Java 应用与开发

JSP (Java Server Page)

王晓东

wangxiaodong@ouc.edu.cn

中国海洋大学

December 3, 2018

学习目标

1. 理解 JSP 和 Servlet 的关系。
2. 掌握 JSP 提供的各类编程元素的使用方式，包括 JSP 指令、JSP 动作、JSP 脚本。
3. 掌握 JSP 提供的内置对象与 Servlet 相关对象的对应，学会各类对象的使用方法。
4. 能够使用 JSP 完成简单的 Java Web 编程。
5. 对 JSP 作为 MVC 设计模式中的视图构建方式有初步的了解。

大纲

接下来…

JSP 基本概念

- JSP(Java Server Page)，即 Java 服务器页面。
- JSP 是 Servlet 的扩展。
- **JSP 将使用 Java 类编写动态 Web 组件的方式转变为使用文本编写（采用标记型语法和过程性语法混合），降低了开发的难度。**
- JSP 提供了一种自然的生成网页的方法。
- 可以使用 GUI 工具来绘制构建 JSP 页面。
- JSP 文件的扩展名必须是.jsp。

JSP 的优点和缺点

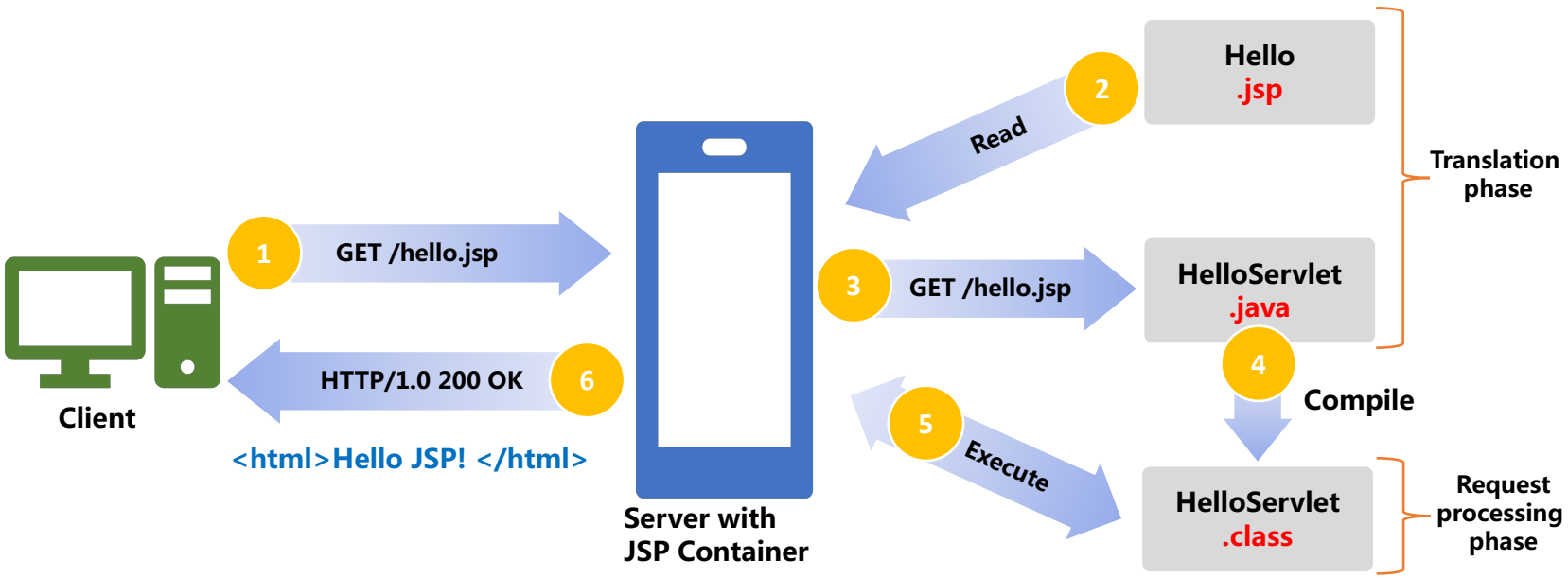
❖ 优点

- 使得编写动态 Web 网页更加容易；
- 降低了开发难度；
- 可以使用工具的拖拉方式生成 JSP 页面。

❖ 缺点

- 非 OO 编程方式；
- Java 代码嵌入到 HTML 代码中，导致维护困难；
- 不适合编写规模比较大的业务处理应用程序。

JSP 的执行过程


Hello
.jsp
2
Read
Translation
phase
1
GET /hello.jsp
3
GET /hello.jsp
HelloServlet
.java
4
Compile
HTTP/1.0 200 OK
6
Client
<html>Hello JSP! </html>
5
Execute
Server with
JSP Container
HelloServlet
.class
Request
processing
phase

