JSP/Servlet

讲义4

第10章JSP开发模式

JSP开发的两种模型

- **模型1的特点：**利用JSP+JavaBeans技术将页面显示和业务逻辑处理分开，JSP负责调用JavaBean中的功能然后把执行结果显示在JSP页面上

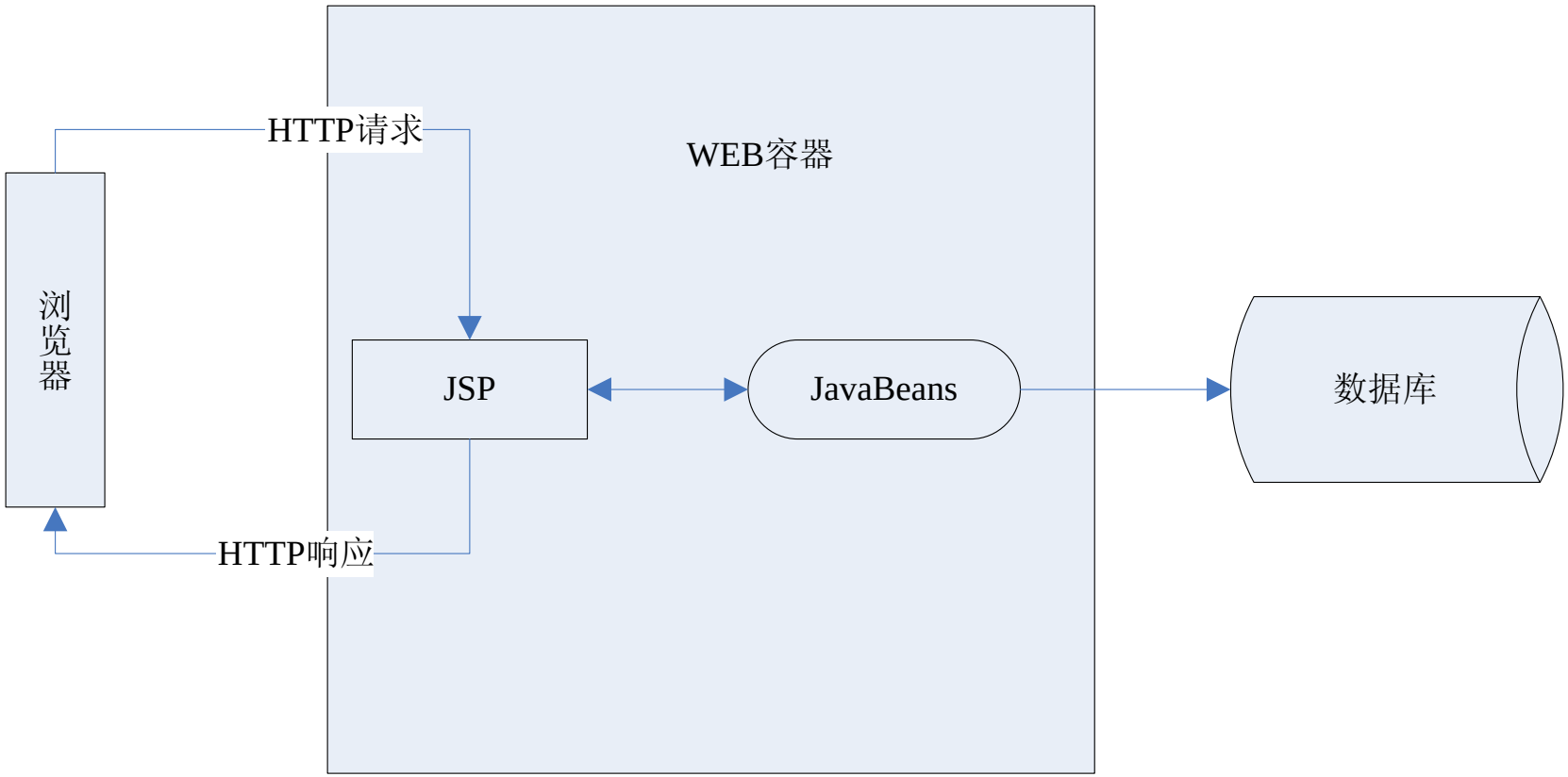
- **模型2的特点：**是三层架构的特点，MVC模型，即模型（model）——视图（view）——控制器（controller）的特点。

