**JSP标签编程简介：**

1. 即便使用了MVC设计模式，在一个JSP页面中也会存在Scriptlet代码，而一个完善 的JSP文件是不应该包含任何的Scriptlet代码的，那么就需要通过标签编程来解决 此问题。

2）由于标签库的开发较难，在实际工作中也并不常见，并且在各种开源组件中已经提 供了大量的可用标签供用户使用。所以只需了解标签编程的基本原理即可。

3）标签编程的主要目的是为了减少JSP页面中的Scriptlet代码，是程序更加便于理解 和修改。

4）JSP标签库也称指定的公益标签库，是使用XML语法格式完成程序操作的一种方法， 其使用的形式类似于JavaBean的使用语法“<jsp:useBean>”。与JavaBean一样都可以 将大量的复杂操作写在类中完成，而且最大的优势是按照HTML标签的形式表现，可 以方便地处理JSP页面的数据显示。

**定义一个简单的空标签：**

1. 要想实现一个标签，可以直接继承javax.servlet.jsp.tagext.TagSupport类。
2. 如果要定义的标签内没有标签体，则直接覆写TagSupport类中的doStartTag( )方法即 可。没有标签体的标签指的是单标签。
3. 一个标签类定义完成之后，就需要编写标签描述文件；在\*.tld文件中，可以描述标 签的名称、简介、处理类和标签使用到的各个属性等，所有标签的具体信息都要通 过此文件定义。
4. 编写完\*.tld文件之后，即可在JSP页面中使用此标签，JSP页面中使用标签的语法如 下：

<%@ taglib prefix=”标签前缀” uri=”TLD文件路径” %>

prefix属性表示的是标签使用时的前缀。

uri表示的是此标签对应的\*.tld文件的路径。

1. 在JSP中使用自定义标签时，是根据uri找到标签描述文件(\*.tld)，但如果此标签文 件的名称过长，则每次编写起来会非常麻烦，所以，可以在web.xml文件中对\*.tld 文件进行名称的映射，以后直接在JSP页面中使用映射名称即可访问标签描述文件。

**自定义标签的组成：**

要想完成一个标签的开发，需要具有以下几个部分：

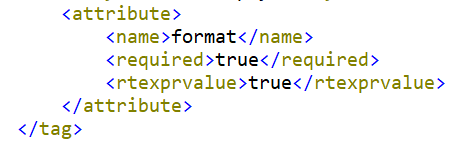
1. 标签处理类
2. 标签描述文件
3. 在JSP页面中通过<%@taglib%>定义标签
4. 在web.xml文件中配置标签描述文件的映射路径（可选）

当一个JSP页面第一运行时，首先会根据JSP文件编写的<%@taglib%>中的uri属性找到 对应的\*.tld文件，并将其加载到JVM中；而如果是第二次运行此标签，由于JVM已经 存在了\*.tld文件，所以不会重复加载。

**定义有属性的标签：**

要定义一个带有属性的标签首先要在定义的标签处理类中定义属性，并为其添加setter 和getter方法；编译完成后，在定义标签描述文件时添加<attribute>标签：

1. <attribute>：表示定义一个标签中所具备的属性，一个<tag>元素中可以定义多个 <attribute>元素。
2. <name>：标签属性的名称，一般与此标签处理类中所对应的属性一致
3. <required>：此属性是否为必须设置，如果为true，则表示必须设置；如果为false， 则表示可选。
4. <rtexprvalue>：是否支持表达式输出或el输出，true表示支持，false表示不支持



**TagSupport类：**

TagSupport类是整个标签编程的一个核心类，通过继承该类可以自定义标签。此类定义 如下：

**public class TagSupport extends Object implements IterationTag , Serializable**

可以发现它实现了IterationTag接口，此接口又是Tag接口的子接口，而Tag接口又是J spTag接口的子接口。由此可见，TagSupport类拥有了这些接口中的方法和属性以及常 量。其中常见的有：

|  |  |  |  |
| --- | --- | --- | --- |
| No | 常量或方法 | 类型 | 描述 |
| 1 | protexted PageContext pageContext | 属性 | 表示PageContext对象 |
| 2 | public static final int SKIP\_BODY | 常量 | 忽略标签体内容，将操作转交给doEndTag( ) |
| 3 | public static final int EVAL\_BODY\_INCLUDE | 常量 | 正常执行标签体操作，但不处理任何的运算 |
| 4 | public static final int SKIP\_PAGE | 常量 | 所有在JSP上的操作都将停止，会将所有输出的内容立刻显示在浏览器 |
| 5 | public staitc final int EVAL\_PAGE | 常量 | 正常执行JSP页面 |
| 6 | public static final int EVAL\_BODY\_AGAIN | 常量 | 重复执行标签体内容，会再次调用doAfterBody( )，直到出现SKIP\_BODY为止 |
| 7 | public int doStartTag( ) throws JspException | 方法 | 处理标签开始部分 |
| 8 | public int doEndTag( ) throws JspException | 方法 | 处理标签结束部分 |
| 9 | public int doAfterBody( ) throws JspException | 方法 | 处理标签主体部分 |
| 10 | Public void release( ) | 方法 | 释放标签资源 |

