JSP高级
（Java Server Page）

王新阳
wxyyuppie@bjfu.edu.cn

主要内容

- JavaBean的定义和优势
- JSP中访问JavaBean
- JavaBean的共享
- 基于Servlet和JSP的MVC架构

JavaBean的定义和优势

JavaBean的定义和优势——定义

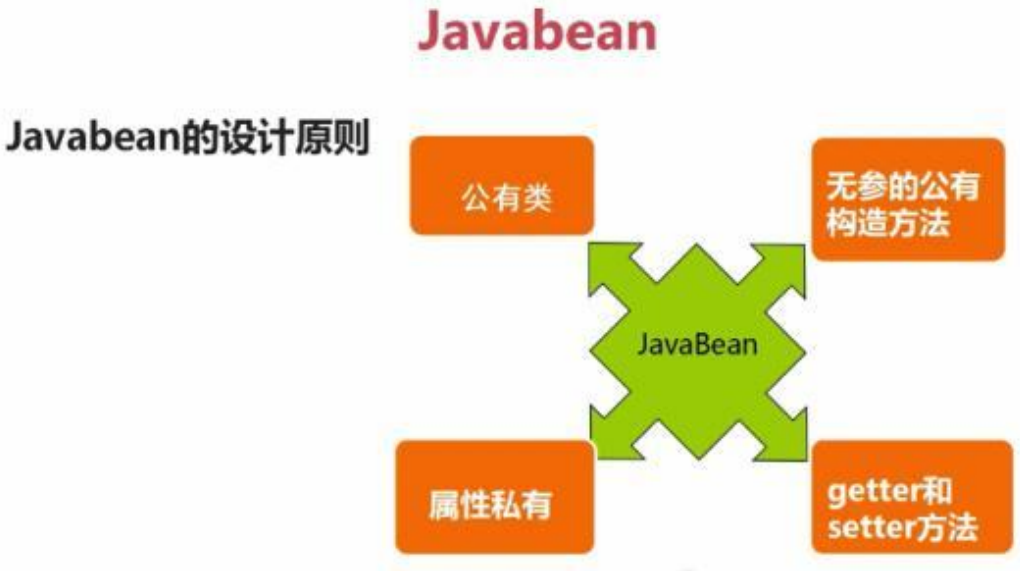
- JavaBean简称为Bean
- 是一种遵循某种定义规则的java类
  - 必须具备空参的构造函数：显式定义这样一个构造函数或者省略类所有的构造函数即可
  - 类变量必须是私有的：使用特定的存取函数对字段直接访问——属性
    - ✓xxx属性的存取函数要定义为getXxx和setXxx方法
    - ✓例如：若类有String类型的title属性，则读取函数需要定义为String getTitle()，设置函数定义为
      void setTitle(String x)
    - ✓ 布尔型的属性使用isXxx，而非getXxx
- 示例：LoginBean.java

xxx属性名称的首字母必须是小写

方法中，字段名第一个字母改为大写

JavaBean的定义和优势——定义

- **JavaBean 与其它 Java 类相比而言独一无二的特征：**
  - 提供一个默认的无参构造函数；
  - 需要被序列化并且实现了 **Serializable** 接口；
  - 可能有一系列可读写属性。
  - 可能有一系列的 **getter** 或 **setter** 方法。



JavaBean的定义和优势——使用Bean的好处

- 可以封装表单各种元素的读取
- 便于表单各种元素在不同JSP和Servlet中共享
- 定义Bean的其它注意事项
  - 一定要把Bean定义在包中，便于共享
  - 一个Bean实现java.io.serializable接口，便于Bean数据保存到外存中

JSP中访问JavaBean

JSP中访问JavaBean

- 创建Bean实例 **<jsp:useBean…>**
  - <jsp:useBean id="beanName“ class="package.Class" />
- 读取Bean属性的值 **<jsp:getProperty…>**
  - <jsp:getProperty name="beanName“ property="propertyName" />
- 设置Bean属性的值 **<jsp:setProperty…>**
  - <jsp:setProperty name="name“ property="property“ value="value" />
- 把**表单参数**和**Bean属性**进行绑定 **<jsp:setProperty…>**
  - <jsp:setProperty name="name" property="property " param=" param " />

Bean属性名称

表单元素名称

JSP中访问JavaBean—实例1

- 实例1　读取Bean属性的值
  - Bean属性文件 mvc.LoginBean.java
  - 结果显示文件 ReadLoginBean.jsp

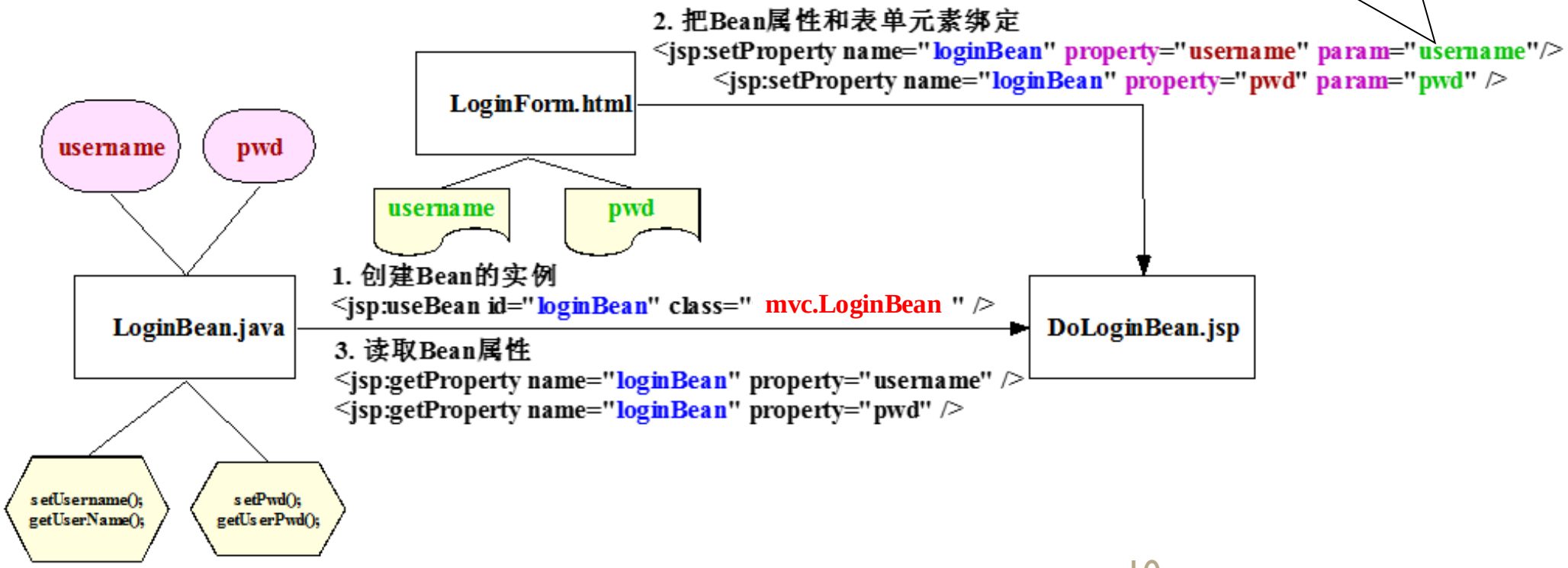
JSPAdvance工程

JSP中访问JavaBean—实例2

- 实例2　读取Bean属性对应的表单参数的值
  - 请求表单文件 LoginForm.html
  - Bean属性文件 mvc.LoginBean.java
  - 结果显示文件 DoLoginBean.jsp

