Advanced Internet Technology Lab

Lab # 11

Model View Controller (MVC)

Eng. Haneen El-masry

May, 2014
Objectives

➢ To be familiar with MVC concepts.

What is MVC?

MVC stands for Model View Controller. It is a design approach to separate data processing code from data presentation code.

Possibilities for Handling a Single Request

1- Servlet only: Works well when:
   ➢ Output is a binary type, e.g., an image.
   ➢ There is no output, e.g., you are doing forwarding or redirection.
   ➢ Format/layout of page is highly variable.

2- JSP only: Works well when:
   ➢ Output is mostly character data, e.g. HTML.
   ➢ Format/layout mostly fixed.

3- Combination (MVC architecture): Needed when:
   ➢ A single request will result in multiple substantially different looking results.
   ➢ You perform complicated data processing, but have a relatively fixed layout.

MVC Flow of Control

![MVC Flow of Control Diagram]
The original request is handled by a servlet. The servlet invokes the business-logic and data-access code and creates beans to represent the results (that’s the model). Then, the servlet decides which JSP page is appropriate to present those particular results and forwards the request there (the JSP page is the view). The servlet decides what business logic code applies and which JSP page should present the results (the servlet is the controller).

**MVC Implementation**

- Frameworks as Struts and Java Server Faces (JSF) are sometimes useful, but they are not required!
- Implementing MVC with the built-in RequestDispatcher works very well for most simple and moderately complex applications.

**Implementing MVC with RequestDispatcher**

1- **Define beans to represent the data**

In this case, since a servlet (never a JSP page) will be creating the beans, the requirement for an empty (zero-argument) constructor is waived.

2- **Write a servlet to handle requests**

In most cases, the servlet reads request parameters. In fact, with the MVC approach the servlets do not create any output; the output is completely handled by the JSP pages. So, the servlets do not call response.setContentType, response.getWriter, or out.println.

3- **Populate the beans**

The results are placed in the beans that were defined in step 1.

   `ValueObject value = new ValueObject(...);`

4- **Store the bean in the request, session, or servlet context**

   - **Storing Data in request:**
     
     `request.setAttribute("key", value);`

   - **Storing Data in session:**
     
     `HttpSession session = request.getSession();
     session.setAttribute("key", value);`
Storing Data in application:
ServletContext application = getServletContext();
    application.setAttribute("key", value);

5- 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.

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

Note:
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.

6- Extract the data from the beans

Once the request arrives at the JSP page, the JSP page uses jsp:useBean and jsp:getProperty to extract the data.

➢ The servlet, not the JSP page, should create all the data objects. So, to guarantee that the JSP page will not create objects, you should use:

    <jsp:useBean ... type="package.Class" />

instead of

    <jsp:useBean ... class="package.Class" />.

➢ The JSP page should not modify the objects. So, you should use jsp:getProperty but not jsp:setProperty.
Example: Bank Account Balances

- BankCustomer Bean

```java
package package1;

import java.util.*;

public class BankCustomer {

    private String id, firstName, lastName;
    private double balance;

    public BankCustomer(String id, String firstName, String lastName, double balance) {
        this.id = id;
        this.firstName = firstName;
        this.lastName = lastName;
        this.balance = balance;
    }

    public String getId() {
        return id;
    }

    public String getFirstName() {
        return firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public double getBalance() {
        return balance;
    }

    public double getBalanceNoSign() {
        return Math.abs(balance);
    }

    public void setBalance(double balance) {
        this.balance = balance;
    }

    static{
        BankCustomer customers = new BankCustomer("id001", "John", "Hacker", -3456.78);
        customers.put("id002", new BankCustomer("id002", "Jane", "Hacker", 1234.56));
        customers.put("id003", new BankCustomer("id003", "Juan", "Hacker", 987654.32));
    }

    public static BankCustomer getCustomer(String id) {
        return customers.get(id);
    }
}
```
• **showBalance.java Servlet**

```java
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    String id = request.getParameter("id");
    BankCustomer customer = BankCustomer.getCustomer(id);
    String address = ";
    if(customer==null){
        address="/WEB-INF/UnKnownCustomer.jsp";
    }
    else{
        double balance = customer.getBalance();
        if(balance<0){
            address="/WEB-INF/NegativeBalanceCustomer.jsp"
            request.setAttribute("negative",customer);
        }
        else if(balance<10000){
            address="/WEB-INF/NormalBalanceCustomer.jsp"
            request.setAttribute("normal",customer);
        }
        else{
            address="/WEB-INF/HighBalanceCustomer.jsp"
            request.setAttribute("high",customer);
        }
    }
    RequestDispatcher dispatcher = request.getRequestDispatcher(address);
    dispatcher.forward(request, response);
}
```

