JSTL标签 参考手册

博客分类：

- Servlet/Jsp

前言

==========

JSTL标签库，是日常开发经常使用的，也是众多标签中性能最好的。把常用的内容，放在这里备份一份，随用随查。尽量做到不用查，就可以随手就可以写出来。这算是Java程序员的基本功吧，一定要扎实。

JSTL全名为JavaServer Pages Standard Tag Library，目前最新的版本为1.1版。JSTL是由JCP(Java Community Process)所制定的标准规范，它主要提供给Java Web开发人员一个标准通用的标签函数库。

Web程序员能够利用JSTL和EL来开发Web程序，取代传统直接在页面上嵌入Java程序(Scripting)的做法，以提高程序的阅读性、维护性和方便性。

JSTL 1.1必须在支持Servlet 2.4且JSP 2.0以上版本的Container才可使用

<%@ taglib %>引入标签库

==========

1、以classPath中，加入jar包： standard-1.1.2.jar ， jstl-1.1.2.jar

2、在相目\WEB-INF\tld\文件夹中放入常用的tld文件：c.tld，fmt.tld

3、在jsp文件的顶部加入以下内容：

Java代码

```
1. <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"
%>
```

```
2. <%@ taglib uri="http://java.sun.com/jsp/jstl/fmt"
prefix="fmt"%>
3. <%@ taglib uri="http://java.sun.com/jsp/jstl/functions"
prefix="fn" %>
```

<c:> 核心标签库

==========

JSTL 核心标签库(C标签)标签共有13个，功能上分为4类：

1.表达式控制标签：out、set、remove、catch

2.流程控制标签：if、choose、when、otherwise

3.循环标签：forEach、forTokens

4.URL操作标签：import、url、redirect

<c:forEach> 标签

为循环控制，它可以将集合(Collection)中的成员循序浏览一遍。

<c:forEach> 标签的语法 说明：

语法1：迭代一集合对象之所有成员

Html代码

```
1. <c:forEach [var="varName"] items="collection"
[varStatus="varStatusName"] [begin="begin"] [end="end"]
[step="step"]>
2. 本体内容
3. </c:forEach>
```

语法2：迭代指定的次数

Html代码

```
1. <c:forEach [var="varName"] [varStatus="varStatusName"]
begin="begin" end="end" [step="step"]>
2. 本体内容
3. </c:forEach>
```

<c:forEach> 标签的 属性说明：

名　称	说　明	EL	类型	必须	默认值
var	用来存放现在指到的成员	N	String	否	无
items	被迭代的集合对象	Y	Arrays Collection Iterator Enumeration Map String	否	无
varStatus	用来存放现在指到的相关成员信息	N	String	否	无
begin	开始的位置	Y	int	否	0
end	结束的位置	Y	int	否	最后一个成员
step	每次迭代的间隔数	Y	int	否	1

<c:forEach> 标签的 属性： varStatus属性： 它的提供另外四个属性:index,count,fist和last，它们个自的意义如下:

Java代码

```
1. 属性        类型        意义
2. index       number      现在指到成员的索引
3. count       number      总共指到成员的总和
4. first       boolean     现在指到成员是否为第一个
5. last        boolean     现在指到成员是否为最后一个
```

<c:forEach> 遍历 List列表：

对于一个基本类型的数组，当前元素将作为相应包装类（Integer、Float等等）的一个实例提供。

Html代码

```
1. <c:forEach items="${domainList }" var="item">
```

```
2. <tr>
3.   <td align="center"
valign="middle">${item["domain"]==null?" ":item["domain"]}
</td>
4.   <td align="center" valign="middle"><fmt:formatDate
value="${item['bind_date']}" pattern="yyyy-MM-dd HH:mm:ss"/></td>
5.   <td align="center" valign="middle">
6.     <c:if test="${item['domain']!=null}">
7.     <a href="javascript:;" id="${item['domain']}"
class="del"> </a>
8.     </c:if>
9.     </td>
10. </tr>
11. </c:forEach>
```

<c:forEach> 遍历Map:

对于一个java.util.Map，当前元素则作为一个java.util.Map.Entry提供。

Html代码

```
1. <c:if test="${!empty permissionMap}">
2. <c:forEach items="${permissionMap}" var="item">
3. <tr>
4.     <td>${item.value.id}</td>
5.     <td>${item.value.urlOnClass}</td>
6.     <td>${item.value.urlOnMethod}</td>
7. </tr>
8. </c:forEach>
9. </c:if>
```

<c:forTokens> 标签

用来浏览一字符串中所有的成员，其成员是由定义符号(delimiters)所分隔的。

<c:forTokens> 标签的语法 说明：

Html代码

```
1. <c:forTokens items="stringOfTokens" delims="delimiters"
```

```
[var="varName"]
2. [varStatus="varStatusName"] [begin="begin"] [end="end"]
[step="step"]>
3. 本体内容
4. </c:forTokens>
```

<c:forTokens> 标签的 属性说明：

名　称	说　明	EL	类　型	必 须	默认值
var	用来存放现在指到的成员	N	String	否	无
items	被迭代的字符串	Y	String	是	无
delims	定义用来分割字符串的字符	N	String	是	无
varStatus	用来存放现在指到的相关成员信息	N	String	否	无
begin	开始的位置	Y	int	否	0
end	结束的位置	Y	int	否	最后一个成员
step	每次迭代的间隔数	Y	int	否	1

