特别说明

目　录

Velocity 语法

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{}

(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。

下面的例子演示了上述的每种类型：

```
#set( $monkey = $bill ) ## 变量引用
#set( $monkey.Friend = "monica" ) ## 字面字符串
#set( $monkey.Blame = $whitehouse.Leak ) ## 属性引用
#set( $monkey.Plan = $spindoctor.weave($web) ) ## 方法引用
#set( $monkey.Number = 123 ) ## 字面数字
#set( $monkey.Say = ["Not", $my, "fault"] ) ## 数组列表
```

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.<br>
#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 t 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 (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 都为真的时候才为真。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。

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" )
$email
\$email
\\$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。

示例部分

1. Hello world 的示例代码：

(1)Velocity 模板(hello.html)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE> New Document </TITLE>
</HEAD>
<BODY>
    hello,$name!                    (注意：这里的 name 与 VelocityTest.java 中的名称要一致)
</BODY>
</HTML>
```

(2)将 velocity 模板的内容转换的类(VelocityTest.java)

```
import java.io.File;
import java.io.FileOutputStream;
import java.io.PrintWriter;
import java.io.Writer;
import org.apache.velocity.Template;
import org.apache.velocity.VelocityContext;
import org.apache.velocity.app.Velocity;
import org.apache.velocity.app.VelocityEngine;

/**
 * Velocity 转换
 * @author
 */
public class VelocityTest
{
    /**
     * 主函数
```

```
 * @param args
 */
public static void main(String[] args)
{
    //获取模板引擎
    VelocityEngine ve = new VelocityEngine();
    //模板文件所在的路径
    String path = "D:/java/jproject/regedit/webroot";
    //设置参数
    ve.setProperty(Velocity.FILE_RESOURCE_LOADER_PATH, path);
    //处理中文问题
    ve.setProperty(Velocity.INPUT_ENCODING,"GBK");
    ve.setProperty(Velocity.OUTPUT_ENCODING,"GBK");
    try
    {
        //初始化模板
        ve.init();
        //获取模板(hello.html)
        Template template = ve.getTemplate("hello.html");
        //获取上下文
        VelocityContext root = new VelocityContext();
        //把数据填入上下文
        root.put("name","world");
        //输出路径
        Strint outpath = "e:/helloworld.html";
        //输出
        Writer mywriter = new PrintWriter(new FileOutputStream(
            new File(outpath)));
        template.merge(root, mywriter);
        mywriter.flush();
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
}
```

Velocity 模板的名称

（注意：与上面的对应）

(3)环境的搭建

在 lib 目录内分别 copy 进：velocity-1.4.jar，velocity-dept.jar;

下载地址：http://jakarta.apache.org/velocity/

(4)运行后的结果如下：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
```

```
<TITLE> New Document </TITLE>
</HEAD>
<BODY>
    hello,world!
</BODY>
</HTML>
```

2. Servlet 和 Velocity 结合示例

(1)example.html

```
<html>
 <head><title>Velocity</title></head>
 <body bgcolor="#ffffff">
   <center>
   <h2>Welcom to Velocity!</h2>
   <i>Here's the list of people</i>
   <table cellspacing="0" cellpadding="5" width="20%" >
   <tr>
   <td bgcolor="#eeeeee" align="center">
   Names:
   </td>
   </tr>
   #foreach ($name in $theList)
    <tr>
    <td bgcolor="#eeeeee" align="center">$name</td>
    </tr>
    #end
    </table>
    </center>
  </body>
</html>
```

(2)servlet

```
package com.koal.velocity;

import java.io.IOException;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.Properties;
import java.util.Vector;
import javax.servlet.ServletConfig;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.velocity.Template;
import org.apache.velocity.context.Context;
import org.apache.velocity.servlet.VelocityServlet;
```

```
import org.apache.velocity.app.Velocity;
import org.apache.velocity.exception.ResourceNotFoundException;
import org.apache.velocity.exception.ParseErrorException;

public class SampleServlet extends VelocityServlet
{
    /**
     *   由 VelocityServlet.init()调用,
     *   在此找出模版的路径
     */
protected Properties loadConfiguration(ServletConfig config )
    throws IOException, FileNotFoundException {
    Properties p = new Properties();
    //取得路径
    String path = config.getServletContext().getRealPath("/");
    if (path == null)
    {
        System.out.println(" SampleServlet.loadConfiguration() : unable to "
                + "get the current webapp root.  Using '/'. Please fix.");
        path = "/";
    }
    //设置路径
    p.setProperty( Velocity.FILE_RESOURCE_LOADER_PATH, path);

    return p;
}

/**
 *  Velocity 主要的商业逻辑处理方法，由 VelocityServlet 自动调用
 *  @param ctx 模板上下文
 *  @return Template 模板信息
 */
public Template handleRequest( HttpServletRequest request,
   HttpServletResponse response, Context ctx )
{
    //主要在此设置演示用的数据，开发中在此调用相应的业务处理流程，

    //并设置返回到页面的数据
    //待展示的列表数据
    String p1 = "第一位：LiuDong";
    String p2 = "第二位：Liang.xf";
    Vector personList = new Vector();
    //中文需要转换
    try {
```

```
        personList.addElement(new String(p1.getBytes(), "ISO-8859-1") );
        personList.addElement(new String(p2.getBytes(), "ISO-8859-1") );
    } catch (Exception e) {
        System.out.println("数据转换异常："+e);
    }
    //设置数据，供页面模版替换成显示的数据
    ctx.put("theList", personList );
    //定义模板
    Template outty = null;
    try
    {
        //取模板
        outty =  getTemplate("example.html");
    }
    catch( ParseErrorException pee )
    {
        System.out.println("SampleServlet: parse error for template " + pee);
    }
    catch( ResourceNotFoundException rnfe )
    {
        System.out.println("SampleServlet: template not found " + rnfe);
    }
    catch( Exception e )
    {
        System.out.println("Error " + e);
    }

    return outty;
}
```

(3)在 web.xml 中的配置：

```
<web-app>
    <servlet>
        <servlet-name>SampleServlet</servlet-name>
        <servlet-class>com.koal.velocity.SampleServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>SampleServlet</servlet-name>
        <url-pattern>/SampleServlet</url-pattern>
    </servlet-mapping>
</web-app>
```

(4)环境的搭建

在 lib 目录内分别 copy 进：commons-collections.jar,velocity-1.4.jar， velocity-dept.jar;

Tomcat 运行环境正常。

启动 Tomcat，在 IE 上输入：http://localhost:8080/example，页面显示数据列表：

Here's the list of people

Names:

第一位：LiuDong

第二位：Liang.xf