**Velocity教程**

参考：<http://blog.csdn.net/nengyu/article/details/6671904>

Velocity是一个基于java的模板引擎（template engine）。它允许任何人仅仅简单的使用模板语言（template language）来引用由java代码定义的对象。 当Velocity应用于web开发时，界面设计人员可以和java程序开发人员同步开发一个遵循MVC架构的web站点，也就是说，页面设计人员可以只 关注页面的显示效果，而由java程序开发人员关注业务逻辑编码。Velocity将java代码从web页面中分离出来，这样为web站点的长期维护提 供了便利，同时也为我们在JSP和PHP之外又提供了一种可选的方案。

一、Velocity脚本语法摘要

1. 变量

(1)变量的定义：

#set($name = "hello") 说明：velocity中变量是弱类型的。

当使用#set 指令时，括在双引号中的字面字符串将解析和重新解释，如下所示：

#set($directoryRoot = "www" )

#set($templateName = "index.vm" )

#set($template = "$directoryRoot/$templateName" )

$template

输出将会是：www/index.vm

注：在velocity中使用$2.5这样的货币标识是没有问题得的，因为velocity中的变量总是以一个大写或者小写的字母开始的。

(2)变量规范的写法

${name} ，也可以写成：$name。提倡用前面的写法。

例如：你希望通过一个变量$vice来动态的组织一个字符串。

 Jack is a $vicemaniac.

本来变量是$vice现在却变成了$vicemaniac，这样Veloctiy就不知道您到底要什么了。所以，

应该使用规范的格式书写 ： Jack is a ${vice}maniac

现在Velocity知道变量是$vice而不是$vicemaniac。

注意：当引用属性的时候不能加{}（me：也就是说将引用的方式都写在一个{}中）

**(3)变量的赋值:**

 $name="hello"

赋值的左边必须是一个变量或者是属性引用。右边可以是下面六种类型之一：

 变量引用,字面字符串,属性引用,方法引用,字面数字,数组列表（me：与PHP很相似）。

下面的例子演示了上述的每种类型：

#set( $monkey = $bill ) ## variable reference

#set( $monkey.Friend = "monica" ) ## string

 #set( $monkey.Blame = $whitehouse.Leak ) ## property reference

#set( $monkey.Plan = $spindoctor.weave($web) ) ## method reference

#set( $monkey.Number = 123 ) ##number

#set( $monkey.Say = ["Not", $my, "fault"] ) ## ArrayList

注意：①如果上述例子中的右值是null, 则左值不会被赋值，也就是说会保留以前的值。

②velocity模板中未被定义的变量将被认为是一个字符串。例如：

#set($foo = "gibbous")

$moon = $foo

输出结果为：

$moon = gibbous

③velocity模板中不会将reference解释为对象的实例变量。例如：$foo.Name将被解释为Foo对象的getName（）方法，而不是Foo对象的Name实例变量。例如：

$foo.getBar() 等同于$foo.Bar ；

$data.getUser("jon") 等同于$data.User("jon") ；

data.getRequest().getServerName() 等同于

$data.Request.ServerName等同于${data.Request.ServerName}

**2. 循环**

#foreach ($element in $list)

This is $element.

   $velocityCount

   #end

例子：

#set( $list = ["pine", "oak", "maple"])

#foreach ($element in $list)

$velocityCount

This is $element.  
#end

输出的结果为：

1 This is pine.

2 This is oak.

3 This is maple.

每次循环$list中的一个值都会赋给$element变量。

$list可以是一个Vector、Hashtable或者Array。分配给$element的值是一个java对象，并且可以通过变量被引用。

例如：如果$element是一个java的Product类，并且这个产品的名字可以通过调用他的getName（）方法得到。

#foreach ( $key in $list.keySet())

Key: $key -> Value: $list.get($key) <br>  
#end

提示：velocity中大小写敏感。

Velocity还特别提供了得到循环次数的方法，$velocityCount变量的名字是Velocity默认的名字。

 例子：

First example:

#foreach ( $foo in [1..5] )

    $foo

    #end

Second example:

#foreach ( $bar in [2..-2] )

    $bar

    #end

Third example:

#set ( $arr = [0..1] )

    #foreach ( $i in $arr )

    $i

   #end

上面三个例子的输出结果为：

 First example：

 1 2 3 4 5

Second example：

  2 1 0 -1 -2

Third example：

 0 1

**3. 条件语句**

#if（me：命令都是以#开头，与预编译指令相似） (condition)

#elseif (condition)

#else

 #end

**4. 语句的嵌套**

#foreach ($element in $list)

 ## inner foreach 内循环

#foreach ($element in $list)

This is $element. $velocityCount <br>inner<br>  
#end

 ## inner foreach 内循环结束

 ## outer foreach

This is $element.