• **UnKnownCustomer.jsp**

```html
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Unknown</title>
</head>
<body>
<center><h1>UnKnown Customer</h1></center>
Unrecognized Customer ID
</body>
</html>
```
- **NegativeBalanceCustomer.jsp**

```html
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Negative</title>
</head>
<body>
<jsp:useBean id="negative" type="package1.BankCustomer" scope="request"></jsp:useBean>
<br><center><h1>Negative Balance !!!!</h1></center>
<br><center>Customer ID: 
<jsp:getProperty property="id" name="negative"/></center>
<br>First Name: 
<jsp:getProperty property="firstName" name="negative"/>
<br>Last Name: 
<jsp:getProperty property="lastName" name="negative"/>
<br>You take from Us: 
<jsp:getProperty property="balanceNoSign" name="negative"/>
</body>
</html>
```

- **NormalBalanceCustomer.jsp**

```html
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<jsp:useBean id="normal" type="package1.BankCustomer" scope="request"></jsp:useBean>
<br><center><h1>Normal Customer</h1></center>
<br><center>Customer ID: 
<jsp:getProperty property="id" name="normal"/></center>
<br>First Name: 
<jsp:getProperty property="firstName" name="normal"/>
<br>Last Name: 
<jsp:getProperty property="lastName" name="normal"/>
<br>Your Balance: 
<jsp:getProperty property="balance" name="normal"/>
</body>
</html>
```
- **HighBalancecustomer.jsp**

```html
<!-- page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"-->
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>High Balance</title>
</head>
<body>
<jsp:useBean id="high" type="package1.BankCustomer" scope="request"></jsp:useBean>
<center><h1>High Balance</h1></center>
CustomerID:
<jsp:getProperty property="id" name="high"/>
<br>
First Name:
<jsp:getProperty property="firstName" name="high"/>
<br>
LastName:
<jsp:getProperty property="LastName" name="high"/>
<br>
Your High Balance:
<jsp:getProperty property="balanceNoSign" name="high"/>
<br>
</body>
</html>
```

**The Output**

![Image of the output](image)

Negative Balance !!!

CustomerID: id001
First Name: John
Last Name: Hacker
You take from Us: 3456.78
Normal Customer

CustomerID: id002
First Name: Jane
Last Name: Hacker
Your Balance: 1234.56

High Balance

CustomerID: id003
First Name: Juan
Last Name: Hacker
Your High Balance: 987654.32

UnKnown Customer

Unrecognized Customer ID
Comparing the Three Data-Sharing Approaches

- **Request-Based Sharing**

Example

Our goal is to display a random number to the user. Each request should result in a new number, so request-based sharing is appropriate.

- **NumberBean.java**

```java
package package1;

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;
    }
}
```

- **RandomNumberServlet.java**

```java
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    NumberBean bean = new NumberBean(Math.random());
    request.setAttribute("randomNum", bean);
    String address = "/WEB-INF/RandomNum.jsp";
    RequestDispatcher dispatcher = request.getRequestDispatcher(address);
    dispatcher.forward(request, response);
}
```
- **RandomNum.jsp**

```jsp
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>request Sharing</title>
</head>
<body>
jsp:useBean id="randomNum" type="package1.NumberBean" scope="request" />
<h2>Random Number:
jsp:getProperty name="randomNum" property="number" />
</h2>
</body>
</html>
```

