# JSTL简介

## 认识JSTL

1. 使用标签库可以避免过多的Scriptlet代码，但是如果采用自定义标签库的做法，会 非常的繁琐且不通用，所以此时可以借助一些开源工具使用一些公共的标签来完成代码 的开发，而JSTL就是这种使用广泛的通用标签。JSTL（JSP Standard Tag Library，JSP标准标签库）是一个开发源代码的标签组件，由Apache开发。
2. JSTL主要由5大标签库组成：

|  |  |  |  |
| --- | --- | --- | --- |
| **JSTL** | **标记名称** | **标签配置文件** | **描述** |
| 核心标签库 | c | c.tld | 定义了属性管理、迭代、判断、输出 |
| SQL标签库 | sql | sql.tld | 定义了查询数据库操作 |
| XML标签库 | xml | x.tld | 用于操作XML数据 |
| 函数标签库 | fn | fn.tld | 提供了一些常用的操作函数，如字符串函数 |
| 118N格式标签库 | fmt | fmt.tld | 格式化数据 |

在实际开发中用到的标签库主要是c.tld和fn.tld，即核心标签库和函数标签库。

## 安装JSTL

JSTL可以从<http://tomcat.apache.org/taglibs/>下载， 这里以1.2版本为例。下载时选择 的是通过jar包的形式下载的，下载完成后，将其中META-INF文件夹中的几个主要标 签配置文件（c.tld、fmt.tld、fn.tld、sql.tld、x.tld）保存在WEB-INF文件夹中。



另外，还需在项目中的导入以下包：



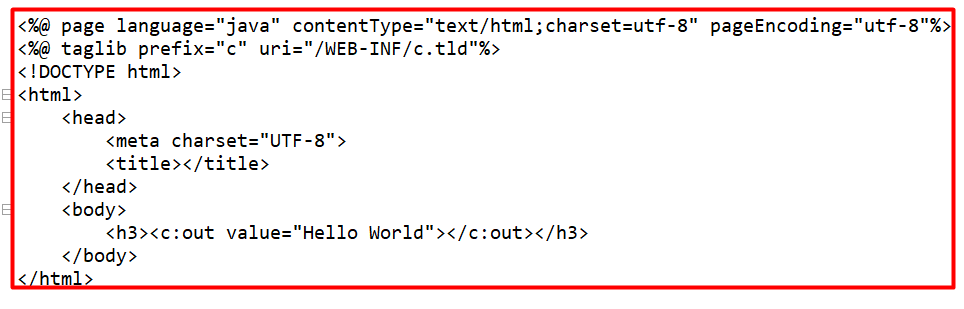
此时，即可在JSP页面使用JSTL标签进行项目开发（但需要注意，配置完JSTL后需要 Tomcat服务器，这样才能够加载到JSTL）。

## 使用JSTL

使用JSTL需要通过taglib指令引入，引入语法如下：

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

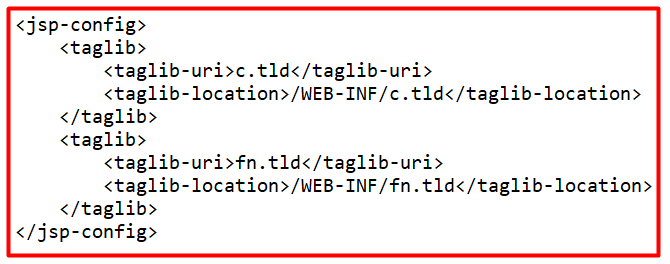
示例如下：hello\_jstl.jsp



本示例通过tablib指令导入了c.tld文件，之后利用JSTL提供的<c:out>标签输出了一个 字符串。

## 映射标签文件

在使用<%@taglib%>标签通过uri属性引入了配置在WEB-INF中的c.tld文件，如果觉得 麻烦，可以在web.xml文件中设置一个标签文件的映射名称。



重启服务器后，再次使用<%@taglib%>标签时uri属性指定别称即可。



# 核心标签库

核心标签库（c.tld）是JSTL中最重要的部分，也是在开发中最长使用到的部分，在核心 标签库中主要完成流程控制、迭代输出等操作。核心标签库中主要标签如下所示：

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **功能分类** | **标签名称** | **描述** |
| 1 | 基本标签 | <c : out> | 输出内容 |
| 2 | <c : set> | 设置属性内容 |
| 3 | <c : remove> | 删除指定属性 |
| 4 | <c : catch> | 异常处理 |
| 5 | 流程控制标签 | <c : if> | 条件判断 |
| 6 | <c : choose> | 多条件判断，可以设置<c:when>和<c:otherwise>标签 |
| 7 | 迭代标签 | <c : forEach> | 输出数组、集合 |
| 8 | <c : forTokens> | 字符串拆分及输出操作 |
| 9 | 包含标签 | <c : import> | 将一个指定的路径包含到当前页面 |
| 10 | 生成URL标签 | <c : url> | 根据路径和参数生成一个新的URL |
| 11 | 客户端跳转 | <c : redirect> | 实现客户端跳转 |