模型：代表的是业务逻辑

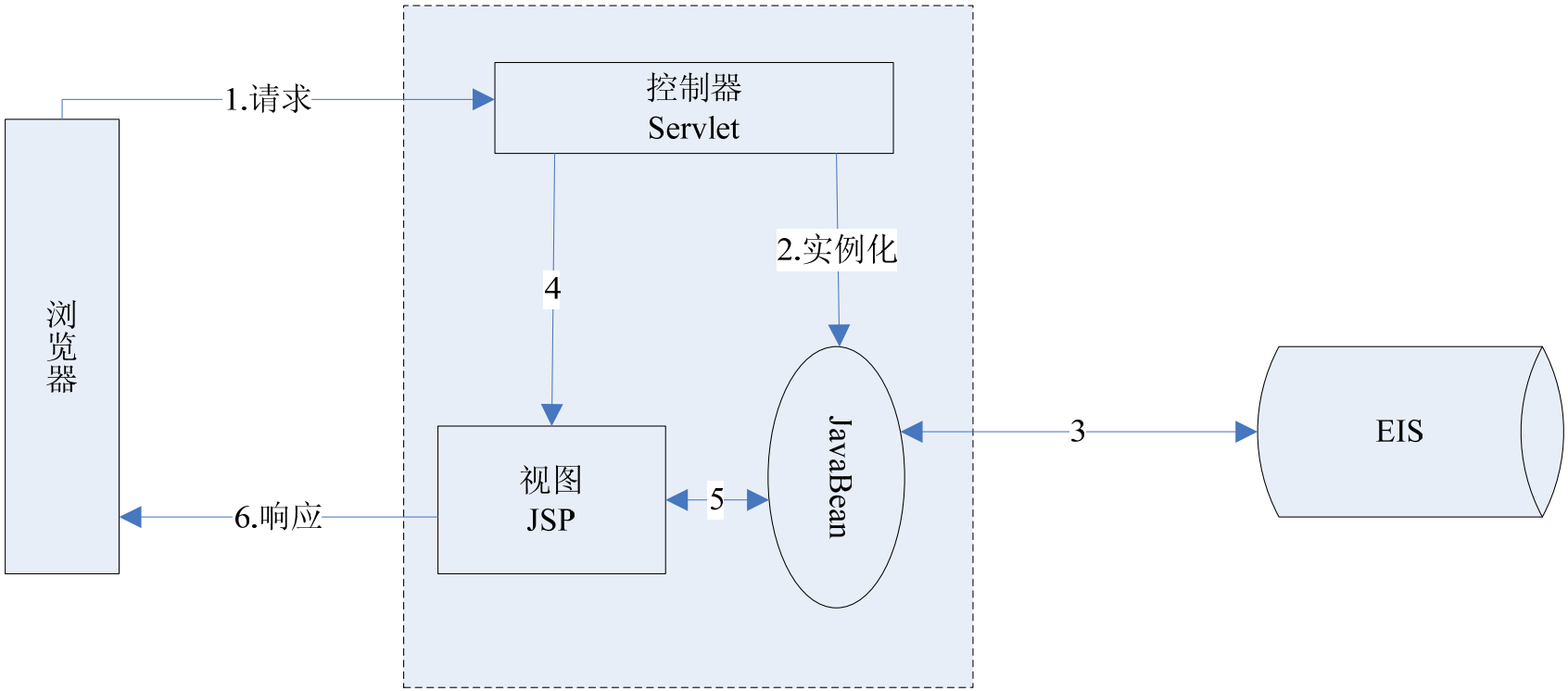
视图：用来表示数据

控制器：是模型和视图之间的桥梁，理论上只负责简单的业务逻辑，一般用来调用模型的业务逻辑以及转发到视图显示数据。

Model1图示



Model2图示



第11章 EL表达式语言

表达式语言（EL）

- EL的语法简单，使用方便。所有的EL表达式都是以“${”开始，以“}”结束
- 例如
  - ${sessionScope.user.name}
- 等价于以下语句

```
<%
User user=(User)session.getAttribute(“user”);
%>
<%=user.getName()%>
```

“[ ]”和“.”操作符

- EL中提供了.和[ ]两种操作符来存取数据。下列两种写法代表的意思是相同的：
- ${sessionScope.user.name}
- 等同于
- ${sessionScope[“user”][“name”]}

EL中的常用隐含对象

隐含对象	类型	说明
pageScope	java.util.Map	取得page范围内属性名称所对应的值
requestScope	java.util.Map	取得request范围内属性名称所对应的值
sessionScope	java.util.Map	取得session范围内属性名称所对应的值
applicationScope	java.util.Map	取得application范围内属性名称所对应的值
param	java.util.Map	如同request.getParameter(name)，返回String类型的值
paramValues	java.util.Map	如同request.getParameterValues(name)，返回String[]类型的值

范围相关的隐含对象

- pageScope
- requestScope
- sessionScope
- applicationScope
- 相当于范围对象的getAttribute方法
- 没有指定范围，默认从page到application的顺序依次查找，已找到第一个为准
- 如果找不到返回空字符串