JSPAdvance工程

Bean属性名称和表单元素名称可以不一致



JSP中访问JavaBean——一次性绑定

- 一次性把所有表单参数和Bean属性绑定
  - **<jsp:setProperty name="name" property="*" />**
  - **前提条件是：表单参数名称和Bean属性名称要完全一致，大小写也要一致**

- 实例3
  - **JSPAdvance工程**
  - 请求表单文件 LoginForm2.html
  - Bean属性文件 mvc.LoginBean2.java
  - 结果显示文件 DoLoginBean2.jsp

JSP中访问JavaBean——使用bean的实质

- 本质：
  - 创建Bean实例
    - ✓ <jsp:useBean id="loginBean" class="mvc.LoginBean" /> 等价于
    - ✓ <% mvc.LoginBean loginBean = new mvc.LoginBean(); %>
  - 读取Bean属性
    - ✓ <jsp:getProperty name="*loginBean*" property="*pwd*" />等价于
    - ✓ <%= loginBean.getPwd() %>
  - 设置Bean属性
    - ✓ <jsp:setProperty name="*loginBean*" property="*pwd*" param="*pwd*" />

    等价于 <% loginBean.setPwd(request.getParameter("pwd")) %>
    - ✓ <jsp:setProperty name="*loginBean*" property="*pwd*" value="*111*" />

    等价于 <% loginBean.setPwd("111") %>
- 好处：
  - 不需用到显式的Java编程
  - 从请求表单中导出表单参数的值更容易
  - 在页面和servlet间共享对象更容易

JSP中访问JavaBean—bean创建的前提

- **Bean创建的前提**
  - 因为Bean可以共享，所以在执行创建Bean语句jsp:useBean之前需要检查是否已经存在同名的Bean
  - 仅当找不到相同id和scope的bean时，jsp:useBean才会引发bean新实例的创建。
- **结论**
  - 不同的Bean要有不同的名字，即jsp:useBean中的id应该不同

JavaBean的共享

Bean的作用域

- 所谓“作用域”就是“信息共享的范围”，也就是说一个信息能够在多大的范围内有效。
- Web交互的最基本单位为HTTP请求。每个用户从进入网站到离开网站这段过程称为一个HTTP会话，一个服务器的运行过程中会有多个用户访问，就是多个HTTP会话。作用域解释如下。
  - ✓**application**：服务器启动到停止这段时间。
  - ✓**session**：HTTP会话开始到结束这段时间。
  - ✓**request**：HTTP请求开始到结束这段时间。
  - ✓**page**：当前页面从打开到关闭这段时间。

Bean的作用域

- 在应用Bean时可以指明Bean的共享范围
  - **<jsp:useBean id="…" class="…" scope="…" />**

- 共享范围
  - **page----默认值**
    - ✓**Bean只能应用在本JSP文件中**
  - **application**
    - ✓**Bean可以为同一Web应用中的所有用户共享**
  - **session**
    - ✓**Bean可以为用户访问会话过程所共享(不关闭浏览器窗口的情况下)，不同用户不能共享**
  - **request**
    - ✓**Bean只在当前请求过程中有效,请求过程可以连续跨越多个页面**

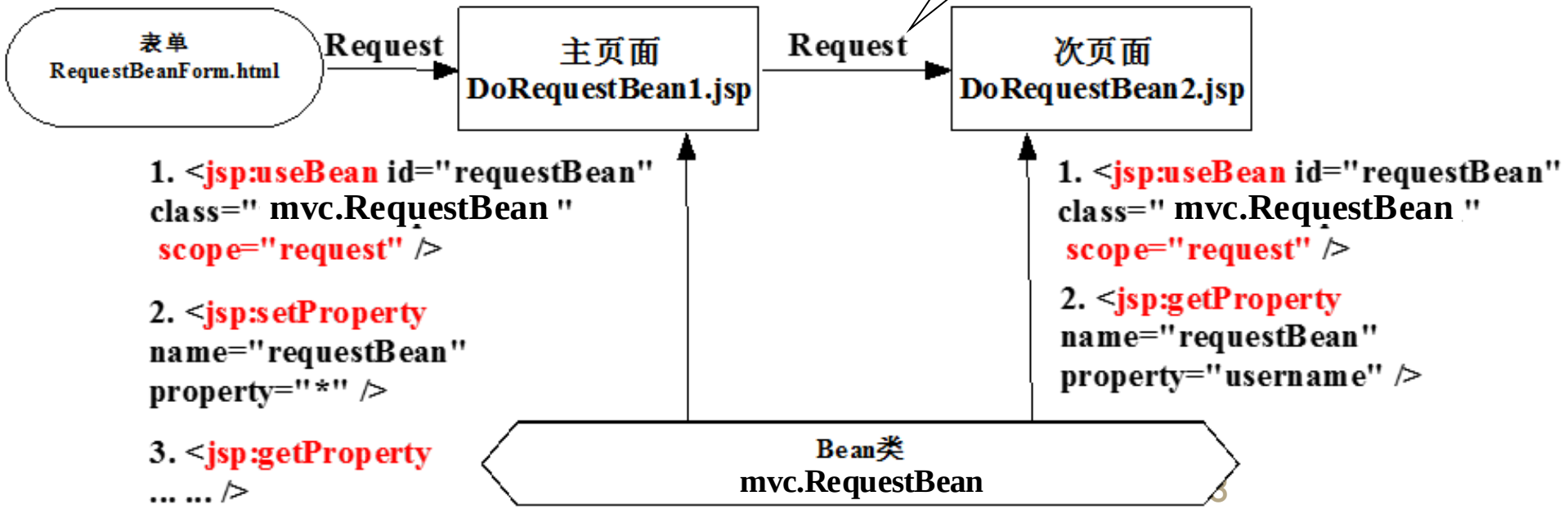
Bean共享实例——page范围

- 本幻灯片中P10-11中实例2和实例3，主要步骤：
  - 在请求表单中定义各种表单元素（元素名称即为表单参数）
  - 在java类中定义一个Bean类（私有属性xxx和其存取方法GetXxx(),SetXxx()）
  - 在处理表单的Jsp文件中
    - ✓创建Bean实例<jsp:useBean … …
    - ✓设置Bean属性<jsp:setProperty … …
    - ✓读取Bean属性<jsp:getProperty … …

Bean共享实例——request范围

- **实例　JSPAdvance工程**
  - 表单 **RequestBeanForm.html**
  - **Bean　RequestBean.java**
  - 处理表单和**Bean**的文件：
    - ✓主页面 **DoRequestBean1.jsp**
    - ✓次页面 **DoRequestBean2.jsp**

主页面跳转到次页面的方法——内部跳转：
<jsp:include …..
<jsp:forward … …
注意：<a href … …>在这里并不可行，因为它是重开一个request请求，是一种外部跳转。



