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 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**test.ftl**模板文件，内容如下：

```
你好啊，${user}，今天你的精神不错！
```

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

```
import freemarker.template.Configuration;
import freemarker.template.DefaultObjectWrapper;
import freemarker.template.Template;

import java.io.File;
import java.io.FileOutputStream;
import java.io.OutputStreamWriter;
import java.io.Writer;
import java.util.Date;
import java.util.HashMap;
import java.util.Map;

public class App {
    public static void main(String[] args) throws Exception {
        Configuration cfg = new Configuration(Configuration.VERSION_2_3_23);
        // 指定模板文件从何处加载的数据源，这里设置成一个文件目录。
        cfg.setDirectoryForTemplateLoading(
                new File("templates"));
        // 指定模板如何检索数据模型，这是一个高级的主题了...
        // 但先可以这么来用：
        cfg.setObjectWrapper(new DefaultObjectWrapper());
```

```
        // 创建根哈希表
        Map root = new HashMap();
        // 在根中放入字符串"user"
        root.put("user", "Big Joe");

        Template temp = cfg.getTemplate("test.ftl");

        Writer out = new OutputStreamWriter(new FileOutputStream("test.html"));
        temp.process(root, out);
        out.flush();
    }
}
```

数据类型

一、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！

二、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：

```
<#ifisBig>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)>。

```
root.put("num0", 18);
<#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>
```

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("name", "neld");

内建函数：
${name?cap_first}
```

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":"greenmouse", "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 < 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

和

）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" />
```