# JSP

## JSP概述

### JSP介绍

由SUN公司提供的动态web资源的开发技术，看起来非常像html，但是可以在JSP页面中写java代码，所以JSP是一种动态web资源开发技术.

JSP本质上就是一个Servlet!

### JSP技术的由来

JSP的出现是为了解决Servlet在响应时不适合向外输出页面的问题

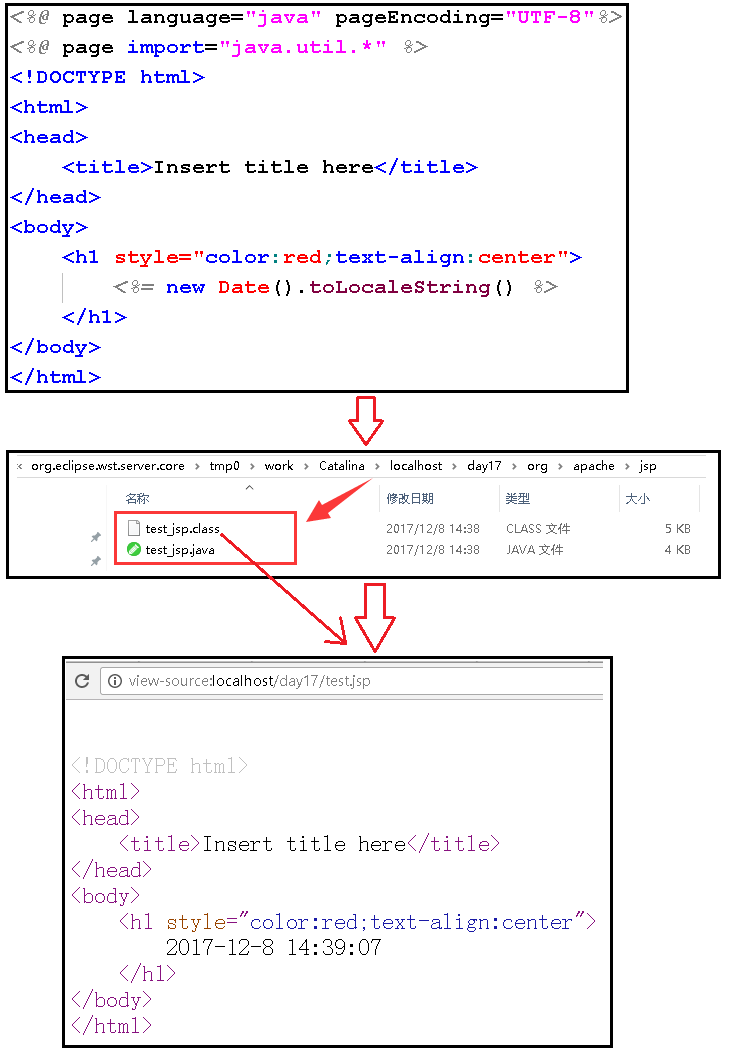
（1）Servlet本质上是一段java代码，非常适合处理逻辑，但是处理的结果不太适合由Servlet向外输出（不适合输出完整的html页面）。

