**JSP标准标签库**

**1.C标签（核心）：最重要的使用最多的标签**

要使用jsp标签编程就要安装JSTL。JSTL是以jar包形式存在的直接将jar包保存在WEB-INF\lib目录之中。之后可以解压该jar文件，然后将标签配置文件（c.tld,fmt.tld,fn.tld,sql.tld,x.tld）保存在WEB-INF文件夹中。

范例：利用c标签进行输出

<head>

<%@ page language=*"java"* contentType=*"text/html; charset=GBK"*

pageEncoding=*"GBK"*%>

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

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=GBK"* />

<title>Hello\_JSTL</title>

</head>

<body>

<h3><c:out value = *"Hello wenpan!!!"*></c:out></h3>

</body>

</html>

本程序通过标签命令导入了c.tld文件，之后利用jstl提供的<c:out>输出了一个字符串。

也可以通过web.xml文件配置一个标签文件的映射名称。以方便<%@ taglib uri = “映射路径/\*.tld文件的路径”/>中uri的编写。

**核心标签库：**在核心标签库中主要就是完成流程控制，迭代输出等操作。

1. <c:out>标签 ：标签中value为要输出的内容值。Default为当要输出的内容不存在的时候，显示的内容。

范例：

<head>

<%@ page language=*"java"* contentType=*"text/html; charset=GBK"*

pageEncoding=*"GBK"*%>

<%@ taglib uri = *"http://www.wenpan.cn/jstl/core"* prefix = *"c"* %>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=GBK"* />

<title><c :out >输出</title>

</head>

<body>

<%

pageContext.setAttribute("info", "<www.wenpan.com>") ;

%>

<h3>属性存在：<c:out value = *"*${info }*"*/></h3>

<h3>属性不存在：<c:out value = *"*${ref }*"* default = *"没有此内容"*/></h3>

<h3>属性不存在：<c:out value = *"*${ref}*"*>

没有此内容

</c:out></h3> <!-- 第二种属性不存在的默认输出方式 -->

</body>

</html>

在以上程序中使用了两种输出方式:一种是有标签体的输出，一种是没有标签体的输出。也使用了两种当要输出的属性内容不存在的时候的默认输出方式。

1. <c:set>标签：可以通过该标签设置相应属性的内容。

范例：

<head>

<%@ page language=*"java"* contentType=*"text/html; charset=GBK"*

pageEncoding=*"GBK"*%>

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

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=GBK"* />

<title></title>

</head>

<body>

<c:set var = *"info"* value = *"文攀你好！"* scope = *"request"*/> <!-- 将value的值设置到var中 -->

<h3>属性内容：${info }</h3>

</body>

</html>

在以上程序中就进行了属性info的设置，然后对info属性的输出。

当然该标签也可以进行将内容设置到javabean中，此时就需要通过target和property进行操作。

范例:将属性设置到Javabean中

</head>

<body>

<% //定义一个Javabean

SimpleInfo sim = **new** SimpleInfo() ; //实例化SimpleInfo对象

request.setAttribute("simple", sim) ; //设置属性

%>

<!-- 将value内容设置到simple对象的content属性中 -->

<c:set value = *"Hello wenpan!!!"* target = *"*${simple }*"* property = *"content"*/>

