1.添加maven依赖： <dependency>

<groupId>javax.servlet.jsp</groupId>

<artifactId>jsp-api</artifactId>

<version>2.2</version>

</dependency>

然后新建一个java类，继承TagSupport。

在该类中：1）private static final long serialVersionUID = 1L; 定义一个版本id；

2）重写doStartTag()

2.写tld文件

3.引入标签：<%@ taglib uri="http://www.ztemt.com/tag/ztemt" prefix="rbac"%>

4.使用：<rbac:privilege text="yes">

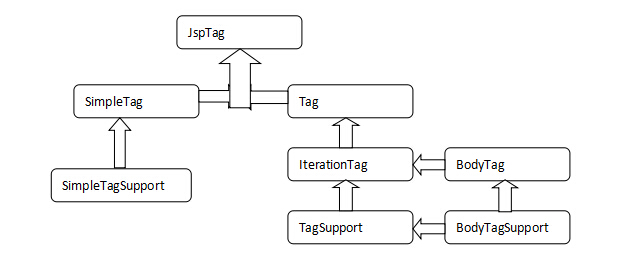
这里是管理员才能看到的内容！

</rbac:privilege>

如果使用rbac的权限，只要把text的值传入对于的权限，比如delOpt删除，然后在doStartTag()里面进行对user是否含有这个delOpt进行判断即可。

从jsp1.1开始就可以在jsp页面中使用自定义标签了，使用自定义标签不但可以实现代码重用，而且可以使jsp代码更简洁。Jsp2.0的标签扩展API中又增加了SimpleTag接口和其实现类SimpleTagSupport。

标签扩展API层次结构：



TLD文件元素详解：

每个自定义标签都必须在TLD文件中声明，TLD文件只不过是一个XML文件。根元素是<taglib>，它包含一个或者多个<tag>标签，该元素用来声明定制标签。<taglib>元素中只有<tlib-version>元素是必须的，其他都是可选的。

TLD文件中的元素

|  |  |
| --- | --- |
| 标　　签 | 含　　义 |
| <taglib> | TLD文件的根元素 |
| <tlib-version> | 此标签库的版本 |
| <jsp-version> | 此标签库依赖的JSP版本。 |
| <short-name>    <description>  <display-name>  <icon>  <uri>  <validator>  <listener>  <function> | 当在JSP中使用标签时，此标签库首选或者建议的前缀。当然可以完全忽略这个建议  描述信息  图形工具可显示的一个简短名称  图形工具可显示的图标  指定使用该标签库中标签的URI  关于该库的TagLibraryValidator信息  指定事件监听器类  定义一个在EL中使用的函数 |
| <tag> | 定义一个标签 |

<tag>元素的子元素

|  |  |
| --- | --- |
| 标　　签 | 含　　义 |
| <description>  <display-name>  <icon>  <name> | 指定针对标签的信息  开发工具用于显示的一个简短名称  可被开发工具使用的图标  标签名称 |
| <tag-class>    <tei-class> | Java标签处理器类的名称。注意这是处理器类的全限定名称，比如com.xx.tag.TableTag  Javax.servlet.jsp.tagext.TagExtraInfo类的一个可选子类 |
| <body-content>    <variable>  <example> | 此标签的主体部分的内容。其值可为scriptless\tagdependent\empty,默认为empty  定义脚本变量信息  使用该标签例子的可选的非正式描述 |
| <attribute> | 包含了此标签的一个属性的元数据 |

<attribute>的子元素

|  |  |
| --- | --- |
| 标　　签 | 含　　义 |
| <description>  <name> | 有关描述的文本信息  在jsp标签中使用的属性名称 |
| <required>      <rtexprvalue>      <type> | 指定属性是必须的还是可选的，默认为false，表示属性可选。如果该值为true，则jsp页面必须为该属性提供一个值。可能的值true、false、yes、no  指定属性是否能接受请求时表达式的值，默认为false，表示不能接受请求时表达式的值。可能值：true、false、yes、no  属性的数据类型,该元素只能用在当<rtexprvalue>设置为true时。它指定当使用请求时属性表达式（<%= %>）返回类型。默认string |

