

Web应用开发 之 JSP技术模型

赵小敏

浙江工业大学计算机科学与技术学院

本节内容

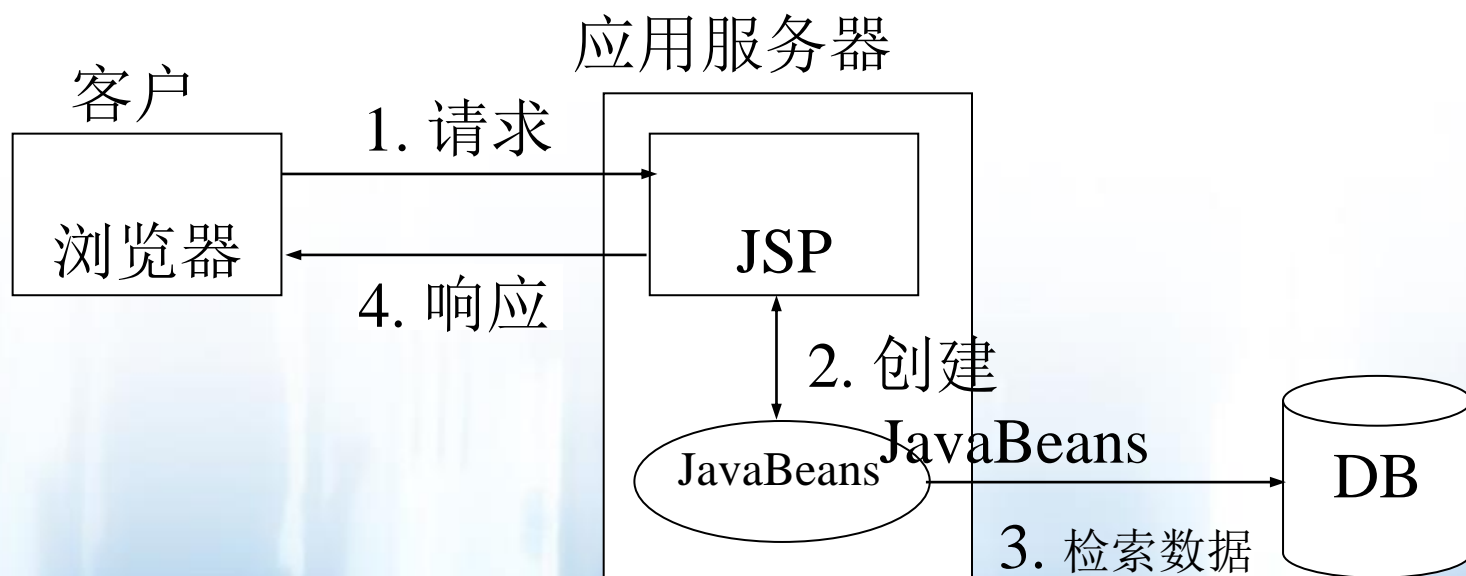
- 3.9 MVC设计模式

3.9.1 MVC设计模式简介

- 原Sun公司推出Servlet技术的主要目的是代替CGI编程。
- 可以把Servlet看成是含有HTML的Java代码。
- 可以把JSP看成是含有Java代码的HTML页面。
- Web应用程序的两种体系结构方法：
 - JSP Model 1体系结构
 - JSP Model 2体系结构

1、Model 1体系结构

- 在Model 1体系结构中，每个请求的目标都是JSP页面。该页面完全负责完成请求所需要的所有的任务，其中包括验证客户、使用JavaBean访问数据库及管理用户状态等。



