**实验十一 JSTL与自定义标签**

一、实验目的

1. 了解JSTL的基本概念；

2. 掌握JSTL的核心库标签的使用；

3. 掌握使用SimpleTag和SimpleTagSuppot开发简单标签；

4. 了解传统自定义标签的接口和类的使用。

二、实验原理

JSP标准标签库（JSP Standard Tag Library，简称JSTL）是一个为实现Web应用程序的常用功能而开发的标签库集合。功能包括基本输入输出、流程控制、XML文件解析、数据库查询、国际化和文本格式化等。JSTL主要由核心标签库、XML标签库、格式化/国际化标签库、数据库标签库以及函数标签库组成。

自定义标签（custom tag）是JSP 1.1技术规范引进的，它也常称为自定义动作。自定义标签实际上是实现了某些特定接口的Java类。在JSP 1.1规范中提供了3个接口和2个实现类，它们是：Tag、IterationTag和BodyTag接口与TagSupport和BodyTagSupport实现类。这些API一般称为传统的标签扩展API。

为了简化标签库的开发，JSP 2.0在标签扩展API中增加了一个SimpleTag接口以及它的一个实现类SimpleTagSupport。使用它们会使开发自定义标签大大简化。创建和使用自定义标签一般包含下面三个步骤：

【步骤1】创建标签处理类(tag handler)；

【步骤2】创建标签库描述文件（TLD）；

【步骤3】在JSP文件中引入标签库和使用标签。

三、实验内容与步骤

**1、JSTL的安装和测试【10分】**

【步骤1】安装JSTL库。可以在Apache网站下载最新版本的JSTL，或者是在Tomcat示例应用程序中获得JSTL，它位于<CATALINA\_HOME>\webapps\examples\WEB-INF\lib目录中，共包含两个文件：jstl.jar和standard.jar。将这两个文件复制到Web应用程序的WEB-INF\lib目录中，完成JSTL的安装。

1. **效果截图【2分】**

【步骤2】完成jstl\_test.jsp页面，使用核心标签库实现显示"Welcome to using JSTL"、获取并显示请求方法以及主机IP地址。（8分）

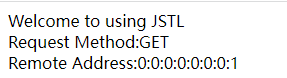


图12.1 jstl\_test.jsp页面的运行结果

1. **代码**

<%@ page contentType="text/html;charset=gb2312"%>

**<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>【2分】**

<html>

<head>

<title>Using JSTL</title>

</head>

<body>

**<c:out value="Welcome to using JSTL" />【2分】**

**<br> Request Method:<c:out value="${pageContext.request.method}"/>【2分】**

**<br> Remote Address:<c:out value="${pageContext.request.remoteAddr}"/>【2分】**

</body>

</html>

说明：使用JSTL我们不用建立标签库描述文件，因为该文件已经包含在 standard.jar文件中了，JSP容器可以找到该文件。另外，也可以不用配置web.xml文件，因为JSP容器使用隐含映射的方式可以将标准的URI映射到适当的标签库上。

**2、表达式标签<c:out>、<c:set>和<c:remove>的使用（25分）**

【步骤1】完成core\_expression.jsp，实现在标签体内分别设置页面作用域、请求作用域以及会话作用域中变量number的值分别为5、15和25。（10分）

