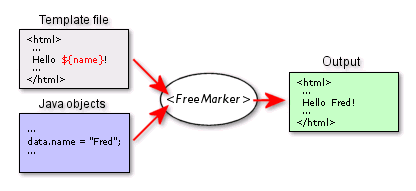
# FreeMarker语言

## FreeMarker语言概述

FreeMarker是一个模板引擎，一个基于模板生成文本输出的通用工具，使用纯Java编写。

FreeMarker被设计用来生成HTML Web页面，特别是基于MVC模式的应用程序

虽然FreeMarker具有一些编程的能力，但通常由Java程序准备要显示的数据，由FreeMarker生成页面，通过模板显示准备的数据（如下图）



FreeMarker不是一个Web应用框架，而适合作为Web应用框架一个组件。

FreeMarker与容器无关，因为它并不知道HTTP或Servlet；FreeMarker同样可以应用于非Web应用程序环境。

FreeMarker更适合作为Model2框架（如Struts）的视图组件，你也可以在模板中使用JSP标记库。

FreeMarker是免费的。

## FreeMarker特性

### 通用目标

能够生成各种文本：HTML、XML、RTF、Java源代码等等

易于嵌入到你的产品中：轻量级；不需要Servlet环境

插件式模板载入器：可以从任何源载入模板，如本地文件、数据库等等

你可以按你所需生成文本：保存到本地文件；作为Email发送；从Web应用程序发送它返回给Web浏览器

### 强大的模板语言

所有常用的指令：include、if/elseif/else、循环结构

在模板中创建和改变变量

几乎在任何地方都可以使用复杂表达式来指定值

命名的宏，可以具有位置参数和嵌套内容

名字空间有助于建立和维护可重用的宏库，或者将一个大工程分成模块，而不必担心名字冲突

输出转换块：在嵌套模板片段生成输出时，转换HTML转义、压缩、语法高亮等等；你可以定义自己的转换

### 通用数据模型

FreeMarker不是直接反射到Java对象，Java对象通过插件式对象封装，以变量方式在模板中显示

你可以使用抽象（接口）方式表示对象（JavaBean、XML文档、SQL查询结果集等等），告诉模板开发者使用方法，使其不受技术细节的打扰

### 为Web准备

在模板语言中内建处理典型Web相关任务（如HTML转义）的结构

能够集成到Model2 Web应用框架中作为JSP的替代

支持JSP标记库

为MVC模式设计：分离可视化设计和应用程序逻辑；分离页面设计员和程序员

### 智能的国际化和本地化

字符集智能化（内部使用UNICODE）

数字格式本地化敏感

日期和时间格式本地化敏感

非US字符集可以用作标识（如变量名）

多种不同语言的相同模板

### 强大的XML处理能力

<#recurse> 和<#visit>指令（2.3版本）用于递归遍历XML树。在模板中清楚和直觉的访问XML对象模型。开源论坛 [JForum](http://www.oschina.net/p/jforum) 就是使用了 FreeMarker 做为页面模板。

## 第一个FreeMarker程序

1. 建立一个普通的java项目：testFreeMarker
2. 引入freemarker.jar包
3. 在项目目录下建立模板目录：templates
4. 在templates目录下，建立a.ftl模板文件，内容如下：

|  |
| --- |
| 你好啊，${user}，今天你的精神不错！ |

1. 建立com.sxt.test.freemarker包，然后建立Test1.java文件，内容如下：

|  |
| --- |
| package com.sxt.test.freemarker;  import java.io.File;  import java.io.OutputStreamWriter;  import java.io.Writer;  import java.util.HashMap;  import java.util.Map;  import freemarker.template.Configuration;  import freemarker.template.DefaultObjectWrapper;  import freemarker.template.Template;  public class Test1 {  public static void main(String[] args) throws Exception {  //创建Freemarker配置实例  Configuration cfg = new Configuration();    cfg.setDirectoryForTemplateLoading(new File("templates"));    //创建数据模型  Map root = new HashMap();  root.put("user", "老高");    //加载模板文件  Template t1 = cfg.getTemplate("a.ftl");    //显示生成的数据,//将合并后的数据打印到控制台  Writer out = **new** OutputStreamWriter(System.*out*);  t1.process(root, out);  out.flush();  //显示生成的数据,//将合并后的数据直接返回成字符串！  // StringWriter out = new StringWriter();  // t1.process(root, out);  // out.flush();  // String temp = out.toString();  // System.out.println(temp); }  } |

