr>
                </c:forEach>
            </tbody>
        </table>
        <p><a href="./ViewShoppingCartServlet">View Shopping Cart</a></p>
    </body>
</html>
```

ShoppingCart.jsp

```
<%@page import="java.util.ArrayList"%>
<%@page import="mvcdemo.dto.CartDTO"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head><meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Shopping Cart</title>
    </head>
    <body>
        <h1>Your Shopping Cart</h1>
        <form action="" method="POST">
        <table border="0">
            <thead><tr><th>Book Title</th><th>Quantity</th><th>Price</th></tr></thead>
```

```
<tbody>
<%
    float totalOrder = 0;
    HttpSession currentSession = request.getSession();
    if (currentSession.getAttribute("shoppingCart") != null) {
        ArrayList<CartDTO> arrC = (ArrayList<CartDTO>)
        currentSession.getAttribute("shoppingCart");
        for (int i = 0; i < arrC.size(); i++) {
            totalOrder += (float) (arrC.get(i).getOrderQuantity() * arrC.get(i).getPrice());
%>
        <tr>
            <td><%= arrC.get(i).getTitle()%></td>
            <td><%= arrC.get(i).getOrderQuantity()%></td>
            <td><%= arrC.get(i).getPrice()%></td></tr>

<%          }
    } else {
%>
```

```
            <h1>YOU DONT BUY ANYTHING</h1>
 <%
     }
 %>
                    </tbody>
            </table>

            <p>Total: <%= totalOrder%></p>
            <p><input type="button" value ="confirm" onclick=""/></p>
            </form>
            <p><a href="./BackToSearchServlet">Back to search</a></p>
    </body>
</html>
```

```
public class DBConnectionService {


    protected static void loadJDBCDriver() throws Exception {...7 lines }

    public static Connection getConnection() throws Exception {
        Connection connect = null;
        if (connect == null) {
            loadJDBCDriver();
            try {
                connect = DriverManager.getConnection(
                "jdbc:mysql://localhost:3306/ebookshop?"
                        + "allowPublicKeyRetrieval=true&useSSL=false",
                    "root", "");
            } catch (java.sql.SQLException e) {
                throw new Exception("Can not access to Database Server ..."
                        + e.getMessage());
            }
        }
        return connect;
    }
}
```

```java
public class DBMapper {
    private Connection connection;
    public DBMapper() throws Exception {
        try {
            connection = DBConnectionService.getConnection();
        } catch (Exception e) {
            System.out.println("Failed in constructor method in MapperDB:" + e);
            e.printStackTrace();
            throw e;
        }
    }

    public DBMapper(Connection con) {...3 lines }

    public void closeConnection() throws Exception {...8 lines }

    public Connection getConnection() {...3 lines }

    public void setConnection(Connection connection) {...3 lines }
}
```

```java
package mvcdemo.mapper;
import ...5 lines

public class BookMapper extends DBMapper {
    public BookMapper() throws Exception {
        super();
    }

    public ArrayList<BookDTO> searchBook(String authorName) {...26 lines }

    public BookDTO getBook(int id) {...24 lines }

}
```

```java
public ArrayList<BookDTO> searchBook(String authorName) {
    ArrayList<BookDTO> books = new ArrayList<>();
    try {
        Statement stmt = getConnection().createStatement();
        String sqlStr = "SELECT * FROM books WHERE author LIKE "
                + "'%" + authorName + "%'"
                + " AND qty > 0 ORDER BY author ASC, title ASC";
        ResultSet rs = stmt.executeQuery(sqlStr);
        // Process the query result
        int count = 0;
        while (rs != null && rs.next()) {
            BookDTO book = new BookDTO();
            book.setId(rs.getInt("id"));
            book.setTitle(rs.getString("title"));
            book.setAuthor(rs.getString("author"));
            book.setPrice(rs.getFloat("price"));
            book.setQty(rs.getInt("qty"));
            books.add(book);
        }
    } catch (Exception ex) {
        ex.printStackTrace();
    }
    return books;
}
```

```java
package mvcdemo.dto;
public class BookDTO {
    int id;
    String title;
    String author;
    float price;
    int qty;

    public int getId() {...3 lines }

    public void setId(int id) {...3 lines }
```

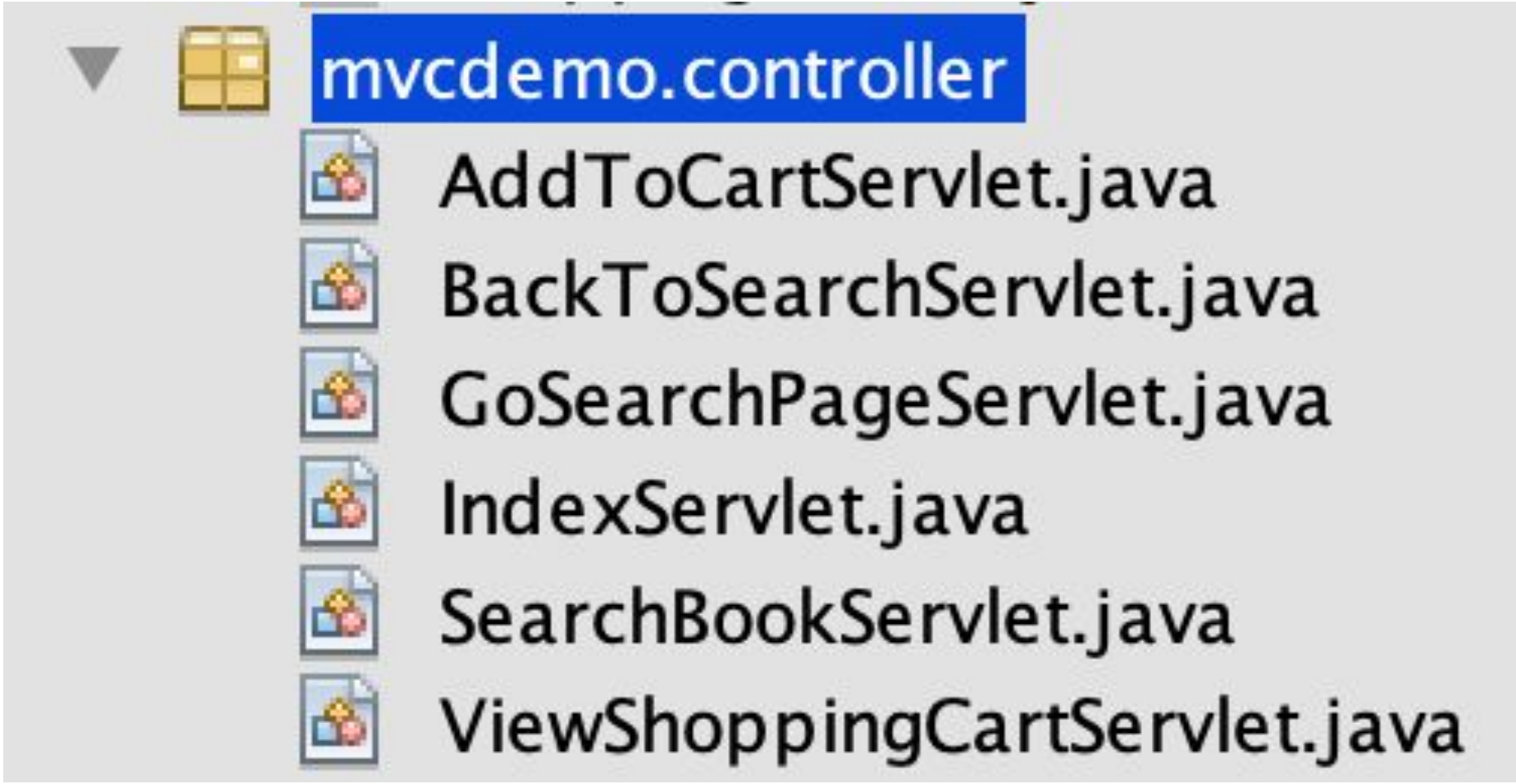
```java
package mvcdemo.dto;

public class CartDTO {
    private int id;
    private String title;
    private String author;
    private float price;
    private int orderQuantity;

    public int getId() {...3 lines }

    public void setId(int id) {...3 lines }

    public String getTitle() {...3 lines }

    public void setTitle(String title) {
        this.title = title;
    }
```

```java
package mvcdemo.bo;
import ...5 lines

public class BookBO {
    public ArrayList<BookDTO> searchBook(String authorName) {
        ArrayList<BookDTO> books = null;
        BookMapper mapper = null;
        try {
            mapper = new BookMapper();
            books = mapper.searchBook(authorName);
        } catch (Exception ex) {
            Logger.getLogger(BookBO.class.getName()).log(Level.SEVERE, null, e
        }
        finally {
            try {
                mapper.closeConnection();
            } catch (Exception ex) {
                Logger.getLogger(BookBO.class.getName()).log(Level.SEVERE, nu
            }
        }
        return books;
    }
    public BookDTO getBook(int id) {...20 lines }
}
```

```
package mvcdemo.bo;

import ...6 lines

public class ShoppingCartBO {

    public ArrayList<CartDTO> updateShoppingCart
        (ArrayList<CartDTO> shoppingCart, int currentBookID) {...56 lines }
}
```