JSP 执行过程描述

1. 客户使用浏览器通过 HTTP 请求 JSP 文件的 URL 地址，例如：http://localhost:8080/webapp/hello.jsp；
2. Web 服务器接收到请求，如果没有此地址，发出错误响应给浏览器；
3. Web 服务器检查 JSP 文件和对应的 Servlet 版本的时间是否一致，如果一致则执行 servlet 的处理请求方法，类似于 doGet 或 doPost，发送响应给浏览器；
4. 版本时间不一致，Web 服务器调用转换系统，将 JSP 的文本代码转换为 Servlet 的 Java 代码；
5. 将 Java 代码编译为 class 文件；
6. 调用 Servlet Class 的相应方法处理请求并返回响应。

JSP 页面的组成

一个 JSP 页面由 HTML **标记代码**和 JSP **元素**组成。HTML 标记用于生成网页的静态部分，JSP 元素用于生成动态内容部分。

❖ JSP 所包含的元素

- JSP 指令
- JSP 动作
- JSP 脚本

JSP 内置对象

JSP 扩展标记

接下来…

JSP 指令

JSP 指令指示一个 JSP 页面的属性和特征，JSP 指令不会产生任何的输出到当前输出流中。

❖ JSP 指令

- page 指令，用于定义 JSP 页面级的其他元素特征。
- include 指令，用于嵌入另一个文本文件的内容到本页面。
- taglib 指令，用于引入第三方 JSP 扩展标记类库。

❖ JSP 指令的语法

```
1    <%@ 指令名 属性名="值" 属性名="值" %>
```

page 指令

page 指令定义应用于整个页面的属性。

❖ page 指令语法

```
1    <%@ page 属性名="属性值" %>
```

❖ page 指令属性

- language=”java”，指定页面语言。
- contentType=”text/html; charset=gb2312”，指定页面的内容类型，默认是 text/html，可以指定显示的字符集，中文是 gb2312 。
- import=”package, pageage”，指定 JSP 页面使用的包和类，可以引用多个包，每个包用逗号分隔。

page 指令

❖ page 指令属性

- buffer=”none | xkb”，指定输出缓冲区容量，默认为 8KB，buffer=”9kb”。
- errorPage=”errorURL”，指定错误页面地址，当页面出现异常时自动跳转到指定的错误页面。
- isErrorPage=”true|false”，指定本页面是否是错误处理页面。
- autoFlush=”true | false”，控制输出缓冲区是否自动清空，默认是 true。

include 指令

include 指令用于在当前网页中嵌入另一个网页，可以是 JSP、HTML 等。

❖ include 指令语法

```
1    <%@ include file="url" %>
```

👉 说明

- file 属性确定要嵌入的页面。
- 嵌入页面的源代码被放置在此指令所在的位置。
- 嵌入的用途是将一个复杂的页面分解为小的页面，然后使用 include 指令将他们再组装在一起。
- 当修改被 include 的文件时，如果不修改主文件，则嵌入的内容不会改变。因此必须也修改主文件才能反映更新的嵌入文件。

taglib 指令

taglib 指令用于引入扩展标签库，如 JSTL、Struts 等标签库。

❖ taglib 指令语法

```
1    <%@ taglib uri="/WEB-INF/tlds/struts-html.tld" prefix="html" %>
2    <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
```

接下来…

JSP 动作

JSP 动作使用特定的符合 XML 格式的标记完成特定的任务。利用 JSP 动作可以完成动态的插入文件、重用 JavaBean 组件、把用户重定向到另外的页面、为 Java 插件生成 HTML 代码。