<h3>属性内容：${simple.content}</h3>

</body>

</html>

在以上程序中，是将value的值设置到target对象的content属性中。

1. <c:remove>标签

格式：<c:remove var = “要删除的属性名称” scope = “属性范围”/>

1. <c:catch>标签

格式：<c:catch var = “保存异常信息的属性名称”>

有可能出现异常的语句

</c:catch>

范例：使用jstl进行异常处理

<%@ page language=*"java"* contentType=*"text/html; charset=GBK"*

pageEncoding=*"GBK"*%>

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

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=GBK"* />

<title>捕获异常</title>

<body>

<c:catch var = *"errmsg"*>

<%=

10/0 //在此处产生异常

%>

</c:catch>

<h3>异常信息${errmsg}</h3>

</body>

</html>

**5.<c:if>标签（最重要的标签之一）**

格式1 没有标签体：<c:if test = “”判断条件 var = “存储判断结果” scope = “范围”/>

格式2 有标签体：<c:if test = “”判断条件 var = “存储判断结果” scope = “范围”>

满足条件时执行的语句。（若条件为true则执行标签体）

</c:if>

范例：判断操作

<head>

<%@ page language=*"java"* contentType=*"text/html; charset=GBK"*

pageEncoding=*"GBK"*%>

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

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=GBK"* />

<title></title>

</head>

<body>

<c:if test=*"*${param.ref=='mldn'}*"* var = *"res1"* scope = *"request"*>

<h2>欢迎${param.ref}光临</h2>

</c:if>

<c:if test=*"*${10 < 30 }*"* var = *"res2"* scope = *"request"*>

<h3>10比30小</h3>

</c:if>

</body>

</html>

6.<c:choose>,<c:when>,<c:otherwise>标签

Choose标签只能作为when标签和otherwise标签的父标签出现。When标签而已出现一次或多次，otherwise标签可以出现0次或1次。When标签相当于switch--case中的case，otherwise相当于default，choose相当于switch。

范例：多条件判断

<head>

<%@ page language=*"java"* contentType=*"text/html; charset=GBK"*

pageEncoding=*"GBK"*%>

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

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=GBK"* />

<title></title>

</head>

<body>

<%

pageContext.setAttribute("num", 20) ;

%>

<!-- 只要有一个when的条件满足就不再往下判断 -->

<c:choose>

<c:when test=*"*${10 > num }*"*>

num的值大于10！

</c:when>

<c:when test = *"*${20 == num }*"*>

num的值等于20！

</c:when>

<c:when test=*"num* < *30 "*>

num的值小于30！

</c:when>

<c:otherwise>

<h3>没有一个条件满足！</h3>

</c:otherwise>

</c:choose>

</body>

</html>

在以上程序中，只要有一个when条件满足则就不会继续往下判断。Otherwise要写在when后。

**7.<c:forEach>标签（最重要的标签之一）**

forEach标签即是对集合或数组中的每一个元素进行操作。

格式：**<c:forEach var = “每一个要操作的对象属性名称” items = “集合” begin = “集合开始位置” end = “集合结束位置” step = “每次增长的步长”>**

**<c:forEach>**

范例：输出数组

<head>

<%@ page language=*"java"* contentType=*"text/html; charset=GBK"*

pageEncoding=*"GBK"*%>

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

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=GBK"* />

<title></title>

</head>

<body>

<%

String inf[] = {"hello","wenpan","nihao","xiaoming","nihaoma","wohenhao"} ; //定义数组

pageContext.setAttribute("info", inf) ; //将数组保存

%>

<h3>输出全部：

<c:forEach items = *"*${info}*"* var = *"param"* step = *"1"*>

${param}、

</c:forEach>

</h3>

<h3>隔一个输出：

<c:forEach var = *"param"* items = *"*${info}*"* step = *"2"*>

${param}、

</c:forEach>

</h3>

<h3>从第二个开始输出：

<c:forEach var = *"param"* items = *"*${info}*"* step = *"1"* begin = *"1"*>

${param}、

</c:forEach>

</h3>

</body>

</html>

范例：输出集合

<?xml version=*"1.0"* encoding=*"GBK"* ?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<%@page import=*"java.util.\*"*%>

<html xmlns=*"http://www.w3.org/1999/xhtml"*>

<head>

<%@ page language=*"java"* contentType=*"text/html; charset=GBK"*

pageEncoding=*"GBK"*%>

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

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=GBK"* />

<title></title>

</head>

<body>

<%

List list = **new** ArrayList() ; //创建一个集合

list.add("hello ") ;

list.add("wenpan") ;

list.add("how ") ;

list.add("are ") ;

list.add("you") ;

pageContext.setAttribute("info", list) ;

%>

<h2>全部输出：

<c:forEach var = *"each"* items = *"*${info }*"* step = *"1"*>

${each}

</c:forEach>

</h2>

<h2>从第一个开始输出二个：

<c:forEach var = *"each"* items = *"*${info }*"* step = *"2"* begin = *"1"* end = *"3"*>

