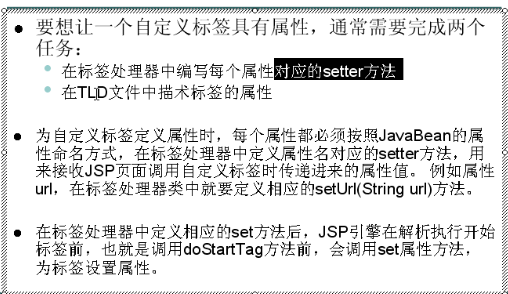
# 一、自定义标签带属性



## 1.下面这个count表示的就是这个标签体执行的次数

**public** **class** KZtagshuxing **extends** SimpleTagSupport {

**int** count ;

**public** **void** setCount(**int** count) {

**this**.count = count;

}

**public** **void** doTag() **throws** JspException, IOException {

//得到jsp标签体

JspFragment jspPage = **this**.getJspBody();

**for** (**int** i = 1; i < count; i++) {

jspPage.invoke(**null**);

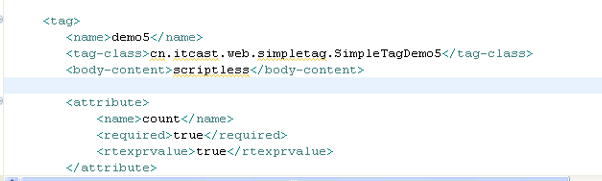
}

**super**.doTag();

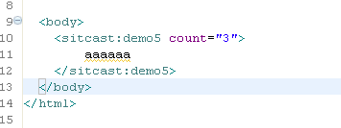
}

}

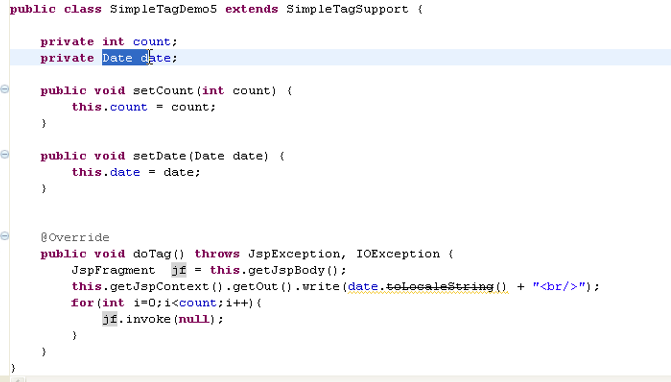
## 2.然后我们到了tId中进行修改，这里的标签体变成了scriptless，而且针对我们的的属性也会添加 内容 attribute ，最后一个是我们的属性count赋值可以是一个表达式，如果是false 那个只能是确定的值



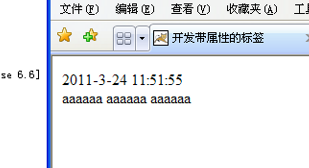
## 3.然后我到了jsp标签的页面



## 4.现在我们还想增加一个属性date

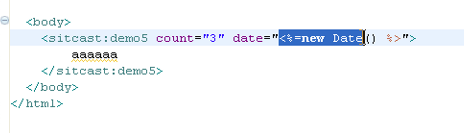




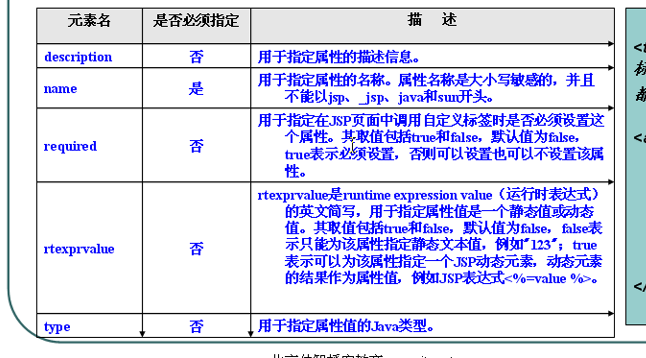




## 4.1然后tid中

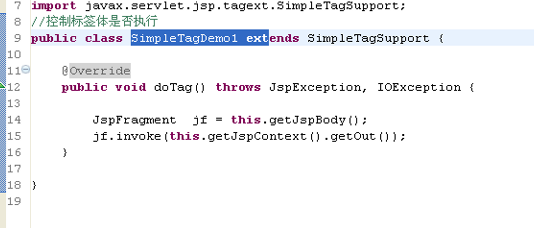


# 二、属性的中tid中的各项参数



# 三、控制标签体是不是执行

## 1.jf.invoke(this.getJspContext( ).getOut() )，没有这个标签体就不会输出了



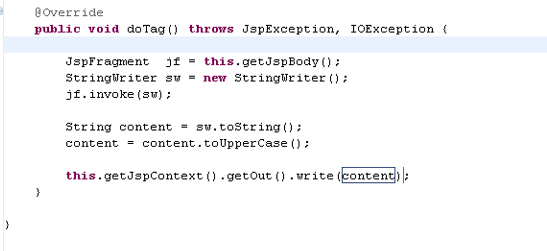
## 2.控制体是scriptless

或者我们的 if.invoke(null);是默认的，为

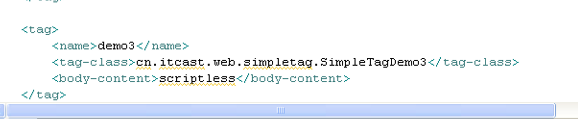
jspPage.invoke(**null**); /jspPage.invoke(this.getJspContext().getOut());

# 四、修改标签体的内容

## 1.先适应StringWriter得到一个缓冲，然后在开始制作一个，通过这个StringWriter的缓冲，去得到东西，



## 2.控制体还是下面的这个



# 五、简单标签控制整个jsp是不是执行

1.其实就是使用一个异常进行抛出，这样就不会使用了

