《JavaEE 实验手册》

Java 教研室

版 本 1.3

文档提供：Java 教研室 孙丽萍

修 改 记 录

修改时间	修改人	修改内容
2009.8.30	孟双英	文档创建
2012.8.27	张立飞	修改
2016.3.7	孙丽萍	修改

目录

第八章 JSP 基础

一、内容概述

本章的教学内容是介绍 JSP 的页面组成元素---6 种标签分别是：声明<%! %>、脚本<% %>、表达式<%= %>、动作<jsp: >、伪指令<%@page %>、注释<%-- --%>，介绍 JSP 生命周期的 7 个阶段分别是：转换、编译、装载、创建、初始化、服务、销毁，详细介绍伪指令的各个属性以及重点区分动作指令 include 和伪指令 include。

通过本章实验应该达到的目的：掌握 JSP 页面的 6 种标签的使用，理解 JSP 生命周期各个阶段，能利用 JSP 相关基础知识编写 JSP 程序，实现与 Servlet 相同的功能。

二、实验一 利用 JSP 的 6 种标签，循环输出彩色条，输出系统时间

2.1 实验目的

利用 JSP 的 6 种标签，创建一个 JSP 程序

实现功能：完成循环输出彩色条，并输出系统时间。

2.2 准备

在进行此实验前首先保证本地机器上配置好了正确的 Java 开发环境(JDK 环境)，已安装 Web 服务器 Tomcat，如果以上设置不正确，请参考《Tomcat 安装手册》进行正确设置。

2.3 实验步骤

步骤一：

在 Eclipse 中新建 Dynamic project “03_tag”，在“03_tag/WebContent”目录下新建 index.jsp，打开 index.jsp 如下图

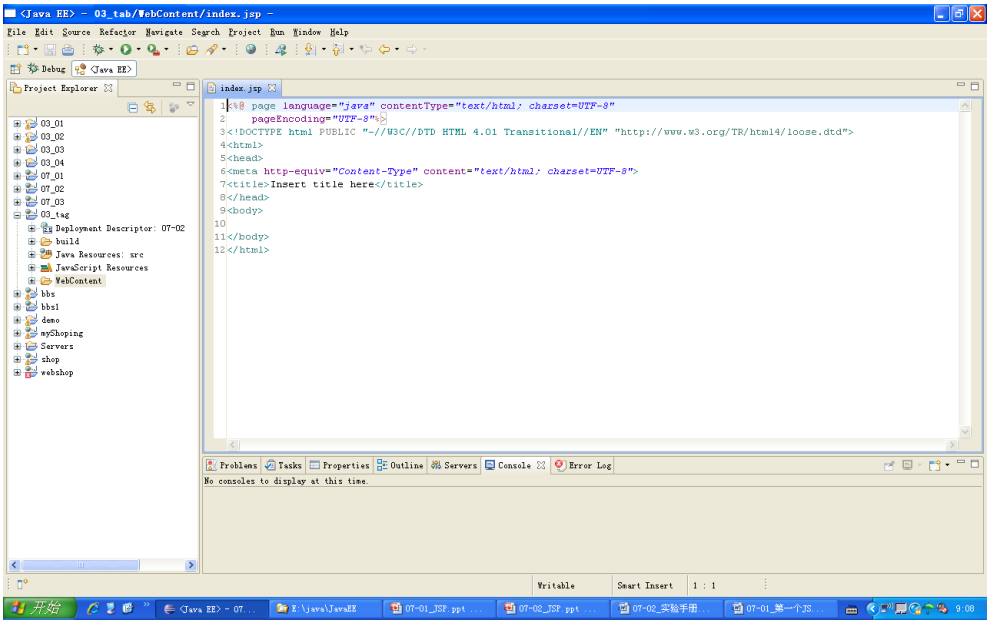

图 2-1

步骤二：

修改 index.jsp 内容，利用 page 指令的 info 属性给出页面说明，声明一个 getDate()方法获取当前系统时间，声明一个变量 count 记录颜色的条数初始值为 10