${each }

</c:forEach>

</h2>

</body>

</html>

范例：输出map集合

<?xml version=*"1.0"* encoding=*"GBK"* ?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<%@page import=*"java.util.\*"*%>

<html xmlns=*"http://www.w3.org/1999/xhtml"*>

<head>

<%@ page language=*"java"* contentType=*"text/html; charset=GBK"*

pageEncoding=*"GBK"*%>

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

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=GBK"* />

<title></title>

</head>

<body>

<%

Map<Integer,String> map = **new** HashMap<Integer,String>() ;

map.put(0, "文攀") ;

map.put(1, "周健") ;

map.put(2, "杨冬") ;

map.put(3, "刘挺") ;

pageContext.setAttribute("info", map) ;

%>

<h2>全部输出：

<c:forEach var = *"each"* items = *"*${info}*"* step = *"1"* >

<h3>${each.key }--->${each.value}</h3>

</c:forEach>

</h2>

</body>

</html>

1. <c:forTokens>标签

该标签用于拆分字符串。就像是Sring类中的split方法和循环输出的一种集合。

格式：<c:forTokens items = “输出的字符串” delims = “字符串分隔符” var = “存放每一个字符串变量” begin = “输出位置” end = “结束位置” step = “ 输出间隔” >

</c:forTokens>

范例：使用<c:forTokens>进行输出

<head>

<%@ page language=*"java"* contentType=*"text/html; charset=GBK"*

pageEncoding=*"GBK"*%>

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

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=GBK"* />

<title></title>

</head>

<body>

<%

String str = "hello.wen.pan.ni.hao.ma" ;

pageContext.setAttribute("info", str) ;

%>

<h2>拆分的结果是：

<c:forTokens items=*"*${info}*"* delims=*"."* var = *"con"*>

${con},

</c:forTokens>

</h2>

<h2>拆分结果：

<c:forTokens var = *"con"* items = *"wen:pan:ni:hao:ma:wo:hen:hao:a"* delims = *":"*>

${con},

</c:forTokens>

</h2>

</body>

</html>

1. <c:import>标签：该标签可以将其他页面的内容包含进来一起显示

格式：<c:impot url = “包含地址的url” var = “保存内容的属性名称” charEncoding = “字符编码”> 标签体内容

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

范例：导入标签

<head>

<%@ page language=*"java"* contentType=*"text/html; charset=GBK"*

pageEncoding=*"GBK"*%>

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

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=GBK"* />

<title></title>

</head>

<body>

<!-- 接收参数的页面 -->

<h3>name参数：${param.name }</h3>

<h3>url参数：${param.url }</h3>

</body>

</html>

<head>

<%@ page language=*"java"* contentType=*"text/html; charset=GBK"*

pageEncoding=*"GBK"*%>

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

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=GBK"* />

<title>传递参数的页面</title>

</head>

<body> <!-- 将其他页面（param.jsp）内容包含进来一起显示 -->

<c:import url=*"param.jsp"* charEncoding = *"GBK"*>

<c:param name = *"name"* value = *"wenpan"*/>

<c:param name = *"url"* value = *"www.wenpan.com"*/>

</c:import>

</body>

</html>

以上程序就将param.jsp页面包含了进来，并且未param.jsp页面进行了属性参数的设置，然后一起显示。

1. <c:redirect>标签

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

范例：进行客户端跳转

<head>

<%@ page language=*"java"* contentType=*"text/html; charset=GBK"*

pageEncoding=*"GBK"*%>

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

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=GBK"* />

<title>实现客户端跳转的标签</title>

</head>

<body>

<c:redirect url = *"param.jsp"*>

<c:param name=*"name"* value = *"wenpan"*></c:param>

<c:param name=*"url"* value = *"pange"*></c:param>

</c:redirect>

</body>

</html>

**国际化标签库：**

1. <fmt:setLocale>标签设置区域

范例：设置时区

<head>

<%@ page language=*"java"* contentType=*"text/html; charset=GBK"*

pageEncoding=*"GBK"*%>

<%@ taglib uri = *"http://www.wenpan.cn/jstl/fmt"* prefix = *"fmt"* %>

<%@ page import = *"java.util.\*"* %>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=GBK"* />

