# JSP

## 一、JSP的基本知识

### 1.基本概念

JSP(Java Server Page)是一种可以混合静态HTML和java代码，并能动态生成HTML的技术。文件以.jsp为后缀名。JSP是一种服务器端脚本语言。JS则是客户端脚本语言。jsp中可以编写html标记和java代码（Java脚本）。

JSP文件有**两种形态**：**设计时态的JSP文件；运行时态的Servlet代码**。

JSP既支持脚本生成动态内容，也支持标签生成静态内容。

JSP文件会被Web容器转化成Servlet，然后编译运行：

①JSP源文件会被编译

②产生JSP Servlet代码

③JSP Servlet会被编译，然后被容器读入，被运行。

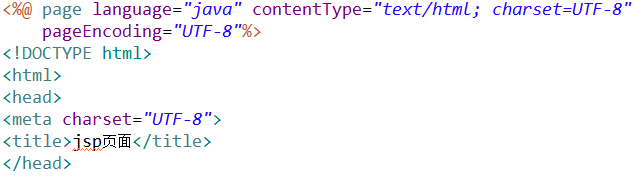
JSP运行在Web容器中。

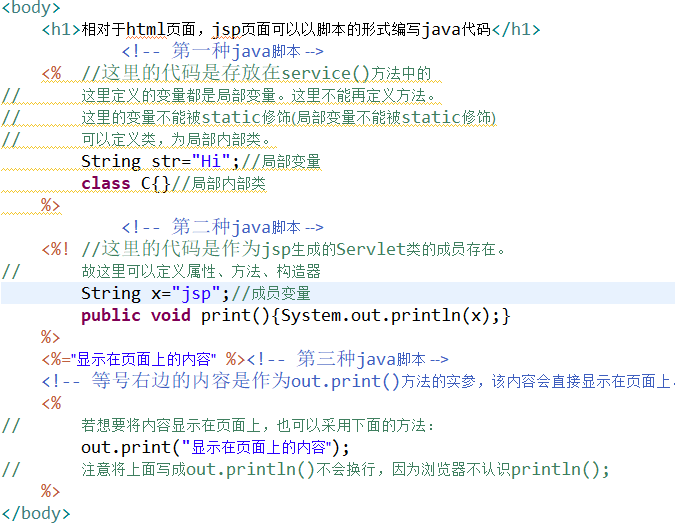
**各种文件如html、css、js、jsp等的编码方式设置：Window-->Preference-->输入对应的文件，找到编码设置。**

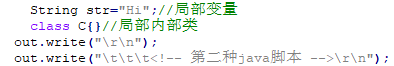
session的数据是保存在服务器端的，cookie的数据是保存在客户端的。

由jsp生成的.java文件中，\r\n代表的是换行，而\t代表的是空格。。由jsp文件 生成的.java文件和.class文件一般存放在tomcat的work目录下的jsp文件所在的项目名目录下。

### 2. jsp页面写java代码

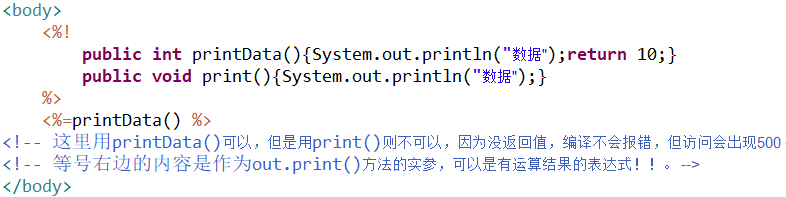




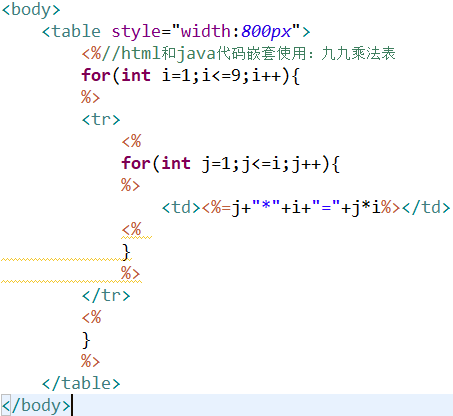




注意一个xxx.jsp文件，发布到tomcat中后生成的类一般类名为xxx\_jsp，该类继承HttpjspBase类。在jsp文件中写的java脚本，一般在对应的.java文件中直接以java代码的形式显示，而写的html的标签内容则是直接被out.write()方法输出到页面中。在网页中查看源代码时，只能看到关于html页面的源代码，而java代码则不会显示。