$velocityCount <br>outer<br>  
#end

语句中也可以嵌套其他的语句，如#if…#else…#end等。

**5. 注释**

(1)单行注释：##

 ## This is a single line comment.

(2)多行注释：#\*…\*#

#\*

 Thus begins a multi-line comment. Online visitors won’t

 see this text because the Velocity Templating Engine will

 ignore it.

 \*#

(3)文档格式：#\*\*…\*#

#\*\*

 This is a VTL comment block and

 may be used to store such information

 as the document author and versioning

information:

@version 1.1

@author xiao

\*#

**6. 关系和逻辑操作符**

Velocity 也具有逻辑AND, OR 和 NOT 操作符。

如

## example for AND

#if($foo && $bar)<strong>This and that</strong>

#end

例子中#if() 指令仅在$foo 和$bar 斗为真的时候才为真。如果$foo 为假，则表达式也为假；

并且 $bar 将不被求值。如果 $foo 为真，Velocity 模板引擎将继续检查$bar的值，如果 $bar 为真，则整个表达式为真。并且输出This AND that 。如果 $bar 为假，将没有输出因为整个表达式为假。

**7.Velocity 中的宏**

Velocity中的宏我们可以理解为函数。

①宏的定义

#macro(宏的名称 $参数1 $参数2 …)

       语句体(即函数体)

#end

②宏的调用

      #宏的名称($参数1 $参数2 …)

                说明：参数之间用空格隔开。

**8．#stop**

 停止执行模板引擎并返回,把它应用于debug是很有帮助的。

9．#include与#parse

#include和#parse的作用都是引入本地文件, 为了安全的原因，被引入的本地文件只能在TEMPLATE\_ROOT目录下。

区别：

(1) 与#include不同的是，#parse只能指定单个对象。而#include可以有多个

如果您需要引入多个文件，可以用逗号分隔就行：

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

在括号内可以是文件名，但是更多的时候是使用变量的：

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

(2) #include被引入文件的内容将不会通过模板引擎解析；

 而#parse引入的文件内容Velocity将解析其中的velocity语法并移交给模板，意思就是说相当与把引入的文件copy到文件中。

#parse是可以递归调用的，例如：如果dofoo.vm包含如下行：

Count down.<br>  
#set ($count = 8)

#parse ("parsefoo.vm")  
<br>All done with dofoo.vm!

那么在parsefoo.vm模板中，你可以包含如下VTL：

$count

#set($count = $count - 1)

#if ( $count > 0 )<br>  
#parse( "parsefoo.vm" )

#else  
<br>All done with parsefoo.vm!

#end的显示结果为：

Count down.

8

7

6

5

4

3

2

1

0

All done with parsefoo.vm!

All done with dofoo.vm!

注意：在vm中使用#parse来嵌套另外一个vm时的变量共享问题。如：

->a.vm 里嵌套 b.vm；

->a.vm 里定义了变量 $param；

->b.vm 里可以直接使用$param，无任何限制。

但需要特别注意的是，如果b.vm里同时定义有变量$param，则b.vm里将使用b.vm里定义的值（me：就近访问原则）。

**10．转义字符'\'的使用**

如果reference被定义，两个’\’意味着输出一个’\’，如果未被定义，刚按原样输出。

如：

#set($email = "foo" )

$email

\$emai

l\\$email

\\\$email

输出：

foo

$email

\foo

\$email

如果$email 未定义

$email

\$email

\\$email

\\\$email

输出：

$email

\$email

\\$email

\\$email

**11．内置对象**

Velocity内置了一些对象，在vm模版里可以直接调用，列举如下：

$request、$response、$session，另外，模板内还可以使用 $msg内的消息工具访问 Struts 的国际化资源，达到简便实现国际化的方法。

**12. 数组访问**

对数组的访问在Velocity中存在问题，因为Velocity只能访问对象的方法，而数组又是一个特殊的Array，所以虽然数组可以进行循环列举，但却不能定位访问特定位置的元素，如 strs[2]，数组对固定位置元素的访问调用了Array的反射方法get(Object array, int index)，而Velocity没能提供这样的访问，所以数组要么改成List等其他类容器的方式来包装，要么就通过公用Util类的方式来提供，传入数组对象和要访问的位置参数，从而达到返回所需值的目的。