<title>设置区域</title>

</head>

<body>

<%

pageContext.setAttribute("date", **new** Date()) ;

%>

<h3>中文显示日期：

<fmt:setLocale value=*"zh\_CN"* />

<fmt:formatDate value=*"*${date }*"* />

</h3>

<h3>英文显示日期：

<fmt:setLocale value=*"en\_US"* />

<fmt:formatDate value=*"*${date }*"* />

</h3>

</body>

</html>

以上程序中value代表地区编码，表示一个java.util.Locale类。

1. <fmt:requestEncoding>标签设置编码

格式：<fmt:requestEncoding value = “编码”/>

**读取资源文件：**

所有的资源文件都是以\*.properties为后缀，所有的内容都按照“key = value”的格式进行编写，在web中资源文件要保存在/WEB-INF/classes文件夹中。

资源文件命名要求和类是一样的，每个单词的首字母要大写。

范例：定义并读取资源文件

1. 定义一个资源文件

name = wenpan

#info = 欢迎{0}光临

info = /u6b22/u8fce{0}/u5149/u4e34/ufff01

读取资源文件时不需要加入后缀

<head>

<%@ page language=*"java"* contentType=*"text/html; charset=GBK"*

pageEncoding=*"GBK"*%>

<%@ taglib uri = *"http://www.wenpan.cn/jstl/fmt"* prefix = *"fmt"* %>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=GBK"* />

<title>读取资源文件</title>

</head>

<body>

<fmt:bundle basename = *"Message"*>

<fmt:message key = *"name"* var = *"nameref"*>

</fmt:message>

</fmt:bundle>

<h3>姓名：

${nameref}

</h3>

<fmt:bundle basename=*"Message"*>

<fmt:message key = *"info"* var = *"inforef"*>

<fmt:param value = *"wenpan"*/>

</fmt:message>

</fmt:bundle>

<h3>信息：${inforef}</h3>

</body>

</html>

在以上程序中通过<fmt:bundle basename>中的basename指定要读取的资源文件名称，通过key指定要读取的资源文件中的属性key值。然后var为保存读取属性的保存变量名称。然后通过value为key设置值。

1. 数字格式化标签（即将数字转换为指定的字符串形式）

**数字格式化：无标签体**

**格式：<fmt:formatNumber value = “数字” type = “number|currency|percent” pattern = “要格式化的格式” minIntegerDigits = “整数位显示的最小长度” maxFractionDigits = “小数位显示的最大长度” minFractionDigits = “小数位显示的最小长度” var = “格式化数字保存属性”/>**

**数字格式化，有标签体**

**格式：<fmt:formatNumber type = “number|currency|percent” pattern = “要格式化的格式” minIntegerDigits = “整数位显示的最小长度” maxFractionDigits = “小数位显示的最大长度” minFractionDigits = “小数位显示的最小长度” var = “格式化数字保存属性”>**

**要格式化的数字**

**</fmt:formatNumber>**

范例：格式化数字

<head>

<%@ page language=*"java"* contentType=*"text/html; charset=GBK"*

pageEncoding=*"GBK"*%>

<%@ taglib uri = *"http://www.wenpan.cn/jstl/fmt"* prefix = *"fmt"* %>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=GBK"* />

<title>格式化数字显示</title>

</head>

<body>

<fmt:formatNumber value = *"325546.12546535552"* maxIntegerDigits = *"6"* maxFractionDigits = *"3"* groupingUsed = *"true"* var = *"num"*>

</fmt:formatNumber>

<h3>格式化数字：${num}</h3> <!-- 最大整数位为6位，最大小数位为3位 -->

<fmt:formatNumber value = *"245545.245454878"* pattern = *"#.###E0"* var = *"num"*/>