```
public ArrayList<CartDTO> updateShoppingCart
    (ArrayList<CartDTO> shoppingCart, int currentBookID) {
    BookMapper bookMapper = null;
    ArrayList<BookDTO> arrBooks = new ArrayList<BookDTO>();
    try {
        bookMapper = new BookMapper();
        BookDTO book = bookMapper.getBook(currentBookID);
        //store product infomantion to session
        if (shoppingCart == null) {
            shoppingCart = new ArrayList<CartDTO>();
            //if not exist session cart, add new book to cart
            CartDTO cartDTO = new CartDTO();
            cartDTO.setId(book.getId());
            cartDTO.setTitle(book.getTitle());
            cartDTO.setAuthor(book.getAuthor());
            cartDTO.setPrice(book.getPrice());
            cartDTO.setOrderQuantity(1);
            shoppingCart.add(cartDTO);
        } else {
            //if ID is exist, increase quantity
            boolean checkID = false;
            for (int i = 0; i < shoppingCart.size(); i++) {
                CartDTO cart = shoppingCart.get(i);
                if (cart.getId() == currentBookID) {
                    cart.setOrderQuantity(cart.getOrderQuantity() + 1);
                    checkID = true;
                    break;
                }
            }
```

```
            //if ID isn't exist
            if (checkID == false) {
                CartDTO cart = new CartDTO();
                cart.setId(book.getId());
                cart.setTitle(book.getTitle());
                cart.setAuthor(book.getAuthor());
                cart.setPrice(book.getPrice());
                cart.setOrderQuantity(1);
                shoppingCart.add(cart);
            }
        }
    } catch (Exception ex) {
        ex.printStackTrace();
    }
    finally {
        if (bookMapper != null) try {
            bookMapper.closeConnection();
        } catch (Exception ex) {
            Logger.getLogger(ShoppingCartBO.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    return shoppingCart;
}
```


mvcdemo.controller
AddToCartServlet.java
BackToSearchServlet.java
GoSearchPageServlet.java
IndexServlet.java
SearchBookServlet.java
ViewShoppingCartServlet.java

```java
package mvcdemo.controller;

import ...6 lines

@WebServlet(name = "IndexServlet", urlPatterns = {""})
public class IndexServlet extends HttpServlet {

    @Override
    protected void service(HttpServletRequest req, HttpServletResponse resp)
            throws ServletException, IOException {
        resp.sendRedirect("./GoSearchPageServlet");
    }
}
```

```java
package mvcdemo.controller;

import ...7 lines

@WebServlet(name = "GoSearchPageServlet", urlPatterns = {"/GoSearchPageServlet"})
public class GoSearchPageServlet extends HttpServlet {

    @Override
    protected void service(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        RequestDispatcher dispatcher
                = request.getRequestDispatcher
        ("./WEB-INF/jsp/SearchBook.jsp");
        dispatcher.forward(request, response);
    }

}
```

```java
package mvcdemo.controller;
import ...8 lines

@WebServlet(name = "BackToSearchServlet",
        urlPatterns = {"/BackToSearchServlet"})
public class BackToSearchServlet extends HttpServlet {
    @Override
    protected void service(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {
        RequestDispatcher dispatcher
                = request.getRequestDispatcher
        ("./WEB-INF/jsp/SearchBook.jsp");
        dispatcher.forward(request, response);
    }
}
```

```java
package mvcdemo.controller;

import ...11 lines

@WebServlet(urlPatterns = {"/SearchBookServlet"})
public class SearchBookServlet extends HttpServlet {
    @Override
    protected void service(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        BookBO bookBO = new BookBO();
        ArrayList<BookDTO> books =
                bookBO.searchBook(request.getParameter("author"));
        request.setAttribute("books", books);
        RequestDispatcher dispatcher
                = request.getRequestDispatcher
        ("./WEB-INF/jsp/ListOfBooks.jsp");
        dispatcher.forward(request, response);
    }
}
```

```java
package mvcdemo.controller;
import ...21 lines

@WebServlet(name = "AddToCartServlet", urlPatterns = {"/AddToCartServlet"})
public class AddToCartServlet extends HttpServlet {
    @Override
    public void service(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        int currentBookID = new Integer(request.getParameter("id"));
        HttpSession session = request.getSession();
        ArrayList<CartDTO> shoppingCart = (ArrayList)
                session.getAttribute("shoppingCart");
        ShoppingCartBO cartBO = new ShoppingCartBO();
        shoppingCart = cartBO.updateShoppingCart(shoppingCart, currentBookID);
        //set session
        session.setAttribute("shoppingCart", shoppingCart);
        RequestDispatcher dispatcher = request.getRequestDispatcher
        ("./WEB-INF/jsp/ShoppingCart.jsp");
        dispatcher.forward(request, response);
    }

}
```

```java
package mvcdemo.controller;

import ...7 lines

@WebServlet(name = "ViewShoppingCartServlet",
        urlPatterns = {"/ViewShoppingCartServlet"})
public class ViewShoppingCartServlet extends HttpServlet {

    @Override
    protected void service(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {
        RequestDispatcher dispatcher
                = request.getRequestDispatcher
        ("./WEB-INF/jsp/ShoppingCart.jsp");
        dispatcher.forward(request, response);
    }
}
```

Summarizing MVC code

- Summarize the code that would be used for request-based, session-based, and application-based MVC approaches.

Summarizing MVC Code

Request-Based Data Sharing

- The servlet stores the beans in the **HttpServletRequest**
- **Servlet**

```
ValueObject value = new ValueObject(...); request.setAttribute("key", value);
RequestDispatcher dispatcher =
request.getRequestDispatcher("/WEB-INF/SomePage.jsp");
dispatcher.forward(request, response);
```

- **JSP Page**

```
<jsp:useBean id="key" type="somePackage.ValueObject" scope="request" />
<jsp:getProperty name="key" property="someProperty" />
```

Summarizing MVC Code

Session-Based Data Sharing

- The servlet stores the beans in the **HttpSession**, where they are accessible to the same client in the destination JSP page or in other pages.
- **Servlet**

```
ValueObject value = new ValueObject(...);
HttpSession session = request.getSession(); session.setAttribute("key", value);
RequestDispatcher dispatcher =
request.getRequestDispatcher("/WEB-INF/SomePage.jsp");
dispatcher.forward(request, response);
```

- **JSP Page**

```
<jsp:useBean id="key" type="somePackage.ValueObject" scope="session" />
<jsp:getProperty name="key" property="someProperty" />
```

Summarizing MVC Code

Application-Based Data Sharing

- The servlet stores the beans in the **ServletContext**, where they are accessible to any servlet or JSP page in the Web application. To guarantee that the JSP page extracts the same data that the servlet inserted, you should synchronize your code as below.

Summarizing MVC Code

Application-Based Data Sharing

Servlet

```
synchronized(this) {
        ValueObject value = new ValueObject(...);
        getServletContext().setAttribute("key", value);
        RequestDispatcher dispatcher =
        request.getRequestDispatcher("/WEB-INF/SomePage.jsp");
        dispatcher.forward(request, response);
 }
```

JSP Page

```
 <jsp:useBean id="key" type="somePackage.ValueObject" scope="application" />
 <jsp:getProperty name="key" property="someProperty" />
```

Applying MVC Example: Bank Account Balances

```
package coreservlets;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
/** Servlet that reads a customer ID and displays
 *  information on the account balance of the customer who has that ID.  */
public class ShowBalance extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException  {
```

Applying MVC Example: Bank Account Balances

