**刚刚学了自定义标签,也知道很多公司,都有自己的自定义的一套标签库,比如擅长做金融产品的公司,一般都有自己的一套自己写的针对金融产品的标签库,但是像我们平时做的中小系统,就没必要都自己去写一套标签库了,我们有现成的会使用就好!比如很常用的JSTL!**

**JSP Standard Tag Library(简称JSTL)，是一套预先定义好、协助程序设计人员简化JSP网页制作的标签函数库，规格包含了各种网页运作所需的运用，例如循环、流程控制、输入/出、文本格式化，甚至XML文件处理及数据库访问操作均为其涵盖范围。**

**到JSP2.0后，通过结合使用JSTL和EL，可以开发出没有java代码的JSP，这使得JSP的开发非常简单，非Java程序员都能胜任。另外，通过使用JSTL可以使编程的代码量大大减少。**



**设置JSTL运行环境**

**下载JSTL的jar包，下载地址：http://tomcat.apache.org/taglibs/standard/**

**Impl: taglibs-standard-impl-1.2.5.jar JSTL实现类库**

**Spec: taglibs-standard-spec-1.2.5.jar JSTL标准接口**

**EL: taglibs-standard-jstlel-1.2.5.jar JSTL1.0标签-EL相关**

**Compat: taglibs-standard-compat-1.2.5.jar 兼容版本**

**如果不使用JSTL1.0标签,可以忽略taglibs-standard-jstlel包,**

**taglibs-standard-compat包,应该是兼容以前版本标签库,**

**所以一般只需要 taglibs-standard-impl 和 taglibs-standard-spec 两个jar包**

**也可以不用下载,不下载可以到tomcat目录里去copy也是可以的!**

**下载的JSTL以jar包形式存在，直接将其保存到WEB-INF\lib当中。**

**用WINRAR软件打开jar包，将其中META-INF文件夹中的几个主要标签配置文件(c.tld(主要是这个核心)、fmt.tld、fn.tld(要用到函数这个也要)、sql.tld、x.tld)保存在WEB-INF文件夹中（或其子目录里）。**

**使用JSTL之前，必须引用taglib指令声明网页所要使用的标签种类 。**

**<%@taglib prefix=tabName uri=uriString %>**

**prefix代表标签种类的前缀词**

**uri代表标签的URI**

**自定义标签库要声明,而声明有下面两种方式：**

**1、直接指定路径**

**<%@ taglib prefix="mytag" uri="/WEB-INF/testcomp.tld" %>**

**2、间接引用**

**<%@ taglib prefix="tt" uri="http://testcomp.com/testcomp/core" %>**

**这样做需要在web.xml中增加下面的内容：**

**<jsp-config>**

**<taglib>**

**<taglib-uri>http://testcomp.com/testcomp/core</taglib-uri>**

**<taglib-location>/WEB-INF/testcomp.tld</taglib-location>**

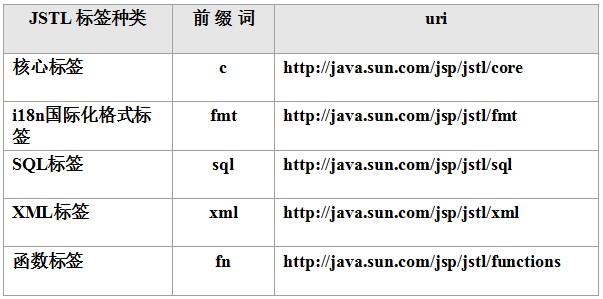
**</taglib>**

**</jsp-config>**

**很明显，间接的方式更加灵活，当.tld文件改变存放位置时只需要修改web.xml，而不用去修改每个jsp页面。**

**注意:在现在的新版本和编程环境, 如果没发现标签出问题的话,环境是可以直接通过反射从jar包里获取tld配置, 可以省略copy配置dlt文件,也可以省略xml中的配置! 只需要在jsp页面头上加标签指令就OK, 但uri地址不能错!**

**标签前缀词以及uri**



**使用核心标签**



**<c:out>**

**语法1：未包含主体(body)**

**<c:out value= "value" [escapeXml= "{true|false}"]**

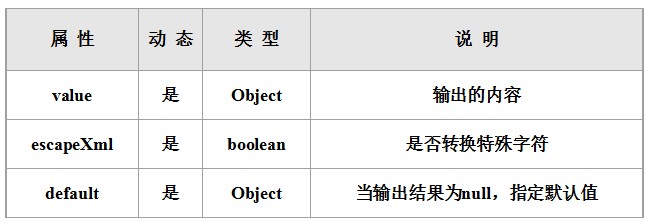
**[default= "defaultValue"] />**

**语法2：包含主体(body)**

**<c:out value= "value" [escapeXml= "{true|false}"]>**

**default value**

**</c:out>**



**举个例子吧,显示:<<Java>>**

**<c:set>**

**语法1：将value值储存至范围变量varName**

**<c:set value="value" var="varName"**

**[scope="{page|request|session|application}"]/>**

**语法2：将本体(body)内容设定储存至范围变量varName**

**<c:set var="varName" [scope="{page|request|session|application}"]>**

**body content**

**</c:set>**

**语法3：将value值储存至目标对象target的属性propertyName**

**<c:set target="target" property="propertyName" value="值"/>**

**语法4:将本体(body)内容储存至目标对象target的属性propertyName**

**<c:set target="target" property="propertyName">**

**body content**

**</c:set>**



**举例吧**

**<c:remove>**

**用来移除某个范围变量的內容值**

**<c:remove**

**var="varName"**

**[scope="{page|request|session|application}"]/>**