1. 编译和运行Test1.java文件，控制台打印：



## 数据类型

1. **直接指定值**

直接指定值可以是字符串、数值、布尔值、集合及Map对象。  
**1. 字符串**  
直接指定字符串值使用单引号或双引号限定。字符串中可以使用转义字符”\"。如果字符串内有大量的特殊字符，则可以在引号的前面加上一个字母r，则字符串内的所有字符都将直接输出。  
  
**2. 数值**  
数值可以直接输入，不需要引号。FreeMarker不支持科学计数法。  
  
**3. 布尔值**   
直接使用true或false，不使用引号。  
  
**4. 集合**  
集合用中括号包括，集合元素之间用逗号分隔。  
使用数字范围也可以表示一个数字集合，如1..5等同于集合[1, 2, 3, 4, 5]；同样也可以用5..1来表示[5, 4, 3, 2, 1]。  
  
**5. Map对象**  
Map对象使用花括号包括，Map中的key-value对之间用冒号分隔，多组key-value对之间用逗号分隔。  
注意：Map对象的key和value都是表达式，但key必须是字符串。

**6. 时间对象**

root.put("date1", **new** Date());

${date1?string("yyyy-MM-dd HH:mm:ss")}

**7. JAVABEAN的处理**

Freemarker中对于javabean的处理跟EL表达式一致，类型可自动转化！非常方便！

1. **输出变量值**

FreeMarker的表达式输出变量时，这些变量可以是顶层变量，也可以是Map对象的变量，还可以是集合中的变量，并可以使用点（.）语法来访问Java对象的属性。  
  
**1. 顶层变量**  
所谓顶层变量就是直接放在数据模型中的值。输出时直接用${variableName}即可。  
  
**2. 输出集合元素**  
可 以根据集合元素的索引来输出集合元素，索引用中括号包括。如： 输出[“1”， “2”， “3”]这个名为number的集合，可以用${number[0]}来输出第一个数字。FreeMarker还支持用**number[1..2]**来表示原 集合的子集合[“2”， “3”]。  
  
**3. 输出Map元素**  
对于JavaBean实例，FreeMarker一样把它看作属性为key，属性值为value的Map对象。  
输出Map对象时，可以使用点语法或中括号语法，如下面的几种写法的效果是一样的：  
             book.author.name                                                                                                                           
             book.author["name"]                                                                                                                        
             book["author"].name                                                                                                                        
             book["author"]["name"]                                                                                                                     
使用点语法时，变量名字有和顶层变量一样的限制，但中括号语法没有任何限制。

**三、字符串操作**

**1. 字符串连接**  
字符串连接有两种语法：  
（1） 使用${..}或#{..}在字符串常量内插入表达式的值；  
(2)  直接使用连接运算符“+”连接字符串。  
如，下面两种写法等效：  
              ${"Hello, ${user}"}                                                                                                                  
              ${"Hello, " + user + "!"}                                                                                                           
有一点需要注意： ${..}只能用于文本部分作为插值输出，而不能用于比较等其他用途，如：  
              <#if ${isBig}>Wow!</#if>                                                                                                                  
              <#if "${isBig}">Wow!</#if>                                                                                                                
应该写成：  
              <#if isBig>Wow!</#if>                                                                                                                       
  
**2. 截取子串**  
截取子串可以根据字符串的索引来进行，如果指定一个索引值，则取得字符串该索引处的字符；如果指定两个索引值，则截取两个索引中间的字符串子串。如：  
              <#assign number="01234">  
              ${number[0]} <#-- 输出字符0 -->  
              ${number[0..3]} <#-- 输出子串“0123” -->