2、Model 2体系结构

- 遵循MVC (Model-View-Controller) 的设计模式

□核心思想是实现功能分离

□MVC是Model-View-Controller的缩写

通过JavaBean组件实现

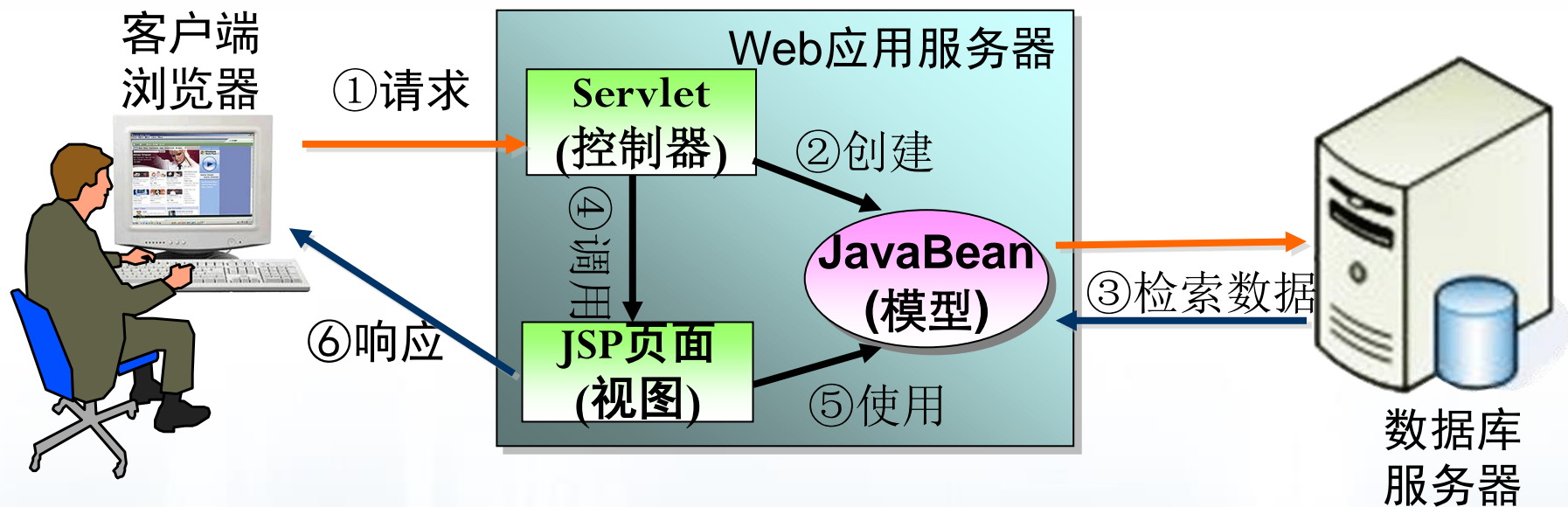
- **模型 (Model)** 是实现控制数据访问与数据处理功能的
应用程序，即负责实现业务逻辑
- **视图 (View)** 是实现采集用户输入的数据并传递给控
制器，或输出控制器中的处理数据给用户的应用程
序，即显示用户界面

JSP实现

控制器 (Controller) 是实现模型与视图之间的数
据流向与数据转换功能的应用程序，即负责View和
Model之间的控制关系

通常是Servlet

MVC模式的工作原理



MVC模式的优点

□ 各司其职，互不干涉

- 在MVC模式中，三个层各司其职，所以如果一旦哪一层的需求发生了变化，就只需要更改相应的层中的代码而不会影响到其它层中的代码。

□ 有利于开发中的分工合作

- 网页设计人员可以进行开发视图层中的JSP，对业务熟悉的开发人员可开发业务层，而其它开发人员可开发控制层。

□ 有利于组件的重用

- 分层后更有利于组件的重用。如控制层可独立成一个能用的组件，视图层也可做成通用的操作界面。

3.9.2 实现MVC模式的一般步骤

- 使用MVC设计模式开发Web应用程序可采用下面的一般步骤：
 - 1. 定义JavaBeans表示数据
 - 2. 使用Servlet处理请求
 - 3. 填写JavaBeans对象数据
 - 4. 结果的存储
 - 5. 转发请求到JSP页面
 - 6. 从JavaBeans对象中提取数据