在jsp页面中是可以混合java代码和html代码（可以嵌套的）：



在html标签之间可以通过<% %>等形式写java代码，但是在每个<%..... %>中间不能写html标签。

### 3.jsp中的注释

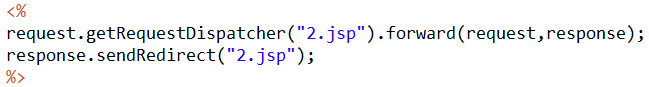
**⑴<!- - - ->**：html的注释，该注释会被作为jsp对应的Servlet类代码中的一部分，会被执行，被out.write()输出流输出到浏览器中，在页面中不会显示，但是在源代码中可以看到该内容被注释。

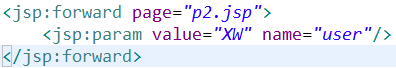
**⑵// /\*\*/**：Java的注释，该注释会被作为jsp对应的Servlet类代码中的一部分，不会被执行。在Serlvet代码中可以看到该内容被注释。

**⑶<%- - - -%>**：jsp的注释，该注释不会被编译到jsp对应的Servlet类代码中，因此在Servlet中和浏览器中的源代码中都看不到该注释。

### 4.jsp页面跳转到其它页面

可以直接使用jsp页面中的html标签进行跳转，即使用<a>标签或者location.href进行跳转。也可以在<%...%>中写java代码进行跳转。也可以使用jsp标签**<jsp:forward page="uri "></jsp:forward>**进行跳转。



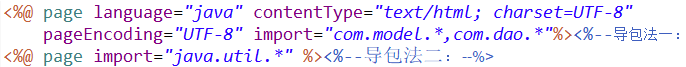


## 二、jsp中的三个指令

JSP的指令不会产生任何可视化的输出，该指令告诉JSP编译器对此页面要做的一些工作。比如导包...。该指令在页面编译成Servlet时会被调用。

**1.<%@page %>**：设置页面的特有属性。

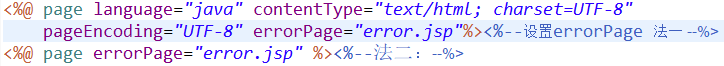
**①import导包：**



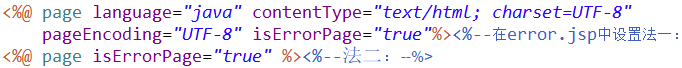
**导包时\*：表示导入当前包中的所有类，但不包括其子包中的类。Class是直接属于java.lang包中的类，这个包中的直属的类不需要导入就可以直接使用。java.lang包中的子包中的类是必须导入才能使用**。

**②错误处理：**

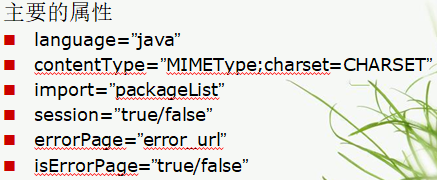
**在出错的地方设置**：**errorPage=”error url”**



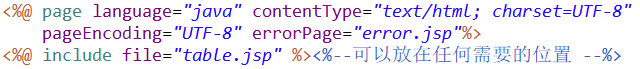
**在解决错误的地方设置**：**isErrorPage=”true”**



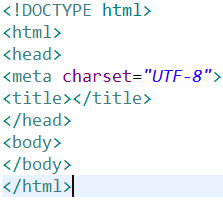
当浏览发生错误时跳转到url页面，而不是在页面上显示500错误。如果在解决错误的页面写：，则可以在页面上显示错误的具体信息，jsp中的exception相当于java代码中异常Exception中的e。