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ page info="a classic JSP"%>
<%@ page import="java.util.*"%>
<%-- This is a classic JSP ,it contain all elements of JSP --%>
<%!String getDate() {
        return (new Date()).toLocaleString();
    }

    int count = 10;
%>
```

图 2-2

步骤三：

在 index.jsp 的 body 标签内编写输出彩条程序，利用<table></table>标签输出彩条，利用 JSP 脚本定义彩条的颜色 color1,color2，循环输出 color1,color2 两种颜色

```
<table>
    <tr bgcolor="777777">
        <td>----------------------------------</td>
    </tr>
    <%
        int i;
        //color
        String color1 = "99ccff";
        String color2 = "88cc33";
        for (i = 1; i <= count; i++) {
            String color = "";
            if (i % 2 == 0)
                color = color1;
            else
                color = color2;
            out.println("<tr bgcolor='" + color
                    + "'><td>----------------------------------</td></tr>");
        }
    %>
</table>
```

图 2-3

步骤四：

完成彩条输出后，利用 JSP 表达式调用声明的 getDate()方法输出系统时间

```
current time:
<%-- expression --%>
<%=getDate() %>
```

图 2-4

至此完成 index.jsp 程序的编写

步骤五：

在 Eclipse 中运行“03_tag”项目，在浏览器中输入“http://localhost:8080/03_tag”,回车，程序运行效果图如下：