1. 定义JavaBeans表示数据

- JavaBeans对象或使用POJO (Plain Old Java Object) 对象一般只用来存放数据，JSP页面从JavaBeans对象或POJO中获得数据并显示给用户。

```
public class Customer implements Serializable{  
    private String customName;  
    private String email;  
    private int age;  
    // 构造方法定义  
    // setter和getter方法定义  
}
```

2. 使用Servlet处理请求

- 在MVC模式中，**Servlet实现控制器**功能，它从请求中读取请求信息（如表单数据）、创建JavaBeans对象、执行业务逻辑、访问数据库等，最后将请求转发到视图组件。
- Servlet并不创建任何输出，输出由JSP页面实现，Servlet中并不调用 `response.setContentType()`、`response.getWriter()` 或 `out.println()` 等方法。

3. 填写JavaBeans对象数据

- 控制器创建JavaBeans对象后需要填写该对象的值。可以通过请求参数值或访问数据库得到有关数据，然后填写到JavaBeans对象属性中。

4. 结果的存储

- 创建了与请求相关的数据并将数据存储到JavaBeans对象中后，接下来应该将这些bean对象存储在JSP页面能够访问的位置。
- 在Servlet中主要可以在三个位置存储JSP页面所需的数据，它们是HttpServletRequest对象、HttpSession对象和ServletContext对象。
- 这些存储位置对应<jsp:useBean>动作scope属性的三个非默认值：request、session和application。

5. 转发请求到JSP页面

- 在使用请求作用域共享数据时，应该使用 `RequestDispatcher` 对象的 `forward()` 将请求转发到JSP页面。
- 获取 `RequestDispatcher` 对象可使用请求对象的 `getRequestDispatcher()` 或使用 `ServletContext` 对象的 `getRequestDispatcher()` 方法。

5. 转发请求到JSP页面

- 得到RequestDispatcher对象后，调用它的forward()将控制转发到指定的组件。
- 在使用会话作用域共享数据时，使用响应对象的sendRedirect()重定向可能更合适。
- 注意，RequestDispatcher的forward()和响应对象的sendRedirect()完全不同。

6. 从JavaBeans对象中提取数据

- 请求到达JSP页面之后，使用
`<jsp:useBean>`和`<jsp:getProperty>`
提取数据。
- 但应注意，不应在JSP页面中创建对象，创建JavaBeans对象是由Servlet完成的。
- 为了保证JSP页面不会创建对象，应该使用动作：

```
<jsp:useBean id="customer"  
type="com.demo.Customer" />
```

- 而不应该使用动作：

```
<jsp:useBean id="customer"  
class="com.demo.Customer" />
```

6. 从JavaBeans对象中提取数据

- 在JSP页面中也不应该修改对象。因此，只应该使用`<jsp:getProperty>`动作，而不应该使用`<jsp:setProperty>`动作。
- 在JSP2.0后，使用表达式语言（EL）输出数据