（2）HTML是用于开发网页的一门技术，可以用来展示数据。但是HTML开发出来的页面本质上就是一个文档(静态资源)，无法展示动态的数据。

（3）JSP非常适合编写HTML代码，适合作为响应页面向外输出，同时JSP里可以写java代码，也可以展示动态的数据。

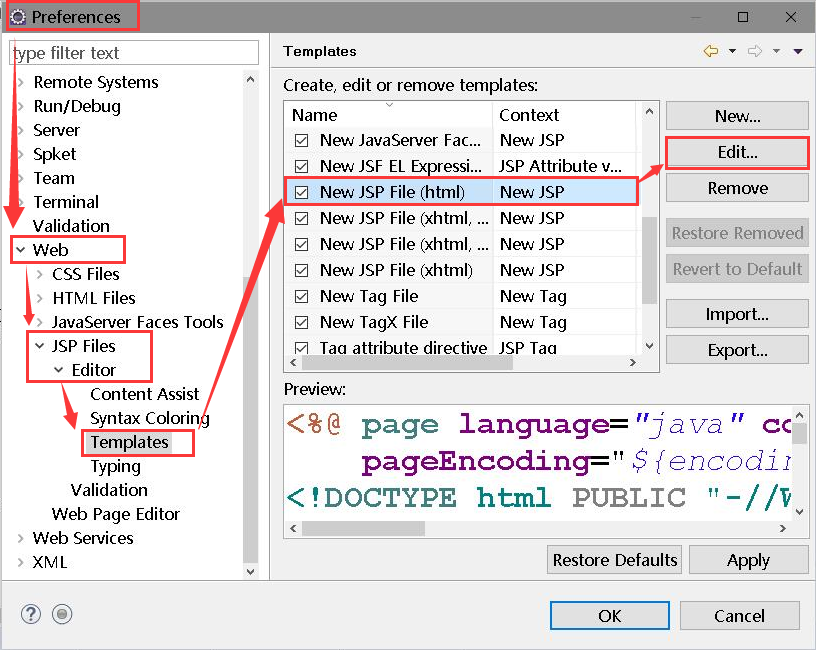
（4）所以JSP的出现既可以解决Servlet不适合向外响应一个完整的页面又可以解决html无法展示动态数据的问题。

（5）JSP在第一次访问时，会被翻译成一个Servlet，对JSP访问后看到的页面 其实就是翻译后的Servlet在向外输出！



### 修改JSP模版

修改JSP模版步骤: 点击菜单栏中的 window --> Preferences，出现如下窗口:



点击edit编辑JSP模版，修改为如下:

<%@ page language="java" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8"/>