```
BankCustomer customer =
BankCustomer.getCustomer(request.getParameter("id"));
String address;
if (customer == null) {
        address = "/WEB-INF/bank-account/UnknownCustomer.jsp";
}
else if (customer.getBalance() < 0)   {
        address = "/WEB-INF/bank-account/NegativeBalance.jsp";
        request.setAttribute("badCustomer", customer);
}
```

Applying MVC Example: Bank Account Balances

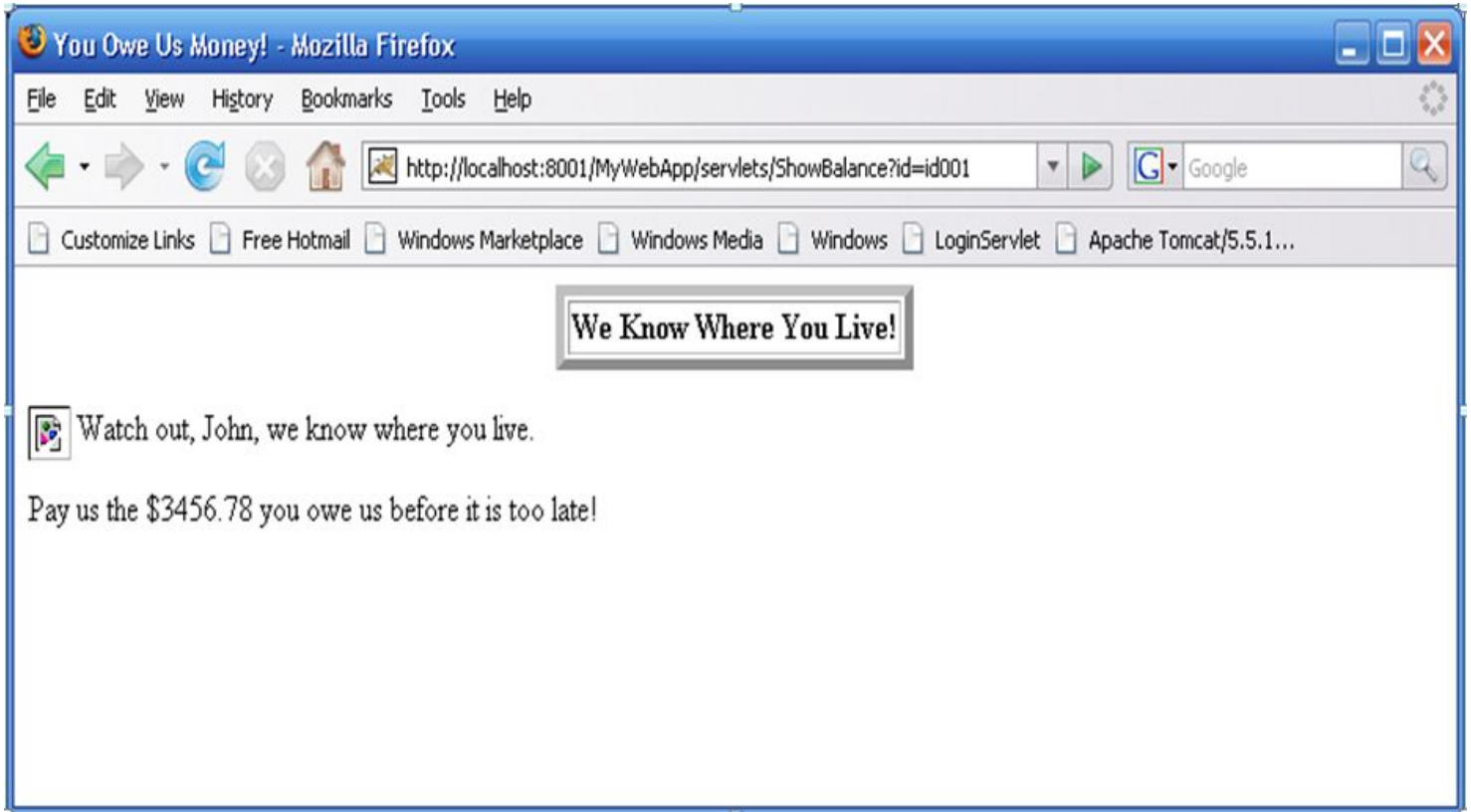
```
        else if (customer.getBalance() < 10000) {
                address = "/WEB-INF/bank-account/NormalBalance.jsp";
                request.setAttribute("regularCustomer", customer);
        } else {
                address = "/WEB-INF/bank-account/HighBalance.jsp";
                request.setAttribute("eliteCustomer", customer);
        }

        RequestDispatcher dispatcher = request.getRequestDispatcher(address);
        dispatcher.forward(request, response);
    }
}
```

Applying MVC Example: Bank Account Balances

NegativeBalance.jsp

```
…
<jsp:useBean id="badCustomer" type="coreservlets.BankCustomer" scope="request" />
Watch out,
<jsp:getProperty name="badCustomer" property="firstName" />,
we know where you live.
<P>
Pay us the
$<jsp:getProperty name="badCustomer" property="balanceNoSign" />
you owe us before it is too late!
…
```

Applying MVC Example: Bank Account Balances

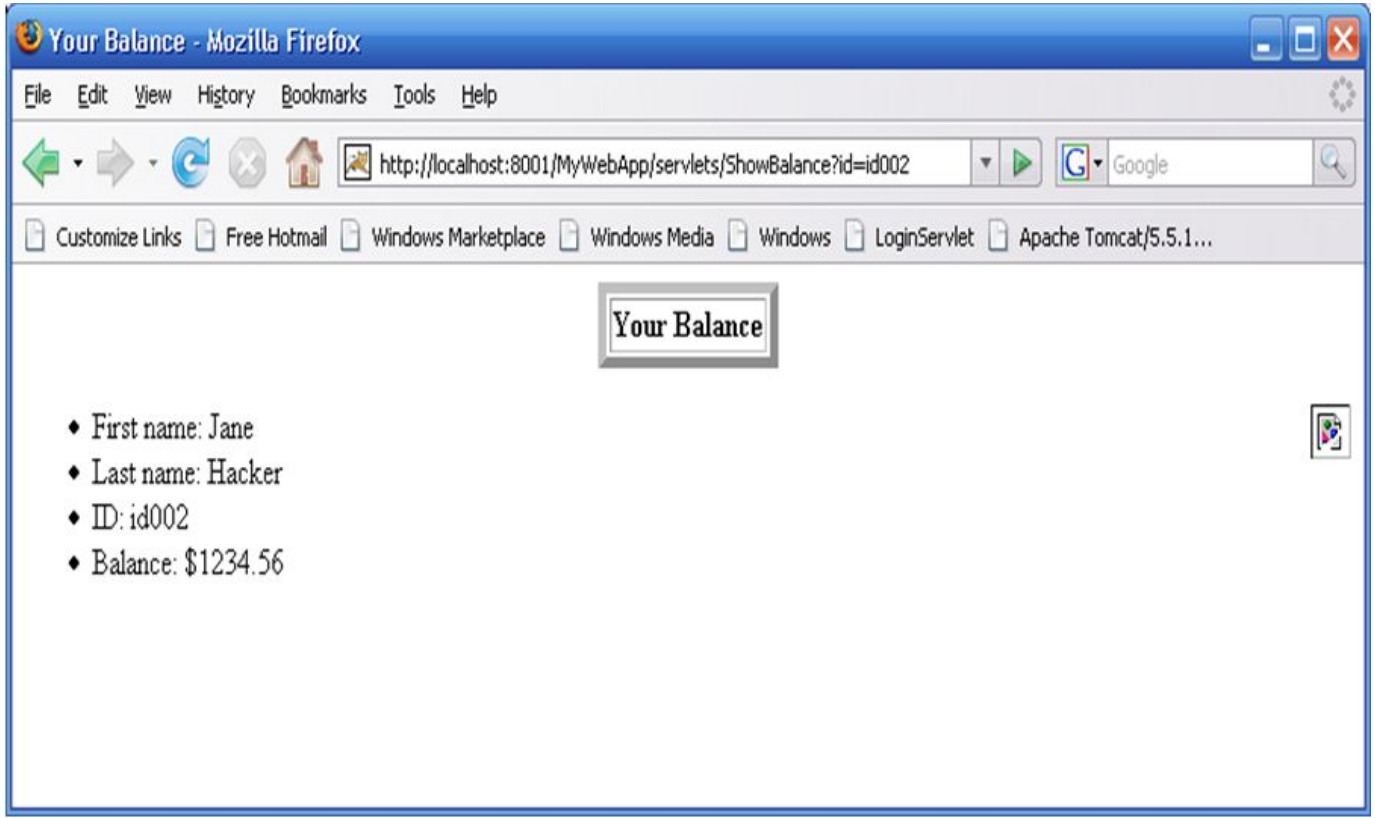


Applying MVC Example: Bank Account Balances

NormalBalance.jsp