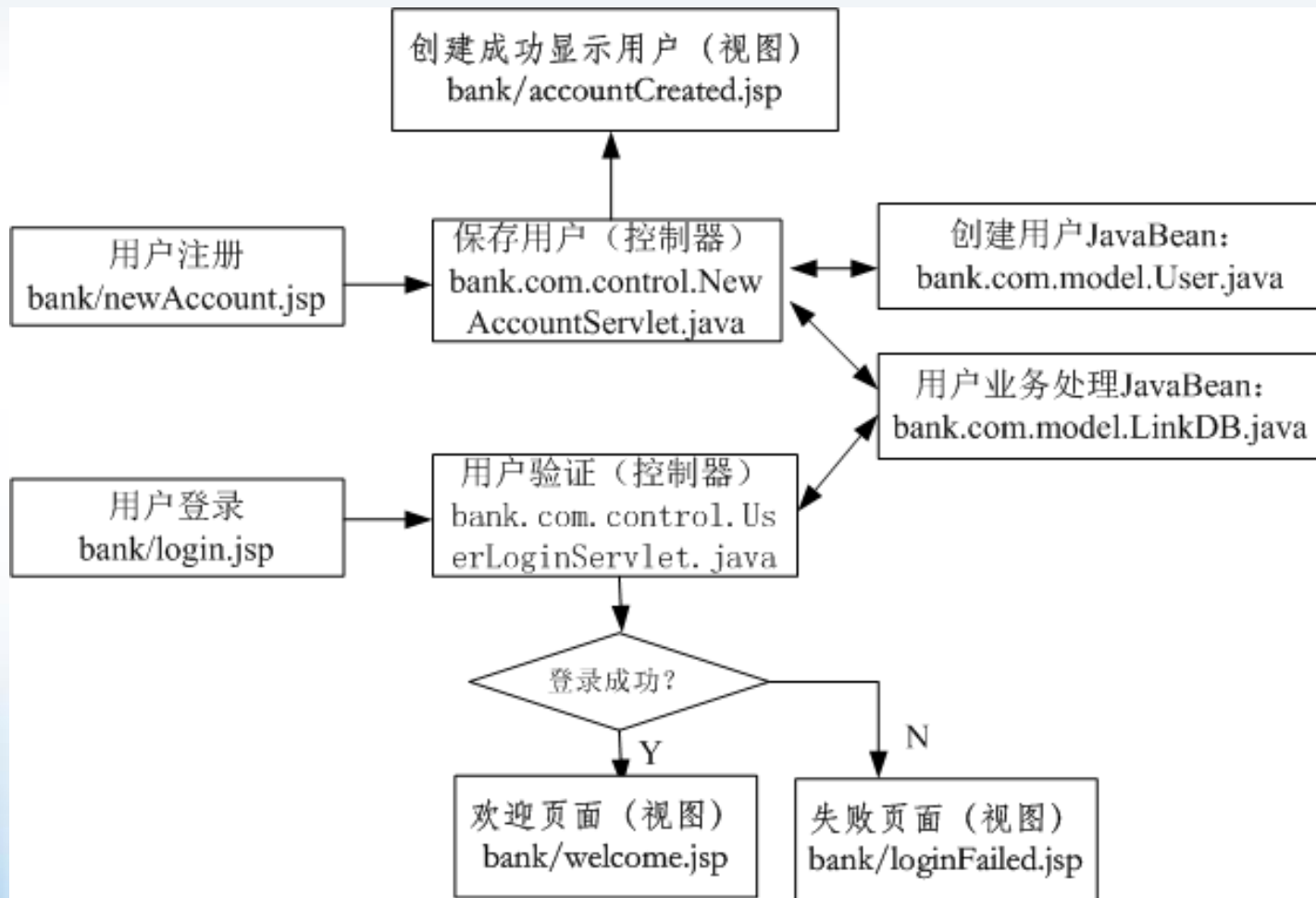
6. 从JavaBeans对象中提取数据

- 另外，在<jsp:useBean>动作中使用的scope属性值应该与在Servlet中将JavaBeans对象存储的位置相对应。
- 如在Servlet中将一个JavaBeans对象存储在请求作用域中，在JSP页面中应该使用下面的<jsp:useBean>动作获得该JavaBeans对象：

```
<jsp:useBean id="customer"  
type="com.demo.Customer" scope="request">
```