❖ JSP 动作的语法

无嵌套封闭格式

```
<jsp:动作名称 属性名="值" 属性名="值" 属性名="值"/>
```

有嵌套封闭格式

```
<jsp:动作名称 属性名="值" 属性名="值" 属性名="值">
  嵌入的其他动作
</jsp:动作名称>
```

JSP 动作的类型

<jsp:include> 嵌入其他页面输出内容动作

<jsp:forward> 转发动作

<jsp:plugin> 引入插件动作

<jsp:param> 提供参数动作

<jsp:useBean> 使用 JavaBean 动作

<jsp:setProperty> 设置 JavaBean 属性动作

<jsp:getProperty> 取得 JavaBean 属性动作

include 动作

include 动作用于嵌入其他页面的输出内容到此动作所在页面。

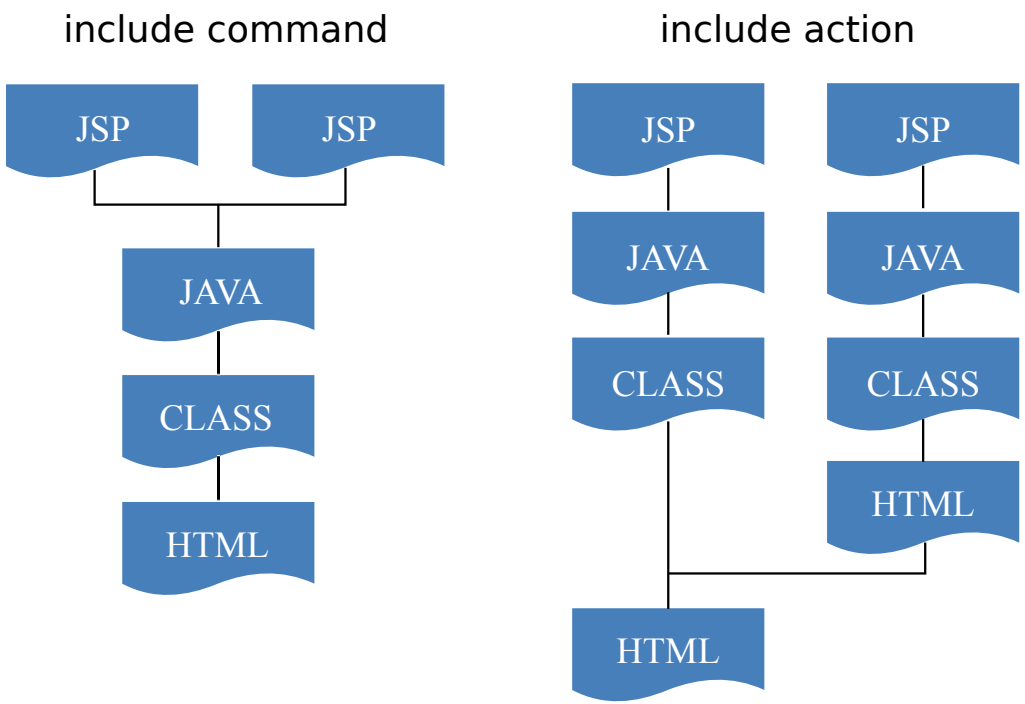
```
1    <jsp:include page="url" flush="true" />
```

```
1    <jsp:include page="url" flush="true">
2      <jsp:param name="参数名" value="参数值" />
3    </jsp:include>
```

- page 属性，指定嵌入页面的 URL 地址；
- flush 属性，指定是否在嵌入页面之前清空响应缓存区，默认为 true；
- jsp:param，为嵌入的页面传递参数，这些参数可以为动态、也可以为静态。

include 指令和 include 动作的差异

- 其根本性不同在于他们被调用的时间。include 指令在页面解析期间被嵌入；inlcude 动作在请求的响应输出时被嵌入。
- **在实现文件包含上，因该尽可能的使用 include 动作。**
- 而 include 指令存在的原因是其功能更加强大，执行速度稍快。



useBean 动作

- 引用并调用 JavaBean 的方法和属性是 JSP 开发时必须面对的主要任务。
- 为了简化 JavaBean 应用，JSP 提供了 useBean 动作标记，使得不必使用 Java 脚本代码，而使用标记方式就可以取得 JavaBean 的对象引用。

useBean 动作

❖ useBean 动作语法

```
1    <jsp:useBean id="name" class="package.ClassName"
2      scope="page|request|session|application" />
```

- id=”name” 实例化变量名
- scope=”scope” 定义实例变量的生存周期范围
  - page 只在本页面中使用，默认值
  - request 在请求范围内有效
  - session 在会话范围内有效
  - application 在整个 Web 启动后有效
- class=”package.ClassName” 指定 JavaBean 的类

useBean 的执行过程

1. 如果在指定范围内找到指定的对象，则得到此对象引用 (即通过 scope 对象的 getAttribute() 方法)。
2. 如果没有找到指定的对象，则实例化一个 class 属性指定的对象 (调用类的无参数构造方法)。
3. 对新建的对象执行嵌入的 <jsp:setProperty /> 中指定的属性值，即调用 JavaBean 的 setXXX 方法，设置类的属性值。
4. 将对象保存到 scope 指定的范围的对象中，即调用内置对象的 setAttribute(name,Object) 方法，name 是 id 指定的名称，object 是类的对象。

useBean 动作示例

JavaBean 类 User：User.java

```
1   package ouc.java.entity;
2   import java.io.Serializable;
3   public class User implements Serializable {
4     private String id = null;
5     private String password = null;
6     private String name = null;
7     private age = 0;
8
9     public String getId() {
10      return id;
11    }
12    public void setId(String id) {
13      this.id = id;
14    }
15    ... ... // (other set/get methods.)
16  }
```