**（1）代码**

<%@ page contentType="text/html;charset=GB2312"%>

**<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>【1分】**

<html>

<head>

<title>表达式相关的标签</title>

</head>

<body>

<h2>使用<c:out value="<c:out> <c:set> <c:remove>" />的例子

</h2>

<hr>

**<c:set scope="page" var="number">【2分】**

**<c:out value="${5}" />【1分】**

**</c:set>**

<br>

**<c:set scope="request" var="number">【2分】**

**<c:out value="${15}" />【1分】**

**</c:set>**

<br>

**<c:set scope="session" var="number">【2分】**

**<c:out value="${25}" />【1分】**

**</c:set>**

</body>

</html>

【步骤2】在步骤1的基础上，使用<c:out>标签设置不同作用域的number变量的初始值，并输出页面作用域、请求作用域以及会话作用域中number变量的值。

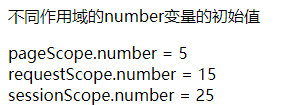


图 core\_expression.jsp

**（2）核心代码**（6分）

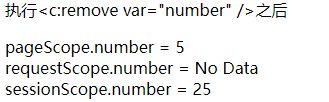
<p>不同作用域的number变量的初始值</p>

**pageScope.number =<c:out value="${pageScope.number}" default="No Data" /><br>【2分】**

**requestScope.number =<c:out value="${requestScope.number}" default="No Data" /><br> 【2分】**

**sessionScope.number =<c:out value="${sessionScope.number}" default="No Data" /><br>【2分】**

【步骤3】在步骤2的基础上，使用<c:remove>标签从请求作用域中删除变量number，并输出页面作用域、请求作用域以及会话作用域中number变量的值。



**（3）核心代码（9分）**

**<p><c:out value='执行<c:remove var="number" />之后' /></p>【1分】**

**<c:remove var="number" scope="request" />【2分】**

**pageScope.number = <c:out value="${pageScope.number}" default="No Data" /><br> 【2分】**

**requestScope.number = <c:out value="${requestScope.number}" default="No Data" /><br> 【2分】**

**sessionScope.number = <c:out value="${sessionScope.number}" default="No Data" /><br>【2分】**

**3、流程控制标签的使用**（**15分）**

【步骤1】完成flowControl.html，提供用户输入用户名（Tom\Jerry\Mike\Ben之一），提交成功后将请求发送至flowControl.jsp。

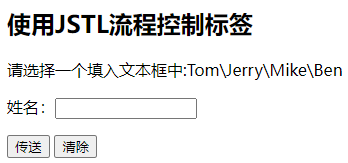


图 flowControl.html的运行效果

**（1）代码**

<html>

<head>

<title>JSTL中用于流程控制的标签</title>

<meta charset="UTF-8">

</head>

<body>

<h2>使用JSTL流程控制标签</h2>

请选择一个填入文本框中:Tom\Jerry\Mike\Ben

**<form method="post" action="flowControl.jsp">【2分】**

<p>

姓名：**<input type="text" name="username" size="15" />【1分】**

</p>

<p>

**<input type="submit" value="传送" /> 【1分】**

**<input type="reset" value="清除" />【1分】**

</p>

</form>

</body>

</html>

**（2）运行截图**

【步骤2】完成flowControl.jsp，分别利用<c:if>标签和<c:choose>标签实现对用户输入内容的判定，并输出“您好，XXX”（XXX为用户输入的姓名）

在文本框中输入Tom后，点击“传送”按钮，运行效果如图11- 所示。

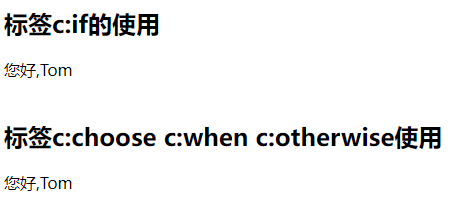


图11.3 flowControl.jsp页面的运行结果

**（3）代码**（10分）

<%@ page contentType="text/html;charset=GB2312"%>

**<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>【1分】**

<html>

<head>

<title>JSTL中用于流程控制的标签</title>

</head>

<body>

<h2>标签c:if的使用</h2>

**<c:if test="${param.username == 'Tom'}" var="condition" scope="session">您好,Tom</c:if>【1分】**

**<c:if test="${param.username == 'Jerry'}" var="condition"scope="session">您好,Jerry</c:if>【1分】**

**<c:if test="${param.username == 'Mike'}" var="condition"scope="session">您好,Mike</c:if>【1分】**

**<c:if test="${param.username == 'Ben'}" var="condition" scope="session">您好,Ben</c:if>【1分】**

<br>

<br>

<h2>标签c:choose c:when c:otherwise使用</h2>

**<c:choose>【1分】**

**<c:when test="${param.username == 'Tom'}">您好,Tom</c:when>【1分】**

**<c:when test="${param.username == 'Jerry'}">您好,Jerry</c:when>【1分】**

**<c:when test="${param.username == 'Mike'}">您好,Mike</c:when>【1分】**

**<c:otherwise>您好,Ben </c:otherwise>【1分】**

</c:choose>

</body>

</html>

**（4）运行截图**

**4、URL相关标签的使用**（15分）

【步骤1】完成included.jsp，本页面是被包含的页面，页面中显示“这是被包含页面中的内容”，利用<c:out>标签实现输出参数用户名userName的值。（5分）

