模板引擎快速指南

什么是 Velocity?

Velocity 是一个基于 **Java** 的模板引擎，它可以使得前端的开发者轻松的引用后端 **Java** 代码中定义的方法。前端和后端的开发可以根据 **MVC** 模型并行的进行，这意味着 **Web** 开发工程师们可以更关注于自己的事情，并使得 **Web** 具有更好的可维护性。**Velocity** 模板使用 **VTL(Velocity** 模板语言**)**编写。

简单例子

这是一个最简单的例子，使用 **set** 指令定义了一个变量 **foo**，变量以“**$**”符开头，当它被赋值后可以在你的 **HTML** 文档中的任何一个地方引用它。

<html>

<body>

#set($foo = "百度**")**

你好啊**$foo!**

</body>

<html>

页面输出结果为：

你好啊百度！

代码中的 **set** 指令一会儿我会更详细的说明。

注释

第一种注释方法是以**##**开头，类似 **C++**代码里的**//**。
##这是一行注释呀这是一行注释
第二种注释方法是在段落的首位分别用**#***和***#**注释，当注释行数较多时用此方法更合适。
#*
这是很多行注释啊很多行注释
这是很多行注释啊很多行注释
这是很多行注释啊很多行注释
这是很多行注释啊很多行注释

这是很多行注释啊很多行注释
这是很多行注释啊很多行注释
#
第三种注释方法主要用于设置一些文档的作者、版本等信息。

#**
这是个很强大很全面的多行注释啊多行注释
这是个很强大很全面的多行注释啊多行注释
这是个很强大很全面的多行注释啊多行注释
@author Bosn
@version 1.0.2
#
引用

在 **VTL** 中有三种引用的类型：变量、属性和方法。在 **VTL** 中所有的类型在模板中都会解析为 **String**，假设有一个对象**$foo** 在 **Java** 代码中是整型，则 **Velocity** 将会调用它的**.toString()** 方法将其转换为 **String**。

变量

VTL 的命名规则很简单，第一个字符必须是字母**(a..z** 或 **A..Z)**，其他的部分限于以下类型：

- 字母（**a..z, A..Z**）
- 数字**(0..9)**
- 减号**(“-“)**
- 下划线**(“_”)**

以下是标识符合法的变量：

- **$foo**
- **$bosnMa**
- **$bosn-ma**
- **$bosn_ma**
- **$bosnMa1**

变量既可以通过 **set** 指令赋值（**FE**），也可以通过 **Java** 代码赋值（**RD**）。例如，如果一个 **Java** 变量**$foo** 的值为”**abc**”，当模板被请求时，所有页面上的**$foo** 都将被赋值”**abc**”。

如果我在模板中使用以下语句：
$set($foo = “abc”)
效果和前者相同，但赋值的位置不同。

属性

VTL 变量的属性和其它语言类似，变量名之后紧接一个**“.“**和另一个变量标识符，一下是

属性的示例：
$customer.Address
$purchase.Total
我们先看第一行，**$customer.Address**。这个语句有两个含义，它可以表示查找标识符为**customer**的**哈希表**并返回键为**Address**相对应的值，但**$customer.Address**也可以意指引用一个方法（引用方法下个部分会详细讨论）当页面被请求时，**Velocity**会决定两种可能哪一项有意义，并返回最合适的值。

方法

VTL的方法**(Method)**的概念和其它语言相差无几，方法也是以**$**开头，后接**VTL**标识符和**VTL**方法体，最后有一对括号，括号内输入可选的参数列表。一下是合法的参数引用：
$customer.getAddress()
$purchase.getTotal()
$page.setTitle(“My Home Page”)
$person.setAttributes([“Strange”,“Weird”,“Excited”])
关于方法，比较有趣的一件事情就是**VTL**属性可以作为**VTL**方法的缩写。比如，属性**$customer.Address**和使用方法**$customer.getAddress()**等价。当发生冲突时（既有**Address**属性，又有**getAddress()**方法），使用**$customer.Address**会优先返回属性**Address**的值。属性和方法的主要区别是只有方法才可以为其输入参数列表。

属性的查找规则

之前我们提到过，**Velocity**会非常聪明的找到被请求的属性所对应的方法，它会基于命名规则尝试找到合适的选项。属性的首字母大写和小写的情况是要区分的，对于小写开头的属性，例如**$customer.address**，查找顺序依次为：
1.getaddress()
2.getAddress()
3.get(“address”)
4.isAddress()
对于大写开头的属性，例如**$customer.Address**，则查找顺序依次为：
1.getAddress()
2.getaddress()
3.get(“Address”)
4.isAddress()

正式的引用法

在以上示例中我们引用变量时，使用“**$**”符紧接变量名，如**$foo**，除此之外还有一种正式的引用方法，如下所示：
${foo}
${customer.Address}
${customer.getAddress()}
对于一般的使用，之前的章节所讲述的方法是完全够用的，但有些特殊情况我们必须使用本

节所讲的正式引用法。例如，我们引用一个变量**$vice**，对于如下语句：

Jack is a $vicemaniac.