<c:out> 标签

主要用来显示数据的内容

<c:out> 标签的语法 说明：

语法1：没有本体(body)内容

Html代码

```
1. <c:out value="value" [escapeXml="{true|false}"]
[default="defaultValue"] />
```

语法2：有本体内容

Html代码

```
1. <c:out value="value" [escapeXml="{true|false}"]>
2. default value
3. </c:out>
```

<c:forEach> 标签的 属性说明： 略

一般来说，<c:out>默认会将<、>、’、” 和 & 转换为 <、>、'、" 和&。假若不想转换时，只需要设定<c:out>的escapeXml属性为fasle就可以了。

<c:set> 标签

主要用来将变量储存至JSP范围中或是JavaBean的属性中。

<c:set> 标签的语法 说明：

语法1：将value的值储存至范围为scope的 varName 变量之中

Html代码

```
1. <c:set value="value" var="varName" [scope="{
page|request|session|application }"]/>
```

语法2：将本体内容的数据储存至范围为scope的 varName 变量之中

Html代码

```
1. <c:set var="varName" [scope="{
page|request|session|application }"]>
2. … 本体内容
3. </c:set>
```

语法3：将 value的值储存至 target 对象的属性中

Html代码

```
1. <c:set value="value" target="target" property="propertyName"
/>
```

语法4：将本体内容的数据储存至target 对象的属性中

Html代码

```
1. <c:set target="target" property="propertyName">
2. … 本体内容
3. </c:set>
```

<c:set> 标签的 属性说明：

名　称	说　明	EL	类型	必须	默认值
value	要被储存的值	Y	Object	否	无
var	欲存入的变量名称	N	String	否	无
scope	var 变量的 JSP 范围	N	String	否	page
target	为一 JavaBean 或 java.util.Map 对象	Y	Object	否	无
property	指定 target 对象的属性	Y	String	否	无

<c:remove> 标签

主要用来移除变量。

<c:remove> 标签的语法 说明：

Html代码

```
1. <c:remove var="varName" [scope="{
age|request|session|application }"] />
```

<c:catch> 标签

主要用来处理产生错误的异常状况，并且将错误信息储存起来。

<c:catch> 标签的语法 说明：

Html代码

```
1. <c:catch [var="varName"] >
2. … 欲抓取错误的部分
3. </c:catch>
```

<c:if> 标签

的用途就和我们一般在程序中用的if一样。

<c:if> 标签的语法 说明：

语法1：没有本体内容(body)

Html代码

```
1. <c:if test="testCondition" var="varName" [scope="
{page|request|session|application}"]/>
```

语法2：有本体内容

Html代码

```
1. <c:if test="testCondition" [var="varName"] [scope="
{page|request|session|application}"]>
2. 本体内容
3. </c:if>
```

示例：

Html代码

```
1. <c:if test="${not empty item.publish_time}">
2. 内容
3. </c:if>
4.
5. <c:if test="${item['domain']!=null}">
6. 内容
7. </c:if>
8.
9. <c:if test="${!empty permissionMap}">
10. 内容
11. </c:if>
```

c:choose> <c:when> <c:otherwise> 标签

<c:choose when otherwise> 标签的语法 说明：

Html代码

```
1. <c:set var="score">85</c:set>
2. <c:choose>
3. <c:when test="${score>=90}">
4. 你的成绩为优秀!
5. </c:when>
6. <c:when test="${score>=70&&score<90}">
7. 您的成绩为良好!
8. </c:when>
9. <c:when test="${score>60&&score<70}">
10. 您的成绩为及格
11. </c:when>
12. <c:otherwise>
13. 对不起，您没有通过考试!
14. </c:otherwise>
15. </c:choose>
```

<fmt:> 格式 化标签库

==========

一：JSTL格式化标签又称为I18N标签库，主要用来编写国际化的WEB应用，使用此功能可以对一个特定的语言请求做出合适的处理。

例如：中国内地用户将显示简体中文，台湾地区则显示繁体中文，使用I18N格式化标签库还可以格式化数字和日期，例如同一数字或日趋，在不同国家可能有不同的格式，使用I18N格式标签库可以将数字和日期格式为当地的格式。

在JSP页面中要使用到格式化标签，需要引入下面的语句：

<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt"% >

二：概览

格式化标签

<fmt:fromatNumber>

<fmt:formatDate>

<fmt:parseDate>

<fmt:parseNumber>
<fmt:setTimeZone>
<fmt:timeZone>

国际化标签
<fmt:setLocale>
<fmt:requestEncoding>
<fmt:bundle>
<fmt:message>
<fmt:param>
<fmt:setBundle>