**（1）代码**

<%@ page contentType="text/html;charset=gb2312"%>

**<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>【2分】**

<html>

<head>

<title>Included Page</title>

</head>

<body>

<h3>这是被包含页面中的内容</h3>

**用户名：<c:out value="${param.userName}" />【3分】**

</body>

</html>

【步骤2】完成import\_test.jsp，本页面是是包含页面，利用<c:import>包含included.jsp页面，同时指定请求参数userName的值为Harry Porter。（5分）

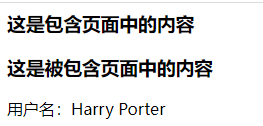


图 import\_test.jsp的运行效果

**（2）代码**

<%@ page contentType="text/html;charset=gb2312"%>

**<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>【1分】**

<html>

<head>

<title>Using URL JSTL</title>

</head>

<body>

<h3>这是包含页面中的内容</h3>

**<c:import url="included.jsp" charEncoding="gb2312">【2分】**

**<c:param name="userName" value="Harry Porter" />【2分】**

**</c:import>**

</body>

</html>

【步骤3】完成redirect\_test.jsp，利用重定向标签<c:redirect>实现定向至import\_test.jsp，同时指定请求参数userName的值为哈里.波特。（5分）

**（3）代码**

<%@ page contentType="text/html;charset=gb2312"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>**【1分】**

<html>

<head>

<title>Using Redirect Tag</title>

</head>

<body>

<h3>使用redirect标签</h3>

**<c:redirect url="import\_test.jsp">【2分】**

**<c:param name="userName" value="哈里.波特" />【2分】**

**</c:redirect>**

</body>

</html>

**5、带有属性标签的开发**（35分）

**实现带有属性的标签的开发。该标签带有一个double型属性，标签功能是求属性x的平方根。**

**【步骤1】完成MathTag.java。在taglib-demo项目的com.mytag包中创建MathTag类，它继承SimpleTagSupport类。**（10分）

1. 代码

package com.mytag;

import java.io.IOException;

import javax.servlet.jsp.JspException;

import javax.servlet.jsp.JspWriter;

import javax.servlet.jsp.tagext.SimpleTagSupport;

**public class MathTag extends SimpleTagSupport {【2分】**

double x;

**public void setX(String x) {【2分】**

double num = 0;

try {

**num = Double.parseDouble(x);【2分】**

} catch (NumberFormatException e) {

// TODO: handle exception

}

this.x = num;

}

@Override

**public void doTag() throws JspException, IOException {【2分】**

// TODO Auto-generated method stub

JspWriter out = getJspContext().getOut();

**out.print(x + "的平方根是：" + Math.sqrt(x));【2分】**

}

}

【步骤2】完成mytaglib.tld，实现自定义标签sqrt。（15分）

<?xml version="1.0" encoding="UTF-8" ?>

<taglib xmlns="http://java.sun.com/xml/ns/j2ee"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-jsptaglibrary\_2\_0.xsd"

version="2.0">

<tlib-version>1.0</tlib-version>

<short-name>TagExample</short-name>

**<uri>http://www.mydomain.com/sample</uri>【2分】**

<tag>

<description>Compute Square Root</description>

**<name>sqrt</name>【2分】**

**<tag-class>com.mytag.MathTag</tag-class>【2分】**

<body-content>empty</body-content>

**<attribute>【2分】**

**<name>x</name>【2分】**

**<required>true</required>【2分】**

</attribute>

</tag>

</taglib>

【步骤3】完成math.jsp页面，使用sqrt标签实现求平方根。（10分）

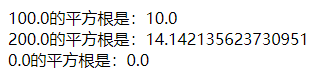


图 math.jsp的运行效果

（3）代码

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

**<%@taglib uri="http://www.mydomain.com/sample" prefix="demo"%>【2分】**

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>求平方根</title>

</head>

<body>

**<demo:sqrt x="100" /> <br>【2分】**

**<demo:sqrt x="200" /> <br>【2分】**

**<demo:sqrt x="0" /> <br>【2分】**

</body>

</html>