$vice 不能够正常解析，因为“**vice**”和文本“**maniac**”紧挨着，造成变量名被解析成**$vicemaniac**。正确的写法，是使用正式引用法：

Jack is a ${vice}maniac.

这样就可以区分特殊情况下的变量名和普通文本。

静引用

什么是静引用？比如，在模板中我们放置一个文本框

<input type=”text” name=”email” value=”$email” />

正常情况还好，后端 **Java Code** 或使用 **set** 都可以为**$email** 赋值，可是一旦**$email** 没有被赋值，字符串”**$email**”就会被显示出来，出于这方面的考虑，我们可以使用静引用，如下所示：

<input type=”text” name=”email” value=”$!email” />

在”**$**”和标识符之间用叹号隔开，这个时候当**$email** 被赋值时和普通变量一样，但当**$email** 没有被赋值时，**$!email** 将默认为空字符串，也就是不会显示在页面上。

另外，正式引用法和静引用时可以同时使用的，如下所示：

<input type=”text” name=”email” value=”$!{email}” />

文本

符号”**$**”和”**#**”是 **VTL** 的特殊字符，它们往往会被解释成 **VTL** 变量或指令的开头，所以在使用它们的时候要格外的小心。在文本中出现这些特殊字符的时候，我们要进行一些处理，告诉 **Velocity** 这些符号和普通文本无异。

例如，当我们写”**Give me $9999 please!**”这句话时不会出现什么问题，因为 **VTL** 标识符总是以字母开头，所以**$9999**不会被误解为变量引用，但当普通文本和现有变量引用存在冲突的时候，我们可以使用右斜杠“****”转义。例如，

#set($email = “foo”)
$email

如果 **Velocity** 在你的模板中碰到**$email**，它将会搜索是否有对应的值。上例的输出为 **foo**，因为**$email** 已定义，如果没有定义，则将输出**$email**（若想未定义时不输出，使用上面讲过的静引用）。现在问题来了，如果**$email** 已经定义了，但是我想输出文本“**$email**”，怎样实现？能解决这个问题的方法不只一种，其中最简单的方法就是使用转义符。

#set ($email = “foo”)
$email
\$email
\\$email
\\\$emailf

输出依次为

foo
$email

\foo
\$email
注意转义符的绑定顺序是从左至右，和 **C++**类似， 前两行很容易看懂，第三行两个“****”将输出一个“****”，在第四行中，我们按照顺序分析，第一个“****”发现临近的一个“****”，结合输出一个“****”，之后第三个“****”临近一个“**$**”，结合，转义，输出一个“**$**”，然后输出普通文本“**email**”，最终结果为“**\$email**”。

对于未定义的引用要注意，比如
#set($foo = “hello baidu”)
$undefined = $foo
这里的输出将会是：**$defined = hello baidu**，由于**$undefined** 没有定义，所以 **Velocity** 会将它当做普通的文本处理，这点**一定要注意**。

等价写法

在上面的属性一节中，我们已经介绍过了等价写法，这些写法主要是取自 **Java** 命名规则的优点，使模板的编写更简单。首先给出以下示例：

$foo

$foo.getBar()
##等价于
$foo.Bar

$data.setUser(“bosn”)
##等价于
$set($data.User = “bosn”)

$data.getRequest().getServerName()
##等价于
$data.Request.ServerName
##等价于
${data.Request.ServerName}

指令

VTL 引用使得模板编写者能够为 **Web** 动态的生成内容，**VTL** 指令总是以**#**开头，类似于 **VTL** 引用，指令也可以用花括号**{}**包围，同样，在某些特殊情况这种表示法会非常有用，如下所示，将会产生错误：

#if($a==1)true enough#elseno way!#end

这里原想当**$a** 等于**1**时输出 **true enough,**否则输出 **no way**，但是**#elseno** 被解析成一个引用，由于**#if** 找不到对应的 **else**，错误便产生。正确的写法如下所示：

#if($a==1)true enough#{else}no way!#end

细节决定成败，这些细节也是一旦出现，最让大家头疼的罪魁祸首，所以我们一定要尽可能

的抓住这些常见的细节问题。

#set 指令

#set 指令用于为变量赋值。变量可以被另外一个变量或属性赋值，例如：

```
#set(  $primate = “monkey” )
#set( $customer.Behavior = $primate )
```