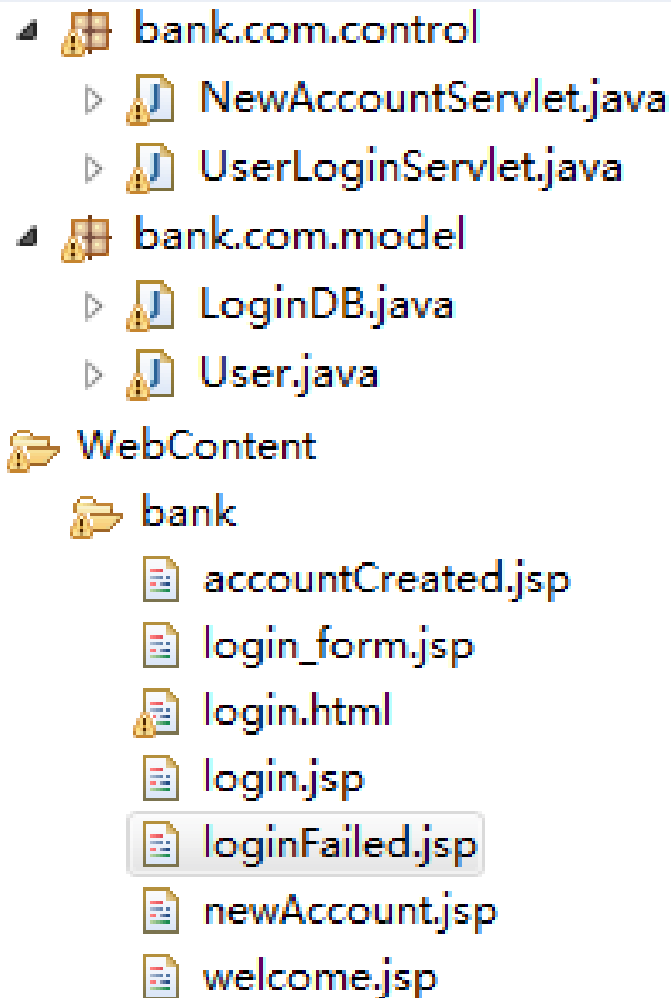
MVC的例子：用户注册/登录系统

- 基于MVC设计模式的用户注册/登录系统流程



3.9.3MVC例子： 用户注册/登录系统

- 基于MVC设计模式的用户注册/登录系统代



```
bank.com.control
├── NewAccountServlet.java
├── UserLoginServlet.java
bank.com.model
├── LoginDB.java
├── User.java
WebContent
├── bank
│   ├── accountCreated.jsp
│   ├── login_form.jsp
│   ├── login.html
│   ├── login.jsp
│   ├── loginFailed.jsp
│   ├── newAccount.jsp
│   └── welcome.jsp
```

The image shows a project structure in an IDE. The project is named 'bank.com'. It has two main packages: 'bank.com.control' and 'bank.com.model'. The 'bank.com.control' package contains two servlets: 'NewAccountServlet.java' and 'UserLoginServlet.java'. The 'bank.com.model' package contains two classes: 'LoginDB.java' and 'User.java'. There is also a 'WebContent' directory, which contains a 'bank' subdirectory. The 'bank' subdirectory contains several JSP files: 'accountCreated.jsp', 'login_form.jsp', 'login.html', 'login.jsp', 'loginFailed.jsp', 'newAccount.jsp', and 'welcome.jsp'. The 'loginFailed.jsp' file is currently selected.

创建用户JavaBean:

bank.com.model.User.java

```
1. package bank.com.model;
2. public class User implements java.io.Serializable {
3.     private final String userName, password, hint;
4.         //final强调此属性初始化后, 不能修改hint是口令提示
5.     public User(String userName, String password, String hint) {
6.         this.userName = userName;
7.         this.password = password;
8.         this.hint = hint;
9.     }
10.    public String getUserName() {
11.        return userName;
12.    }
13.    public String getPassword() {
14.        return password;
15.    }
16.    public String getHint() {
17.        return hint;
18.    }
19.    //判断当前对象用户名和密码是否相等
20.    public boolean equals(String uname, String upwd) {
21.        return getUserName().equals(uname) && getPassword().equals(upwd);
22.    }
23. }
```