**The Output**

![Random Number: 0.4371131122788865](image1)

![Random Number: 0.11359227411692685](image2)

➤ **Session-Based Sharing**

**Example**

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.
- **NameBean.java**

```java
package package1;

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;
    }

    public void setFirstName(String newFirstName) {
        firstName = newFirstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String newLastName) {
        lastName = newLastName;
    }
}
```

- **RegistrationServlet.java**

```java
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    HttpSession session = request.getSession();
    NameBean nameBean = (NameBean)session.getAttribute("nameBean");
    if (nameBean == null) {
        nameBean = new NameBean();
        session.setAttribute("nameBean", nameBean);
    }
    String firstName = request.getParameter("firstName");
    if ((firstName != null) && (!firstName.trim().equals(""))) {
        nameBean.setFirstName(firstName);
    }
    String lastName = request.getParameter("lastName");
    if ((lastName != null) && (!lastName.trim().equals(""))) {
        nameBean.setLastName(lastName);
    }
    String address = "/WEB-INF/ShowName.jsp";
    RequestDispatcher dispatcher = request.getRequestDispatcher(address);
    dispatcher.forward(request, response);
}
```
ShowName.jsp

```html
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>session sharing</title>
</head>
<body>
<center><h1>Thanks for Registering</h1></center>
<jsp:useBean id="nameBean" type="package1.NameBean" scope="session" />
<h2>First Name: $nameBean.firstName$</h2>
<h2>Last Name: $nameBean.lastName$</h2>
</body>
</html>
```

The Output

http://localhost/MVC/RegistrationServlet

Thanks for Registering

First Name: Missing first name

Last Name: Missing last name

http://localhost/MVC/RegistrationServlet?firstName=Haneen&lastName=Ibrahim

Thanks for Registering

First Name: Haneen

Last Name: Ibrahim
Thanks for Registering

First Name: Haneen
Last Name: Ibrahim

- Application-Based Sharing

Example

Our goal is to display a user mood. If the user fails to tell us his mood, we want to use whatever mood we most recently computed for any user. Data is shared among multiple clients, so application-based sharing is appropriate.

- MoodBean.java

```java
package package1;

public class MoodBean {
    private String mood="No Specified Mood";

    public MoodBean (){
    }

    public String getMood(){
        return (mood);
    }

    public void setMood(String mood){
        this.mood=mood;
    }
}
```
- **MoodServlet.java**

```java
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    ServletContext application = getServletContext();
    String mood = request.getParameter("mood");
    synchronized (this) {
        MoodBean moodBean = (MoodBean) application.getAttribute("moodBean");
        if (moodBean == null) {
            moodBean = new MoodBean();
            application.setAttribute("moodBean", moodBean);
        }
        if (mood != null) {
            moodBean.setMood(mood);
        }
        RequestDispatcher dispatcher = request.getRequestDispatcher("WEB-INF/MoodShow.jsp");
        dispatcher.forward(request, response);
    }
}
```

- **MoodShow.jsp**

```jsp
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1"/>
<title>application sharing</title>
</head>
<body>
<jsp:useBean id="moodBean" type="package1.MoodBean" scope="application"/>
<h2>Mood :</h2>
<jsp:getProperty name="moodBean" property="mood"/>
</body>
</html>
```

**The Output**

![Mood: No Specified Mood](http://localhost/MVC/MoodServlet)

![Mood: bad](http://localhost/MVC/MoodServlet?mood=bad)
Notes:

➢ If the jsp destination page uses relative URLs for images or style sheets, it needs to make them relative to the servlet URL or the server root, not to the destination page’s actual location.

➢ POST requests cannot be forwarded to normal HTML pages. The solution to this problem is to simply rename the HTML page to have a .jsp extension.

➢ The most common request-forwarding scenario is one in which the request first goes to a servlet and the servlet forwards the request to a JSP page.

➢ It is certainly possible for the destination page to be a servlet. Similarly, it is quite possible for a JSP page to forward requests elsewhere.

➢ The servlet can combine its output with that of one or more JSP pages, using include method of RequestDispatcher instead of forward method.