**四、集合连接操作**      连接集合的运算符为“+”  
  
**五、Map连接操作**  
 Map连接操作的运算符为“+”  
  
**六、算术运算符**  
 FreeMarker表达式中支持“+”、“－”、“\*”、“/”、“%”运算符。  
  
**七、比较运算符**

表达式中支持的比较运算符有如下几种：  
1. =（或者==）： 判断两个值是否相等；  
2. !=： 判断两个值是否不相等；  
注： =和!=可以用作字符串、数值和日期的比较，但两边的数据类型必须相同。而且FreeMarker的比较是精确比较，不会忽略大小写及空格。  
3. >（或者gt）： 大于  
4. >=（或者gte）： 大于等于  
5. <（或者lt）： 小于  
6. <=（或者lte）： 小于等于  
注： 上面这些比较运算符可以用于数字和日期，但不能用于字符串。大部分时候，使用gt比>有更好的效果，因为FreeMarker会把>解释成标签的结束字符。可以使用括号来避免这种情况，如：<#if (x>y)>。

|  |
| --- |
| if else 语句测试：  <#if num0 gt 18> <#--不是使用>，大部分时候，freemarker会把>解释成标签结束！ -->  及格！  <#else>  不及格！  </#if> |
| root.put("num0", 18); |

**八、逻辑运算符**

1. &&： 逻辑与；  
2. ||： 逻辑或；  
3. !： 逻辑非  
逻辑运算符只能用于布尔值。

**九、内建函数**

FreeMarker提供了一些内建函数来转换输出，可以在任何变量后紧跟?，?后紧跟内建函数，就可以通过内建函数来转换输出变量。  
  
字符串相关常用的内建函数：  
1. html： 对字符串进行HTML编码；  
2. cap\_first： 使字符串第一个字母大写；  
3. lower\_case： 将字符串转成小写；  
4. upper\_case： 将字符串转成大写；  
  
集合相关常用的内建函数：  
1. size： 获得集合中元素的个数；  
  
数字值相关常用的内建函数：  
1. int： 取得数字的整数部分。

举例：

|  |
| --- |
| root.put("htm2", "<b>粗体</b>"); |
| 内建函数：  ${htm2?html} |

**十、空值处理运算符**

FreeMarker的变量必须赋值，否则就会抛出异常。而对于FreeMarker来说，null值和不存在的变量是完全一样的，因为FreeMarker无法理解null值。  
FreeMarker提供两个运算符来避免空值：  
1. !： 指定缺失变量的默认值；  
2. ??：判断变量是否存在。  
!运算符有两种用法：variable!或variable!defaultValue。第一种用法不给变量指定默认值，表明默认值是空字符串、长度为0的集合、或长度为0的Map对象。  
使用!运算符指定默认值并不要求默认值的类型和变量类型相同。

|  |
| --- |
| 测试空值处理：  <#-- ${sss} 没有定义这个变量，会报异常！ -->  ${sss!} <#--没有定义这个变量，默认值是空字符串！ -->  ${sss!"abc"} <#--没有定义这个变量，默认值是字符串abc！ --> |

??运算符返回布尔值，如：variable??，如果变量存在，返回true，否则返回false。



## 数据类型常见示例

直接指定值

 字符串 ： "Foo"或 者'Foo'或"It's \"quoted\""或r"C:\raw\string"

 数字：123.45

 布尔值：true, false

 序列：["foo", "bar", 123.45], 1..100

 哈希表：{"name":"green mouse", "price":150}

 检索变量  顶层变量：user

 从哈希表中检索数据：user.name, user[“name”]

 从序列中检索：products[5]

 特殊变量：.main

 字符串操作

 插值（或连接）："Hello ${user}!"（或"Free" + "Marker"）

 获取一个字符：name[0]

 序列操作

 连接：users + ["guest"]

 序列切分：products[10..19] 或 products[5..]

 哈希表操作

 连接：passwords + {"joe":"secret42"}

 算数运算: (x \* 1.5 + 10) / 2 - y % 100

 比 较 运 算 ： x == y, x != y, x < y, x > y, x >= y, x <= y,

x &lt; y, 等等

 逻辑操作：!registered && (firstVisit || fromEurope)

 内建函数：name?upper\_case

 方法调用：repeat("What", 3)

 处理不存在的值

 默认值：name!"unknown" 或者(user.name)!"unknown" 或者

name! 或者 (user.name)!

 检测不存在的值：name?? 或者(user.name)??