<title></title>

</head>

<body>

${cursor}

</body>

</html>

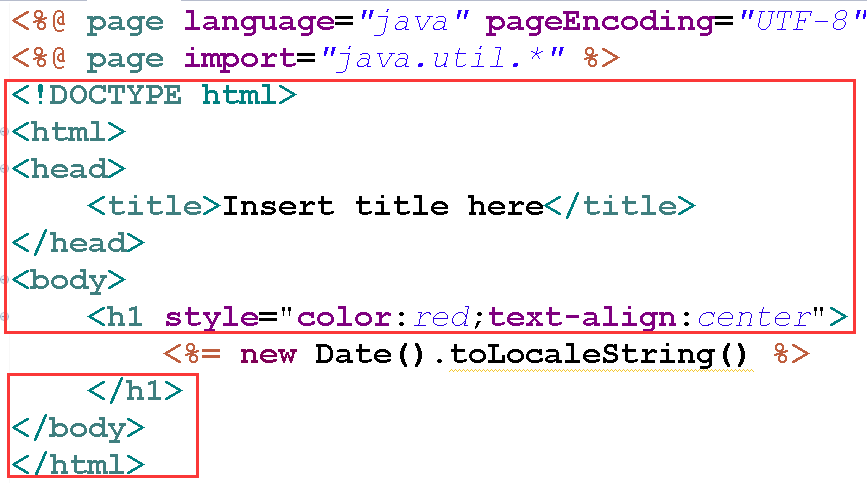
## JSP语法

### 模版元素

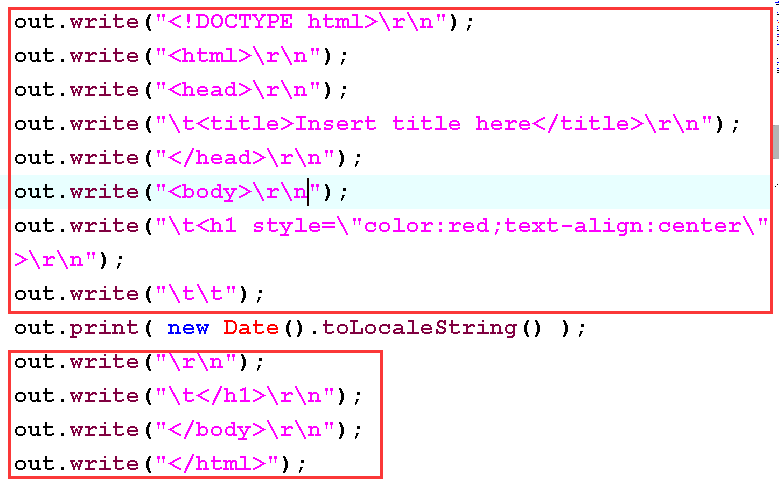
直接写在jsp页面中的html内容称之为jsp页面中的模版元素

模版元素在翻译过来的Servlet中被out.write()原样输出到浏览器中

比如:



模版元素在JSP翻译后的Servlet中直接被out.write原样输出. 如下:



### JSP表达式

格式: **<%= 表达式 %>** 其中可以书写常量、变量、表达式。

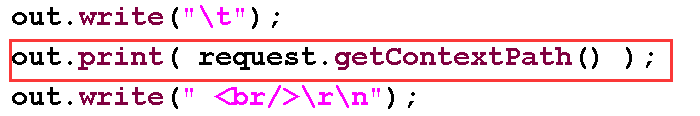
JSP表达式在翻译后的Servlet中是执行表达式的值再原样输出到浏览器，例如

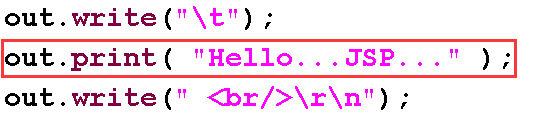
（1）在JSP页面编写内容如下：

<%= request.getContextPath() %> <br/>

<%= "Hello...JSP..." %> <br/>

（2）在翻译后的Servlet中翻译如下：





### JSP脚本片段

格式：**<% 若干java语句 %>**

在翻译过来的servlet中，脚本片段被复制粘贴到对应位置执行

翻译前:

<% for(int i=0; i<5; i++){ %>

Hello JSP~~~~~<br/>

<% } %>

翻译后:

for(int i=0; i<5; i++){

out.write("\r\n");

out.write("\t\tHello JSP~~~~~<br/>\r\n");

out.write("\t");

}

在某一个脚本片段中的java代码可以是不完整的，但是要求在翻译过来的servlet中整体的代码必须是完整符合java语法的.

### JSP注释:

格式: **<%-- JSP注释 --%>**

例如：<%-- out.write("aaa"); --%> 被JSP注释注释的内容，在翻译的过程中被抛弃，不会被翻译。

### JSP指令

JSP指令格式：**<%@ 指令名称 若干属性声明... %>**

-- 不会直接产生输出，用来指挥JSP解析引擎如何来翻译当前JSP页面中其他部分的内容（除指令意外的其他内容，例如模版元素、jsp表达式等）

**1.page指令**

-- 用来声明当前JSP页面的基本属性的，page指令可以写在JSP页面的任意位置，但是为了可读性考虑，一般情况下最好放在JSP页面的最前面

格式: **<%@ page ... %>**

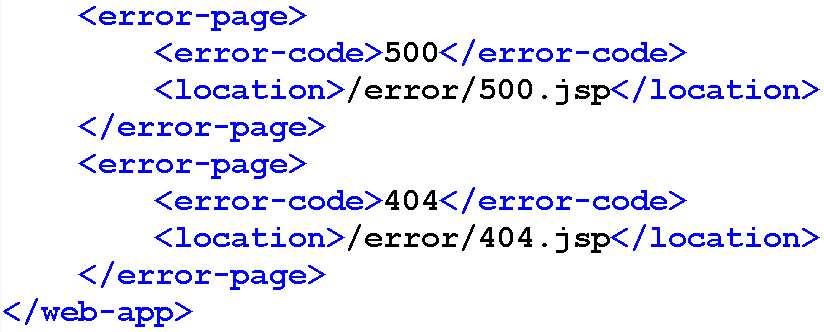
（1）<%@ page **language="java"** %> -- 指定当前JSP使用的语言是java.(了解即可)