创建/取得指定的 JavaBean 对象，并保存在 Request 对象中

```
1   <jsp:useBean id="user" class="ouc.java.entity.User" scope="request" \>
2   <%
3     String name = user.getName();
4     out.println(name);
5   %>
```

使用 useBean 动作 VS. JSP 脚本创建并取得对象引用

❖ 使用 JSP 脚本

```
1    <% User user = new User(); %>
```

☛ 注意

以上使用 Java 脚本创建的 JavaBean，只能在本页面中使用，且每次都是创建新的 JavaBean 对象，不会保存在任何范围对象中。

使用 useBean 动作 VS. JSP 脚本创建并取得对象引用

❖ 使用 JSP 动作

```
1    <jsp:useBean id="user" class="ouc.java.entity.User" scope="request" \>
```

以上使用 JSP 动作代码，相当于如下 Java 脚本：

```
1    <%
2      User user = (User) request.getAttribute("user");
3      if (user == null) {
4        user = New User();
5        request.setAttribute("user", user);
6      }
7    %>
```

setProperty 动作

setProperty 动作用于设定 userBean 动作取得的 Bean 对象的属性，相当于执行 Bean 对象的 setXxx 方法。

```
1    <jsp:setProperty name="beanId" property="*|name" param="参数名" value="value" />
```

```
1    <jsp:useBean id="user" class="com.city.oa.value.User" scope="request" />
2    <jsp:setProperty name="user" property="name" value="吴明" >
```

getProperty 动作

用于取得 Bean 对象指定的属性值，转换为 String 类型，并显示在此动作所在的位置。

```
1    <jsp:getProperty name="beanId" property="属性名" />
```

- name 指定 bean 对象的名称，与 useBean 动作的 id 值对应；
- property 指定属性名，与 JavaBean 的 getXxx 方法对应。

forward 动作

forward 动作把请求转到另外的页面。

无嵌入参数的转发动作

```
1    <jsp:forward page="URL" />
```

例如：

```
1    <jsp:forward page="main.jsp" />
```

有嵌入参数的转发动作

```
1    <jsp:forward page="URL">
2      <jsp:param name="参数名" value="值" />
3    </jsp:forward>
```

例如：

```
1    <jsp:forward page="main.jsp">
2      <jsp:param name="id" value="kevin" />
3      <jsp:param name="password" value="1000" />
4    </jsp:forward>
```

param 动作

param 动作不能单独使用，需要嵌套在其他动作中，为其他动作提供参数，它可以嵌入到 include、forward 动作中为目标地址提供参数。

```
1    <jsp:forward page="main.jsp">
2      <jsp:param name="id" value="kevin" />
3      <jsp:param name="password" value="1000" />
4    </jsp:forward>
```

接下来…

JSP 脚本

❶ 代码脚本

```
1    <%
2      int a=0; // 可以放置任何 Java 代码
3    % >
```

- 可以同时嵌入多个脚本代码段，但它们都表示在一个方法内，属于一个代码段，一个脚本内定义的变量，另一个脚本也可以使用。
- 在代码脚本中定义的变量类似于 Servlet 的 doGet 方法中的局部变量，未初始化使用是非法的。

例如，连接数据库的代码脚本片段如下：

```
1    <%
2    Connection cn = null;
3    try {
4      Class.forName("sun.jdbc.odbc.JdbcOdbc.Driver");
5      cn  = DriverManager.getConnection("jdbc:odbc:cityoa");
6      String sql = "select␣*␣from␣EMP";
7      ... ...
8    }
9    % >
```

JSP 脚本

❷ 表达式脚本

```
1    <%= a %> 用于输出 Java 表达式的值
```

☛ 注意

表达式后不能有分号，= 号与 <% 之间不能有空格。例如：

```
1    <%= rs.getString("NAME") %>
```

JSP 脚本

❸ 声明脚本

```
1    <%!
2      int m=0; // 声明 JSP 类变量和方法
3    % >
```

用于声明 JSP 页面的类变量和方法。
由于 JSP 在运行时会转换为 Servlet 类，声明的脚本部分会成为类中定义的一部分。例如：

```
1  <%!
2    int num = 0;
3    public void addNum() {
4      num++;
5    }
6  %>
```