赋值时，左值必须是变量的引用或属性的引用，而右值可以是以下列表中的任意类型：

- 变量引用
- 字符串文本
- 属性引用
- 方法引用
- 数字文本
- 数组
- Map

下面让我们一起来看例子吧：

```
#set( $monkey = $bill )##变量引用
#set( $monkey.Friend = “monica” )##字符串文本
#set( $monkey.Blame = $whitehouse.Leak )##属性引用
#set( $monkey.Plan = $spindoctor.weave($web) )##方法引用
#set( $monkey.Number = 123 )##数字文本
#set( $monkey.Say = [“Not”, $my, “fault”] )##数组
#set( $monkey.Map = {“banana”:”good”, “roast beef”: “bad”})## Map
```

注意：用**[..]**定义的数组可以用 **ArrayList** 类定义的方法访问，例如上面的例子中的数组的第一个元素可以用**#monkey.Say.get(0)**访问。类似的，用**{}**操作符定义的 **Map** 可以使用 **Map** 类定义的方法，例如上面的例子中的 **Map** 的第一个元素可以用 **#monkey.Map.get(“banana”)**访问。

除此之外右值还可以是一个简单的数学表达式：

```
#set( $value = $foo + 1 )
#set( $value = $bar - 1 )
#set( $value = $foo * $bar )
#set( $value = $foo / $bar )
```

这里又要提到一个需要注意的细节，当右值是一个为空或未定义的属性或方法的引用时，左值将不会被赋值，这个非常容易搞混。我们先看例子：

#set($query = {“name”:” Bosn”} ##这里定义了一个 **Map**，只有一个键”**name**”，并且值为”**Bosn**”

```
#set( $result = $query(“name”) )
The result of the first query is $result
#set( $result = $query(“address”) )##注意 query 并没有”address”键
The result of the second query is $result
```

输出结果为：

```
The result of the first query is Bosn
The result of the second query is Bosn
```

这很容易令人疑惑，由于**$query(“address”)**为空，**$result** 并没有被赋值，而不是像 **C++** 等其他语言那样将 **Null** 赋给**$result**，因此**$result** 依旧保存旧值。我们再看一个例子：

```
#set( $query = {“name”:”Bosn”,“address”:”F5-BE329”}
#set ($keys = [“name”,“address”,“age”]##定义了一个数组，存放要查找的所有键。
#foreach( $i in $keys)
        #set( $result = $query.get($i) )
        #if($result)
                Query was successful
        #end
#end
```

输出为：

```
Query was successful
Query was successful
Query was successful
```

有些同学可能会迷惑，**key** ”**age**”在 **query** 中未定义，为什么会输出三句“**Query was successful**”**?**由于循环在第二轮时**$result** 被赋为键”**address**”对应的值”**F5-BE329**”，而**query.get(“age”)**为空，所以并不赋值。因此第三次循环**$result** 依然为”**F5-BE329**”，所以依然会输出结果。为了解决这个问题，我们再每次循环的开头将**$result** 赋值为 **false**，如下所示：

```
#set( $query = {“name”:”Bosn”,“address”:”F5-BE329”})
#set ($keys = [“name”,“address”,“age”])##定义了一个数组，存放要查找的所有键。
#foreach( $i in $keys)
        #set( $result = false )
        #set( $result = $query.get($i) )
        #if($result)
                Query was successful
        #end
#end
```

输出为：

```
Query was successful
```

Query was successful

像**#foreach**、**#if** 等指令都要由**#end**，而**#set** 指令并不需要。

文本

VTL 的文本处理因为和其它语言有些细微差异，所以容易搞混，请各位同学认真区分。首先，在用**#set** 指令时，双引号“”包围的文本中如果含有变量引用是会被解析的（**C/C++/Java/C#**都不会），例如：

```
#set( $directoryRoot = “www” )
#set( $templateName = “index.vm” )
#set( $template = “$directoryRoot / $templateName” )
```

输出为：

www / index.vm

但是以单引号’’包围的字符串将不会被解析：

```
#set( $directoryRoot = “www” )
#set( $templateName = “index.vm” )
#set( $template = ‘$directoryRoot / $templateName’ )
```

输出为

$directoryRoot / $templateName

默认情况下，用单引号避免文本被解析的功能是开启的，但这个默认设置也可以在 **velocity.properties** 中修改，例如 **stringliterals.interpolate = false**。