```
…
<jsp:useBean id="regularCustomer“ type="coreservlets.BankCustomer"
scope="request" />
<UL>
<LI>First name: <jsp:getProperty name="regularCustomer" property="firstName" />
  <LI>Last name: <jsp:getProperty name="regularCustomer" property="lastName" />
  <LI>ID: <jsp:getProperty name="regularCustomer"property="id" />
  <LI>Balance: $<jsp:getProperty name="regularCustomer"property="balance" />
</UL>
…
```

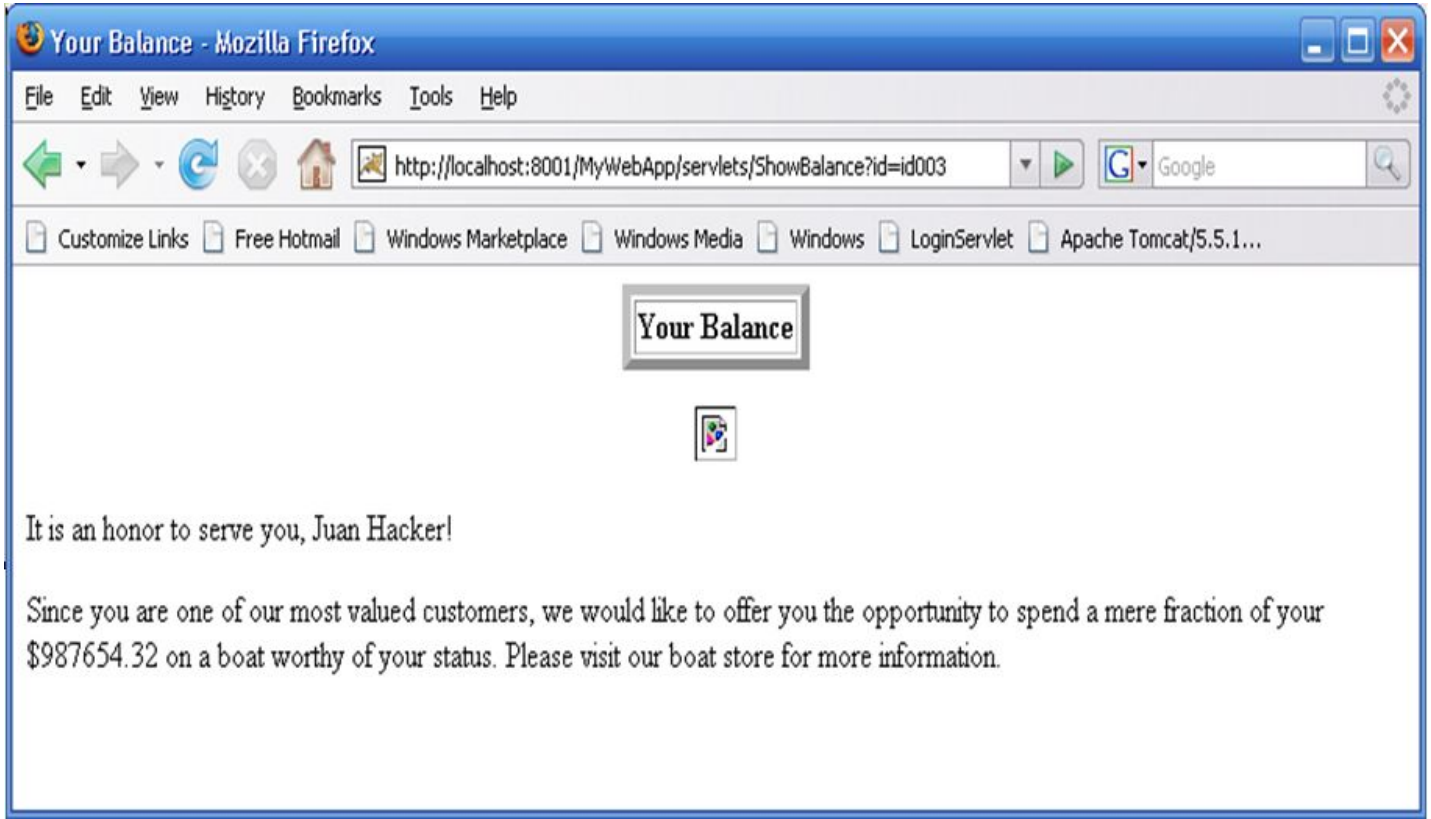
Applying MVC Example: Bank Account Balances



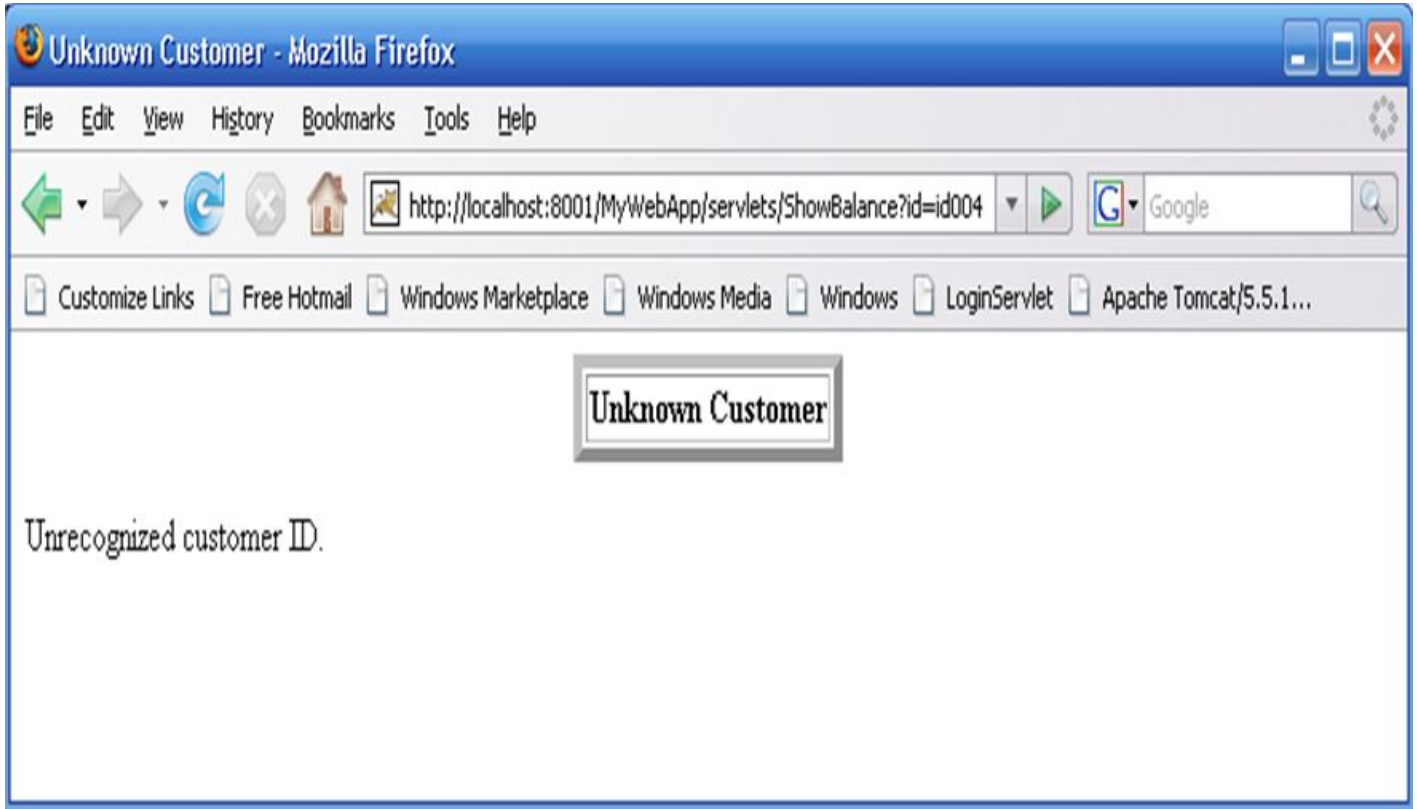
Applying MVC Example: Bank Account Balances

```
<jsp:useBean id="eliteCustomer" type="coreservlets.BankCustomer"
scope="request" />
It is an honor to serve you,
<jsp:getProperty name="eliteCustomer" property="firstName" />
<jsp:getProperty name="eliteCustomer" property="lastName" />!
<P>
Since you are one of our most valued customers, we would like to offer you the
opportunity to spend a mere fraction of your $
<jsp:getProperty name="eliteCustomer"  property="balance" />
on a boat worthy of your status. Please visit our boat store for more information.
```

Applying MVC Example: Bank Account Balances



Applying MVC Example: Bank Account Balances



Applying MVC Example: Show Name (Session-Based Sharing)

- Our goal is to display users' first and last names. If the users fail to tell us their name, we want to use whatever name they gave us previously. If the users do not explicitly specify a name and no previous name is found, a warning should be displayed. Data is stored for each client, so session-based sharing is appropriate.

Applying MVC Example: Show Name (Session-Based Sharing)

```
package coreservlets;
public class NameBean {
    private String firstName = "Missing first name";
    private String lastName = "Missing last name";
    public NameBean() {}
    public NameBean(String firstName, String lastName) {
        setFirstName(firstName);
        setLastName(lastName);
    }
    public String getFirstName() {
        return(firstName);
    }
```

Applying MVC Example: Show Name (Session-Based Sharing)

```
    public void setFirstName(String newFirstName) {
        firstName = newFirstName;
    }
    public String getLastName() {
        return(lastName);
    }
    public void setLastName(String newLastName) {
        lastName = newLastName;
    }
}
```

Applying MVC Example: Show Name (Session-Based Sharing)

```
package coreservlets;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
/** Reads firstName and lastName request parameters and forwards to JSP page to
display them. Uses session-based bean sharing to remember previous values. */
public class RegistrationServlet extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
  HttpSession session = request.getSession();
  NameBean nameBean = (NameBean)session.getAttribute("nameBean");
```

Applying MVC Example: Show Name (Session-Based Sharing)

```
 if (nameBean == null) {
        nameBean = new NameBean();
        session.setAttribute("nameBean", nameBean);
}

 String firstName = request.getParameter("firstName");
if ((firstName != null) && (!firstName.trim().equals(""))) {
   nameBean.setFirstName(firstName);
}
```

Applying MVC Example: Show Name (Session-Based Sharing)

```
        String lastName = request.getParameter("lastName");
        if ((lastName != null) && (!lastName.trim().equals(""))) {
            nameBean.setLastName(lastName);
        }
        String address = "/WEB-INF/mvc-sharing/ShowName.jsp";
        RequestDispatcher dispatcher = request.getRequestDispatcher(address);
        dispatcher.forward(request, response);
    }
}
```

Applying MVC Example: Show Name (Session-Based Sharing)

ShowName.jsp