## <c:out>

<c:out>标签用于输出内容，与表达式语言、表达式输出Scriptlet功能相同。语法为：

**<c:out value=”内容” [default=”默认值“] />**

或

**<c:out value=”内容”>**

**默认值**

**</c:out>**

标签属性如下：

|  |  |  |
| --- | --- | --- |
| 属性名称 | 是否支持EL | 描述 |
| value | true | 设置要显示的内容 |
| default | true | 如果要显示的value内容为null，则显示default定义的内容 |
| escapeXml | true | 是否转换为字符串，例如将“>”转换成“&gt;”，默认为true |

## <c:set>

<c:set>标签用于将属性保存在4种属性范围中，与setAttribute( )方法功能类似。语法 为：

**<c:set var=”属性名称” value=”属性内容” [scope=”page | request | session | application”] />**

或

**<c:set var=”属性名称” [scope=”page | request | session | application”]>**

**属性内容**

**</c:set>**

以上是设置属性到4大保存范围中，如果要将某个内容设置为对象的某个属性的值，前 提是该对象是被保存在4中属性范围中，此时语法为：

**<c:set value=”属型内容” target=”属性名称” property=”属性名称”/>**

或

**<c:set target=”属性名称” property=”属性名称”>**

**属性内容**

**</c:set>**

标签属性如下：

|  |  |  |
| --- | --- | --- |
| **属性名称** | **是否支持EL** | **描述** |
| value | true | 设置属性的内容，如果为null则表示删除属性 |
| var | false | 设置属性名称 |
| scope | true | 设置属性的保存范围，如不设置，则默认为page |
| target | true | 存储的目标属性 |
| property | true | 指定的target属性 |

## <c:remove>

<c:remove>标签用于删除指定范围中的属性，功能与removeAttribute( )方法相同，语 法为：

**<c:remove var=”属性名称” [scope=”page|request|session|application”/>**

属性如下：

|  |  |  |
| --- | --- | --- |
| **属性名称** | **是否支持EL** | **描述** |
| var | false | 要删除的属性的名称 |
| scope | false | 要删除的属性的所属范围，如不指定，则默认为page |

## <c:catch>

<c : catch>标签用于处理程序中产生的异常，并进行相关的异常处理。语法如：

**<c : catch [var=”异常信息对象”]>**

**有可能发生异常的代码**

**</c : catch>**

|  |  |  |
| --- | --- | --- |
| **属性名称** | **是否支持EL** | **描述** |
| var | false | 用来保存异常信息的属性名称 |

标签属性如下：

## <c:if>

<c : if>标签用于完成单分支if语句的功能，其语法为：

**<c:if test=”判断条件” var=”属性名称” scope=”page|request|session|application”/>**

或

**<c:if test=”判断条件” var=”存储判断结果” [scope=”page|request|session|application”]>**

**判断条件为真时执行的语句；**

**</c:if>**

标签属性如下：

|  |  |  |
| --- | --- | --- |
| **属性名称** | **是否支持EL** | **描述** |
| test | true | 用于指定判断条件，如果条件为true，则执行标签体的内容 |
| var | false | 保存判断的结果 |
| scope | false | 指定判断结果的保存范围，如不指定，则默认为page |

## <c:choose>、<c:when>、<c:otherwise>

<c : choose>、<c : when>、<c : otherwise>可以实现多分支判断语句。语法为：

**<c:choose>**

**<c:when>**

**<c:otherwise>**

**</c:choose>**

<c:choose>只能作为<c:when>、<c:otherwise>的父标签，其本身并不进行判断，要想 进行判断还需要依靠<c:when>、<c:otherwise>标签。一个<c:choose>标签中可以包含 多个< c :when>标签和一个<c : otherwise>标签。

<c : when>标签语法为：

**<c:when test=”判断条件”>**

**满足条件时执行的语法**

**</c:when>**

<c : otherwise>标签语法为：

**<c:otherwise>**

**当所有<c : when>都不满足时，执行本标签体内容**

**</c:otherwise>**

可以看出，<c:choose>可以完成类似于switch语句的功能，<c:when>好比一个个case 语句，<c:otherwise>好比default语句，并且它必须写在<c:when>标签之后。