（2）<%@ page **pageEncoding="UTF-8"**%> -- 用来通知JSP解析引擎使用指定的编码来翻译JSP。如果想防止JSP乱码，应该要保证JSP文件保存时的编码和pageEncoding指定的编码保持一致。

（3）<%@ page **session="true"**%> -- 用来指定当前页面是否使用session，如果设置为true，则翻译过来的servlet中将会有对session对象的引用，于是可以直接在JSP中使用session隐式对象。但是这将导致一旦访问JSP就会调用request.getSession()方法，可能导致不必要的空间浪费。如果确定JSP中不需要session可以设为false

（4）<%@ page **errorPage="filepath"**%> -- 为当前JSP页面指定友好错误提示页面，即当前JSP如果抛出了异常，将会跳转到errorPage属性指定的页面。

扩展：如果整个WEB应用中有很多JSP都需要指定错误提示页面，如果挨个为每一个JSP通过errorPage来指定，非常繁琐，所以推荐在web.xml文件中为整个网站的异常统一配置友好错误提示页面，配置如下:



（5）<%@ page **import="..."**%> -- 为JSP翻译后的Servlet指定所依赖的jar包.例如:

<%@page import="java.util.Date"%>

<%@page import="java.io.File" %>

<%@page import="java.sql.DriverManager"%>

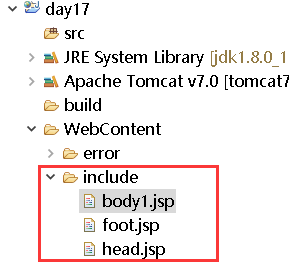
**2.include指令**

-- 可以实现页面包含的效果

格式：<%@ include file="" %>

示例: 创建三个JSP文件(分别为head.jsp，body.jsp，foot.jsp)，在body.jsp中包含head.jsp和foot.jsp，最终实现将三个文件的内容一起响应给浏览器.

（1）创建三个JSP文件(分别为head.jsp，body.jsp，foot.jsp)



（2）分别实现三个JSP文件的内容如下:

a)\_head.jsp

<%@ page language=*"java"*

pageEncoding=*"utf-8"*%>

<h1 style="background-color:*pink*;height:*200px*;">

网页的头部部分

</h1>

b)body.jsp

<%@ page language=*"java"*

pageEncoding=*"utf-8"*%>

<!DOCTYPE html>

<html>

<head>

<title></title>

</head>

<body>

<!-- 引入头部 -->

<%@include file=*"/include/\_head.jsp"* %>

<h1 style="background-color:*red*;height:*400px*;">

网页的主体部分

</h1>

<!-- 引入尾部 -->

<%@include file=*"/include/\_foot.jsp"* %>

</body>

</html>

c)\_foot.jsp

<%@ page language=*"java"*

pageEncoding=*"utf-8"*%>

<h1 style="background-color:*cyan*;height:*100px*;">

网页的尾部部分

</h1>

（3）测试: 浏览器访问的结果为：



**3.taglib指令**

导入tld标签库文件

## JSP的九大隐式对象

JSP翻译引擎在将JSP翻译成servlet的过程中，在servlet里预先定义了九个对象，因此我们可以在JSP页面中直接使用这九个对象

page this

request ServletRequest

response ServletResponse

application ServletContext

config ServletConfig

session HttpSession

exception

out 等价于response.getWriter()

pageContext

### pageContext对象

代表当前JSP页面的运行环境的对象，通过该对象可以访问页面中的共享数据

1.pageContext对象可以获取其他八大隐式对象

（1）pageContext.getPage(); -- 获取page隐式对象

（2）pageContext.getRequest(); -- 获取request隐式对象

（3）pageContext.getResponse(); -- 获取response隐式对象

（4）pageContext.getServletContext(); -- 获取application隐式对象

（5）pageContext.getServletConfig(); -- 获取config隐式对象

（6）pageContext.getSession(); -- 获取session隐式对象

（7）pageContext.getException(); -- 获取exception隐式对象

（8）pageContext.getOut(); -- 获取out隐式对象

2.pageContext是一个域对象

setAttribute(String name，Object value) – 添加一个域属性

getAttribute(String name) – 根据指定的属性名获取属性值.