放置文本被解析的另一种方式是使用**#literal** 命令，这在将多行字符串设置为当做文本处理时非常有用，例子如下：

```
#literal()
#foreach( $i in $myMap)
        nothing will happen to $i
#end
#end
```

结果为：

```
#foreach( $i in $myMap)
        nothing will happen to $i
#end
```

条件分支

If/ElseIf/Else

VTL 中的**#if** 指令类似于其它语言的 **if** 语句，例如：

#if($foo)
Hello Baidu!
#end

当变量**$foo** 为 **true**（**$foo** 是 **boolean** 型，值为 **true**，或者 **value** 为其它类型，值不为空时都将在 **if** 中返回 **true**）时，结果为**Hello Baidu!**，否则结果为空。下面是使用 **elseif** 的例子，用法和其它编程语言相同，不再累述：

```
#if( $foo < 10 )
   <strong>Go North</strong>
#elseif( $foo == 10 )
   <strong>Go East</strong>
#elseif( $bar == 6 )
   <strong>Go South</strong>
#else
   <strong>Go West</strong>
#end
```

逻辑或、逻辑与、逻辑非

直接上例子：

```
## logical AND
#if( $foo && $bar )
  <strong> This AND that</strong>
#end

当且仅当$foo 和$bar 都返回 true 时，输出<strong> This AND that</strong>
，否则输出为空。

## logical OR
#if( $foo || $bar )
   <strong>This OR That</strong>
#end

只要$foo 和$bar 有任何一个为真（包括都为真），输出<strong> This OR that</strong>
，否则输出为空。

##logical NOT
#if( !$foo )
  <strong>NOT that</strong>
#end

当$foo 为 true 时，输出为空，否则，输出<strong>NOT that</strong>。
```

循环

#foreach 循环，例子如下：

```
<ul>
#foreach( $product in $allProducts )
   <li>$product</li>
#end
</ul>

该循环遍历数组$allProducts 中的所有元素，和 C#的 foreach 语句相同。
```

Include 和 Parse 指令

#include 指令可以将本地文件导入到模板中该条指令所在位置。使用**#include** 导入的内容将不会通过模板引擎，为了安全，只有在 **TEMPLATE_ROOT(**模板的根目录及该目录的所有子目录、子子目录等**)**下的文件才可以使用**#include** 导入。下面是例子：

#include(“one.text”)

如果要导入的文件不只一个，可以用逗号”**,**”隔开：

#include(“one.gif” , “two.txt” , “three.htm”)

我们也可以使用变量，在运行时识别导入的模板的文件名：

#include(“greetings.txt” , $seasonalstock)

#parse 和**#include** 类似，但是它允许导入的本地文件中包含有 **VTL** 指令，**Velocity** 将会解析 **VTL** 指令。例如：

#parse(“me.vm”)

如同**#include** 指令，**#parse** 也可以将变量作为参数。任何使用**#parse** 导入的文件都必须在 **TEMPLATE_ROOT** 下。不同于**#include**，**#parse** 指令只允许带有一个参数。

停止处理

使用**#stop** 指令可以停止模板引擎的执行，这在调试的时候会很有用。

宏

#macro 指令可以用来定义宏，无论对于简单还是复杂的情形，宏都是广泛使用的非常强大的工具，我们可以这样定义一个宏：

```
#macro(d)
        <tr><td></td></tr>
#end
```

括号中的 **d** 是所定义的宏的标识符（简单的说就是名字），我们可以这样调用一个宏：

#d()

当这个模板被请求时，**Velocity** 会将**#d()**替换为宏 **d** 中的内容，即“**<tr><td></td></tr>**”。

宏是可以带任意数量的参数的，包括**0**个（如上面的例子），在下面的例子中，定义了一个带有两个参数的宏，第一个**$color** 是 一个变量，第二个**$somelist** 是一个数组。

```
#macro( tablerows $color $somelist )
#foreach( $something in $somelist )
   <tr><td bgcolor=$color>$something</td></tr>
#end
#end

下面是调用这个宏的例子：
#set( $greatlakes = ["Superior","Michigan","Huron","Erie","Ontario"] )
#set( $color = "blue" )
<table>
   #tablerows( $color $greatlakes )
</table>

最后将输出：

<table>
   <tr><td bgcolor="blue">Superior</td></tr>
   <tr><td bgcolor="blue">Michigan</td></tr>
   <tr><td bgcolor="blue">Huron</td></tr>
   <tr><td bgcolor="blue">Erie</td></tr>
   <tr><td bgcolor="blue">Ontario</td></tr>
</table>
```

参考文献


```
http://velocity.apache.org/engine/releases/velocity-1.5/user-guide.html
```