三：<fmt:formatNumber>

此标签会根据区域定制的方式将数字格式化成数字，货币，百分比。
此标签的属性：
value:要格式化的数字
type：按照什么类型格式化
pattern：自定义格式化样式
currencyCode:ISO-4721货币代码，只适用于按照货币格式化的数字
currencySymbol： 货币符号,如￥,只适用于按照货币格式化的数字
groupingUsed： 是否包含分隔符
maxIntegerDigits： 整数部分最多显示多少位
mixIntegerDigits： 整数部分最少显示多少位
maxFractionDigits： 小数部分最多显示多位位
minFractionDigits： 小数部分最少显示多位位
var:存储格式化后的结果
scope: 存储的范围

示例1：

Java代码

```
<%@ page language="java" pageEncoding="utf-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
```

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <title>chapter4.jsp</title>
  </head>
  <body>
    <div>
      <div>
        <fmt:setLocale value="fr_fr"/>
        <fmt:formatNumber value="123456789.012"/>
        <br/>
        <fmt:setLocale value="zh_cn"/>
        <fmt:formatNumber value="123456789.012"/>
        <br />
        <fmt:setLocale value="de_de"/>
        <fmt:formatNumber value="123456789.012"/>
        <br />
      </div>
    </div>
  </body>
</html>

<%@ page language="java" pageEncoding="utf-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <title>chapter4.jsp</title>
  </head>
  <body>
    <div>
      <div>
```

```
            <fmt:setLocale value="fr_fr"/>
            <fmt:formatNumber value="123456789.012"/>
            <br/>
            <fmt:setLocale value="zh_cn"/>
            <fmt:formatNumber value="123456789.012"/>
            <br />
            <fmt:setLocale value="de_de"/>
            <fmt:formatNumber value="123456789.012"/>
            <br />
        </div>
    </div>
</body>
</html>
```
注意:如果要实现国际化,那么编码格式要设置为utf-8.

从程序运行效果可以看出,设定的区域不同,格式化数字的显示也会不同.

四：type属性：可以是数字(number),货币(currency),百分比(percent)

示例2：

Java代码

```
<%@ page language="java" pageEncoding="utf-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <title>chapter4.jsp</title>
  </head>
  <body>
    <div>
      <div>
        <fmt:setLocale value="zh_cn"/>
```

```
            <fmt:formatNumber value="0.3" type="number"/><br />
            <fmt:formatNumber value="0.3" type="currency"/><br />
            <fmt:formatNumber value="0.3" type="percent"/><br />
        </div>
    </div>
  </body>
</html>

<%@ page language="java" pageEncoding="utf-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <title>chapter4.jsp</title>
  </head>
  <body>
    <div>
      <div>
        <fmt:setLocale value="zh_cn"/>
        <fmt:formatNumber value="0.3" type="number"/><br />
        <fmt:formatNumber value="0.3" type="currency"/><br />
        <fmt:formatNumber value="0.3" type="percent"/><br />
      </div>
    </div>
  </body>
</html>
```

currencyCode为货币代码,例如美元为USD,人民币为CNY等

currencySymbol为货币符号例如,人民币为￥,美元为$。

如果不指定区域,则会根据语言区域自动选择currencySymbol

示例3:

Java代码

```
<%@ page language="java" pageEncoding="utf-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <title>chapter4.jsp</title>
  </head>
  <body>
    <div>
      <div>
        <fmt:setLocale value="zh_cn"/>
        <fmt:formatNumber value="0.3" type="currency"/><br />
        <fmt:setLocale value="en_Us"/>
        <fmt:formatNumber value="0.3" type="currency"/><br />
      </div>
    </div>
  </body>
</html>

<%@ page language="java" pageEncoding="utf-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <title>chapter4.jsp</title>
  </head>
  <body>
    <div>
      <div>
        <fmt:setLocale value="zh_cn"/>
        <fmt:formatNumber value="0.3" type="currency"/><br />
        <fmt:setLocale value="en_Us"/>
```

```
            <fmt:formatNumber value="0.3" type="currency"/><br />
          </div>
        </div>
    </body>
</html>
```

currencySymbol属性还可以自定义要显示的头标识,但是一定得type="currency"才会生效,例如:

Java代码

```
<%@ page language="java" pageEncoding="utf-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
    <head>
        <title>chapter4.jsp</title>
    </head>
    <body>
        <div>
            <div>
                <fmt:setLocale value="zh_cn"/>
                <fmt:formatNumber value="0.3" type="currency"
currencySymbol="#"/><br />
                <fmt:setLocale value="en_Us"/>
                <fmt:formatNumber value="0.3" type="currency"
currencySymbol="#"/><br />
            </div>
        </div>
    </body>
</html>

<%@ page language="java" pageEncoding="utf-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
```

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <title>chapter4.jsp</title>
  </head>
  <body>
    <div>
      <div>
        <fmt:setLocale value="zh_cn"/>
        <fmt:formatNumber value="0.3" type="currency"
currencySymbol="#"/><br />
        <fmt:setLocale value="en_Us"/>
        <fmt:formatNumber value="0.3" type="currency"
currencySymbol="#"/><br />
      </div>
    </div>
  </body>
</html>
```

自定义数字样式

```
<fmt:formatNumber value="12.31" pattern=".0000"/><br/>
<fmt:formatNumber value="1234" pattern="###.##E0"/>
```

会显示:

12.3100

1.234E3

会四舍五入

var:定义一个变量,存储格式化后的结果,scope指定变量存储的范围.用法和前面讲的标签一致.