**2.<%@include file=”fileName” %>**：静态包含，会将包含的内容原样全部拿过来。可放在任何需要的位置如放在html标签之中。



用在需要包含一个外部文件的地方，可以是任何文件如.txt，.html，.jsp文件等。一般情况被包含的文件不能出现<html><head><body>标签。不然的话会使下面这部分内容也出现在jsp的Servlet代码中(原来的jsp对应的Servlet代码中就有下面这些内容，当包含文件中含有这些标签时会重复出现，因此**可以将包含文件中的这些标签直接删除**)。



**3.<%@taglib %>**：引入标签库。内容好像缺少了。

## 三、九个内嵌变量

HttpServletRequest(request)、PageContext(pageContext)、HttpServletResponse(response)、HttpSession(session)、ServletContext(application)、ServletConfig(config)、JspWriter(out)、Object(page)、Throwable(exception)

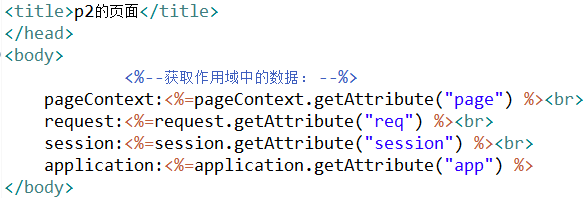
**①四个作用域**：**pageContext(页面作用域，只作用于当前页面)、request、session、application(全局作用域)**

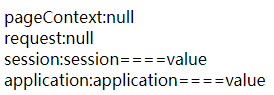
**②响应和输出流**：**response、out(response.getWriter())**

**③其它三个**：**page(this)、config**(初始化参数等配置信息)、**exception**(在错误页面上，exception指向发生的异常对象)。

1.在jsp中，pageContext的作用域仅限于当前页面，当由页面一跳转到页面二时，不管是使用服务器跳转还是客户端跳转，pageContext中保存的数据都会丢失。







也可通过pageContext操作其它作用域：



2.对配置页面进行配置之后，可以在配置页面中输入代码：

即可获得初始化参数。

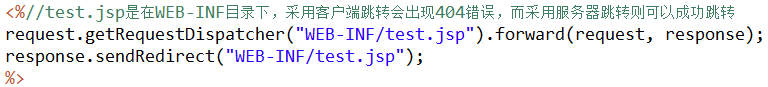
3.在解决错误的页面中输入代码：即可获得错误的具体信息。

## 四、客户端访问WEB-INF目录下的内容

**WEB-INF下的内容只有服务器端才能访问，客户端是没有权限访问的。Web-Content目录下的内容才是客户端可以访问的**。

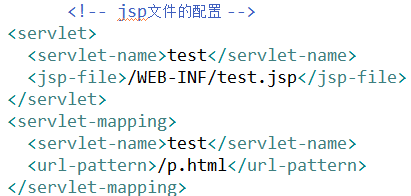
。

因此直接在页面上输入localhost:8080/jsp\_First(项目名称)/test.jsp会出现404错误。



**直接在Web-Content目录下的文件的相对路径直接写成文件名就行，如index.html。而在Web-Content目录下的子目录中的文件的相对路径要写成子目录名称/文件名，如下面的WEB-INF/test.jsp。**

因此，有的时候会将一些文件放到WEB-INF目录下面，这样可以防止别人知道跳转的文件名而进行恶意操作。**将文件放置到WEB-INF目录下面，要使客户端能够访问该文件，则需要在web.xml中进行配置。该配置方法和配置Servlet基本一样，只是将<servlet-class>改为了<jsp-file>。**如下所示：



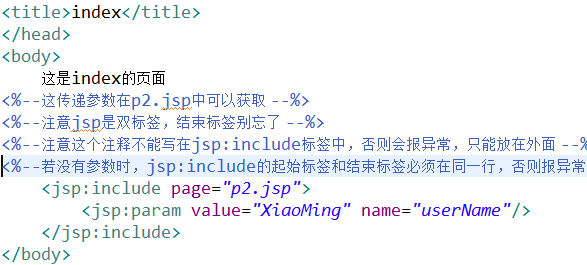
配置好之后，重启tomcat，然后在浏览器中输入localhost:8088/jsp\_First/p.html则可以直接访问到WEB-INF目录下的test.jsp内容。可以看到url-pattern的路径名称可以是迷惑性的，路径显示的文件是p.html而实际文件是test.jsp。