```
…
<jsp:useBean id="nameBean" type="coreservlets.NameBean"
             scope="session" />
<H2>First Name:
<jsp:getProperty name="nameBean" property="firstName" /></H2>
<H2>Last Name:
<jsp:getProperty name="nameBean" property="lastName" /></H2>
…
```

HOMEWORK

Example: JSPs, Servlets, JavaBeans?

- We need to implement one module of the specified application with functions as following:
  - System Administrator can
    - Log in the system
    - Create a new account for a new user
    - Update user's information
    - Delete the specified user
    - View the list of users, search users
    - Logout the system
  - User
    - Can log in the system
    - Update his/her information
    - Logout the system

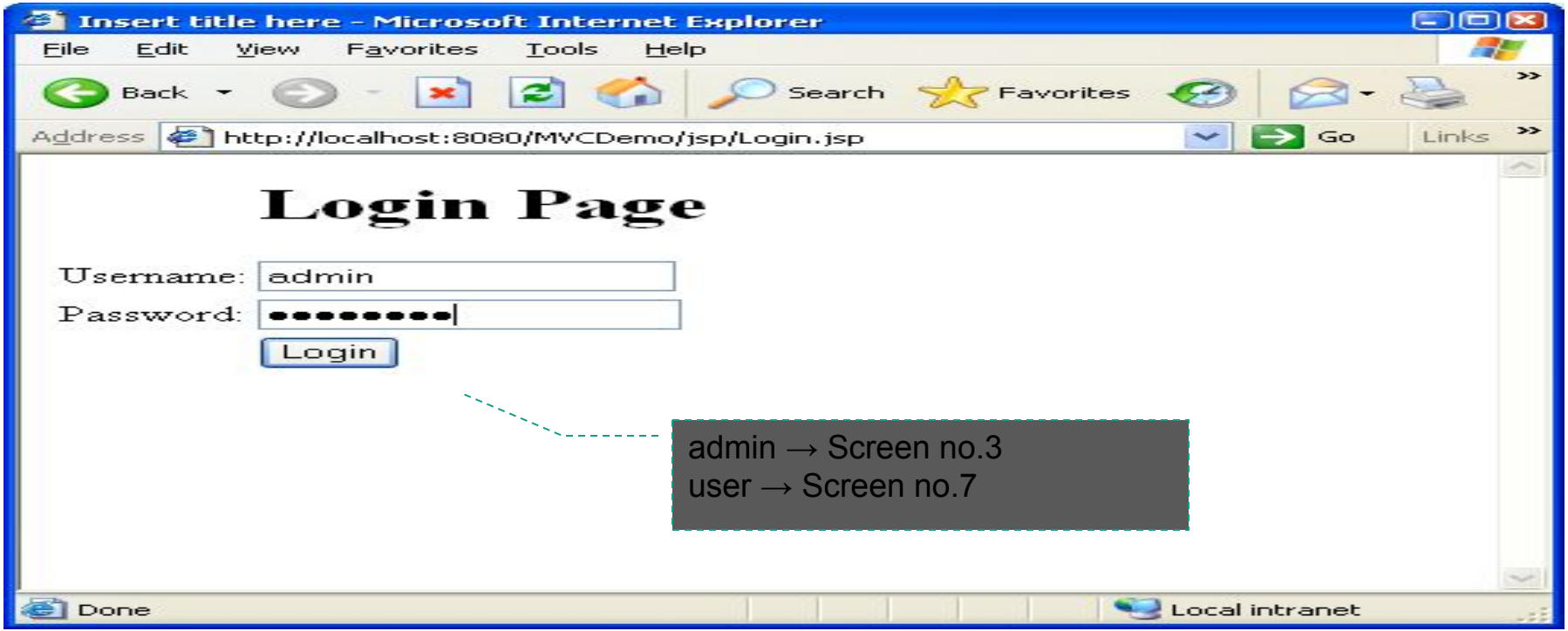
Example - Requirements

- Listing Pages/Screens for this module and the detailed information of these page. List all JSPs, Servlets, JavaBeans that we need to implement the above functions.

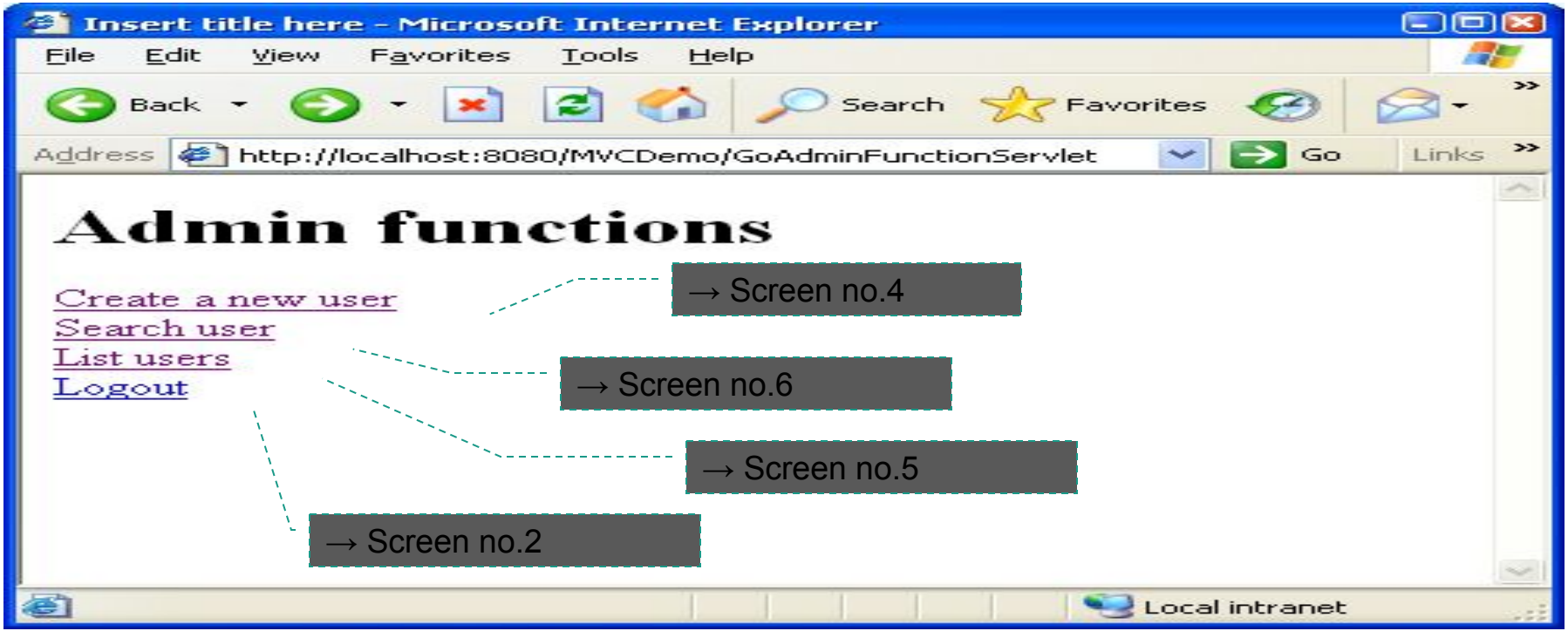
Example - Requirements

- JSPs?
- Servlets?
- JavaBean?
- How will you organize your source code?

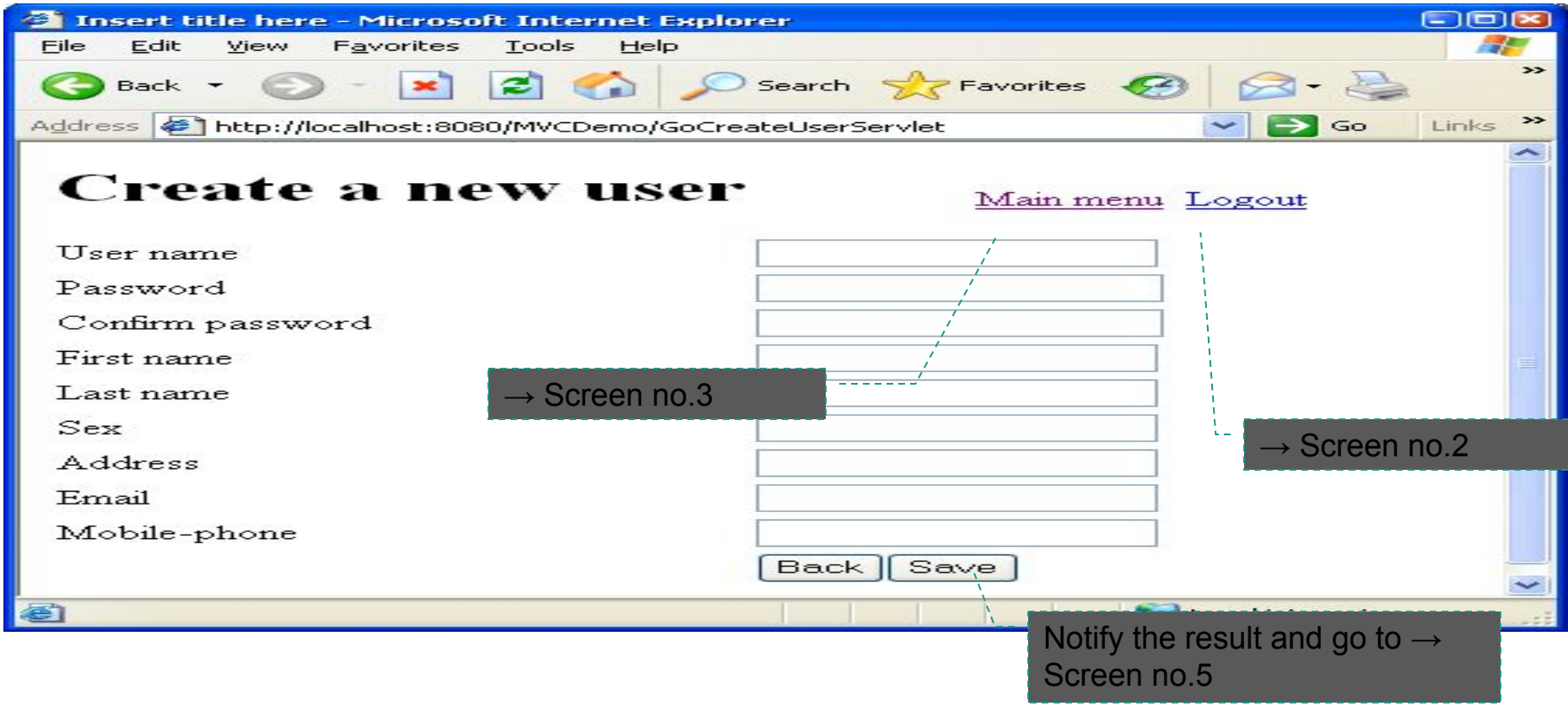
Example – Screen no.2



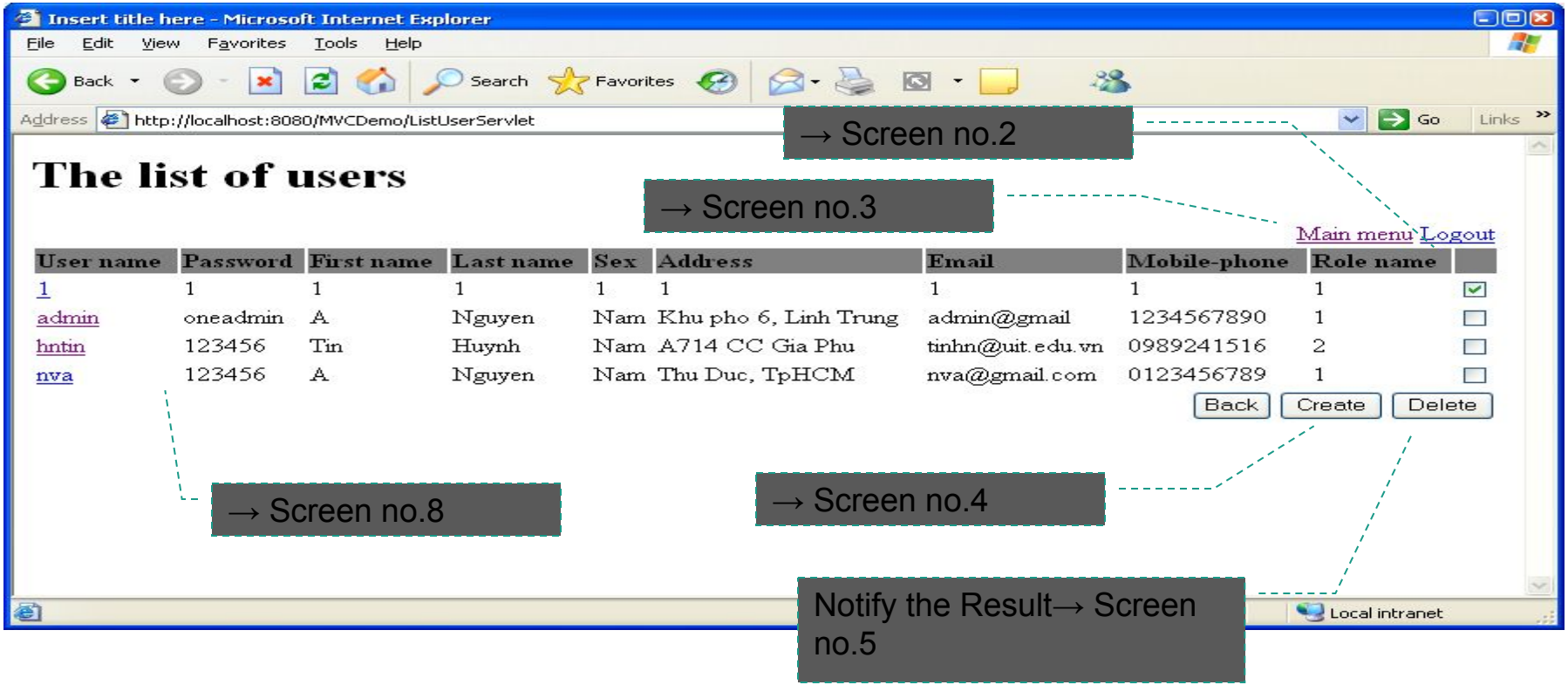