1. EVAL\_BODY\_INCLUDE：把Body读入存在的输出流中，doStartTag()函数可用（告诉服务器正文的内容，并把这些内容送入输出流 ）
2. EVAL\_PAGE：继续处理页面，doEndTag()函数可用   （让服务器继续执行页面 ）
3. SKIP\_BODY：忽略对Body的处理，doStartTag()和doAfterBody()函数可用  （告诉服务器不要处理正文内容 ）
4. SKIP\_PAGE：忽略对余下页面的处理，doEndTag()函数可用   （让服务器不要处理剩余的页面 ）
5. EVAL\_BODY\_TAG：已经废止，由EVAL\_BODY\_BUFFERED取代
6. EVAL\_BODY\_BUFFERED：申请缓冲区，由setBodyContent()函数得到的BodyContent对象来处理tag的body，如果类实现了BodyTag，那么doStartTag()可用，否则非法   （）

标签中静态常量:   
EVAL\_BODY\_INCLUDE：告诉服务器正文的内容，并把这些内容送入输出流   
SKIP\_BODY：告诉服务器不要处理正文内容   
EVAL\_PAGE：让服务器继续执行页面   
SKIP\_PAGE：让服务器不要处理剩余的页面   
EVAL\_BODY\_AGAIN：让服务器继续处理正文内容，只有doAfterBody方法可以返回   
EVAL\_BODY\_BUFFERED：BodyTag接口的字段，在doStartTag()返回   
EVAL\_BODY\_INCLUDE、SKIP\_BODY一般由doStartTag()返回，而EVAL\_PAPGE、SKIP\_PAGE由doEndTag()返回。

他们执行顺序如下：   
doStartTag()→doInitBody()→setBodyContent()→doAfterBody()→doEndTag()

doStartTag()方法可返回EVAL\_BODY\_INCLUDE或SKIP\_BODY，   
如果返回EVAL\_BODY\_INCLUDE则继续执行；   
如果返回SKIP\_BODY则接下来的doInitBody(),setBodyContent(), doAfterBody()三个方法不会被执行，   
而直接执行doEndTag()方法。

setBodyContent()方法用于设置标签体内容，如果在此之前要作一些初始化工作，则在doInitBody()方法中完成。   
标签体内容执行完后，会调用doAfterBody()方法，此方法可返回EVAL\_BODY\_TAG, SKIP\_BODY, EVAL\_PAGE或SKIP\_PAGE。   
如果返回EVAL\_BODY\_TAG则会再次设置标签体内容，直到返回SKIP\_BODY；   
如果返回EVAL\_PAGE则标签体执行完后会继续执行JSP页面中接下来的部分；   
如果返回SKIP\_PAGE，则JSP页面的后续内容将不再执行。

------------------------------------------------------------------------------------------------------------------------------

[**TagSupport与BodyTagSupport的区别**](http://dongguoh.iteye.com/blog/100782)

1 TagSupport与BodyTagSupport的区别  
  TagSupport与BodyTagSupport的区别主要是标签处理类是否需要与标签体交互，如果不需要交互的就用TagSupport，否则如果不需要交互就用BodyTagSupport。  
  交互就是标签处理类是否要读取标签体的内容和改变标签体返回的内容。  
  用TagSupport实现的标签，都可以用BodyTagSupport来实现，因为BodyTagSupport继承了TagSupport。  
  
 2 doStartTag(),doEndTag()  
  
   doStartTag()方法是遇到标签开始时会呼叫的方法，其合法的返回值是EVAL\_BODY\_INCLUDE与SKIP\_BODY,前者表示将显示标签间的文字，后者表示不显示标签间的文字；doEndTag()方法是在遇到标签结束时呼叫的方法，其合法的返回值是EVAL\_PAGE与 SKIP\_PAGE，前者表示处理完标签后继续执行以下的JSP网页，后者是表示不处理接下来的JSP网页  
  
   doAfterBody(),这个方法是在显示完标签间文字之后呼叫的，其返回值有EVAL\_BODY\_AGAIN与SKIP\_BODY，前者会再显示一次标签间的文字，后者则继续执行标签处理的下一步。  
  
   预定的处理顺序是：doStartTag()返回SKIP\_BODY,doAfterBodyTag()返回SKIP\_BODY,doEndTag()返回EVAL\_PAGE.  
  
  如果继承了TagSupport之后，如果没有改写任何的方法，标签处理的执行顺序是：  
   doStartTag() ->不显示文字 ->doEndTag()->执行接下来的网页//不显示内容时不运行doAfterBodyTag()  
  
  如果您改写了doStartTag(),则必须指定返回值，如果指定了EVAL\_BODY\_INCLUDE,则执行顺序是  
   doStartTag()->显示文字->doAfterBodyTag()->doEndTag()->执行下面的网页