范围相关的EL隐含对象示例

```
<%
//在4个不同范围中存储属性键值对
pageContext.setAttribute("user","tom");
request.setAttribute("user","jack");
session.setAttribute("user","rose");
application.setAttribute("user","smith");
%>
page范围：${pageScope.user}<br>
request范围：${requestScope.user}<br>
session范围：${sessionScope.user}<br>
application范围：${applicationScope.user}<br>
默认范围从page~application查找：${user}<br>
```

示例结果

执行后输出结果：

page范围：tom

request范围：jack

session范围：rose

application范围：smith

默认范围从page~application查找：tom

传递参数相关的EL隐含对象

- param
  - 相当于request.getParameter方法，如果参数不存在，返回空字符串
- paramValues
  - 相当于request.getParameterValues方法，返回一个字符串数组，参数不存在，返回空字符串

EL隐含对象pageContext用法

表达式	说明
${pageContext.request.queryString}	请求参数字符串，例如 username=tom&age=20
${pageContext.request.requestURL}	请求URL，例如 http://127.0.0.1:7777/TestJSP/index.jsp
${pageContext.request.contextPath}	WEB应用名
${pageContext.request.remoteAddr}	客户端IP

第12章 JSTL

JSTL（JSP Standard Tag Library ）

- JSP标准标签库是一个开源的标签库，最新版本是1.2
- 可以取代传统的在页面中嵌入java程序的做法，提高程序的可读性和维护性
- 需要把jstl-1.2.jar文件拷贝到WEB应用的WEB-INF/lib目录
- 在JSP页面中需要加入taglib指令
  - <%@ taglib uri=“标签库URI” prefix=“标签前缀"%>
  - 例如

    <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

JSTL标签库的分类

JSTL标签分类	前缀名称	URI	范例
核心标签库	c	http://java.sun.com/jsp/jstl/core	<c:out>
I18N格式标签库	fmt	http://java.sun.com/jsp/jstl/fmt	<fmt:formatDate>
SQL标签库	sql	http://java.sun.com/jsp/jstl/sql	<sql:query>
XML标签库	x	http://java.sun.com/jsp/jstl/xml	<x:forBach>
函数标签库	fn	http://java.sun.com/jsp/jstl/functions	fn:split

<c:out>

- 显示数据，和<%=……%>功能相似
- 示例

```
<c:out value="hello,china"/>
<hr>
<c:out value="${param.str}">hello,world</c:out>需要传递参数值
<hr>
<c:out value="<h1>hello,world</h1>"/>
<hr>
<c:out value="<h1>hello,world</h1>" escapeXml="false"/>
```

<c:out>属性说明

名称	说明	EL	类型	必须	默认值
value	需要显示的值	Y	Object	是	无
default	如果value为null，则显示default的值	Y	Object	否	无
escapeXml	是否转换特殊字符， 如：<转换为<	Y	boolean	否	true

<c:if>

- 用途和我们Java程序中的if语句基本相同，只是没有对应的else标签

- 示例：传入一个username参数，判断值是否等于king，如果是打印“你好，老大”

```
<!--此处字符串可以用单引号 -->
<c:if test="${param.username=='king'}" var="result" scope="request">
    你好，老大
</c:if>
<br>
表达式结果：${requestScope.result}
```

<c:if>属性说明

名称	说明	EL	类型	必须	默认值
test	如果表达式的结果为true，则执行本体内容，false则不执行	Y	boolean	是	无
var	用来存储test运算的结果	N	String	否	无
scope	var变量的JSP范围	N	String	否	page

<c:choose>

- 类似java中的if elseif或switch 语句，只能和<c:when>和<c:otherwise>配合使用

- 示例：传入一个成绩数字，根据数字判断返回“优秀”“一般”“差劲”

```
<c:choose>
    <c:when test="${param.num>=85}">
        优秀
    </c:when>
    <c:when test="${param.num>=60}">
        一般
    </c:when>
    <c:otherwise>
        差劲
    </c:otherwise>
</c:choose>
```

<c:forEach>

- 为循环控制标签，可以遍历集合，数组的元素
- 有两种典型用法
  - 遍历集合或数组元素
  - 进行一个计数循环

<c:forEach>标签的属性说明

名称	说明	EL	类型	必须	默认值
var	用来存放现在指到的成员	N	String	否	无
items	被迭代的集合对象	Y	Arrays Collection Iterator Enumeration Map String	否	无
varStatus	用来存放现在指到的成员的相关信息	N	String	否	无
begin	开始位置	Y	int	否	0
end	介绍位置	Y	int	否	最后一个成员
step	每次迭代的间隔数	Y	int	否	1

遍历集合或数组元素

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%
//创建一个字符串数组
String[] names = new String[] {"tom","jack","rose","smith","john"};
//将数组保存到request范围中
request.setAttribute("names",names);
%>
<!-- 遍历所有数组元素 -->
<c:forEach items="${names}" var="item">
    <c:out value="${item}"/>
</c:forEach>
```

进行一个计数循环