Example – Screen no.3



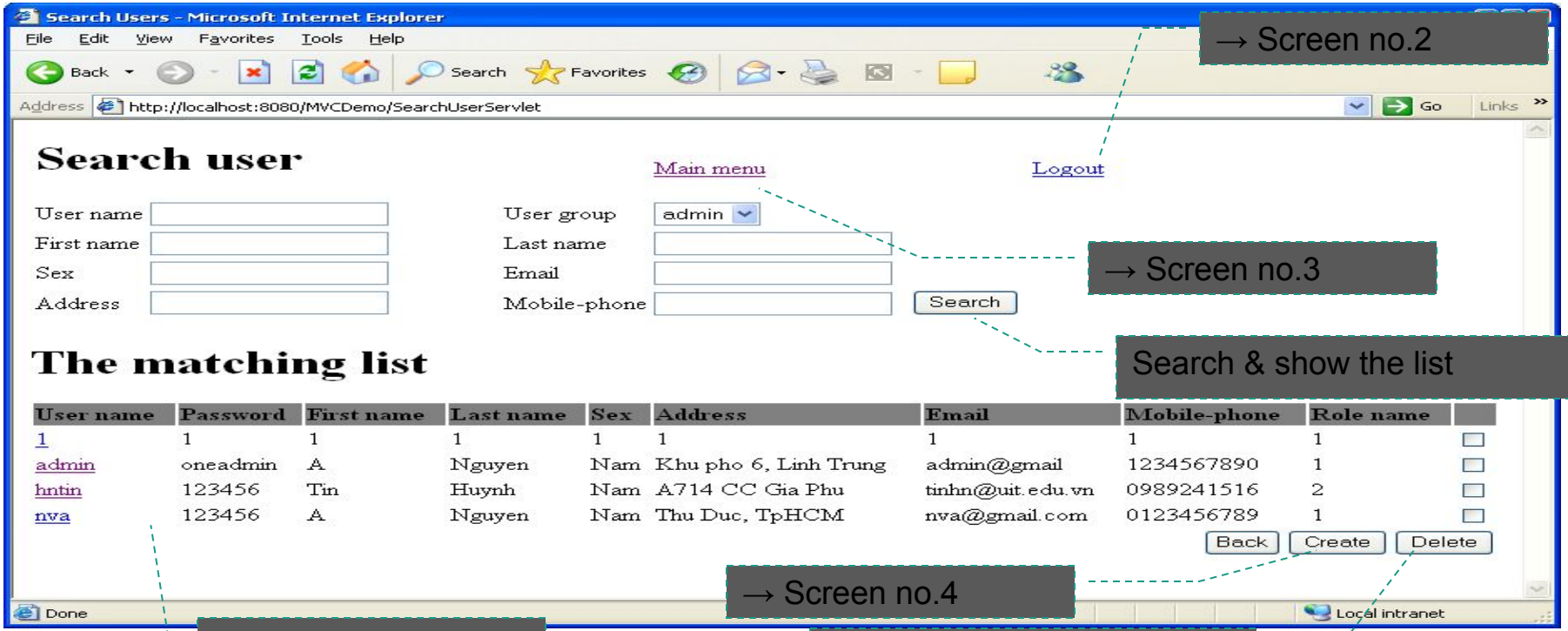
Example – Screen no.4



Example – Screen no.5

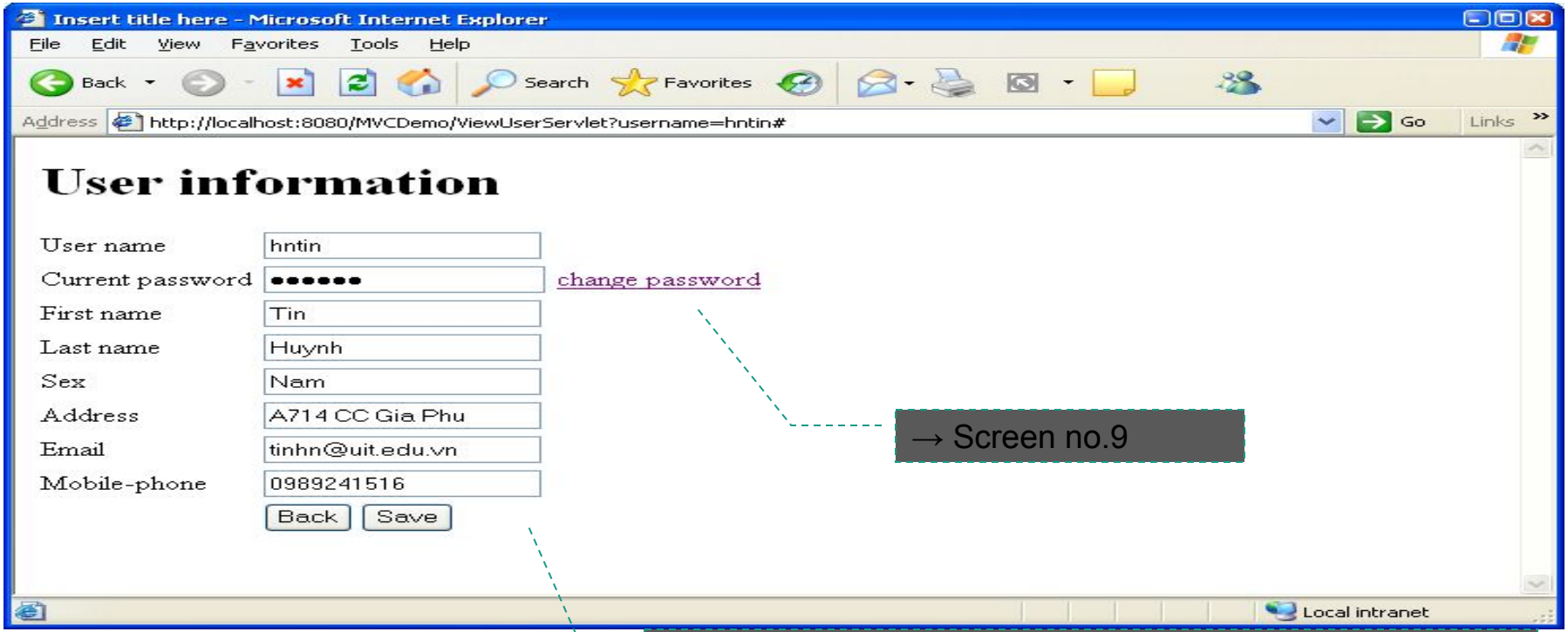


Example – Screen no.6

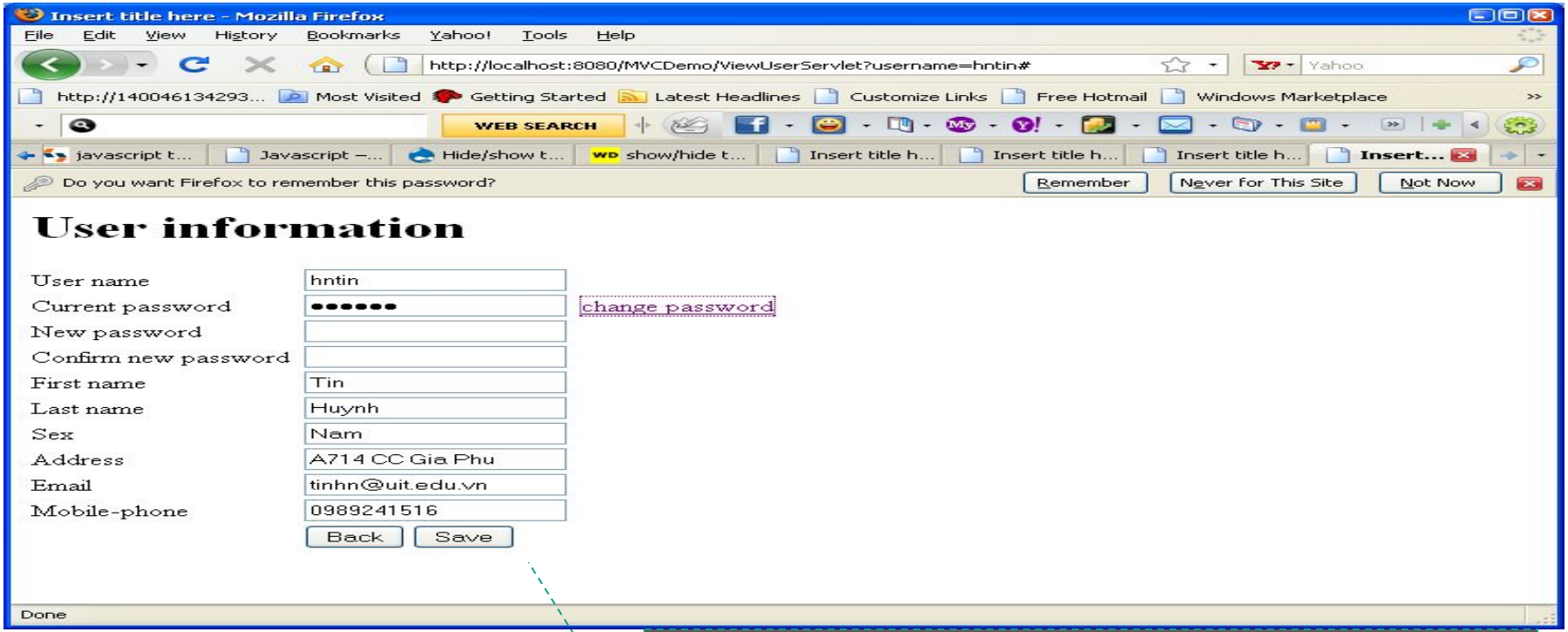


Example – Screen no.7

Example – Screens no.8

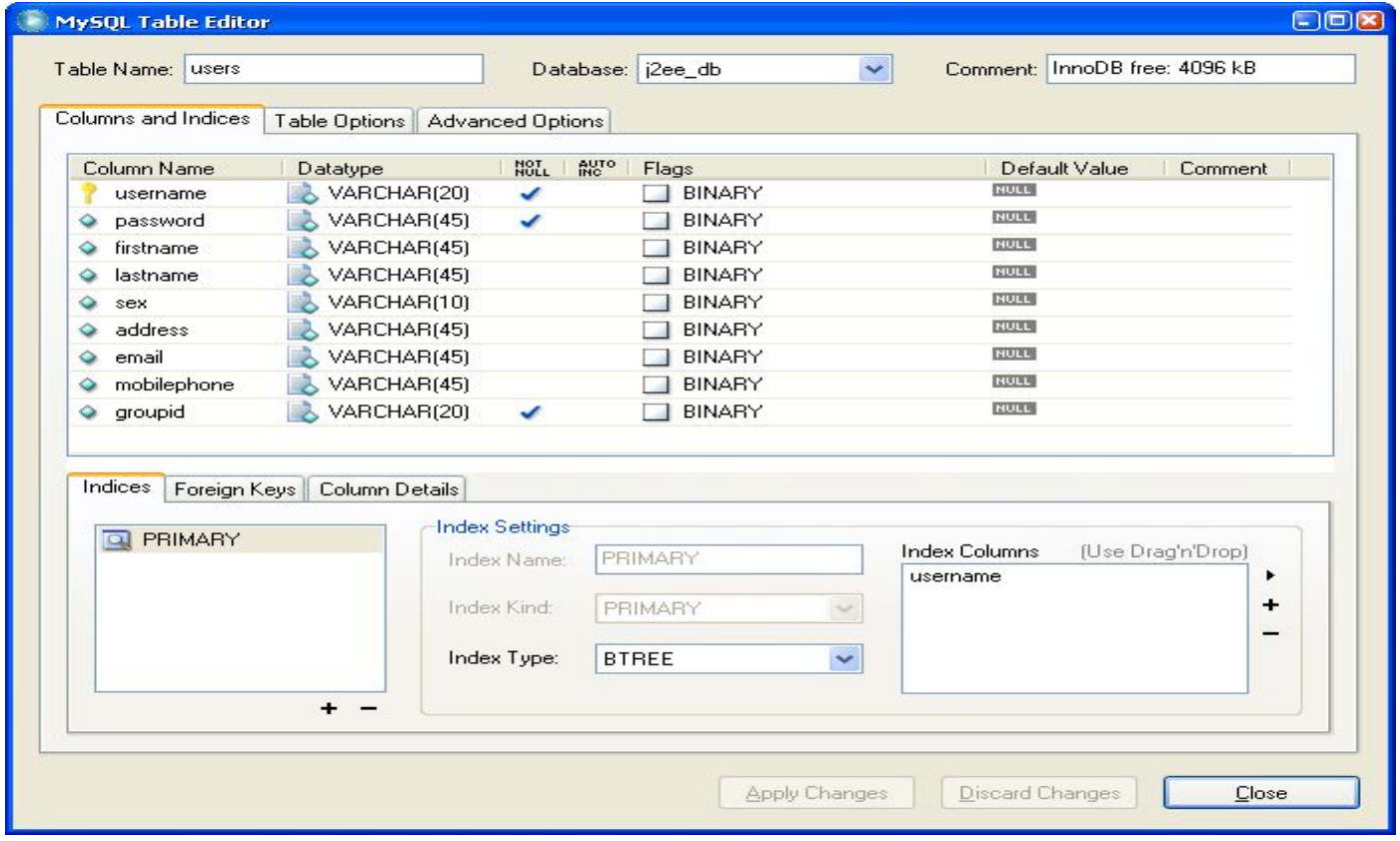


Example – Screens no.9

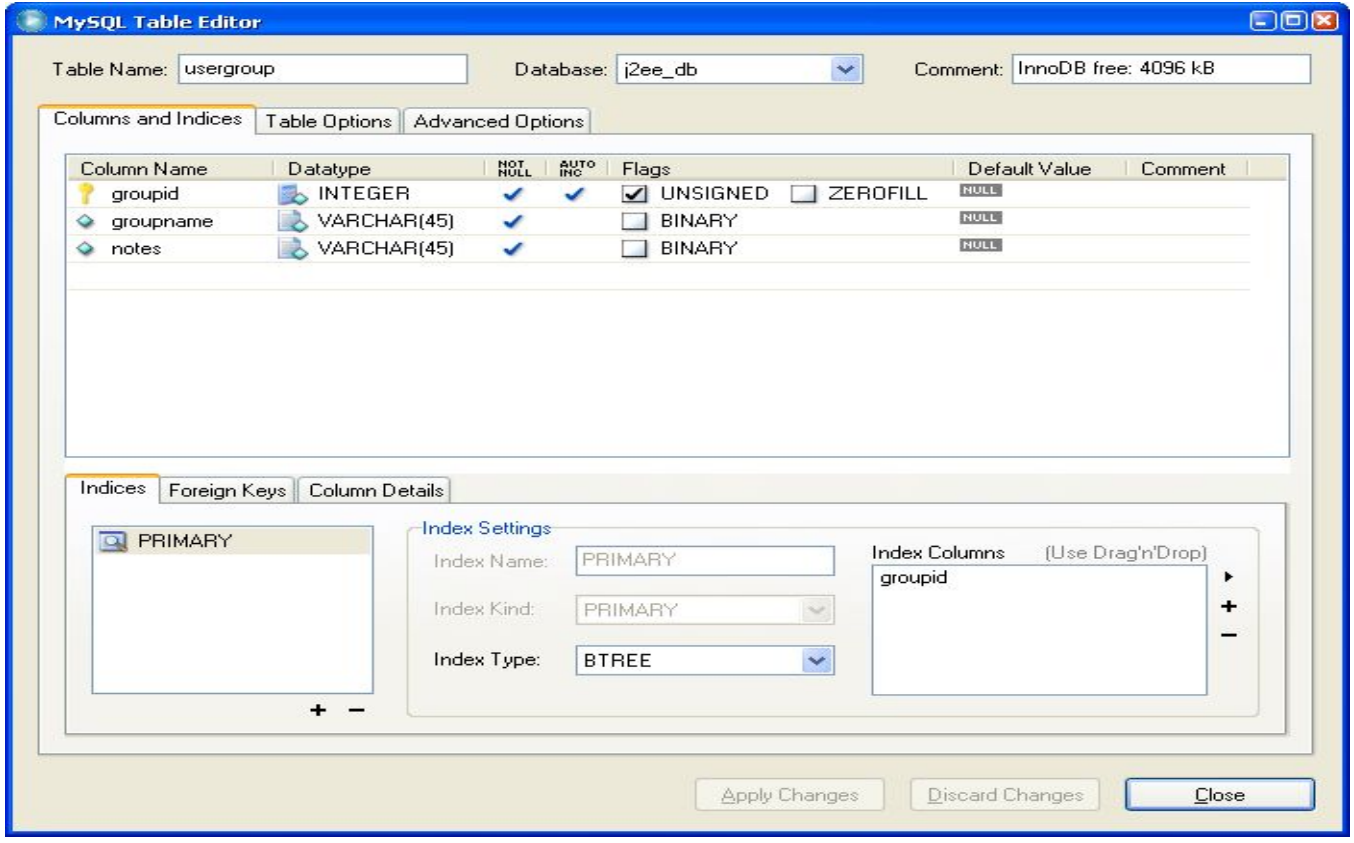


Update user's information → Screen no.5

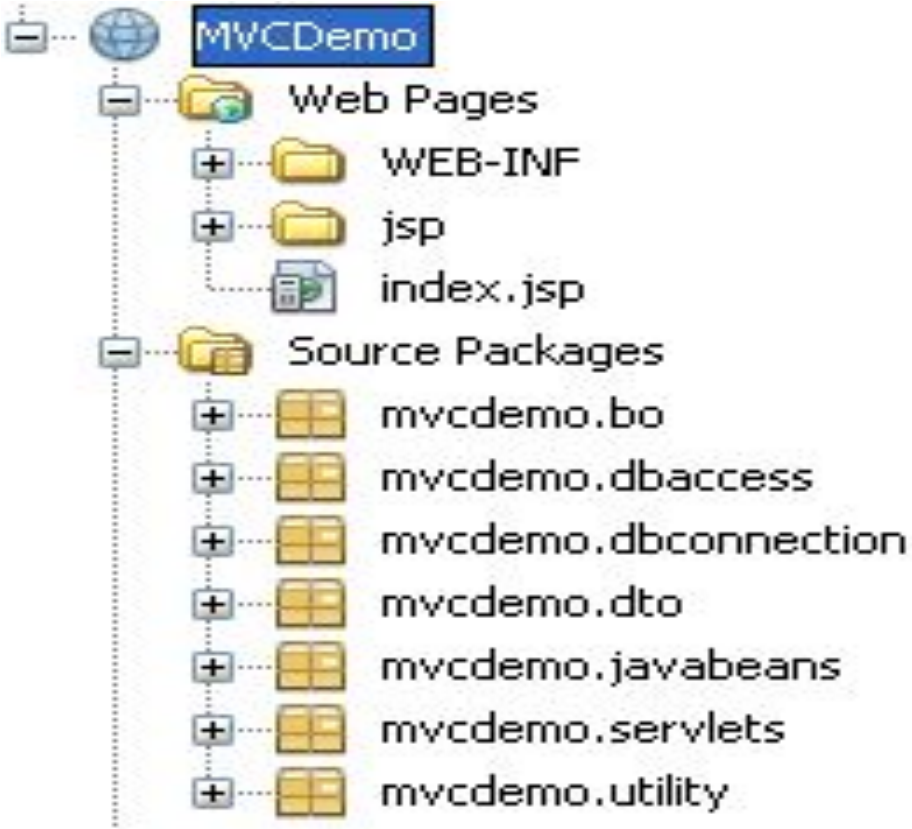
Example - Database

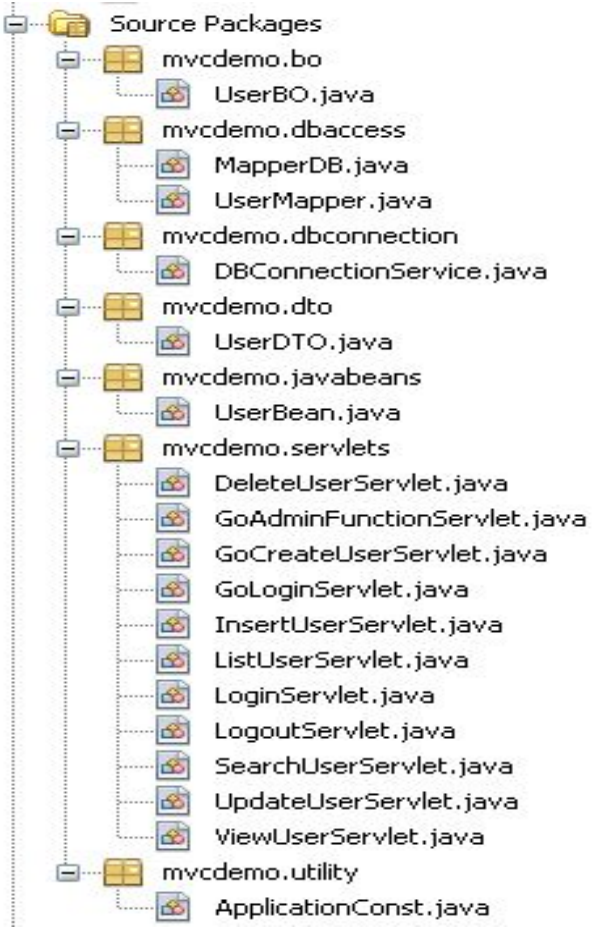


Example - Database



Packages


MVCDemo
Web Pages
WEB-INF
jsp
index.jsp
Source Packages
mvcdemo.bo
mvcdemo.dbaccess
mvcdemo.dbconnection
mvcdemo.dto
mvcdemo.javabeans
mvcdemo.servlets
mvcdemo.utility


Source Packages
mvcdemo.bo
UserBO.java
mvcdemo.dbaccess
MapperDB.java
UserMapper.java
mvcdemo.dbconnection
DBConnectionService.java
mvcdemo.dto
UserDTO.java
mvcdemo.javabeans
UserBean.java
mvcdemo.servlets
DeleteUserServlet.java
GoAdminFunctionServlet.java
GoCreateUserServlet.java
GoLoginServlet.java
InsertUserServlet.java
ListUserServlet.java
LoginServlet.java
LogoutServlet.java
SearchUserServlet.java
UpdateUserServlet.java
ViewUserServlet.java
mvcdemo.utility
ApplicationConst.java