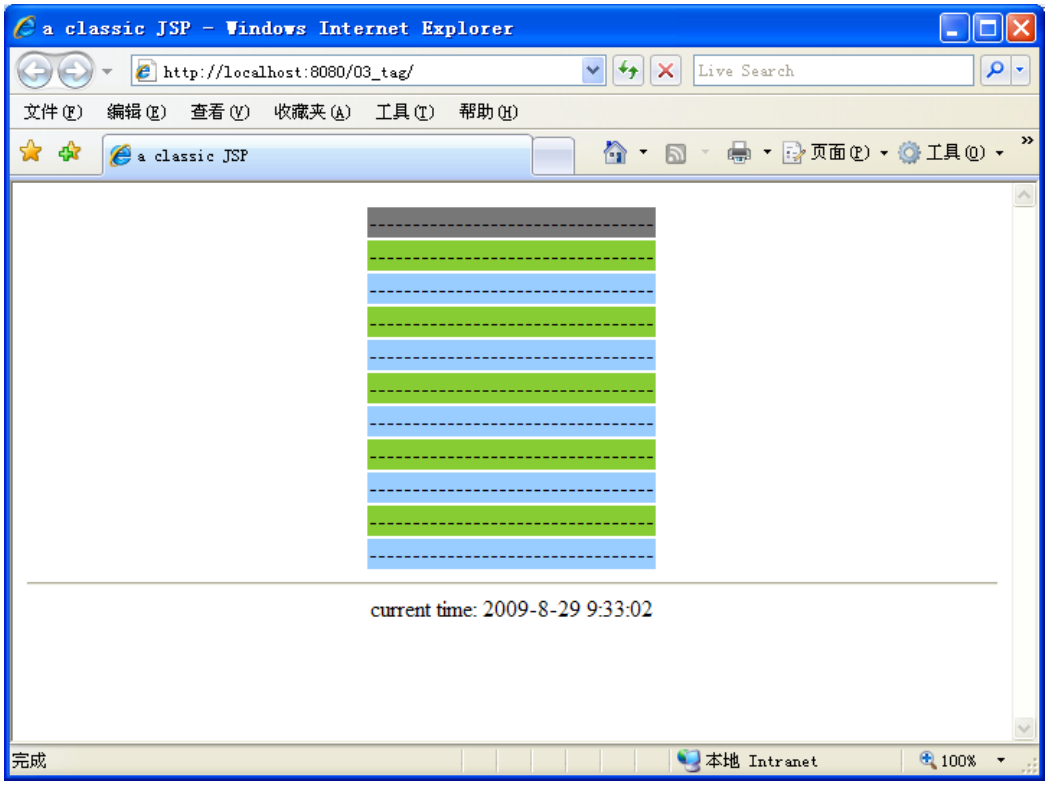

图 2-5

2.4 实验结论

通过试验，成功在浏览器中输出彩条，并显示系统当前时间，整个 index.jsp 程序的编写过程中，熟悉了 JSP 的伪指令中 info 以及 import 属性、声明、脚本、表达式、注释等标签的使用

三、实验二 练习 JSP 中 page 伪指令属性的使用

3.1 实验目的

使用 Eclipse 编写 JSP 程序，练习 JSP 中 page 伪指令的各个属性的使用
实现功能：编写 page.jsp 程序，利用 page 伪指令指定页面的属性

3.2 准备

保证 Eclipse 已经正确配置 Tomcat 服务器，可以进行 JavaWeb 项目编写、编译与调试。（如果以上设置不正确，请参考《Eclipse 部署配置 Tomcat 手册》进行正确设置）

3.3 实验步骤

步骤一：

在实验一的基础上，在“03_tag/WebContent”目录下新建 page.jsp 文件，page.jsp 中利用 page 伪指令的 language 指定页面使用的脚本语言是 Java、import 导入 java.util.Date 类、session 指出页面需要一个 HTTP 会话、buffer 指定客户端缓冲区的大小事 12K、autoFlush 指定当缓冲区满时到客户端的输出自动被刷新、info 描述页面信息、errorPage 指定当页面出现异常时应调用错误提示页面、isErrorPage 指明不可以使用 exception 对象、contentType 指定字符编码格式

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ page import="java.util.*"%>
<%@ page session="true"%>
<%@page buffer="12kb"%>
<%@page autoFlush="true"%>
<%@page info="a test directtive jsp page"%>
<%@page isErrorPage="false"%>
<%@page errorPage="error.jsp"%>
<!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=UTF-8">
<title>a test directtive jsp page</title>
</head>
<body>
<h1>use page directive</h1>
<%=new Date().toLocaleString()%>
</body>
</html>
```