思考：

1、将DoRequestBean1中的scope换成“page”结果会变成怎样

2、DoRequestBean1中<jsp:forwardpage=“DoRequestBean2.jsp”>

换成

<%response.sendRedirect(“DoRequestBean2.jsp”); %>

可不可以，为什么？

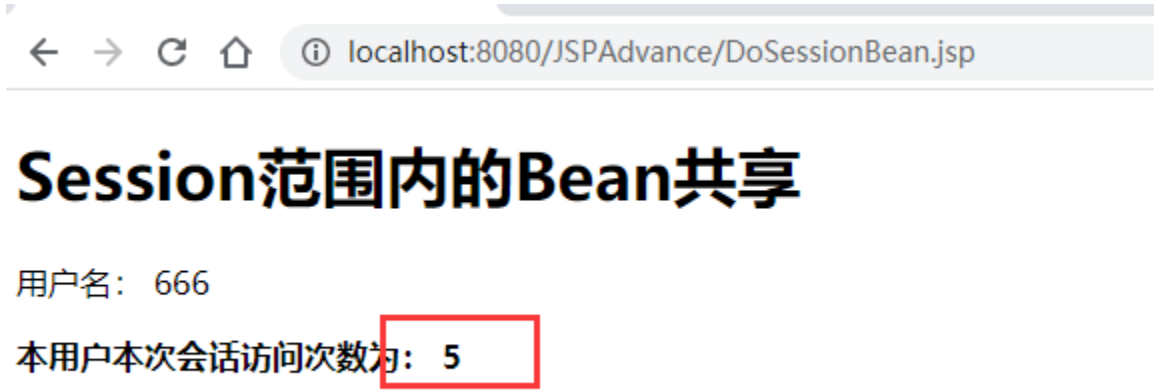
Bean共享实例——session范围

- **实例　JSPAdvance工程**
  - 表单 **SessionBeanForm.html**
  - **Bean　SessionBean.java**
  - 处理表单和**Bean**的页面文件 **DoSessionBean.jsp**
    - ✓创建Bean实例
      - **<jsp:useBean id="*sessionBean*" class="*mvc.SessionBean*" scope="*session*" />**
    - ✓设置Bean属性和表单参数对应
      - **<jsp:setProperty name="*sessionBean*" property="*" />**
    - ✓设置其它Bean属性
      - **<jsp:setProperty name="*sessionBean*" property="*accesscount*" value="*1*" />**
    - ✓读取Bean属性**<jsp:getProperty … …**

Bean共享实例——session范围

```jsp
<h1>Session范围内的Bean共享</h1>
<jsp:useBean id="sessionBean" class="mvc.SessionBean" scope="session" />
<jsp:setProperty name="sessionBean" property="*" />
<jsp:setProperty name="sessionBean" property="accesscount" value="1"  />
```

```java
public void setAccesscount(int t){
    this.accessount =this.accessount+t ;
}
```



结论：

(1) jsp:setProperty本质上调用的是setXXX方法

(2) session会话周期内都是同一个bean实例

只要在会话周期内，每次同一客户端访问该页面，计数都会递增

Bean共享实例——application范围

- **实例　JSPAdvance工程**
  - 表单 **ApplicationBeanForm.html**
  - **Bean　ApplicationBean.java**
  - 处理表单和**Bean**的页面文件 **DoApplicationBean.jsp**
    - ✓创建Bean实例
      - **<jsp:useBean id=“application*Bean*" class="*mvc.* Application*Bean*" scope=*“application"* />**
    - ✓设置Bean属性和表单参数对应
      - **<jsp:setProperty name=" application*Bean* " property="*" />**
    - ✓设置其它Bean属性
      - **<jsp:setProperty name=" application*Bean* " property="*accesscount*" value="*1*" />**
    - ✓读取Bean属性<jsp:getProperty … …

只要服务器开启，每次不同客户端访问该页面，计数都会递增

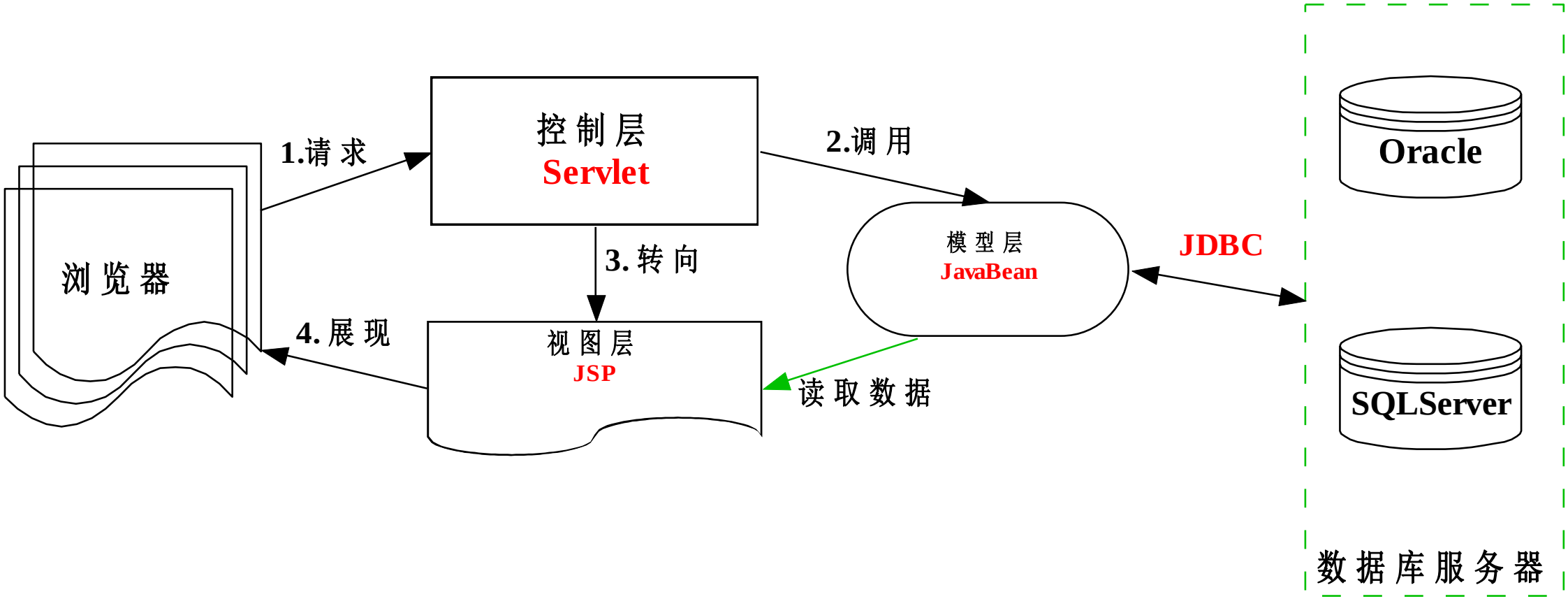
小结

- JavaBean的优点
  - 隐藏了Java语法
  - 将请求参数与Java对象关联起来更加容易(bean属性)
  - 使得在多个请求或servlets/JSP间共享对象得到简化
- JavaBean的定义
  - 私有属性xxx
  - 存取方法getXxx()和setXxx()
  - 定义在包中
- 创建或使用bean
  - jsp:useBean
- 输出bean的属性
  - jsp:getProperty
- 设置bean属性(用来把bean实例和表单元素绑定在一起)
  - jsp:setProperty