五：<fmt:parseNumber>

此标签用来将字符串类型的数字,货币或百分比转换成数字类型,和<fmt:formatNumber>标签的作用正好相反.

value: 要转换的字符串
type: 指定要转换的字符串为什么类型,可取值:number,percent,currency
pattern: 自定义格式化样式
parseLocale: 指定区域来转换字符串
IntegerOnly: 转换后的数字是否只显示整数部分
var: 存储转换后的结果
scope: 存储的范围

示例1:
<fmt:parseNumber value="500,800,200"/>
显示: 500800200

示例2:
<fmt:parseNumber value="52%" type="percent" />
显示: 0.52 (52%在这里是一个字符串, type指定这个字符串是什么类型的值)

示例3:
<fmt:parseNumber value="￥123" type="currency" />
显示123, ￥123在这里是一个字符串, type指定这个字符串是什么类型的值

示例4:
<fmt:parseNumber value="123.333" type="number" />
<fmt:parseNumber value="123.333" type="number" integerOnly="true"/>
显示:
123.333
123
integerOnly确定是否只显示整数部分.

示例5:

<fmt:parseNumber value="￥123.333" type="currency" parseLocale="zh_CN"/>
<fmt:parseNumber value="$123.333" type="currency" parseLocale="en_US"/>
parseLocale="en_US"主要是配合当type="currency"时用的,
如果要转换货币的字符串类型为value="￥123.333",不设置语言环境的话,会取当前浏览器的默认设置,否则就要加上parseLocale="zh_CN",指定环境为中文环境
如果要转换货币的字符串类型为value="$123.333",不设置语言环境的话,会取当前浏览器的默认设置,如果默认为zh_cn的话,程序会报错的,否则就要加上parseLocale="en_US",指定环境为英文美国环境

六：<fmt:formatDate />

此标签可以将日期格式化.
属性介绍:
value 用来格式化的时间或日期
type 指定格式化的是日期还是时间,或者两者都是取值范围:date,time,both
pattern 自定义格式化样式
dateStyle 日期的格式化样式
timeStyle 时间的格式化样式
timeZone 指定使用的时区
var 存储格式化后的结果
scope 指定存储的范围

自定义格式:

Java代码

```
<fmt:setLocale value="zh_cn" />
<fmt:formatDate value="<%=new Date()%>" type="both" pattern="yyyy/MM/dd hh:mm:ss" />
<br />
<fmt:formatDate value="<%=new Date()%>" type="both" pattern="yyyy-MM-dd HH:mm:ss" />
<br />
```

```
<fmt:formatDate value="<%=new Date()%>" type="both" pattern="yyyy年MM月dd日 hh小时mm分钟ss秒" />
<br />
<fmt:formatDate value="<%=new Date()%>" type="both" pattern="yy/MM/dd hh:mm:ss" />
<br />

<fmt:setLocale value="zh_cn" />
<fmt:formatDate value="<%=new Date()%>" type="both" pattern="yyyy/MM/dd hh:mm:ss" />
<br />
<fmt:formatDate value="<%=new Date()%>" type="both" pattern="yyyy-MM-dd HH:mm:ss" />
<br />
<fmt:formatDate value="<%=new Date()%>" type="both" pattern="yyyy年MM月dd日 hh小时mm分钟ss秒" />
<br />
<fmt:formatDate value="<%=new Date()%>" type="both" pattern="yy/MM/dd hh:mm:ss" />
<br />     注意这里小时 hh表示12小时制, HH代表24小时制
```

示例1：

Java代码

```
<fmt:setLocale value="zh_cn" />
<fmt:formatDate value="<%=new Date()%>" />
<br />
<fmt:setLocale value="zh_tw" />
<fmt:formatDate value="<%=new Date()%>" />

<fmt:setLocale value="zh_cn" />
<fmt:formatDate value="<%=new Date()%>" />
<br />
<fmt:setLocale value="zh_tw" />
```

<fmt:formatDate value="<%=new Date()%>" />
大家可以看到大陆和台湾显示日期的格式是有区别的.

显示结果:
2009-12-7
2009/12/7

示例2:

Java代码
<fmt:setLocale value="zh_cn" />
<fmt:formatDate value="<%=new Date()%>" type="time"/>

<fmt:setLocale value="zh_tw" />
<fmt:formatDate value="<%=new Date()%>" type="time"/>

<fmt:setLocale value="zh_cn" />
<fmt:formatDate value="<%=new Date()%>" type="time"/>

<fmt:setLocale value="zh_tw" />
<fmt:formatDate value="<%=new Date()%>" type="time"/>
显示结果:
14:59:28
下午 02:59:28

type可取值及意义:
date 格式化日期
time格式化时间
both格式化日期时间

示例3:

Java代码
<fmt:setLocale value="zh_cn" />

```
<fmt:formatDate value="<%=new Date()%>" type="both" />
<br />
<fmt:setLocale value="zh_tw" />
<fmt:formatDate value="<%=new Date()%>" type="both" />

<fmt:setLocale value="zh_cn" />
<fmt:formatDate value="<%=new Date()%>" type="both" />
<br />
<fmt:setLocale value="zh_tw" />
<fmt:formatDate value="<%=new Date()%>" type="both" />
```
输出结果:

2009-12-7 21:24:26

2009/12/7 下午 09:24:26

dateStyle用来设定日期显示的样式,其值可以是default, short, medium, long, full,请看示例:

Java代码

```
<fmt:setLocale value="zh_cn" />
<fmt:formatDate value="<%=new Date()%>" type="both" dateStyle="default"
/>
<br />
<fmt:formatDate value="<%=new Date()%>" type="both" dateStyle="short"
/>
<br />
<fmt:formatDate value="<%=new Date()%>" type="both"
dateStyle="medium" />
<br />
<fmt:formatDate value="<%=new Date()%>" type="both" dateStyle="long" />
<br />
<fmt:formatDate value="<%=new Date()%>" type="both" dateStyle="full" />
<br />

<fmt:setLocale value="zh_cn" />
```

```
<fmt:formatDate value="<%=new Date()%>" type="both" dateStyle="default"
/>
<br />
<fmt:formatDate value="<%=new Date()%>" type="both" dateStyle="short"
/>
<br />
<fmt:formatDate value="<%=new Date()%>" type="both"
dateStyle="medium" />
<br />
<fmt:formatDate value="<%=new Date()%>" type="both" dateStyle="long" />
<br />
<fmt:formatDate value="<%=new Date()%>" type="both" dateStyle="full" />
<br />
```
显示结果如下:

2009-12-7 21:30:49

09-12-7 21:30:49

2009-12-7 21:30:49

2009年12月7日 21:30:49

2009年12月7日 星期一 21:30:49

可以看到dateStyle属性只对日期部分起作用,时间部分没有作用.

timeStyle用来显示时间部分的样式,取值范围同上

Java代码

```
<fmt:setLocale value="zh_cn" />
<fmt:formatDate value="<%=new Date()%>" type="both"
timeStyle="default" />
<br />
<fmt:formatDate value="<%=new Date()%>" type="both" timeStyle="short"
/>
<br />
<fmt:formatDate value="<%=new Date()%>" type="both"
timeStyle="medium" />
<br />
```

```
<fmt:formatDate value="<%=new Date()%>" type="both" timeStyle="long" />
<br />
<fmt:formatDate value="<%=new Date()%>" type="both" timeStyle="full" />
<br />

<fmt:setLocale value="zh_cn" />
<fmt:formatDate value="<%=new Date()%>" type="both"
timeStyle="default" />
<br />
<fmt:formatDate value="<%=new Date()%>" type="both" timeStyle="short"
/>
<br />
<fmt:formatDate value="<%=new Date()%>" type="both"
timeStyle="medium" />
<br />
<fmt:formatDate value="<%=new Date()%>" type="both" timeStyle="long" />
<br />
<fmt:formatDate value="<%=new Date()%>" type="both" timeStyle="full" />
<br />
```
输出:

2009-12-7 21:35:52

2009-12-7 下午9:35

2009-12-7 21:35:52

2009-12-7 下午09时35分52秒

2009-12-7 下午09时35分52秒 CST

timeZone用来设定时区,时区的意思类似于酒店里大堂放的几个时钟,比如现在时间会有北京时间,东京时间,纽约时间,伦墩时间,

取值范围为:EST, CST, MST, PST

Java代码

```
<fmt:setLocale value="zh_cn" />
<fmt:formatDate value="<%=new Date()%>" type="time" timeStyle="full" />
<br />
```

```
<fmt:formatDate value="<%=new Date()%>" type="time" timeStyle="full"
timeZone="EST" />
<br />
<fmt:formatDate value="<%=new Date()%>" type="time" timeStyle="full"
timeZone="CST" />
<br />
<fmt:formatDate value="<%=new Date()%>" type="time" timeStyle="full"
timeZone="MST" />
<br />
<fmt:formatDate value="<%=new Date()%>" type="time" timeStyle="full"
timeZone="PST" />
<br />

<fmt:setLocale value="zh_cn" />
<fmt:formatDate value="<%=new Date()%>" type="time" timeStyle="full" />
<br />
<fmt:formatDate value="<%=new Date()%>" type="time" timeStyle="full"
timeZone="EST" />
<br />
<fmt:formatDate value="<%=new Date()%>" type="time" timeStyle="full"
timeZone="CST" />
<br />
<fmt:formatDate value="<%=new Date()%>" type="time" timeStyle="full"
timeZone="MST" />
<br />
<fmt:formatDate value="<%=new Date()%>" type="time" timeStyle="full"
timeZone="PST" />
<br />     输出结果:
    下午09时41分37秒 CST
    上午08时41分37秒 EST
    上午07时41分37秒 CST
    上午06时41分37秒 MST
    上午05时41分37秒 PST
```

七：<fmt:parseDate>

将字符串类型的时间转换为日期类型.

value 用来格式化的时间或日期的字符串
type 指定格式化的是日期还是时间,或者两者都是取值范围:date,time,both
pattern 自定义格式化样式
dateStyle 日期的格式化样式
timeStyle 时间的格式化样式
timeZone 指定使用的时区
var 存储格式化后的结果
scope 指定存储的范围

示例:

```
<fmt:setLocale value="zh_cn" />
<fmt:parseDate type="date" value="2008-4-5"/>
```

输出: Sat Apr 05 00:00:00 CST 2008,

这里已经将字符串” 2008-4-5”转换为了日期对象了.转换一定得注意,类似于2008-4-5这样的字符串,type必须为date,类似于12:34:56的字符串,type必须为time类似于2008-4-5 12:34:56这样的字符串,type必须为both还要注意浏览器的语言环境的设置,如果为zh_tw,那么字符串就必须得符合当地的标准,如为2009/12/7 下午09:24:26就正确转换为日期对象,否则就会报错.