用户业务处理JavaBean:

bank.com.model.LoginDB.java

```
1. package bank.com.model;
2. import java.util.Iterator;
3. import java.util.Vector;
4. public class LoginDB implements java.io.Serializable {
5.     private Vector users = new Vector();
6.     //Vector类是同步的，所以addUser就不需要同步了
7.     public void addUser(String name, String pwd, String hint) {
8.         users.add(new User(name, pwd, hint));
9.     }
10.    //下面方法判断是否存在正确的user
11.    public User getUser(String name,String pwd) {
12.        Iterator it = users.iterator();
13.        User user;
14.        synchronized(users) { //迭代需要同步
15.            while(it.hasNext()) {
16.                user = (User)it.next();
17.                if(user.equals(name, pwd))
18.                    return user; //如果返回真，就返回当前user
19.            }
20.        }
21.        return null;
22.    }
23. }
```

用户注册bank/newAccount.jsp

<h4>输入注册信息</h4>

<form action=" ../NewAccountServlet" method="post">

<table>

<tr>

<td>客户名: </td>

<td><input type="text" name="userName"></td>

</tr>

<tr>

<td>密码: </td>

<td><input type="text" name="userPwd"></td>

</tr>

<tr>

<td>口令提示: </td>

<td><input type="text" name="hint"></td>

</tr>

<tr>

<td><input type="submit" value="确定"></td>

<td><input type="reset" value="重置"></td>

</tr>

</table>

</form>

用户注册bank/accountCreated.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; UTF-8">
<title>创建用户</title>
</head>

<body>
新用户已经创建!
<font color="#0000FF"> <%=request.getParameter("userName")%></font>
<hr><a href="bank/Login.jsp">马上去登录</a>
</body>
</html>
```

用户登录 *bank/login.jsp*

1. `<html>`
2. `<head>`
3. `<meta http-equiv="Content-Type" content="text/html; charset=utf-8">`
4. `<title>Login Page</title>`
5. `</head>`
6. `<body>`
7. `<%@ include file="/bank/login_form.jsp" %>`
8. `</body>`
9. `</html>`

bank/login_form.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<p><font color="#6666CC">请登陆</font></p>
<hr>
<form name="form1" method="post" action="../UserLoginServlet">
    <table width="68%" border="0" cellpadding="2" cellspacing="2">
        <tr>
            <td width="33%" align="right">用户名: </td>
            <td width="67%">
                <input type="text" name="userName"></td>
        </tr>
        <tr>
            <td align="right">密码: </td>
            <td><input type="text" name="userPwd" ></td>
        </tr>
        <tr align="center">
            <td colspan="2">
                <input type="submit" name="Submit" value="登陆">
                <a href='newAccount.jsp'>未注册? </a>
            </td>
        </tr>
    </table>
</form>
```

保存用户（控制器）

bank.com.control.NewAccountServlet.java

```
package bank.com.control;

import javax.servlet.*;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.*;
import java.io.IOException;
import bank.com.model.LoginDB;

@WebServlet("/NewAccountServlet")
public class NewAccountServlet extends HttpServlet {
    public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        LoginDB loginDB = (LoginDB) getServletContext().getAttribute("loginDB");
        if (loginDB == null) {
            loginDB = new LoginDB();
        }
        loginDB.addUser(request.getParameter("userName"),
request.getParameter("userPwd"),
        request.getParameter("hint"));
        getServletContext().setAttribute("loginDB", loginDB);
        request.getRequestDispatcher("/bank/accountCreated.jsp").forward(request,
response);
    }
}
```