Servlet+JSP+Bean MVC

基于Servlet+JSP+Bean的MVC架构

- M—模型层
- V —视图层
- C —控制层



使用MVC架构的原因

- 只使用Servlet
  - 虽然在java代码中很容易实现业务流程控制
  - 但不足以承担页面展现的工作
- 只使用JSP
  - 虽然很容易简化HTML内容的开发与维护
  - 但难以处理复杂的业务流程控制，尤其是针对根据需求跳转到不同页面的问题，因为JSP文件使用的目的就是展现一个页面

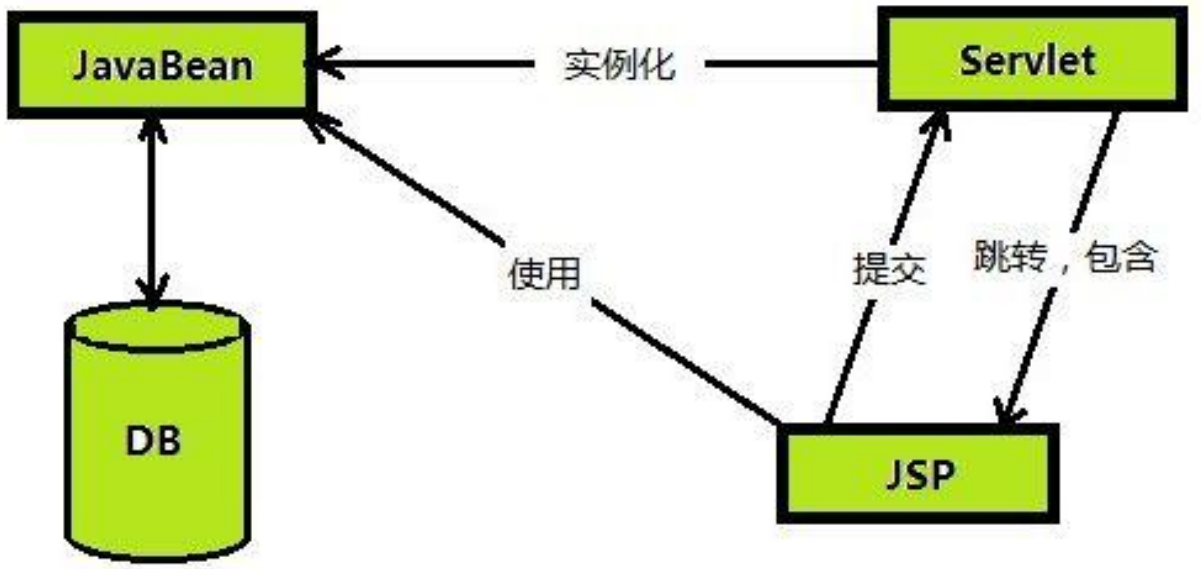
Servlet和JSP结合的好处

- **一个请求表单提交可能会得到外观相差较大的多种页面展示结果。**

- **可以轻易实现数据处理过程复杂，但布局相对固定的应用**
  - **Servlet**负责实现数据处理和流程控制
  - **JSP**负责不同结果的展现

MVC框架结构的实现方式

- 基于Servlet和JSP的实现
  - 使用RequestDispatcher接口的forward方法实现控制流程转移到不同的展现结果页面
- 基于Stucts的实现
- 基于Spring MVC的实现



基于Servlet+JSP+Bean的MVC实现步骤(1)

1. 定义用以表示数据的bean

2. 使用一个servlet处理请求
   - servlet读取请求参数，检查数据的缺失或异常等

3. 填充bean
   - 该servlet调用业务逻辑（与具体应用相关的代码）或数据访问代码得到最终的结果。得出的结果与bean绑定。

4. 将bean存储在request、session或servlet的上下文中，方便共享
   - 该servlet调用请求、会话或servlet上下文对象的setAttribute存储bean。

基于Servlet+JSP+Bean的MVC实现步骤(2)

5. 将请求转发到JSP页面

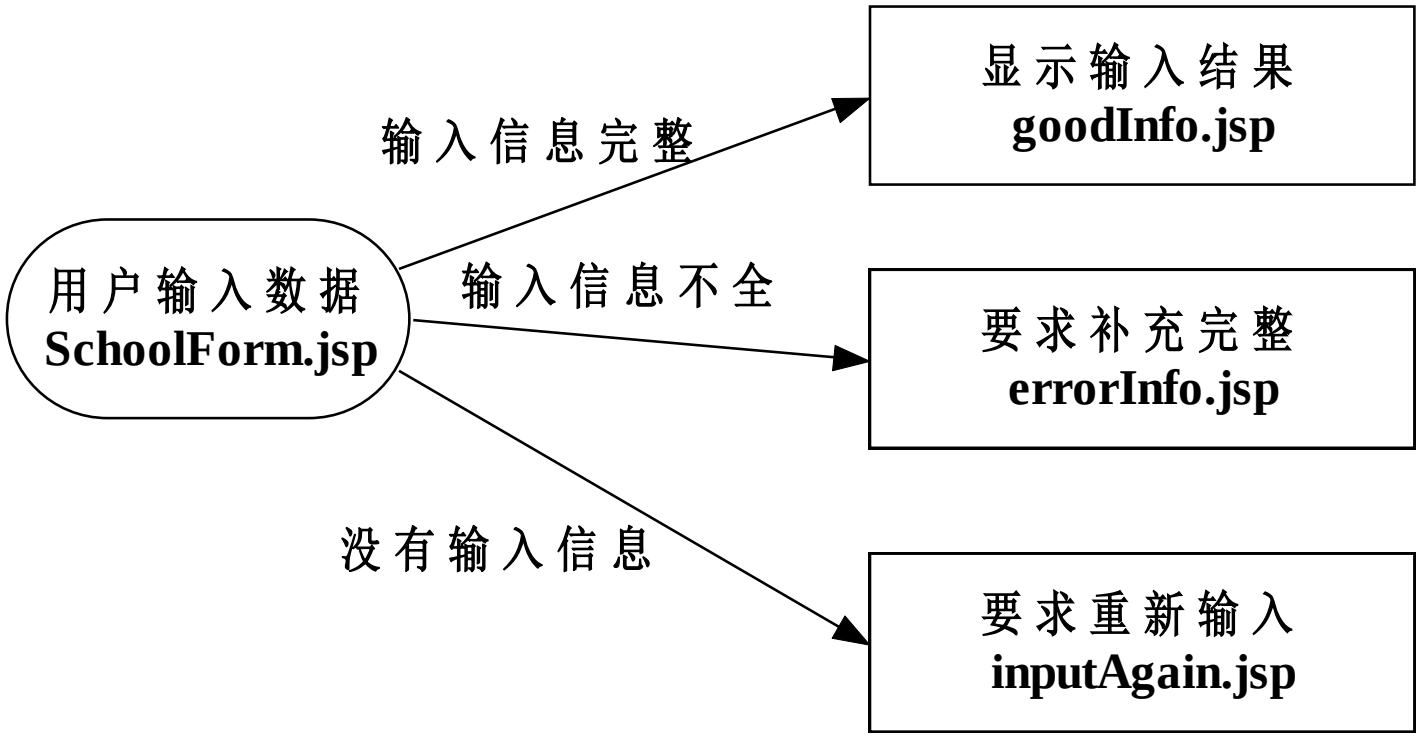
- 该servlet确定哪个JSP页面适合于处理当前的情形，并使用RequestDispatcher的forward方法将控制转移到那个页面。