八：<fmt:setTimeZone>

---

value 设定时区
var 存储设定的时区
scope 存储的范围

value用来设定时区,可以是EST,CST,MST,PST等,如果有var属性,则将结果存储在所设定的范围之内.在属性范围内的页面都会使用该时区为默认时区.

Java代码

```
<fmt:setLocale value="zh_cn" />
```

```
<fmt:setTimeZone value="EST" />
<fmt:formatDate value="<%=new Date()%>" type="time" timeStyle="full" />
<br />
<fmt:formatDate value="<%=new Date()%>" type="time" timeStyle="full" />
<br />
<fmt:formatDate value="<%=new Date()%>" type="time" timeStyle="full" />
<br />

<fmt:setLocale value="zh_cn" />
<fmt:setTimeZone value="EST" />
<fmt:formatDate value="<%=new Date()%>" type="time" timeStyle="full" />
<br />
<fmt:formatDate value="<%=new Date()%>" type="time" timeStyle="full" />
<br />
<fmt:formatDate value="<%=new Date()%>" type="time" timeStyle="full" />
<br />
```
输出:

上午09时25分12秒 EST
上午09时25分12秒 EST
上午09时25分12秒 EST
此时区在该页面内都有效

九：<fmt:timeZone>

用来暂时设置时区.

Java代码

```
<fmt:setLocale value="zh_cn" />
<fmt:timeZone value="EST">
<fmt:formatDate value="<%=new Date()%>" type="time" timeStyle="full" />
<br />
<fmt:formatDate value="<%=new Date()%>" type="time" timeStyle="full" />
<br />
</fmt:timeZone>
```

```
<fmt:formatDate value="<%=new Date()%>" type="time" timeStyle="full" />

<fmt:setLocale value="zh_cn" />
<fmt:timeZone value="EST">
<fmt:formatDate value="<%=new Date()%>" type="time" timeStyle="full" />
<br />
<fmt:formatDate value="<%=new Date()%>" type="time" timeStyle="full" />
<br />
</fmt:timeZone>
<fmt:formatDate value="<%=new Date()%>" type="time" timeStyle="full" />
```
此标签的时区只是部分,在标签开始至标签结束内有效,其它地方无效,其它地方还是会使用默认时区

<fn:> Function标签 库

==========

JSTL Functions 标签库中提供了一组常用的 EL 函数，主要用于处理字符串，在 JSP 中可以直接使用这些函数。
在 JSP 文件中使用 Functions 标签库，要先通过 taglib 指令引入该标签库：
<%@taglib uri=”http://java.sun.com/jsp/jstl/functions” prefix=”fn” %.

18.1fn:contains 函数

fn:contains 函数用于判断在源字符串中是否包含目标字符串，其语法为：

fn:contains(String source,String target) -------boolean;

以上 source 参数指定源字符串， target 参数指定目标字符串，返回类型为 boolean 。

例如对于以下 EL 表达式：

${fn:contains(“Tomcat”,”cat”)}

${fn:contains(“Tomcat”,”CAT”)}

第一个 EL 表达式的值为 true ， 第二个 EL 表达式的值为 false 。

18.2fn:containsIgnoreCase 函数

fn:containsIgnoreCase 函数用于判断在源字符串中是否包含目标字符串，并且在判断时忽略大小写，其语法为：

fn: containsIgnoreCase (String source,String target) -------boolean;

以上 source 参数指定源字符串， target 参数指定目标字符串，返回类型为 boolean 。

例如对于以下 EL 表达式：

${fn: containsIgnoreCase (“Tomcat”,”CAT”)}

${fn: containsIgnoreCase (“Tomcat”,”Mike”)}

第一个 EL 表达式的值为 true ， 第二个 EL 表达式的值为 false 。

18.3 fn:startsWith 函数

fn:startsWith 函数用于判断源字符串是否以指定的目标字符串开头，其语法为：

fn:startsWith(String source,String target) ----boolean

以上 source 参数指定源字符串， target 参数指定目标字符串，返回类型为 boolean 。

例如对于以下 EL 表达式：

${fn: startsWith (“Tomcat”,”Tom”)}

${fn: startsWith (“Tomcat”,”cat”)}

第一个 EL 表达式的值为 true ， 第二个 EL 表达式的值为 false 。

18.4 fn:endsWith 函数

fn: endsWith 函数用于判断源字符串是否以指定的目标字符串结尾，其语法为：

fn: endsWith (String source,String target) ----boolean

以上 source 参数指定源字符串， target 参数指定目标字符串，返回类型为 boolean 。

例如对于以下 EL 表达式：

${fn: endsWith (“Tomcat”,”cat”)}

${fn: endsWith (“Tomcat”,”Tom”)}

第一个 EL 表达式的值为 true ， 第二个 EL 表达式的值为 false 。

18.5 fn:indexOf 函数

fn:indexOf 函数用于在源字符串中查找目标字符串，并返回源字符串中最先与目标字符串匹配的第一个字符的索引，如果在源字符串中不包含目标字符串，就返回 -1 ，源字符串中的第一个字符的索引为 0 。 fn:indexOf 函数的语法为：