图 3-1

步骤二：

page 指令中的 errorPage 指定的错误页面是 error.jsp，error.jsp 页面的代码如下图

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8" isErrorPage="true"%>
<!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=UTF-8">
<title>error!</title>
</head>
<body>
error!
<br />
Here has a error:
<br />
<hr />
<font color="red"><%=exception.getMessage()%></font>
</body>
</html>
```

图 3-2

步骤三：

在 Eclipse 中运行“03_tag”项目，打开浏览器，输入：“http://localhost:8080/03_tag/page.jsp”，回车,运行效果图如下

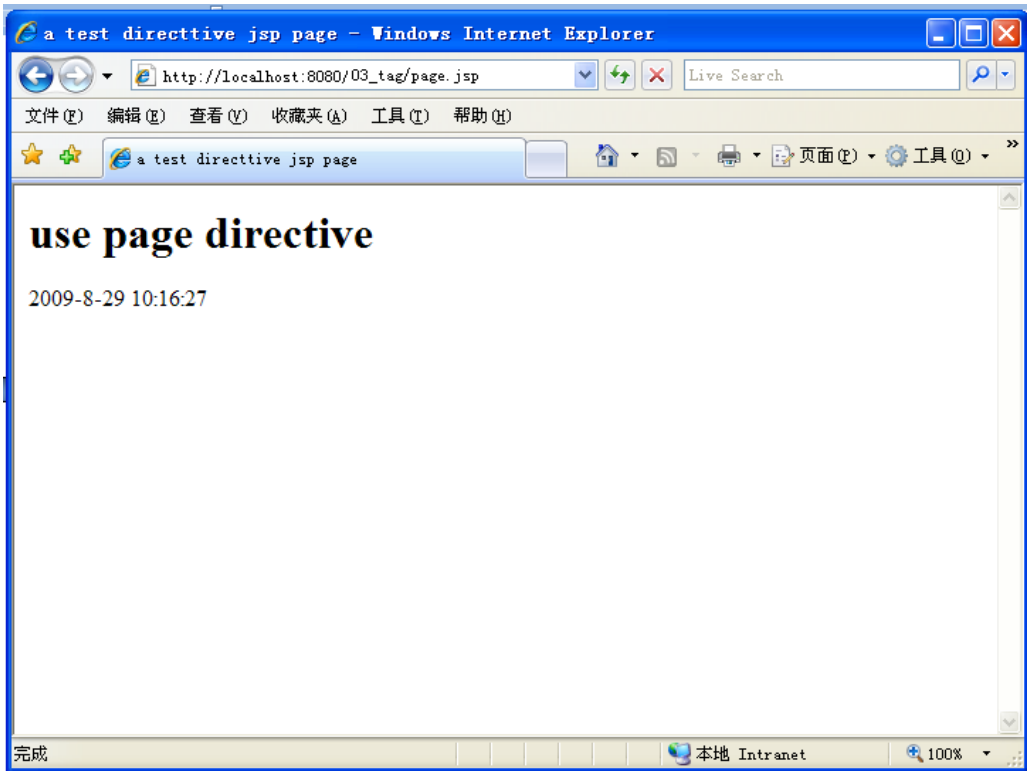

图 3-3

3.4 实验结论

通过 page.jsp 页面编写深入了解 page 伪指令的其他属性的作用与意义，也熟悉了 JSP 程序的编写

四、实验三 区分 JSP 中伪指令静态包含与 JSP 动作指令动态包含的区别

4.1 实验目的

使用 Eclipse 编写 JSP 程序，区分 JSP 中伪指令 include 静态包含与 JSP 动作指令 include 动态包含的区别

实现功能：编写 header.jsp，side.jsp,body.jsp,footer.jsp，mian.jsp 页面，在 main.jsp 页面中分别用 include 静态包含，和 include 动态包含包含其他四个页面，运行看看效果，找出区别

4.2 准备

保证 Eclipse 已经正确配置 Tomcat 服务器，可以进行 JavaWeb 项目编写、编译与调试。（如果以上设置不正确，请参考《Eclipse 部署配置 Tomcat 手册》进行正确设置）

4.3 实验步骤

步骤一：

在实验一的基础上，在“03_tag/WebContent”目录下新建 header.jsp 文件，内容如图所示

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

<%!String name = "header";%>
<table height="20%" width="100%" bgcolor="99ccff">
    <tr>
        <td align="center">==<%=name %>==</td>
    </tr>
</table>
```

图 4-1

步骤二：

在“03_tag/WebContent”目录下新建 side1.jsp,其中采用<%@ include file=“header.jsp” %>包含 header.jsp 并对 header 中定义的 name 重新赋值 name=“side”，如下图

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@include file="header.jsp" %>
<% name = "side";%>
<table height="20%" width="100%" bgcolor="5577ff">
    <tr>
        <td align="center">==<%=name %>==</td>
    </tr>
</table>
```

图 4-2

步骤三：

在“03_tag/WebContent”目录下新建 body1.jsp,其中也采用<%@ include file=“side1.jsp”%>包含 side1.jsp 并对 header 中定义的 name 重新赋值 name=“body”，如下图

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@include file="side1.jsp" %>
<% name = "body";%>
<table height="40%" width="100%" bgcolor="9900ff">
    <tr>
        <td align="center">==<%=name %>==</td>
    </tr>
</table>
```

图 4-3

步骤四：

在“03_tag/WebContent”目录下新建 footer1.jsp,其中也采用<%@ include file=“body1.jsp”%>包含 body1.jsp 并对 header 中定义的 name 重新赋值 name=“footer”，如下图