6. 从bean中提取数据

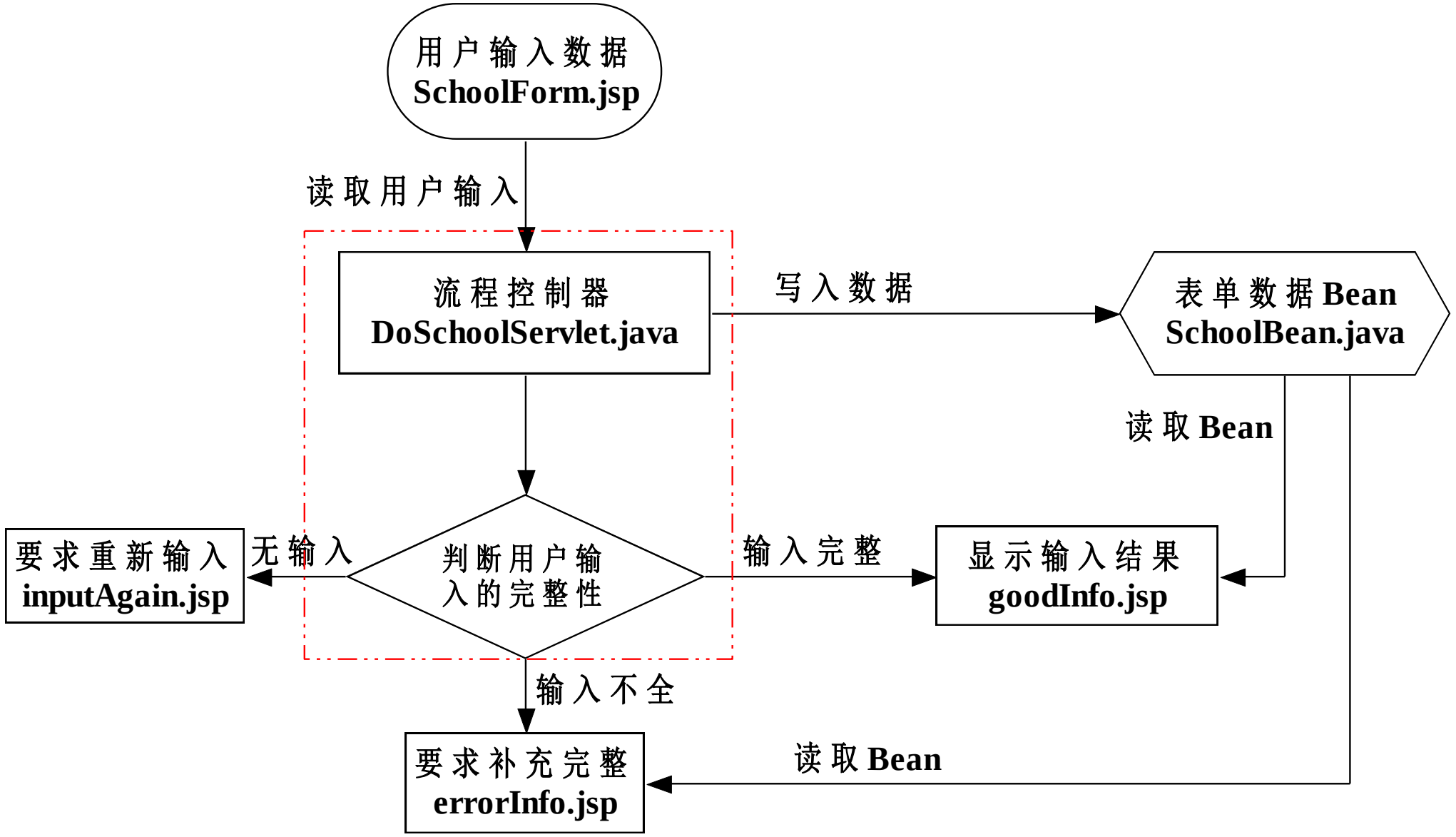
- JSP页面使用jsp:useBean和与第4步匹配的位置访问之前存储的bean，然后使用jsp:getProperty输出bean的属性。
- JSP页面并不创建或修改bean；它只是提取并显示由servlet创建的数据。

实例1

- **实例1 ----MVCTest工程**
  - 用户输入姓名和学校名称，根据用户输入的不同分别跳转到不同结果页面
  - 除了图中所示的四个视图文件外，还有
  - 处理表单数据的Bean: **mvc.SchoolBean.java**
  - 处理控制流程的：**DoSchoolServlet.java**



实例1的处理流程


用户输入数据
SchoolForm.jsp
读取用户输入
流程控制器
DoSchoolServlet.java
写入数据
表单数据Bean
SchoolBean.java
读取Bean
要求重新输入
inputAgain.jsp
无输入
判断用户输入的完整性
输入完整
显示输入结果
goodInfo.jsp
输入不全
要求补充完整
errorInfo.jsp
读取Bean

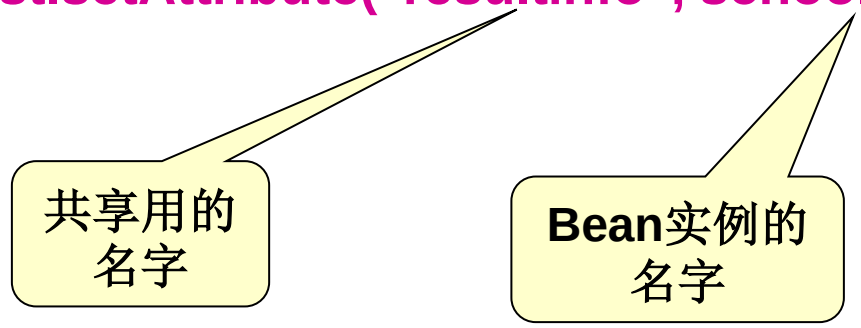
实例1的关键代码分析

- **schoolForm.jsp**
  - **<FORM ACTION="*DoSchoolServlet*" method="*post*" >**
  - 表单提交给控制文件***DoSchoolServlet.java***

实例1的关键代码分析

- **DoSchoolServlet.java（待续）**

1. 设置请求的编码格式，以兼容中文
   - ✓ request.setCharacterEncoding("utf-8");
2. 读取用户请求
   - ✓ String n=request.getParameter("name");
3. 创建数据Bean实例，并设置Bean属性
   - ✓ SchoolBean schoolbean=new SchoolBean();
   - ✓ schoolbean.setName(n);
4. 存储Bean，目的是和其他文件共享该Bean
   - ✓ request.setAttribute("resultinfo", schoolbean);