<h3>科学计数法：${num}</h3> <!-- 将数字格式化为pattern的形式输出 -->

</body>

</html>

数字的反格式化：（即将格式化好的数字还原为数字本身）

格式：

<fmt:parseNumber value = “格式化好的数字” pattern = “要反格式化的形式” parseLocale = “区域编码” var = “存储结果的属性名称” scope = “范围”/>

范例：数字反格式化

<head>

<%@ page language=*"java"* contentType=*"text/html; charset=GBK"*

pageEncoding=*"GBK"*%>

<%@ taglib uri = *"http://www.wenpan.cn/jstl/fmt"* prefix = *"fmt"* %>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=GBK"* />

<title>数字的反格式化</title>

</head>

<body>

<!-- 即将字符串或科学计数法表示格式的数字转换为数字 -->

<fmt:parseNumber value = *"3,15,5169.4856"* var = *"num"*>

</fmt:parseNumber>

<h3>${num }</h3>

<fmt:parseNumber value = *"3.255E5"* var = *"num"*/>

<h3>${num }</h3>

</body>

</html>

日期时间格式化标签：（即将日期时间格式化为字符串形式，方便人们认读）

格式：<fmt:formatDate value = “要格式化的时间” type = “日期|时间|两者” dateStyle = “日期格式” timeStyle = “时间格式” scope = “属性范围” var = “存储结果的属性名称” timeZone = “时区”/>

范例：日期时间格式化

<head>

<%@ page language=*"java"* contentType=*"text/html; charset=GBK"*

pageEncoding=*"GBK"*%>

<%@ taglib uri = *"http://www.wenpan.cn/jstl/fmt"* prefix = *"fmt"* %>

<%@ page import = *"java.util.\*"* %>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=GBK"* />

<title>格式化日期</title>

</head>

<body>

<%

pageContext.setAttribute("date", **new** Date()) ;

%>

<fmt:formatDate value=*"*${date}*"* dateStyle = *"default"* timeStyle = *"default"* var = *"date"* type = *"both"* />

<h2>默认显示的日期时间：${date }</h2>

<%-- <fmt:formatDate value="${date}" dateStyle = "medium" timeStyle = "medium" var = "date" type = "both" />

<h2>默认显示的日期时间：${date }</h2>

<fmt:formatDate value="${date}" type = "both" dateStyle = "short" timeStyle = "short" var = "date" />

<h2>short显示的日期时间：${date}</h2>

<fmt:formatDate value="${date }" type = "both" dateStyle = "long" timeStyle = "long" var = "date" />

<h2>long显示的日期时间：${date }</h2> --%>

<%-- <fmt:formatDate value="${date }" dateStyle = "full" timeStyle = "full" type = "both" var = "date" />

<h2>full显示的日期时间：${date }</h2> --%>

<fmt:formatDate value=*"*${date }*"* pattern = *"yyyy年MM月dd日 HH时mm分ss秒SSS毫秒"* var = *"date"* type = *"both"*/>

<h2>pattern显示的日期：${date }</h2>

</body>

</html>

**日期时间反格式化：**

格式：<fmt:parseDate value = “date” pattern = “反格式化样式” dateStyle = “日期样式” timeStyle = “时间样式” type = “时间|日期|两者” scope = “属性范围” var = “存储结果的属性名称” timeZone = “时区”>

范例：时间日期的反格式化

<head>

<%@ page language=*"java"* contentType=*"text/html; charset=GBK"*

pageEncoding=*"GBK"*%>

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

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=GBK"* />

<title>反日期格式化</title>

</head>

<body>

<!-- 即将字符串日期转化为日期 -->

<fmt:parseDate value = *"2009 年 12 月 12 日 星期三 下午 03 时 15 分 05 秒 CST"* dateStyle = *"default"* timeStyle = *"default"*

var = *"date"*/>

<h2>字符串变为日期：${date }</h2>

<fmt:parseDate value = *"2009 年 12 月 12 日 星期三 下午 16 时 15 分 05 秒 129毫秒"* pattern = *"yyyy年 MM 月 dd 日 HH 时 mm 分 ss 秒 SSS 毫秒"*

var = *"date"*/>

<h2>字符串边为日期：${date }</h2>

</body>

</html>

设置时区：

格式：<fmt:setTimeZone value = “时区” >

标签体内容

</fmt:setTimeZone>

**SQL标签库：**

在jstl中也提供了与数据库操作相关的标签，可以利用这些标签方便的进行数据库的更新及查询操作。但是在web中开发一定是分层的，所以所有的数据库操作都要放在Javabean(DAO)中完成，而jstl是直接在jsp上操作数据库的，这种做法在开发中不是很推荐。