removeAttribute(String name) – 根据指定的属性名删除一个属性.

生命周期:当访问JSP开始时创建pageConext对象，当访问JSP结束时销毁pageContext对象.

作用范围: 整个JSP页面

主要功能: 在整个JSP页面中实现数据的共享

## JSP标签技术

在JSP页面中写入大量的java代码会导致JSP页面中html代码和java代码混杂在一起，会造成页面非常的混乱，难于维护

于是在JSP的2.0版本中，sun提出了JSP标签技术，推荐使用标签来代替JSP页面中java代码，并且推荐，JSP2.0以后不要在JSP页面中出现任何一行java代码。

### EL表达式

EL表达式在JSP中可以非常方便的获取数据，可以代替JSP页面中的JSP表达式(<%= %>)

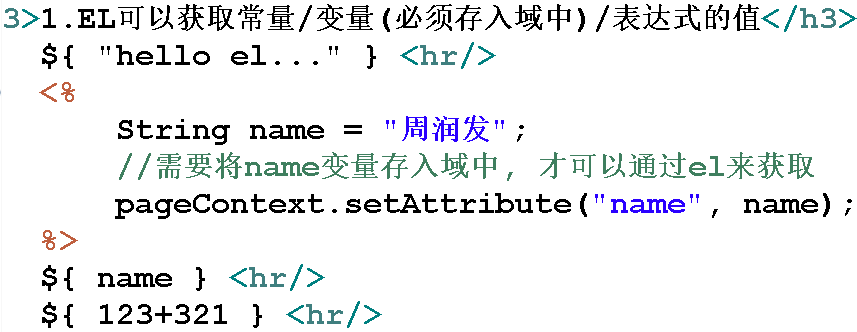
基本结构: **${ 表达式 }**

EL只能获取不能设置!!!

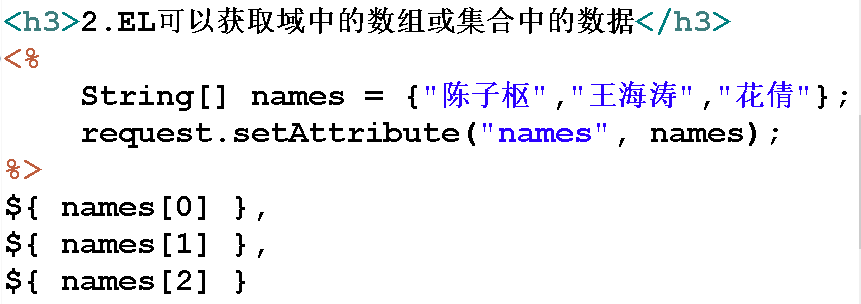
EL只能获取不能遍历!!!