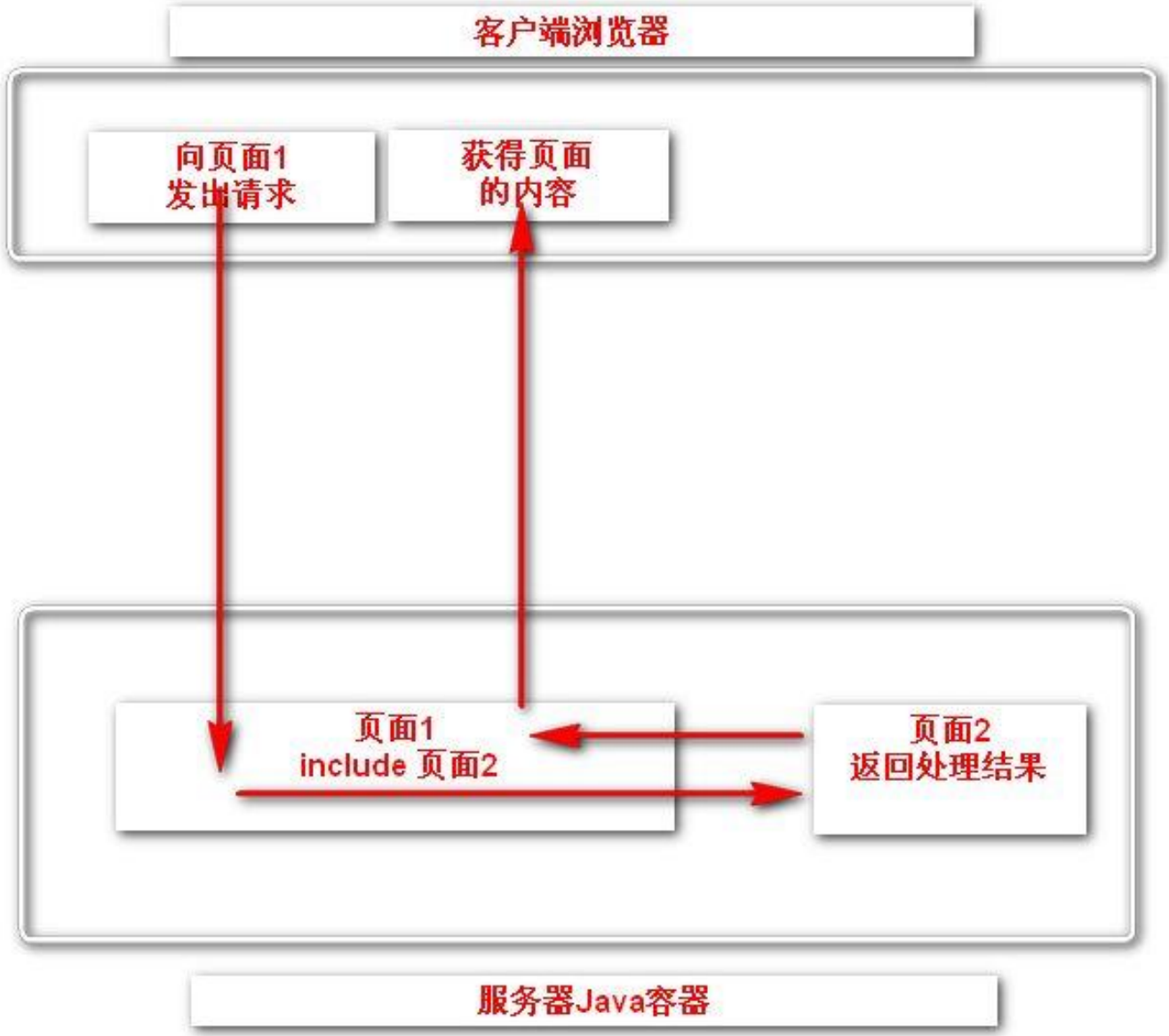
实例1的关键代码分析

- **DoSchoolServlet.java（续）**

5. 流程判断，以确定不同的转发结果地址
   - ✓ forwardUrl="/goodInfo.jsp";
6. 实现转发
   - ✓ RequestDispatcher dispatcher =request.getRequestDispatcher(forwardUrl);
   - ✓ dispatcher.forward(request, response);
   - ✓ 页面的转发对客户来说是透明的。初始的Servlet的URL是浏览器惟一知道的URL。

注意：第4步中， request.setAttribute(“resultinfo”, schoolbean);
默认把bean存储在request的上下文中，所以必须在第6步中使用RequestDispatcher 进行转发，这样目的网页才能共享到resultinfo这个Bean

request.getRequestDispatcher转发



实例1的关键代码分析

- goodInfo.jsp(输入完整的结果页面)
  - 声明要使用的Bean
    - ✓ <jsp:useBean id="resultinfo" type="wxy.beans.SchoolBean" scope="request" />
  - 读取Bean中的数据
    - ✓ <jsp:getProperty name="resultinfo" property="name" />
- erroInfo.jsp (输入不全的结果页面)
  - 声明要使用的Bean(同goodInfo.jsp文件)
  - 读取已有的Bean数据，放在输入框中
    - ✓ <INPUT TYPE="TEXT" NAME="name" VALUE='<jsp:getProperty name="resultinfo" property="name" />' >