**TagSupport类中常用方法：**

在整个TagSupport类中，doStartTag( )、doEndTag( )、doAfter( )和release( )这4个方法是 重要的方法，它们都是从TagSupport的父接口中继承而来的。

1. doStartTag( )：此方法在标签开始时执行，有如下两种返回值。

·SKIP\_BODY：表示忽略标签体的内容，将执行权转交给doEndTag( )方法

·EVAL\_BODY\_INCLUDE：表示执行标签体的内容

1. doAfterBody( )：此方法用来重复执行标签的内容，有如下两种返回值。

·SKIP\_BODY：表示忽略标签体的内容，将执行权转交给doEndTag( )方法

·EVAL\_BODY\_AGAIN：表示重复执行标签体的内容，会重复调用doAfterBody( )方 法，一直循环下去，直到doAfterBody( )方法返回SKIP\_BODY 为止

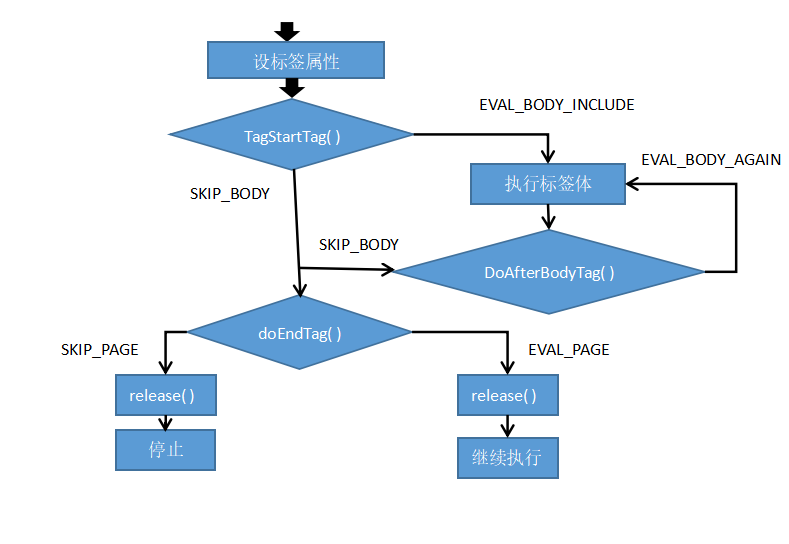
1. doEndTag( )：此方法在标签结束时执行，有如下两种返回值。

·SKIP\_PAGE：表示JSP页面应该立刻停止执行，并将所有的输出立刻回传到浏 览器上

·EVAL\_PAGE：表示JSP页面可以正常地运行完毕

1. release( )：将标签处理类所产生或是获得的资源全部释放，并等待用户下次继续 使用。

这些方法构建起来的执行流程如下所示：



**定义有标签体的标签库：**

要想定义一个带有标签体的标签，在编写\*.tlb文件时<body-content>中的内容应为JSP。带有标签体的标签指的是双标签。

**开发迭代标签：**

在程序开发中迭代输出是常见的一种输出形式，如在输出集合时，需要使用迭代输 出，但这样一来JSP页面就出现了Scriptlet，为了达到JSP页面简洁的效果，可以编 写迭代标签完成迭代输出的功能。

**BodyTagSupport类：**

BodyTagSupport类是TagSupport类的子类，通过继承BodyTagSupport类实现的标签可 以直接处理标签体内容的数据。

**TagExtraInfo类和VariableInfo类：**

使用TagExtraInfo类和VariableInfo类可以完成类似于<jsp:useBean>标签功能的标签开发。

**使用BodyTagSupport开发迭代输出标签：**

使用BodyTagSupport也可以方便的开发迭代标签。

**简单标签（SimpleTagSupport类）：**

1. 在JSP1.2之前如果想要进行标签库的开发，需要继承TagSupport或BodyTagSuppor t类，同时还必须覆写其中的doStartTag( )、doAfterBody( )、doEndTag( )方法，并且 还需要非常的清除这些方法的返回值类型及所表示的含义，如SKIP\_BODY等，这样 开发标签实在是太麻烦了。为此，到了JSP2.0之后，为了简化标签开发的复杂度， 增加了一个制作简单标签库的SimpleTagSupport类，直接覆写其中的doTag( )方 法 即可开发标签。
2. 如果直接使用SimpleTagSupport类完成迭代标签的开发，也有一个困难，在之前讲 解过，如果现在是通过TagSupport类完成迭代标签的开发，则可以通过控制返回值 的方式，让程序循环执行doAfterBody( )方法，但是SimpleTagSupport类实现的doTag( ) 方法并没有返回值，如果想要达到循环的效果，就必须通过JspFragment类完成控 制，在SimpleTagSupport类中存在一个getJspBody( )方法，此方法可以返回一个 JspFragment对象，利用此对象中的invoke( )方法即可完成标签体内容的输出。

**DynamicAttributes接口：**

1. 可以为自定义的标签定义属性，但当使用这个标签时，只能编写指定个数及名称的 属性。为此，JSP专门提供了DynamicAttributes接口，此接口的主要功能是用于完 成动态属性的设置，即用于在使用一个标签时可以根据自己的需要，任意设置多个 属性。
2. 如果想要实现动态属性，仅靠实现DynamicAttribute接口还是不够的，还需要在标 签描述文件中增加动态属性的标签。