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@include file="body1.jsp" %>
<%name = "footer";%>
<table height="20%" width="100%" bgcolor="777777">
    <tr>
        <td align="center">==<%=name %>==</td>
    </tr>
</table>
```

图 4-4

步骤五：

在“03_tag/WebContent”目录下新建 main1.jsp,其中也采用<%@ include file=“footer1.jsp”%>包含 footer1.jsp 如下图

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!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=UTF-8">
<title>main</title>
</head>
<body>
<%@include file="footer1.jsp" %>
</body>
</html>
```

图 4-5

步骤六：

运行“03_tag”项目，在浏览器中输入：

http://localhost:8080/03_tag /main1.jsp，回车，运行效果图如下

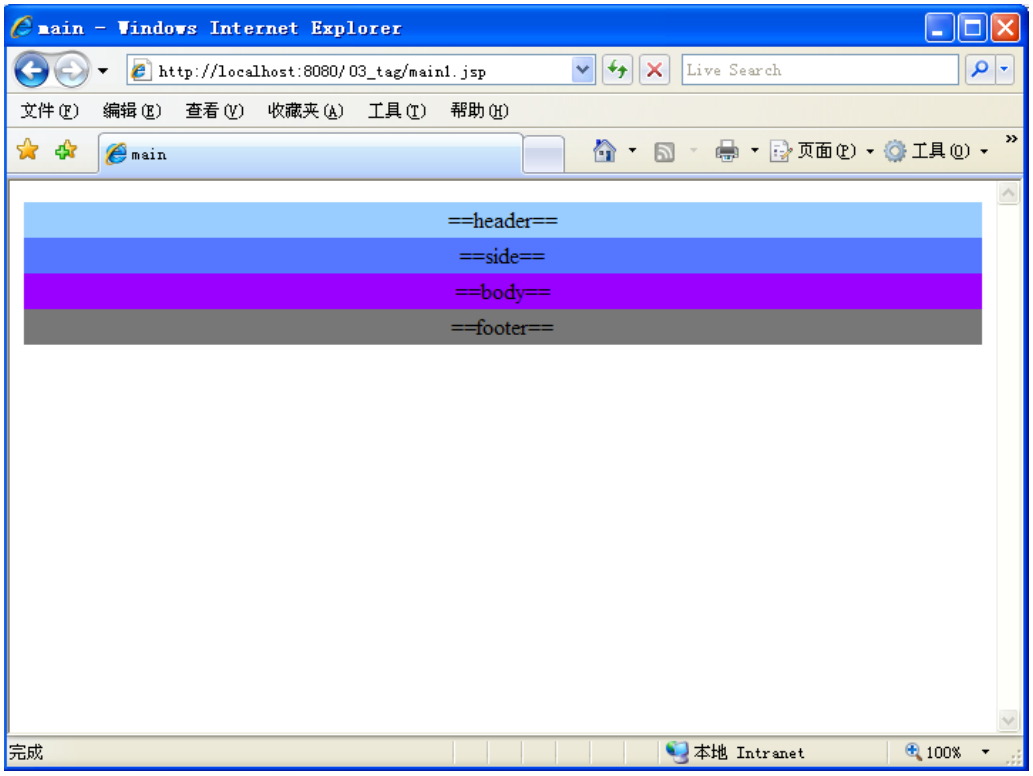

图 4-6

至此<%@include %>静态包含实验结束。

步骤七：

在“03_tag/WebContent”目录下新建 side2.jsp，其中采用<jsp:include page=“header.jsp”/>包含 header.jsp 并对 header 中定义的 name 重新赋值 name=“side”，如下图

图 4-7

我们会发现编译都通不过，会报错，name 没有声明

4.4 实验结论

通过以上实验的完成我们会总结出静态包含<%@include file=”” %>与动态包含<jsp:include page=”” />的区别：静态包含是机械的将一个 A 页面放到另一个 B 页面中，跟把 A 的代码直接粘贴到 B 页面效果是一样的；动态包含是在程序运行过程中动态的将 A 页面加载到 B 页面中去的，在页面代码中写入<jsp:include />代码时并不把 A 的内容加载到 B 页面上（所以 side2.jsp 中找不到 header.jsp 中定义的 name，报错）