参考：运算符的优先级

## 模板开发语句

最简单的模板是普通 HTML 文件（或者是其他任何文本文件—FreeMarker 本身不属于HTML）。当客户端访问页面时，FreeMarker 要发送 HTML 代码至客户端浏览器端显示。如果想要页面动起来，就要在 HTML 中放置能被 FreeMarker 所解析的特殊部分。

${…}：FreeMarker 将会输出真实的值来替换花括号内的表达式，这样的表达式被称为

interpolations 插值，可以参考第上面示例的内容。

FTL tags 标签（FreeMarker 模板的语言标签）：FTL 标签和 HTML 标签有一点相似，但是它们是 FreeMarker 的指令而且是不会直接输出出来的东西。这些标签的使用一般以符号#开头。（用户自定义的 FTL 标签使用@符号来代替#，但这是更高级的主题内容了，后面会详细地讨论）

Comments 注释：FreeMarker 的注释和 HTML 的注释相似，但是它用<#--和-->来分隔的。任何介于这两个分隔符（包含分隔符本身）之间内容会被 FreeMarker 忽略，就不会

输出出来了。

其他任何不是 FTL 标签，插值或注释的内容将被视为静态文本，这些东西就不会被

FreeMarker 所解析，会被按照原样输出出来。

directives指令：就是所指的 FTL 标签。这些指令在 HTML 的标签（如<table>和

</table>）和 HTML 元素（如 table 元素）中的关系是相同的。（如果现在你还不能区

分它们，那么把“FTL 标签”和“指令”看做是同义词即可。）

### if指令

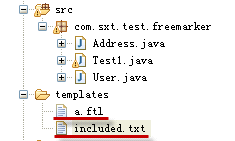
|  |
| --- |
| root.put("random", **new** Random().nextInt(100)); |
| ------------------------------------------------  if语句测试：  ${user}是<#if user=="老高">我们的老师</#if>  ------------------------------------------------  if else 语句测试：  <#if num0 gt 18> <#--不是使用>，大部分时候，freemarker会把>解释成标签结束！ -->  及格！  <#else>  不及格！  </#if>  ---------------------------------------------------  if else if else语句测试：  <#if random gte 90>  优秀！  <#elseif random gte 80>  良好！  <#else>  一般！  </#if>  ---------------------------------------------------- |

### list指令

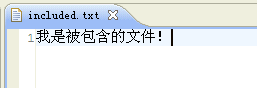
|  |
| --- |
| List list = **new** ArrayList();  list.add(**new** Address("中国","北京"));  list.add(**new** Address("中国","上海"));  list.add(**new** Address("美国","纽约"));  root.put("lst", list); |
| 测试list指令：  <#list lst as dizhi >  <b>${dizhi.country}</b> <br/>  </#list>  **思考问题：**<c:forEach> status属性。在此处如何实现? |
| 控制台打印：  测试list语句：  <b>中国</b> <br/>  <b>中国</b> <br/>  <b>美国</b> <br/> |

### include指令

增加被包含文件，放于templates目录下：



文件内容如下：



模板文件中代码如下：

|  |
| --- |
| 测试include指令：  <#include "included.txt" /> |

### 自定义指令(macro指令)

|  |
| --- |
| <#macro m1> <#--定义指令m1 -->  <b>aaabbbccc</b>  <b>dddeeefff</b>  </#macro> |
| <@m1 /><@m1 /> <#--调用上面的宏指令 --> |

定义带参的宏指令：

|  |
| --- |
| <#macro m2 a b c >  ${a}--${b}--${c}  </#macro> |
| <@m2 a="老高" b="老张" c="老马" /> |

nested指令：

|  |
| --- |
| <#macro border>  <table border=4 cellspacing=0 cellpadding=4><tr><td>  <#nested>  </td></tr></table>  </#macro> |
| <@border >表格中的内容！</@border> |

**欧阳鸿：**宏指令中，有没有类似于方法的返回值?

## 命名空间

当运行 FTL 模板时，就会有使用 assign 和 macro 指令创建的变量的集合（可能是空的），可以从前一章节来看如何使用它们。像这样的变量集合被称为 namespace 命名空间。在简单的情况下可以只使用一个命名空间，称之为 main namespace 主命名空间。因为通常只使用本页上的命名空间，所以就没有意识到这点。

如果想创建可以重复使用的宏，函数和其他变量的集合，通常用术语来说就是引用

library 库。使用多个命名空间是必然的。只要考虑你在一些项目中，或者想和他人共享使用的时候，你是否有一个很大的宏的集合。但要确保库中没有宏（或其他变量）名和数据模型中变量同名，而且也不能和模板中引用其他库中的变量同名。通常来说，变量因为名称冲突也会相互冲突。所以要为每个库中的变量使用不同的命名空间。