## <c:forEach>

<c:forEach>标签的主要功能是进行迭代，可以对数组或集合进行迭代，功能与foreach 语句类似，其语法如下：

**<c:forEach [var=”每一个对象的属性名称”] items=”集合” varStatus=”保存相关成 员信息”] [begin=”集合的开始输出位置” [end=”集合的结束输出位置”] [step=”每次增长的步长”]>**

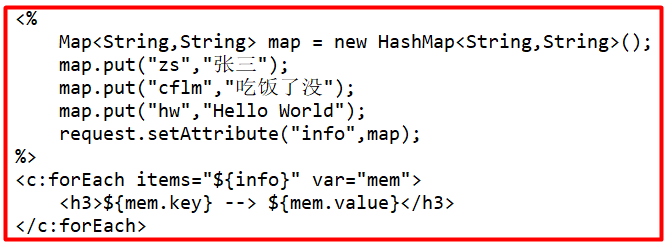
**标签体内容**

**</c:forEach>**

标签属性如下：

|  |  |  |
| --- | --- | --- |
| **属性名称** | **是否支持EL** | **描述** |
| var | false | 用来存放集合中的每一个对象 |
| items | true | 可以是数组、Collection（List、Set）及Map |
| varStatus | false | 用于存放当前对象的成员信息 |
| begin | true | 集合的开始位置，默认从0开始 |
| end | true | 集合的结束位置，默认为集合的最后一个元素 |
| step | true | 每次迭代的间隔数，默认为1 |

在使用<c:forEach>标签迭代Map集合时需要注意，所以保存在Map集合中的对象都是 通过Map.Entry的形式保存的，所以要想分离出key和value，则需要通过Map.Entry提 供的getKey( )和getValue( )方法。示例如下：



## <c:forTokens>

<c : forTokens>标签用于拆分字符串，类似于String类中的split( )方法，可以按照拆分 符拆分一个字符串并进行迭代输出。语法为：

**<c : forTokens items=”要拆分的字符串” delims=”字符串分割符” [var =”存放每一个 拆分的字符串”] [varStatus=”存放当前对象的相关信息” [begin=”输出位置”] [end=”结束位置”] [step=”输出间隔”]>**

**标签体内容**

**</c:forTokens>**

标签属性如下：

|  |  |  |
| --- | --- | --- |
| **属性名称** | **是否支持EL** | **描述** |
| var | false | 用来存放每一个被拆分的字符串 |
| items | true | 要拆分的字符串 |
| delims | false | 定义分隔字符串的内容 |
| varStatus | false | 存放当前对象的相关信息 |
| begin | true | 开始的输出位置，默认从0开始 |
| end | true | 结束的输出位置，默认为最后一个成员 |
| step | true | 迭代输出的间隔 |

## <c:import>

<c:import>标签可以将其他页面的内容包含进来一起显示，与<jsp:include>标签的功能类 似，但是与<jsp:include>标签不同的是，<c:import>还可以包含外部的页面（跨域，即第 三方网站页面）。其语法如下：

**<c:import url=”URL地址” [context=”上下文路径”] [var=”保存内容的属性名称”]**

**[scope=”[page|request|session|application]”] [charEncoding=”字符编码”]**

**[varReader=”以Reader方法读取内容”]>**

**标签体内容**

**[<c:param name=”参数名称” value=”参数内容”/>]**

**</c:import>**

标签属性如下：

|  |  |  |
| --- | --- | --- |
| **属性名称** | **是否支持EL** | **描述** |
| url | true | 指定要包含的文件路径 |
| context | true | 如果要访问在同一Web容器下的其他资源时设置，必须以“/”开头 |
| var | false | 储存导入的文件内容 |
| scope | false | 定义var的保存范围，如不指定，则默认为page范围 |
| charEncoding | true | 定义的字符编码 |
| varReader | false | 储存导入的文件内容，以Reader类型导入 |