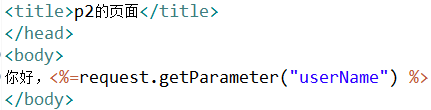
## 五、jsp标签：<jsp:xxx...>

**①<jsp:include page=”uri”></jsp:include>**：动态包含，可传递参数。原页面的内容和包含页面的内容都可以显示。

**静态包含：<include> 动态包含：<jsp:include....>**

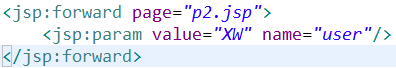
**要特别注意当jsp:include标签没有传递参数时，其对应的起始标记和结束标记必须在同一行，否则在运行时会出异常。还有注释不能放在这个标签之中，否则也会报异常。访问时出现500错误，并报**Expecting "jsp:param" standard action with "name" and "value" attributes**异常**。



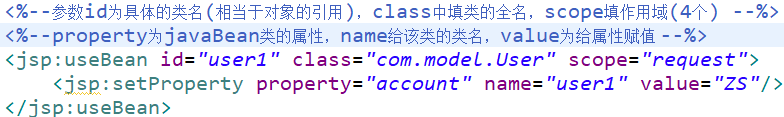




**②<jsp:forward page="uri "></jsp:forward>：属于服务器跳转。**直接跳转到目标页面的内容，原页面的内容不显示。用法和前一个jsp标签差不多。



**③<jsp:useBean ...>**:该jsp标签在jsp对应的Servlet类中创建或取出一个javaBean类的实例。



## 六、标准的JavaBean规范

**①私有属性的get、set方法必须是get属性名和set属性名，且属性名第一个字母大写。若属性是boolean类型，则get方法必须是is属性名**。

**②必须有public无参构造方法。**

**③需实现序列化的接口implements Serializable。**(若没有特殊要求的功能，可以不写这个)。

不管是否用到public 类型的无参构造器，都要将该无参构造器写出来，因为在很多地方都用得到这个无参构造器。

## 七、JSP总结

JSP：Java Server Pages，一个java服务端的页面。JSP是运行在服务端的，需要一个Web容器才能运行。没有Web容器是不能直接打开的。

### 1.JSP的运行模型

**①设计时态：是以.jsp结尾的文本文件。**

**动态：java程序**

**指令：<%@ page ...%>、<%@ taglib ...%>、<%@ include...%>(静态引入，将引入的内容和当前jsp的内容合成一个源码输出)**

**java脚本：<%...%>(定义在service()方法中，有九大内置对象可用)、<%!...%>(定义在类中)、<%=...%>(常量或表达式或带返回值的方法的调用)**

**jsp标签：<jsp:include...>(动态引入：在jsp运行时将内容引入)、<jsp:forward...>(属于服务器跳转)**

**动态内容都是在服务端执行的。注意上面的这些都是在服务端生成的，都是在客户端发送请求之前已经执行过的，且只执行一次。所以不能在html、css、js中写上面这些内容。客户端拿到的内容只是服务端已经执行过的内容。**

**静态：html+css+js(注意这些内容都是客户端执行的)**

**②运行时态：以一个Servlet对象的形式运行。**

### 2.JSP的生命周期

①第一次去访问一个jsp的文件的时候，服务端根据jsp页面中的内容，去生成一个.java的一个文件(遇到动态的java程序，服务器将解析成自己识别的代码，遇到静态的如html等则直接用out.write()输出)。这个文件就是一个Servlet的一个实现类，然后编译成字节码文件.class文件。

②创建jsp对应的servlet对象，调用init方法初始化

③每次请求jsp页面时，实际上调用的就是jsp对应的servlet的对象的service方法

④服务器关闭时，销毁jsp的servlet对象，销毁之前调用destory方法。