fn: indexOf (String source,String target) ----int

以上 source 参数指定源字符串， target 参数指定目标字符串，返回类型为 int 。

例如对于以下 EL 表达式：

```
1  ${fn: indexOf (“Tomcat”,”cat”)}<br/>
2  ${fn: indexOf (“2211221”,”21”)} <br/>
3  ${fn: indexOf (“Tomcat”,”Mike”)} <br/>
```

其输出结果为：

```
1  3
2  1
3  -1
```

18.6 fn:replace 函数

fn:replace 函数用于把源字符串中的一部分替换为另外的字符串，并返回替换后的字符串。 fn:replace 函数的语法为：

fn: replace (String source,String before,String after) ----String

以上 source 参数指定源字符串， before 参数指定源字符串中被替换的子字符串， after 参数指定用于替换的子字符串，返回类型为 String 。

例如对于以下 EL 表达式：

```
1  ${ fn: replace(“TomcAt”,”cAt”,”cat”)}<br/>
2  ${ fn: replace(“2008/1/9”,”/”,”-”)}<br/>
```

其输出结果为：

```
1  Tomcat
2  2008-1-9
```

18.7 fn:substring 函数

fn:substring 函数用于获取源字符串中的特定子字符串，它的语法为：

fn:substring(String source,int beginIndex,int endIndex) ------String

以上 source 参数指定源字符串， beginIndex 参数表示子字符串中的第一个

字符在源字符串中的索引，endIndex 参数表示子字符串的最后一个字符在源字符串中的索引加 1 ，返回类型为 String ，源字符串中的第一个字符的索引为 0 。

例如对于以下 EL 表达式：

```
1  ${ fn: substring (“Tomcat”,0,3)}<br/>
2  ${ fn: substring (“Tomcat”,3,”6”)}<br/>
```

其输出结果为：

```
1  Tom
2  cat
```

18.8 fn:substringBefore 函数

fn:substringBefore 函数用于获取源字符串中指定子字符串之前的子字符串，其语法为：

```
fn:substringBefore(String source,String target) ----String
```

以上 source 参数指定源字符串， target 参数指定子字符串，返回类型为 String 。如果在源字符串中不包含特定子字符串，就返回空字符串。

例如对于以下 EL 表达式：

```
1  ${ fn: substringBefore (“Tomcat”,”cat”)}<br/>
2  ${ fn: substringBefore (“mydata.txt”,”.txt”)}<br/>
```

其输出结果为：

```
1  Tom
2  mydata
```

18.9 fn:substringAfter 函数

fn: substringAfter 函数用于获取源字符串中指定子字符串之后的子字符串，其语法为：

```
fn: substringAfter (String source,String target) ----String
```

以上 source 参数指定源字符串， target 参数指定子字符串，返回类型为 String 。如果在源字符串中不包含特定子字符串，就返回空字符串。

例如对于以下 EL 表达式：

```
1  ${ fn: substringAfter (“Tomcat”,”Tom”)}<br/>
2  ${ fn: substringAfter (“mydata.txt”,” mydata.”)}<br/>
```

其输出结果为：

```
1  cat
2  txt
```

18.10 fn:split 函数

fn:split 函数用于将源字符串拆分为一个字符串数组，其语法为：

fn: split (String source,String delimiter) ----String[]

以上 source 参数指定源字符串， delimiter 参数指定用于拆分源字符串的分隔符，返回类型为 String[] 。如果在源字符串中不包含 delimiter 参数指定的分隔符，或者 delimiter 参数为 null ，那么在返回的字符串数组中只有一个元素，为源字符串。

例如对于以下 EL 表达式：

```
<c:set value='${ fn: split ("www.mywebsite.org",".")}' var="strs"/>
<c:forEach var="token" item="${strs}">
    ${token}<br/>
</c:forEach>
```

其输出结果为：

www
mywebsite
org

再例如对于以下代码：

```
<c:set value='${ fn: split ("www.mywebsite.org","-")}' var="strs"/>
${strs[0]}
```

其输出结果为：

www.mywebsite.org

18.11 fn:join 函数

fn:join 函数用于将源字符串数组中的所有字符串连接为一个字符串，其语法为：

fn:join(String source[],String separator) ----String

以上 source 参数指定源字符串数组， separator 参数指定用于连接源字符串数组中的各个字符串的分隔符，返回类型为 String 。

例如对于以下代码：

```
<%
String strs[] = {"www","mywebsite","org"};
%>
<c:set value="<%=strs%>" var="strs"/>
```