jsp:useBean在MVC中与在独立JSP页面中的不同

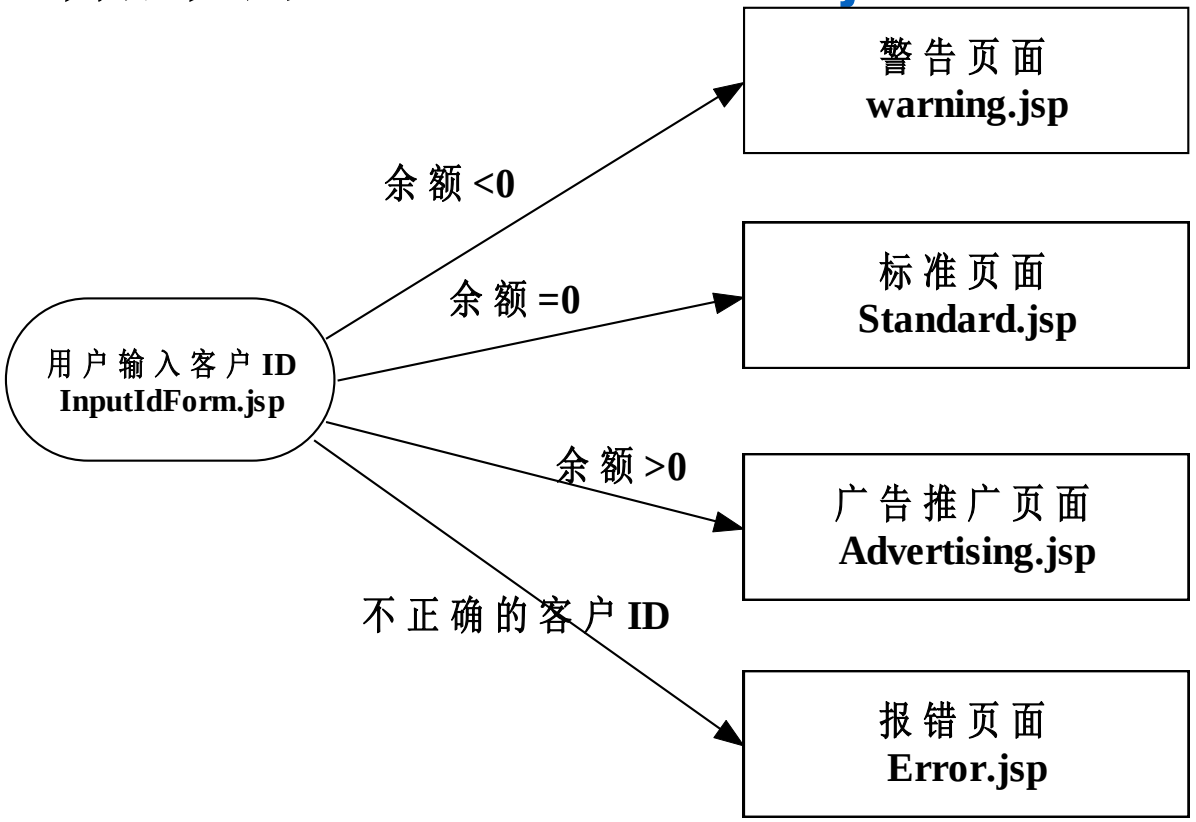
- **首先明确MVC架构的特点：JSP页面仅负责结果视图的展现，Servlet文件负责业务处理和流程控制**
  - JSP页面不应该创建对象
  - 应该由servlet，而非JSP页面，创建所有的数据对象。
- **因此，为了保证JSP页面不会创建对象，我们应该使用<jsp:useBean ... type=“package.Class” />，而不是<jsp:useBean ... class="package.Class" />**
  - <jsp:useBean ... type=“package.Class” />表示使用已经存在的Bean实例
- **JSP页面也不应该修改已有的对象**
  - 因此，jsp文件中只能使用jsp:getProperty，而不能使用jsp:setProperty。

```
<jsp:useBean id="resultinfo" type="wxy.beans.SchoolBean"
    scope="request" />
<p> 您的姓名是：<jsp:getProperty name="resultinfo" property="name" /> <p>
    您的学校是：
    <jsp:getProperty name="resultinfo" property="school" />
```

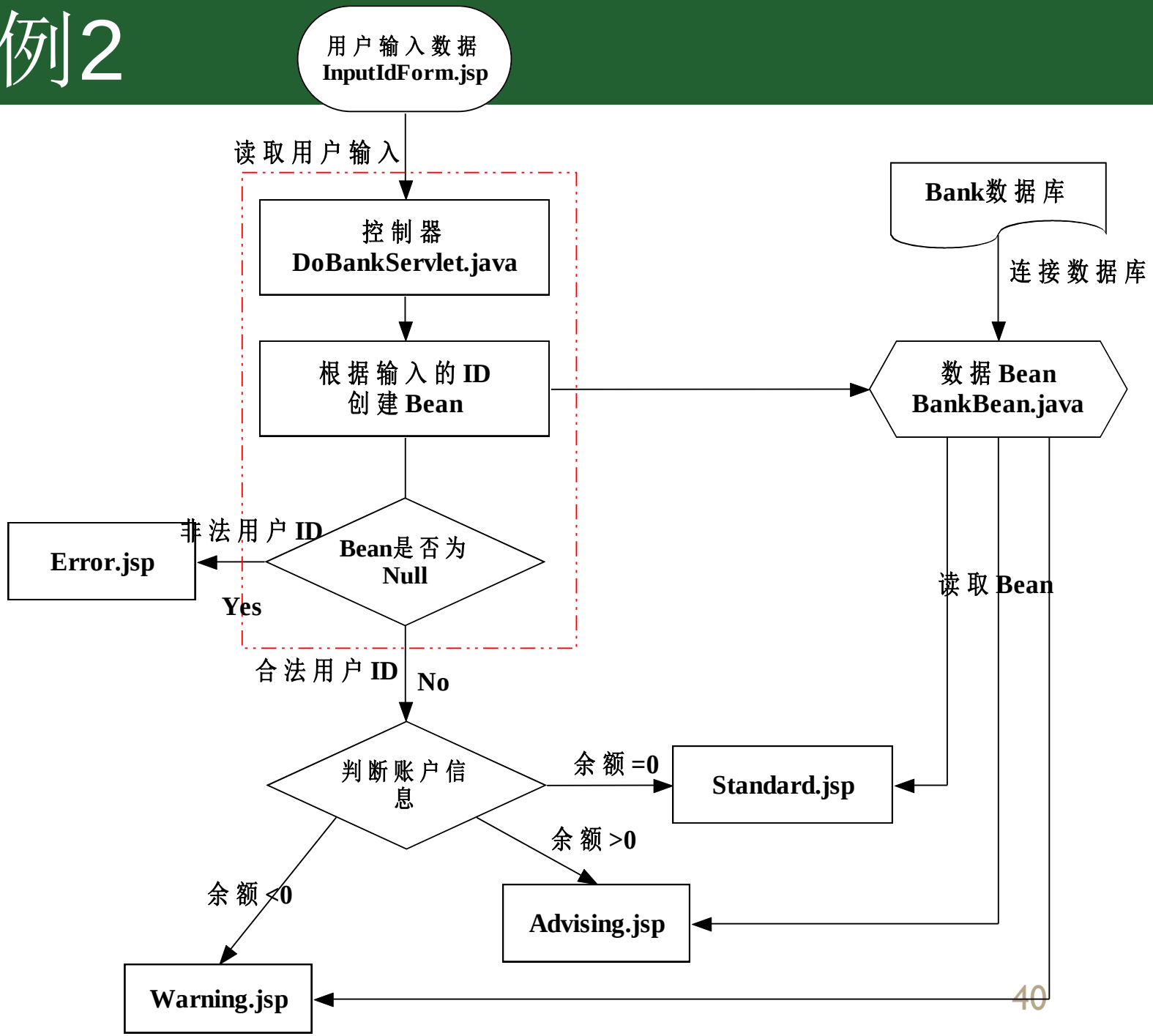
```
<jsp:useBean id="applicationBean" class="mvc.ApplicationBean"  scope="application" />
<jsp:setProperty name="applicationBean" property="*" />
<jsp:setProperty name="applicationBean" property="accesscount" value="1"  />
```

实例2

- 实例2 ----MVCTest工程
  - 根据用户银行账户余额，分别跳转到不同结果页面
  - 除了图中所示的四个视图文件外，还有
  - 处理表单数据的Bean: wxy.beans.BankBean.java
  - 处理控制流程的：DoBankServlet.java



实例2


用户输入数据
InputIdForm.jsp
读取用户输入
控制器
DoBankServlet.java
根据输入的ID
创建Bean
Bank数据库
连接数据库
数据Bean
BankBean.java
Bean是否为
Null
非法用户ID
Error.jsp
Yes
合法用户ID
No
读取Bean
判断账户信息
余额=0
Standard.jsp
余额>0
Advising.jsp
余额<0
Warning.jsp