用户验证（控制器）

bank.com.control.UserLoginServlet.java

```
1. package bank.com.control;
2. import javax.servlet.*;
3. import javax.servlet.annotation.WebServlet;
4. import javax.servlet.http.*;
5. import java.io.IOException;
6. import bank.com.model.User;
7. import bank.com.model.LoginDB;

8. @WebServlet("/UserLoginServlet")
9. public class UserLoginServlet extends HttpServlet {
10.     private LoginDB loginDB;
11.     public void init(ServletConfig config) throws ServletException {
12.         super.init(config);
13.         loginDB = (LoginDB) getServletContext().getAttribute("loginDB");
14.         if (loginDB == null) {
15.             loginDB = new LoginDB();
16.             config.getServletContext().setAttribute("loginDB", loginDB);
17.         }
18.
19.     }
```

用户验证（控制器）

bank.com.control.UserLoginServlet.java

```
20. public void doGet(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {
21.     String name = request.getParameter("userName"); // 从login_form 表单得到值
22.     String pwd = request.getParameter("userPwd");
23.     User user = loginDB.getUser(name, pwd);
24.     if (user != null) { // 说明存在用户
25.         request.getSession().setAttribute("user", user); // 放到session 里面
26.         // 成功转发到welcome.jsp
27.         request.getRequestDispatcher("/bank/welcome.jsp").forward(request, response);
28.     } else {
29.         request.getRequestDispatcher("/bank/loginFailed.jsp").forward(request, response);
30.     }
31. }
32. public void doPost(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {
33.     doGet(request, response);
34. }
35. }
```

欢迎页面（视图）bank/welcome.jsp

```
1. <%@ page language="java" contentType="text/html; charset=UTF-8"
2.     pageEncoding="UTF-8"%>
3. <%@ page import="bank.com.model.User"%>
4. <html>
5. <head>
6. <meta http-equiv="Content-Type" content="text/html; charset=gb2312">
7. <title>Welcome Page</title>
8. </head>
9. <body>
10. <jsp:useBean id="user" scope="session" type="bank.com.model.User"/>
11. <!--也可以
12. <%
13.     User u = (User)session.getAttribute("user");
14. %>
15. -->
16. 欢迎你: <font color=red><%=u.getUserName()%></font>
17. </body>
18. </html>
```

失败页面（视图） bank/loginFailed.jsp

```
<%@ page language="java" contentType="text/html; charset=GB18030"
    pageEncoding="GB18030"%>

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>Login Failed</title>
</head>

<body>
登录失败, <a href="bank/Login.jsp">返回重新登陆</a>
<hr>
<a href="bank/newAccount.jsp">创建一个新用户 </a>
</body>
</html>
```

小 结

- MVC设计模式是Web应用开发中最常使用的设计模式，它将系统中的组件分为模型、视图和控制器，实现了业务逻辑和表示逻辑的分离，使用该模式开发的系统具有可维护性和代码重用性。

作业

- 1、某银行客户管理系统采用MVC模式设计实现，其中有一个注册客户信息和登陆的功能，请按如下要求编写程序。
 - (1) 编写一个类为`bank.com.model.Customer`的JavaBeans，包括4个属性：`email`表示客户邮箱，`password`表示密码，`custName`表示客户姓名，`phone`表示客户手机号；
 - (2) 编写一个输入客户信息页面`bank/inputCustomer.jsp`，通过表单输入客户信息，将请求转发到映射地址为`CustomerServlet.do`的`bank.com.control.CustomerServlet`类进行处理；
 - (3) 编写一个`bank.com.control.CustomerServlet`类，从输入客户信息页面得到客户信息存入文件`customerinfo.txt`文件，并将客户信息通过作用域共享后转发至客户显示页面`bank/displayCustomer.jsp`；
 - (4) `bank/displayCustomer.jsp`显示某客户的邮箱、姓名和手机号；
 - (5) `bank/displayAllCustomer.jsp`显示所有注册客户的信息；
 - (6) `bank/login.jsp`为登陆页面，可输入邮箱和密码
 - (7) 编写一个`bank.com.control.LoginServlet`类，获取登陆页面的邮箱和密码并进行判断，登陆成功后跳转至`welcome.jsp`页面，否则跳转至失败页面