```
<c:forEach begin="1" end="10" var="i">
    <c:out value="${i}"/> <br>
</c:forEach>
```

varStatus属性

- 主要用来存放现在指到成员的相关信息。主要有以下属性

属性	类型	说明
index	number	现在指到成员的索引
count	number	总共指到成员的总数
first	boolean	现在指定的成员是否为第一个成员
last	boolean	现在指到的成员是否为最后一个成员

varStatus属性的使用

```
<%
//创建一个字符串数组
String[] names = new String[] {"tom","jack","rose","smith","john"};
//将数组保存到request范围中
request.setAttribute("names",names);
%>

<c:forEach items="${requestScope.names}" var="item"
    varStatus="status">
    第${status.count}个， 下标：${status.index}，元素： ${item}
    第一个${status.first }，最后一个${status.last}<br>
</c:forEach>
```

<c:import>

- 类似jsp中的动作元素<jsp:include>，但是区别在于，<c:import>可以包含不同的web应用中的文件
- 常用属性：
  - url 被包含文件的URL
  - charEncoding 被包含的文件的编码方式
- 示例代码：演示包含其他站点内文件和本WEB应用的文件
- <c:import url="http://www.baidu.com" charEncoding="gb2312"/>

<c:url>

- 产生一个URL，在禁用cookie的情况下，自动生成带JSESSIONID的URL
- 主要属性：
  - value 产生的URL地址

- 示例代码：禁用coolie后（其实不禁用也能看到效果）链接地址会自动带jsessionid。必须是链接同一个WEB应用的URL
  - <a href="<c:url value="c_out.jsp"/>">打开
  - 执行后的html源代码：
  - 打开

自定义JSP标签

- 一个标签的创建只要提供一个类，然后在tld文件中设置一下这个类就是一个tag，然后在jsp文件中引用使用。
- 可以分成三个步骤
  - 首先创建一个类，这个类必须继承TagSupport
  - 配置tld文件，在WEB-INF目录中，或者在WEB-INF的子目录中创建一个tld文件，配置标签
  - 在JSP文件中使用这个标签

步骤1举例

```
package mytag;
import java.io.IOException;
import javax.servlet.jsp.JspWriter;
import javax.servlet.jsp.JspException;
import javax.servlet.jsp.tagext.TagSupport;
```

```
public class ShowTxtTag extends TagSupport{
    //声明txt属性，这也是标签中的txt属性
    private String txt;

    //设置属性值,必须按照标准set方法命名，用在标签中设置属性值
    public void setTxt(String txt) {
        this.txt = txt;
    }

    //当执行到标签>结束时执行的方法，打印增加<h1></h1>的文本
    public int doEndTag() throws JspException {
        //获得JSP输出流对象
        JspWriter out = pageContext.getOut();
        //输出数据到JSP页面上
        try {
            out.print("<h1>" + this.txt + "</h1>");
        } catch (IOException e) {
            throw new JspException(e);
        }
        //这个常量表示继续执行下面的内容
        return EVAL_PAGE;
    }
}
```

步骤2举例

```
<!-- /WEB-INF/mytag.tld  -->

<?xml version="1.0" encoding="UTF-8"?>
<taglib>
    <tlib-version>1.0</tlib-version>
    <jsp-version>2.0</jsp-version>
    <short-name>my</short-name>
    <tag>
        <name>show</name>
        <tag-class>mytag.ShowTxtTag</tag-class>
        <body-content>empty</body-content>
        <attribute>
            <name>txt</name>
            <required>true</required>
            <rtexprvalue>true</rtexprvalue>
        </attribute>
    </tag>
</taglib>
```

步骤2解释

```
<taglib> 根标签
    <tlib-version>版本写1.0就可以了</tlib-version>
    <jsp-version>JSP版本写1.2或2.0都行</jsp-version>
    <short-name>前缀。比如“c:out value=""/” 里的“c” </short-name>
    <tag> 具体标签设置
        <name>tag的名字。例如“c:out value=""/” 里的"out”</name>
        <tag-class>指向的完整类名mytag.ShowTxtTag</tag-class>
        <body-content>标签之间是否有内容empty表示无内容</body-content>
        <attribute> 属性设置
            <name>属性名txt</name>
            <required>是否是必须设置的属性</required>
            <rtexprvalue>属性值是否支持EL表达式</rtexprvalue>
        </attribute>
    </tag>
</taglib>
```

步骤3举例

```
<%@ page language="java" pageEncoding="UTF-8"%>
<%@ taglib uri="/WEB-INF/mytag.tld" prefix="my"%>
<html>
    <head>
        <title>自定义标签使用</title>
    </head>
    <body>
        <my:show txt="hello"/>
    </body>
</html>
```

本课程结束

- 谢谢大家
  - 记得给我写信啊