EL表达式提供了如下功能：

（1）EL可以获取常量/变量(必须存入域中)/表达式的值

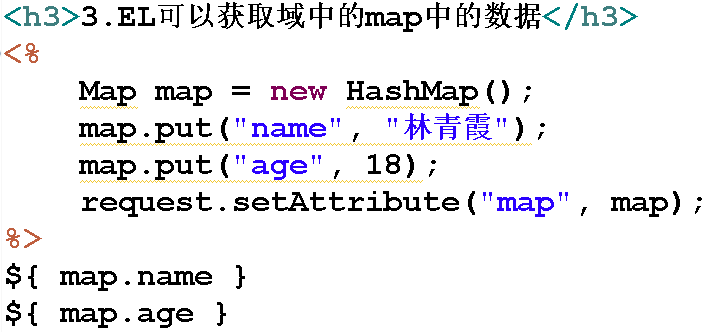


（2）EL可以获取域中的数组或集合中的数据



获取域中集合中的数据和获取数组的方式相同. 这里不再举例

（3）EL可以获取域中的Map集合中的数据。

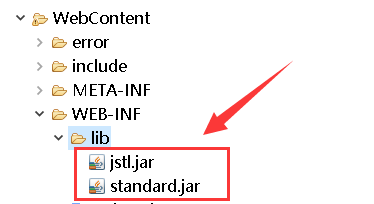


### JSTL标签库

JSTL标签库是为JavaWeb开发人员提供的一套标准通用的标签库，JSTL标签库和EL配合使用取代JSP中大部分的Java代码.

在使用JSTL标签库提供的标签之前，必须在JSP中通过taglib指令引入JSTL标签库(如果缺少JSTL库相关的jar，还需要提前导入jar包)

1、导入JSTL的jar包



2、引入JSTL库

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

其中常用的标签如下:

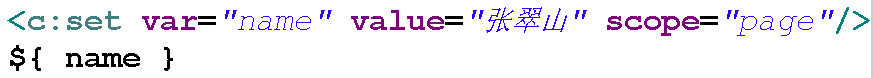
**1、<c:set></c:set>** -- 往四大作用域中添加域属性，或者修改四大作用域中已有的属性

（1）往四大作用域中添加一个域属性

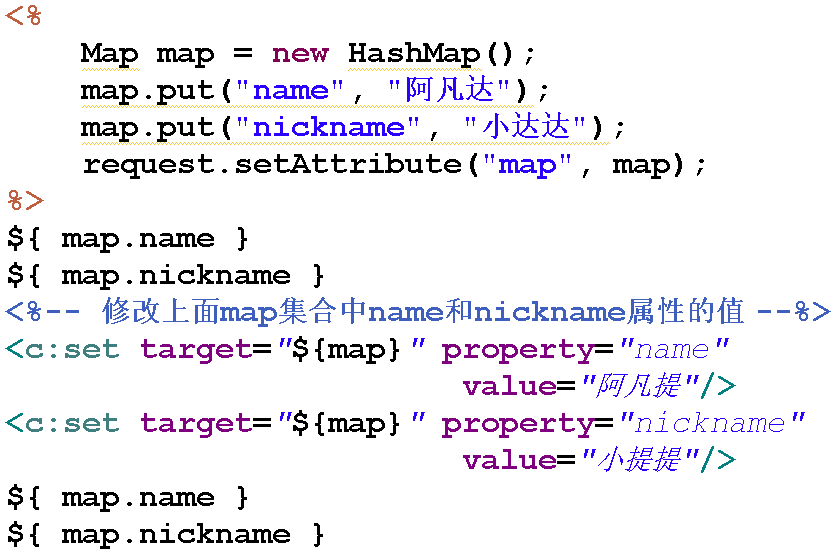


（2）修改四大作用域中已有的属性

如果重复添加相同的属性，值会发生覆盖，相当于修改.



（3）修改作用域中Map集合中的属性



**c\_set标签属性总结:**

(1)var – 指定存入域中属性的名称

(2)value -- 指定存入域中属性的值

(3)scope – 指定将属性如此哪一个域中

(4)target – 指定修改域中的哪一个集合，比如:<c:set target="${map}"...