## <c:url>

<c:url>可以在给定的url地址的基础上产生一个新的URL地址，其语法为：

**<c : url value=”操作的url” [var=”保存属性的名称”] [scope=”属性范围”]/>**

如果想通过地址重写的方式向给定的url追加参数的话，可以使用有参形式：

**<c : url value=”操作的url” [ var=”保存属性的名称”] [scope=”属性范围”]>**

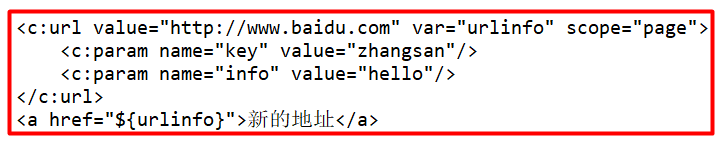
**<c : param name=”参数名” value=”参数值”/>**

**</c : url>**

标签属性如下：

|  |  |  |
| --- | --- | --- |
| **属性名称** | **是否支持EL** | **描述** |
| value | true | 给定的URL地址 |
| context | true | 如果要访问在同一个Web容器下的其他资源时设置，必须以“/”开头 |
| var | false | 储存导入的文件内容 |
| scope | false | 定义var的保存范围，如不指定，则默认为page范围 |

示例如下：



生成的地址栏采用地址栏重写的方式将设置的两个参数加在地址后面，生成的新地址是 “http://www.baidu.com?key=zhangsan&info=hello”。

## <c:redirect>

在学习JSP内置对象时讲解果可以通过response.sendRedirect( )方法进行客户端跳转操 作，<c:redirect>标签的功能也可以完成这样的功能。语法如下：

**<c:redirect url=”跳转的地址” context=”上下文路径”/>**

或

**<c:redirect url=”跳转的地址” context=”上下文路径”>**

**<c:param name=”参数名称” value=”参数内容”/>**

**</c:redirect>**

标签属性如下：

|  |  |  |
| --- | --- | --- |
| **属性名称** | **是否支持EL** | **描述** |
| url | true | 跳转的地址 |
| context | true | 如果要访问在同一个Web容器下的其他资源时设置，必须以“/”开头 |

# 函数标签库

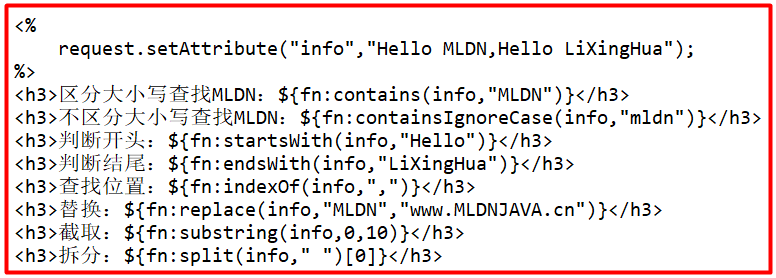
## 常用函数

函数标签库（fn.tld）大部分的操作都是用来处理字符串的，类似于String类中提供的各 个方法，也正是因为这一点，所以在开发中函数标签库也经常使用。函数标签库的主要 操作函数如下：

|  |  |
| --- | --- |
| **函数标签名称** | **描述** |
| ${fn:contains( )} | 查询某个字符串是否存在，区分大小写 |
| ${fn:containsIngoreCase( )} | 查询某个字符串是否存在，不区分大小写 |
| ${fn:startsWith( )} | 判断是否以指定的字符串开头 |
| ${fn:endsWith( )} | 判断是否以指定的字符串结尾 |
| ${fn:toUpperCase( )} | 全部转为大写显示 |
| ${fn:toLowerCase( )} | 全部转为小写显示 |
| ${fn:substring( )} | 字符串截取 |
| ${fn:split()} | 字符串拆分 |
| ${fn:join( )} | 字符串连接 |
| ${fn:escapeXml( )} | 将<、>、” 、’等替换成转义字符 |
| ${fn:trim( )} | 去掉字符串左右空格 |
| ${fn:replace( )} | 字符串替换操作 |
| ${fn:indexof( )} | 查找指定的字符粗位置 |
| ${fn:substringBefore( )} | 截取指定字符串之前的内容 |
| ${fn:substringAfter( )} | 截取指定字符串之后的内容 |

这些函数都是在EL中使用的。

## 示例



运行结果如下：



# 国际化标签库

国际化标签库（fmt.tld）可以对程序进行国际化显示、对数字进行格式化显示、对时间 日期进行格式化性显示。但国际化标签库在实际开发中的使用频率非常低，所以有兴趣 的话可以参考教材书。

# SQL标签库

SQL标签库（sql.tld）是与数据库操作有关的标签，使用这些标签轻松的进行数据库的 更新和查询操作以及事务管理等。但Web项目的开发是分层的，所以在实际开发中SQL 标签库几乎不会使用，有兴趣的话可以参考教材书。

# XML标签库

XML标签库（x.tld）可以对XML文件进行解析处理。但在实际开发中XML标签库几乎 不会使用，有兴趣的话可以参考教材书。