**SQL主要包含的标签：**

1. **数据源标签：<sql:setDataSource> 设置要使用的数据源名称**
2. **数据库操作标签： <sql:query>,<sql:update>**
3. **事务出路标签：<sql:transaction>**
4. <sql:setDateSource>标签

该标签用于设定数据源，语法如下：

<sql:setDataSource dataSource = “数据源名称” var = “保存的属性名称” scope= “属性范围”/>

范例：使用数据源

<%@ taglib uri = “http://wenpan.cn/jstl/sql” prefix = “sql”%>

<sql:setDataSource dataSource = “jdbc/mldn” var = “mldns”/>

在以上范例中直接使用了第十一章配置好的jdbc/mldn数据源进行连接，取得数据库连接的对象就保存在mldns属性名称中。

1. 数据库操作标签

1.数据库查询（无标签体）

<sql:query sql = “sql语句” var = “保存查询结果的属性名称” dataSource = “数据源名称” maxRows = “最多显示的记录条数” startRow = “记录的开始行数” scope = “属性范围” />

2.数据库查询（有标签体）

<sql:query var = “保存查询结果的属性名称” dataSource = “数据源名称” maxRows = “最多显示的记录条数” startRow = “记录的开始行数” scope = “属性范围” >

标签体

</sql:query>

1. 数据库更新：无标签体

<sql:update sql = “sql 语句” var = “保存更新的记录数” scope = “属性范围” dataSource = “数据源名称”/>

数据库更新：有标签体

<sql:update var = “保存更新的记录数” scope = “属性范围” dataSource = “数据源名称”>

sql语句

</sql:update>

在JDBC操作中，可以通过PreparedStatment完成预处理操作，所有要设置的内容都通过?进行占位，然后使用setXxx()方法具体设置每一个占位符的具体数据。在jstl中也可以这样操作。只需要在<sql:query>或<sql:update>标签中使用?，但是所有的内容都要通过<sql:param>和<sql:dateParam>标签进行设置，这两个标签的语法如下：

<sql:param value = “参数内容” />

<sql:dateParam value = “参数内容” type = “date的种类”/> 主要针对日期

事务处理：

格式：<sql:transaction dateSource = “数据源” >

<sql:query>或则<sql:update>

</sql:transaction>

即当事务中所有的内容执行完毕后再一起提交，只要有一个语句未正常执行，那么都不能提交。

**XML标签库:**

在开发中XML解析的操作是非常频繁的，幸运的是在jstl中专门提供了用于XML解析的操作，这样用户就可以不用费力的去研究SAX或DOM等操作的使用。且可以轻松的进行XML文件的解析处理。

XPath的主要功能是在XML文档中能够准确的找到某一个节点元素，可以通过XPath找到一个XML文档中定义的节点内容或属性等。

/：代表根元素。

../:父节点

//：任何路径下的节点

@：属性

\*：通配符

<x:parse>标签：此标签主要是进行XML的解析操作，语法如下：

<x:parse var = “保存名称” doc = “XML文件内容” scope = “属性范围” varDom = “dom解析后的XML对象” systemid = “XML文件的URL”/>

有标签体：

<x:parse var = “保存名称” scope = “属性范围” varDom = “dom解析后的XML对象” systemid = “XML文件的URL”>