MVC中bean的共享范围

jsp:useBean在MVC中的共享范围

- **request**
  - ✓ **<jsp:useBean id="..." type="..." scope="request" />**
- **session**
  - ✓ **<jsp:useBean id="..." type="..." scope="session" />**
- **application**
  - ✓ **<jsp:useBean id="..." type="..." scope="application" />**
- 没有**page**范围

MVC中基于request共享的bean

- **Servlet文件**
  - ✓ BeanObject value = new BeanObject(...);
  - ✓ value.setXxx(…);
  - ✓ request.setAttribute("key", value);
  - ✓ RequestDispatcher dispatcher =request.getRequestDispatcher("/SomePage.jsp");
  - ✓ dispatcher.forward(request, response);
- **JSP文件**
  - ✓ <jsp:useBean id="key" type="somePackage.ValueObject“ scope="request" />
  - ✓ <jsp:getProperty name="key" property="someProperty" />

MVC中基于session共享的bean

- **Servlet**
  - ✓ BeanObject value = new BeanObject(...);
  - ✓ value.setXxx(…);
  - ✓ HttpSession session = request.getSession();
  - ✓ session.setAttribute("key", value);
  - ✓ RequestDispatcher dispatcher =request.getRequestDispatcher(“/SomePage.jsp");
  - ✓ dispatcher.forward(request, response);
- **JSP**
  - ✓ <jsp:useBean id="key" type="somePackage.ValueObject" scope="session" />
  - ✓ <jsp:getProperty name="key" property="someProperty" />

先创建一个session对象，把bean实例添加为session对象的共享属性

Session共享中使用bean：另一种转发方式(待续)

- 使用response.sendRedirect(url地址)取代RequestDispatcher.forward(url地址)
- 差别：
  - 使用sendRedirect时
    - ✓用户可以看到JSP的URL（使用RequestDispatcher.forward时用户只能看到servlet的URL）
    - ✓客户程序要经过两次往返（而forward只需一次）
  - 转发目的URL地址表示上
    - ✓RequestDispatcher.forward中的url地址是以WebContent为当前目录的
    - ✓response.sendRedirect中的url地址是以ip地址为当前目录的
    - ✓例如：forwardUrl="/bankresult/Error.jsp
      - RequestDispatcher dispatcher =request.getRequestDispatcher(forwardUrl);
      - response.sendRedirect("/MVCTest/"+forwardUrl);

Session共享中使用bean：另一种转发方式(续)

sendRedirect——重定向
RequestDispatcher.forward——转发
转发不是重定向，转发是在Web应用内部进行的

转发对浏览器是透明的，也就是说，无论在服务器上如何转发，浏览器地址栏中显示的仍然是最初那个Servlet的地址

- sendRedirect的优点
  - 用户可以单独访问JSP页面
  - 用户能够保存JSP页面的地址
- sendRedirect的缺点
  - 由于用户可以在不首先经过servlet的情况下访问JSP页面，所以，JSP页面所需的数据有可能不存在。
  - 因此，JSP页面需要编写代码检查这种情况。
- RequestDispatcher.forward的特点
  - 页面的转发对客户来说是透明的。初始的Servlet的URL是浏览器惟一知道的URL。

MVC中基于application共享的bean

- **Servlet**
  - ✓ synchronized(this) {
  - ✓ BeanObject value = new BeanObject (...);
  - ✓ value.setXxx(…);
  - ✓ getServletContext().setAttribute("key", value);
  - ✓ }
  - ✓ RequestDispatcher dispatcher =request.getRequestDispatcher("/SomePage.jsp");
  - ✓ dispatcher.forward(request, response);
- **JSP**
  - ✓ <jsp:useBean id="key" type="somePackage.ValueObject"scope="application" />
  - ✓ <jsp:getProperty name="key" property="someProperty" />

同步多个用户对Bean的写入，以免出现冲突

通过getServletContext()方法获取application上下文，并把bean实例设置为application共享

小结：MVC架构中bean线程安全问题

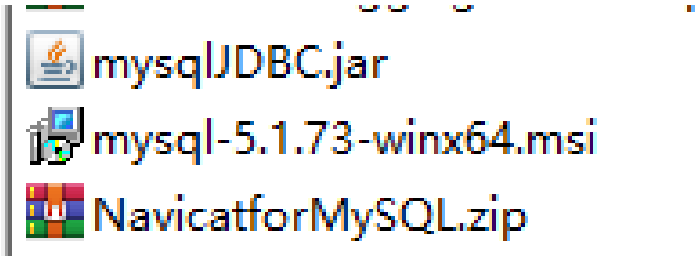
- **在独立JSP页面中**
  - 使用<jsp:useBean id="..." class="..." .../>创建Bean实例不会引发线程安全问题
  - 因为，系统不允许执行创建同id的Bean操作
- **在MVC架构中**
  - Bean实例由servlet创建
  - 当Bean的共享域为session和application时，多次同时访问Servlet就可能会引发线程安全问题
    - ✓例如：在session共享时，用户同时多次按“提交”，就会引发Servlet创建多次Bean的线程冲突
    - ✓例如：在application共享时，多个用户同时按“提交”或同时打开该Servlet页面，都可能引起线程冲突
    - ✓因此，在实际网站开发过程中session和application共享都需要使用线程同步块，以保证Bean的创建不出现冲突

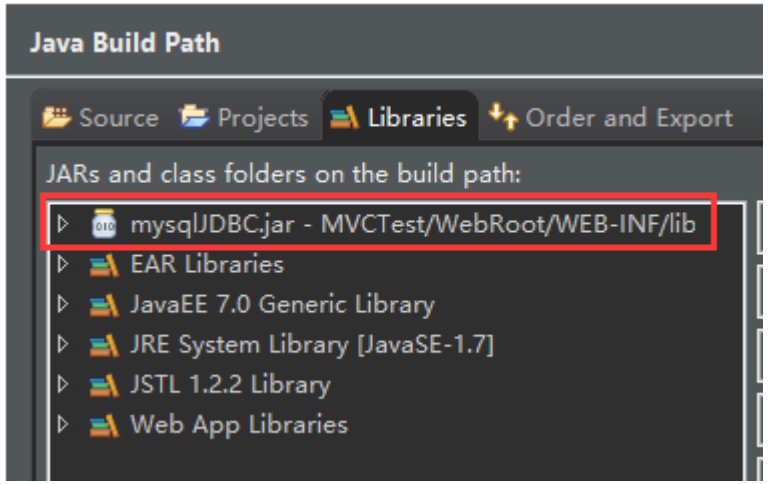
小结

- 掌握理解基于Servlet+JSP+Bean的MVC架构
  1. 定义用以表示数据的bean
  2. 使用一个servlet处理请求
  3. 填充bean
  4. 将bean存储在request、session或servlet的上下文中,方便共享
  5. 将请求转发到JSP页面
  6. 从bean中提取数据
- 理解bean在MVC架构中共享范围的设计

下次课准备

- MySQL数据库
- JDBC jar包
- Navicat






```
package wxy.beans;

import java.io.Serializable;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
```

感谢聆听

Thanks For Your Listening!