(5)property – 指定修改域中集合中的哪一个属性

**2、<c:if></c:if> -- 构造if…else…语句**

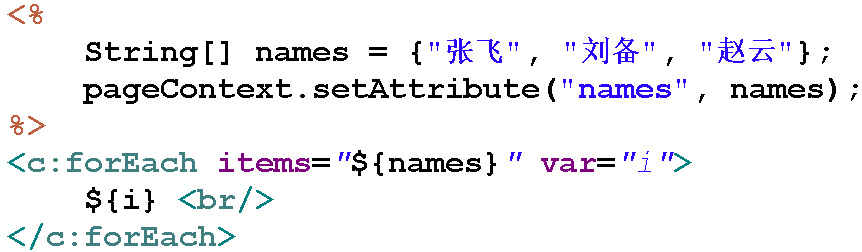
<c:if test="${3>5}">yes</c:if>

<c:if test="${!(3>5)}">no</c:if>

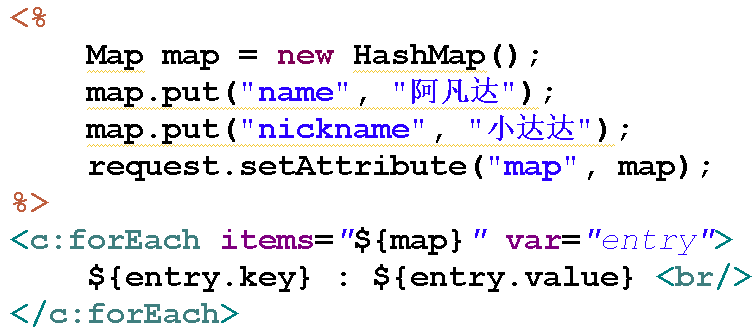
test属性用于指定判断的条件，注意:JSTL中没有提供else对应的标签

**3、<c:forEach></c:forEach> -- 对集合或数组等中元素进行循环遍历或者是执行指定次数的循环.**

（1）遍历域中数组或集合中的元素



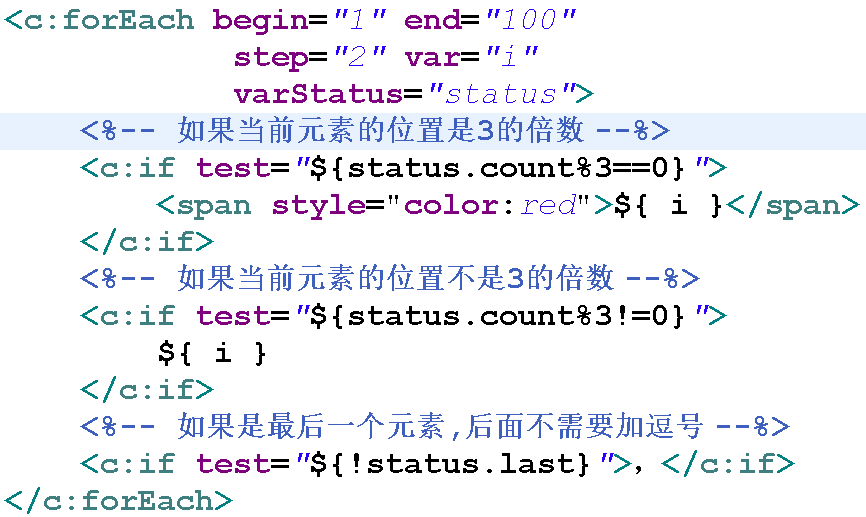
（2）遍历域中map集合中的元素



（3）打印1~100之间的奇数，并指定间隔符号



（4）打印1~100之间的奇数并指定间隔符号，并将位置是3的倍数的数值指定颜色为红色.



**c\_forEach标签属性总结:**

(1)items: 指定需要遍历的集合或数组

(2)var: 指定用于接收遍历过程中的元素

(3)begin: 指定循环从哪儿开始

(4)end: 指定循环到哪儿结束

(5)step: 指定循环时的步长

(6)varStatus="status" 返回一个表示循环状态的对象,该对象还具有如下属性:

a)count:表示当前遍历的元素是第几个 b)first:表示当前遍历的元素是否为第一个

c)last:表示当前遍历的元素是否为最后一个 d)index:表示当前遍历元素的索引(从零开始)