自定义标签

步骤:

a.编写标签处理类

b.编写标签描述符

c.导入并使用

a.编写标签处理类

传统方式(JSP1.1):实现javax.servlet.jsp.tagext.Tag接口　　doStartTag()

简单方式(JSP2.0):实现javax.servlet.jsp.tagext.SimpleTag接口　doTag()

如果jsp在编译阶段发现了自定义标签,就会交给doStartTag()和doTag()

b.编写标签描述符t　l　d

编写建议:可以仿照一个 其他语言标签(el,jstl)的tld文件

c.导入并使用

myTag.tld只能导入到WEB-INF或子目录下 (特例排除,不能是WEB-IN/lib和classes)

使用:

引入具体的要使用的tld文件

<%@taglib uri="..." prefix="..."%>

uri:每个tld文件的唯一标识符

prefix:使用tld标签时的前缀

具体的使用:

空标签(没有标签体的标签)

<d:foreach></d:foreach>

<d:foreach/>

带标签体

<d:foreach>xxx</d:foreach>

带属性

<d:foreach 属性名="属性值">xxx</d:foreach>

<d:foreach collection="${students}">xxx</d:foreach>

嵌套:

<d:foreach>

<d:foreach>xxx</d:foreach>

</d:foreach>

实际开发步骤

a.编写标签处理类

确保项目有tomcat环境

Tag接口：

doStartTag()：标签处理类的核心方法

该方法有以下两个返回值0/1

int SKIP_BODY=0　标签体不会被执行

int EVAL_BODY_INCLUDE=1　标签体会被执行

doEndTag()：标签执行完毕之后的方法，例如可以让标签在执行完毕后再执行一次

int SKIP_PAGE=5　后面的jsp页面内容不被执行

int EVAL_PAGE=6　后面的JSP页面内容继续执行

Tag接口中的所有方法执行顺序

当JSP容器(Tomcat，jetty)在将.jsp翻译成.servlet(.java)的时候

就会依次执行setPageContext() setParent() doStartTag() doEndTag() realease()

javax.servlet.jsp.tagext.IterationTag接口(Tag的子接口)

如果有循环；用IterationTag

如果没有循环：用Tag

方法：

doAfterBody()：当标签体执行完毕后的操作，通过返回值决定：

要么重复执行 EVAL_BODY_AGAIN =2

要么不再执行 KSIP_BODY=0

b.编写标签描述符

myTag.tld

c.导入并使用

BodyTag接口：如果在标签体 被显示之前，进行一些其他的“额外”的操作

hello hello hello--->HELLO HELLO HELLO

包含属性

int EVAL_BODY_BUFFERED=2；是doStartTag()的第三个返回值，代表缓冲区(BodyContent)

BodyContent是abstract，具体使用时需要使用实现类 BodyContentImpl(jasper.jar)

缓冲区(BodyContent)的使用用途:hello->HELLO

如果返回值是EVAL_BODY_BUFFERED，则服务器会自动将标签体需要显示的内容放入缓冲区中

因此,如果要更改最终显示结果,只需要从缓冲区获取原来的数据进行修改即可

就是通过getxxx()获取原来的数据，自己修改，输出getEnclosingWriter();

简单方式

SimpleTag

最大的简化:

将传统的doStartTag() doEnding doAfterTag() 等方法 简化成一个通用的doTag()方法

doTag():传统方式可以对标签的最终显示进行修改,hello-->HELLO，核心是有一个缓冲区。

但是,简单方式没有提供“缓冲区”。

JspFragment类:代表一块JSP元素(除了scriptlet,因此简单方式的tld文件不能是jsp)

JspFragment中有一个invoke(Writer varl)方法,入参是流,即如果显示内容,只需修改

此流.每执行一次invoke(),会执行一次标签体

SimpleTagSupport可以通过getJspBody()获取JspFragment对象。

SimpleTagSupport可以通过getJspContext()方法 可以获取 jsp一些内置对象:

getJspContext()->JspContext()->转成子类PageContext->PageContext就是所有JSP内置对象

的入口,即可以获取一切内置对象.