定义b.ftl文件：

|  |
| --- |
| <#macro copyright date>  <p>Copyright (C) ${date} 北京尚学堂.</p>  </#macro>  <#assign mail = "bjsxt@163.com"> |

在a.ftl文件中引入b.ftl，从而可以使用b.ftl中定义的宏和变量：

|  |
| --- |
| 测试命名空间：  <#import "b.ftl" as bb />  <@bb.copyright date="2010-2011" />  ${bb.mail}  <#assign mail="my@163.com" />  ${mail}  <#assign mail="my@163.com" in bb />  ${bb.mail} |

执行后，控制台打印：

|  |
| --- |
| 测试命名空间：  <p>Copyright (C) 2010-2011 北京尚学堂.</p>  bjsxt@163.com  my@163.com  [my@163.com](mailto:my@163.com) |

## 命名空间命名规则

如果你为 Example 公司工作，它们拥有 www.example.com 网的主页，你的工作是开发

一个部件库，那么要引入你所写的 FTL 的路径应该是：

/lib/example.com/widget.ftl

注意到 www 已经被省略了。第三次路径分割后的部分可以包含子目录，可以像下面这

样写：

/lib/example.com/commons/string.ftl

一个重要的规则就是路径不应该包含大写字母，为了分隔词语，使用下划线\_，就像

wml\_form（而不是 wmlForm）。

如果你的工作不是为公司或组织开发库，也要注意，你应该使用项目主页的 URL，比如

/lib/example.sourceforge.net/example.ftl或/lib/geocities.com/jsmith/example.ftl。

## 在Servlet中使用Freemarker

参考Freemarker包中example目录下webapp1项目！

## struts2中整合FreeMarker

1.解压struts2-core-X.X.X.jar文件，把在META-INF文件夹下面的struts-tags.tld文件复制到WEB-INF文件夹下。   将freemark的jar导入到工程中

２.在web.xml文件中配置freemark同时启动JSPSupportServlet.代码如下：

http://www.blogjava.net/Images/OutliningIndicators/None.gif<servlet>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        <servlet-name>freemarker</servlet-name>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        <servlet-class>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif            freemarker.ext.servlet.FreemarkerServlet  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        </servlet-class>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        <!--下面的配置freemarke的ftl文件的位置 -->  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        <init-param>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif            <param-name>TemplatePath</param-name>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif            <param-value>/</param-value>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        </init-param>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        <!-- 是否和服务器（tommcat)一起启动。-->  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        <load-on-startup>1</load-on-startup>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif    </servlet>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif  
http://www.blogjava.net/Images/OutliningIndicators/None.gif    <servlet-mapping>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        <servlet-name>freemarker</servlet-name>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        <url-pattern>\*.ftl</url-pattern>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif    </servlet-mapping>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif  
http://www.blogjava.net/Images/OutliningIndicators/None.gif  
http://www.blogjava.net/Images/OutliningIndicators/None.gif<servlet>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif  <!-- define a JspSupportServlet Object -->  
http://www.blogjava.net/Images/OutliningIndicators/None.gif  <servlet-name>JspSupportServlet</servlet-name>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif  <servlet-class>org.apache.struts2.views.JspSupportServlet</servlet-class>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif  <!-- setting JspSupportServlet auto start -->  
http://www.blogjava.net/Images/OutliningIndicators/None.gif  <load-on-startup>1</load-on-startup>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif </servlet>

3.在FreeMarker模板中使用assign指令导入标签库。代码如下

<#assign s=JspTaglibs["/WEB-INF/struts-tags.tld"] /> 注：这里我把struts-tags.tld放在WEB-INF下面

4.现在我们可以在FreeMarker模板中使用标签了。

详情可见web项目：testFreeMarker2

作业：

1. 使用list标签遍历list容器时，如何获取索引(下划线加index即可获取)：

|  |
| --- |
| -------------------------------------  测试list索引\_index  <#list list as city>  ${city}<br/>${city\_index}  <#if city\_index==1>  <#break>  </#if>  </#list>  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  测试list中国\_has\_next  <#list list as city>  ${city}<br/>${city\_index}  <#if city\_has\_next>  我有下一项！------- ${city\_index}  </#if>  </#list> |

1. 调用对象或类的方法?
2. .main特殊变量的作用
3. 定义的macro能否有返回值，有的话，怎么处理?