${fn:join(strs,".")}

其输出结果为：

www. mywebsite. org

18.12 fn:toLowerCase 函数

fn:toLowerCase 函数用于将源字符串中的所有字符改为小写，其语法为：

fn:toLowerCase(String source) -----String

以上 source 参数指定源字符串，返回类型为 String 。

例如对于以下 EL 表达式：

fn:toLowerCase("TomCat")

其输出结果为：

tomcat

18.13 fn:toUpperCase 函数

fn: toUpperCase 函数用于将源字符串中的所有字符改为大写，其语法为：

fn: toUpperCase (String source) -----String

以上 source 参数指定源字符串，返回类型为 String 。

例如对于以下 EL 表达式：

fn: toUpperCase ("TomCat")

其输出结果为：

TOMCAT

18.14 fn:trim 函数

fn:trim 函数用于将源字符串中的开头和末尾的空格删除，其语法为：

fn:trim(String source) ----String

以上 source 参数指定源字符串，返回类型为 String 。

例如对于以下 EL 表达式：

fn:trim(" Tomcat ")

以上 EL 表达式的值为" Tomcat "。

18.15 fn:escapeXml 函数

fn:escapeXml 函数用于将源字符串中的字符" < "、" > "、" ' "和" & "等转换为转义字符，本书第 1 章的 1.2 节（ HTML 简介）介绍了转义字符的概念。fn:escapeXml 函数的行为与 <c:out> 标签的 escapeXml 属性为 true 时的转换行为相同， fn:escapeXml 函数的语法为：

fn:escapeXml(String source) ----String

以上 source 参数指定源字符串，返回类型为 String 。

例程 18-1 的 out.jsp 演示了 fn:escapeXml 函数的用法。

```
<%@ page language="java" contentType="text/html;
charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/functions"
prefix="fn"%>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=UTF-8" />
<title>out</title>
</head>
<body>
1.${fn:escapeXml("<b> 表示粗体字 </b>") }<br/>
2.<c:out value="<b> 表示粗体字 </b>" escapeXml="true">
</c:out><br/>
3.${"<b> 表示粗体字 </b>"}<br/>
</body>
</html>
```

对于 out.jsp 中的以下代码：

```
1.${fn:escapeXml("<b> 表示粗体字 </b>") }<br/>
2.<c:out value="<b> 表示粗体字 </b>" escapeXml="true"></c:out><br/>
3.${"<b> 表示粗体字 </b>"}<br/>
```

其输出结果为：

```
1.&lt;b&gt; 表示粗体字 &lt;/b&gt;<br/>
2.&lt;b&gt; 表示粗体字 &lt;/b&gt;<br/>
3.<b> 表示粗体字 </b><br/>
```

18.16 fn:length 函数

fn:length 函数用于返回字符串中的字符的个数，或者集合和数组的元素的个数，其语法为：

```
fn:length(source) ---- int
```

以上 source 参数可以为字符串、集合或者数组，返回类型为 int 。

```
<%@ page language="java" contentType="text/html;
charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/functions"
prefix="fn"%>
<%@page import="java.util.ArrayList"%>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=UTF-8" />
<title>length</title>
</head>
<body>
<%
int[] array = {1,2,3,4};
ArrayList list = new ArrayList();
list.add("one");
list.add("two");
list.add("three");
%>
<c:set value="<%=array%>" var="array"></c:set>
<c:set value="<%=list%>" var="list"></c:set>
数组长度： ${fn:length(array)}<br/>
集合长度： ${fn:length(list)}<br/>
字符串长度： ${fn:length("Tomcat")}<br/>
</body>
</html>
```

Functions 标签库概览

l fn:contains 函数： 用于判断在源字符串中是否包含目标字符串。

l fn:containsIgnoreCase 函数： 用于判断在源字符串中是否包含目标字符串，并且在判断时忽略大小写。

l fn:startsWith 函数： 用于判断源字符串是否以指定的目标字符串开头。

l fn: endsWith 函数：用于判断源字符串是否以指定的目标字符串结尾。

l fn:indexOf 函数：用于在源字符串中查找目标字符串，并返回源字符串中最先与目标字符串匹配的第一个字符的索引。

l fn:replace 函数：用于把源字符串中的一部分替换为另外的字符串，并返回替换后的字符串。

l fn:substring 函数：用于获取源字符串中的特定子字符串。
l fn:substringBefore 函数：用于获取源字符串中指定子字符串之前的子字符串。
l fn: substringAfter 函数：用于获取源字符串中指定子字符串之后的子字符串
l fn:split 函数：用于将源字符串拆分为一个字符串数组。
l fn:join 函数：用于将源字符串数组中的所有字符串连接为一个字符串。
l fn:toLowerCase 函数：用于将源字符串中的所有字符改为小写。
l fn: toUpperCase 函数：用于将源字符串中的所有字符改为大写。
l fn:trim 函数：用于将源字符串中的开头和末尾的空格删除。
l fn:escapeXml 函数：用于将源字符串中的字符“ < ”、“ > ”、“ ” ”和“ & ”等转换为转义字符。
l fn:length 函数：用于返回字符串中的字符的个数，或者集合和数组的元素的个数