要解析的XML文件

</x:parse>

<x:out>标签：进行XML文件内容的输出

<x:out select = “Xpath路径”/>

范例：输出name和email元素内容

1. 定义要解析的XML文件

<?xml version = *"1.0"* encoding = *"GBK"*?>

<addresslist>

<linkman>

<name id = *"wp"*>文攀</name>

<email>www.wenpan@163.com</email>

</linkman>

</addresslist>

1. 元素的输出

<head>

<%@ page language=*"java"* contentType=*"text/html; charset=GBK"*

pageEncoding=*"GBK"*%>

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

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

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=GBK"* />

<title></title>

</head>

<body>

<c:import url=*"address.xml"* var = *"add"* charEncoding = *"GBK"*/> <!-- 导入要引用的文件 -->

<x:parse var = *"addressXml"* doc = *"*${add}*"*/> <!-- doc表示xml的内容 x:parse标签是进行xml文件的解析-->

<h3>姓名：<x:out select=*"$addressXml/addresslist/linkman/name"* /> <!-- select = "Xpath路径": -->

(编号：<x:out select = *"$addressXml/addresslist/linkman/name/@id"*/>)

</h3>

<h3>邮箱：<x:out select = *"$addressXml/addresslist/linkman/email"*/></h3>

</body>

</html>

<x:set>标签：可以将从XML文件取出的内容保存在指定的属性范围中。语法如下：

<x:set select = “Xpath路径” var = “属性保存的名称” scope = “属性范围”/>

**范例：使用<x:set>标签保存解析的结果**

<head>

<%@ page language=*"java"* contentType=*"text/html; charset=GBK"*

pageEncoding=*"GBK"*%>

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

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

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=GBK"* />

<title>set标签保存解析结果</title>

</head>

<body>

<c:import url = *"address.xml"* var = *"add"* charEncoding = *"GBK"*/> <!-- 导入要操作的界面 -->

<x:parse var = *"addressXml"* doc = *"*${add}*"*/> <!-- 进行xml文件的解析 -->

<x:set var = *"nameXml"* scope = *"request"* select = *"$addressXml/addresslist/linkman"*/>

<h3>

姓名：<x:out select = *"$nameXml/name"*/> <!-- 取得保存的路径下的姓名信息 -->

</h3>

</body>

</html>

<x:if>标签:主要功能是判断XPath指定内容是否符合判断的条件。语法如下：

无标签体：<x:if select = “Xpath路径” var = “存放判断的结果” scope = “属性范围”/>

有标签体：<x:if select = “Xpath路径” var = “存放判断的结果” scope = “属性范围”>

满足条件时执行的语句

</x:if>

范例：使用<x:if>判断输出：

<head>

<%@ page language=*"java"* contentType=*"text/html; charset=GBK"*

pageEncoding=*"GBK"*%>

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

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

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=GBK"* />

<title>xml中的if标签</title>

</head>

<body>

<c:import url = *"address.xml"* charEncoding = *"GBK"* var = *"add\_if"*/> <!-- 导入要操作的文件 -->

<x:parse var = *"addressXml"* doc = *"*${add\_if }*"*/> <!-- 进行xml解析 -->

<x:if select = *"$addressXml//name/@id = 'wp'"*> <!-- 查找在任何节点下的name节点中的id属性如果存在则执行标签体 -->

<h2>在所有节点下存在编号是wp的属性!姓名 :<x:out select = *"$addressXml/addresslist/linkman/name"*/>

</h2>

</x:if>

</body>

</html>

<x:choose>,<x:when>,<x:otherwise>标签：

**范例：**

<head>

<%@ page language=*"java"* contentType=*"text/html; charset=GBK"*

pageEncoding=*"GBK"*%>

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

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

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=GBK"* />

<title>xml中的if标签</title>

</head>

<body>

<c:import url=*"address.xml"* charEncoding = *"GBK"* var = *"choose"*></c:import>

<x:parse var = *"addressXml"* doc = *"*${choose}*"*/>

<x:choose>

<x:when select = *"$addressXml/addresslist/linkman/name = '文攀'"*>

<h2>存在姓名为文攀的信息！姓名：<x:out select = *"$addressXml/addresslist/linkman/name"*></x:out></h2>

</x:when>

<x:otherwise>

不存在文攀的信息！

</x:otherwise>

</x:choose>

</body>

</html>