如下声明脚本有错误：

```
1  <%!
2    int num = 0;
3    num++; // 错误，声明脚本中不能直接有非声明代码
4  %>
```

JSP 脚本

❹ 注释脚本

```
1    <%-- JSP Comment --%>
```

在 JSP 页面中可以使用 HTML 注释：

```
1    <!-- HTML Comment -->
```

但是，使用 HTML 注释不安全，因为 HTML 注释随着 JSP 生成的 HTML 响应下载到客户端浏览器，客户可以看到。

JSP 注释是服务器端技术，在服务器端处理，不会发送到客户端，比较安全。

接下来…

JSP 内置对象

为了与 Web 容器以及其他 Web 组件进行通信和协作，JSP 提供了相关内置的对象，这些对象不需要定义和引用，可以在 JSP 代码脚本和表达式脚本中可以直接使用。

request	请求对象
response	响应对象
session	会话对象
application	应用服务器对象
page	JSP 本身页面类对象
pageContext	页面级环境变量，作为页面级容器
out	输出对象
exception	异常对象
config	配置对象，用于读取 web.xml 配置信息

响应对象 response

- JSP 页面使用文本方式实现 HTTP 响应，所以 JSP 内部不经常使用 response 对象。
- 在 JSP 中实现响应，直接将响应内容写在 JSP 页面就可以，不需要使用响应对象取得 PrintWriter 对象进行响应输出。

会话对象 session

- JSP 的 session 对象对应 Servlet 中的**javax.servlet.http.HttpSession**。
- JSP 页面使用 page 指令指示此 JSP 页面是否可以使用 session 内置对象。

```
1  <%@ page session="true␣|␣false" %>
```

页面对象 page

page 对象就是指向当前 JSP 页面本身，有点像类中的 this 指针，它是 java.lang.Object 的实例，在 JSP 编程中应用较少。

常用方法

- int hashCode() 返回此 Object 的 hash 码

输出对象 out

out 内置对象即 JSP 页面向浏览器发出响应流 PrintWriter 的实例对象。但 JSP 页面可以直接放入响应文本，因此 JSP 页面基本不使用 out 进行文本响应。

一般情况下使用**JSP 表达式脚本**来代替使用 out 对象。

```
1    <%= %>
```

异常对象 exception

JSP 内置对象 exception 对应 Java 中的 java.lang.Throwable 接口对象。它自动封装 JSP 页面中出现的异常。
异常对象只有在**错误页面**中可以使用，即：使用 page 指令的属性isErrorPage=”true”声明的页面。

sample.jsp

```
1   <%@ page language="java" contentType="text/html;␣charset=utf-8"
2     errorPage="error.jsp" import="java.util.*" pageEncoding="utf-8" %>
3   <html>
4     <head>
5       <meta http-equiv="Content-Type" content="text/html;␣charset=utf-8">
6       <title>Test Exception</title>
7     </head>
8     <body>
9       <%
10        int s = 0;
11        int t = 0;
12        int p = s / t; // 出现异常的语句
13      %>
14    </body>
15  </html>
```

异常对象 exception

error.jsp

```
1  <%@ page language="java" contentType="text/html;␣charset=utf-8" isErrorPage="true"
2    pageEncoding="utf-8"%>
3  <html>
4    <head>
5      <meta http-equiv="Content-Type" content="text/html;␣charset=utf-8">
6      <title>Error Page</title>
7    </head>
8    <body>
9      <h1>异常显示信息</h1>
10     错误原因: <%= exception.getMessage() %>
11   </body>
12 </html>
```

使用 errorPage 方式实现的页面跳转，使用的是转发方式，而不是重定向模式。

接下来…

本节习题

❖ 简答题

1. 什么是 JSP？JSP 运行在哪里？JSP 一般的输出是什么？
2. JSP 有哪些优点和缺点？
3. JSP 的执行过程是怎样的？
4. 能够基于一个简单的 JSP 页面代码进行分析，找出 JSP 转换为 Servlet 后，相关 JSP 元素在 Servlet 中都是如何转换的。

本节习题

❖ 小编程

1. 整理一份你最喜欢的歌单，最少包括 10 首歌曲，并将歌单存入 MySQL 数据库。每首歌曲需要包括曲名、演唱者、类型风格等信息。
2. 编写 JSP 代码，直接在 JSP 中连接数据库并查询歌单，将所有歌曲信息以 HTML table 的形式显示在页面上。
3. 注意可能会用到 JSTL 标记实现 table 的循环生成行，请自行检索简单用法。

THE END

wangxiaodong@ouc.edu.cn