[MVC框架显示层——Velocity技术](http://blog.csdn.net/fasttalk/article/details/398059)

标签： [velocity](http://www.csdn.net/tag/velocity)[mvc](http://www.csdn.net/tag/mvc)[框架](http://www.csdn.net/tag/%e6%a1%86%e6%9e%b6)[struts](http://www.csdn.net/tag/struts)[jsp](http://www.csdn.net/tag/jsp)[servlet](http://www.csdn.net/tag/servlet)

2005-06-20 00:02 39054人阅读 [评论](http://blog.csdn.net/fasttalk/article/details/398059#comments)(15) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/fasttalk/article/details/398059#report)

http://static.blog.csdn.net/images/category_icon.jpg 分类：

Velocity/JSP（2） http://static.blog.csdn.net/images/arrow_triangle%20_down.jpg

版权声明：本文为博主原创文章，未经博主允许不得转载。

Velocity，名称字面翻译为：速度、速率、迅速，用在Web开发里，用过的人可能不多，大都基本知道和在使用Struts，到底Velocity和Struts(Taglib和Tiles)是如何联系？在技术上Velocity要比Struts Struts(Taglib和Tiles)先进，单从技术上考虑可以考虑Velocity；而Struts用的非常普遍，这样的人才也很多，所以对于公司来说，选择Struts的人才容易点。毕竟Velocity提供了一个很好的思维方式，给大家换了一种思考的方式。

Velocity是一种Java模版引擎技术，该项目由Apache提出，由另外一种引擎技术Webmacro引深而来。那什么是官方的Velocity定义呢？Apache对它的定义是：一种基于Java的模板引擎，但允许任何人使用简单而强大的模板语言来引用定义在Java代码中的对象。目前最新的版本是1.4，可以在<http://jakarta.apache.org/velocity/index.html>查找更多信息。

其实Velocity就是MVC架构的一种实现，但它更多的是关注在Model和View之间，作为它们的桥梁。对于MVC的最流行架构Struts来说，相信大家都不陌生，很多开发人员已经大量在使用Struts架构，包括IBM的Websphere 5以上的管理平台版本，Struts技术很好的实践了MVC，它有效的减少Java代码在View（Jsp）中的出现，但在Model和View之间还是依靠Struts的Taglib技术来实现，试想如果前台开发的网页设计师对Struts乃至Taglib不熟（相信也挺难熟的，包括后期的维护人员也一样），将会对网页设计师和前台开发工程师的相互协作开发带来很大的难度，现实开发中也还是存在这样事实，网页设计师和前台开发之间的工作或多或少还是存在一定的耦合，怎样最大限度的解决这个难题呢？还是让我们来看看Velocity或者说这个概念吧。

下面是一个简单的Velocity例子：

1、创建velocity模版（其实和html一样），文件名为：hellovelocity.vm

<html>   
<title>Hello Velocity</title>   
<body>   
Welcome  $name  to Javayou.com!   
today is  $date.   
</body>  
</html>

2、创建java文件，HelloVelocity.java ：

package com.fasttalk. velocity;   
import java.io.StringWriter;   
import java.util.\*;   
import org.apache.velocity.app.VelocityEngine;   
import org.apache.velocity.Template;   
import org.apache.velocity.VelocityContext;

public class HelloVelocity {   
public static void main(String[] args) throws Exception {  
//初始化并取得Velocity引擎  
VelocityEngine ve = new VelocityEngine();  
ve.init();  
//取得velocity的模版   
Template t = ve.getTemplate("hellovelocity.vm");   
//取得velocity的上下文context   
VelocityContext context = new VelocityContext();  
//把数据填入上下文   
context.put("name", "Liang");   
context.put("date", (new Date()).toString());  
//为后面的展示，提前输入List数值   
List temp = new ArrayList();  
temp.add("1");   
temp.add("2");   
context.put("list", temp);   
//输出流   
StringWriter writer = new StringWriter();  
//转换输出   
t.merge(context, writer);   
System.out.println(writer.toString());  
}   
}

在<http://jakarta.apache.org/site/binindex.cgi>上下载Velocity 1.4 zip

4、把1上的hellovelocity.vm copy到运行的当前目录下，运行结构如下：

<html>   
<title>Hello Velocity</title>   
<body>   
Welcome Liang to Javayou.com!   
today is Tue Dec 14 19:26:37 CST 2004.   
</body>   
</html>

以上是最简单的运行结果，怎么样，知道个大概吧，模版hellovelocity.vm里的2个定义变量$name和$date分别被context.put("name", "Liang")和context.put("date", (new Date()).toString())所设的值替代了。

由此看来业务流程处理包括业务结果基本在model这层全部解决，而view这一层基本只用使用简单的VTL（Velocity Template Language）来展示。这样，Jsp岂不是不用了么？是的，这样的使用模式有点象早前的CGI方式：）由Velocity自动输出代码，并且Velocity在这方面的能力也很强，Turbine里就采用了Velocity来产生很多代码。

在Velocity中，变量的定义都是使用“$”开头的，$作为Velocity的标识符。字母、数字、中划和下划线都可以作为Velocity的定义变量。

还需要注意的是Velocity特色的变量定义，如：$student.No、$student.Address，它有2层含义：

         第1种是如果student是hashtable，则将从hashtable中提取key为No和Address的值，

         另外第2种就是它有可能是调用方法，即上面2个变量将被转换为student.getNo()和student.getAddress()。

Velocity对在servlet中的java code返回的值有对象，还可以调用对象的方法，如$ student.getAddress()等等，在此就不一一举例和深入了。

上面的例子只是简单的举例，现在当然不少人已经不满足这样的例子了，实际的应用中我们还常常需要作些选择性展示和列举一些迭代数据，如List列表，当然Velocity（具体来说应该是VTL模版语言）也支持这项功能，此外还支持其他一些常用的展示，如模版内部的变量（如Jsp内的变量），还有强大一些的如创建宏以实现自动化，让我们继续接着往下看吧。

我们还是使用上面的例子，把模版hellovelocity.vm中的内容改为：

#set( $iAmVariable = "good!" )  
Welcome $name to csdn.net!   
today is $date.   
$iAmVariable

重新执行上面的运行命令，结果：

Welcome Liang to csdn.net!   
today is Tue Dec 14 22:44:39 CST 2004.  
good!

可以看得模版中的变量定义为# set开头的语句，不是很难理解，执行后模版中的变量$iAmVariable都转换成定义的值：good!

再来看看简单的选择，把模版hellovelocity.vm中的内容改为：

#set ($admin = "admin")   
#set ($user = "user")    
#if ($admin = = $user)   
Welcome admin!   
#else    
Welcome user!   
#end

执行运行命令，结果：

Welcome user!

可以看到判断语句只是简单的#if ()、#else、#end，不是很复杂。

接着继续来看看迭代数据吧，把模版hellovelocity.vm中的内容改为：

#foreach( $product in $list )   
<li>$product</li>   
#end

执行运行命令，结果：

<li>1</li>

<li>2</li>

把在例子中预先保存在VelocityContext的List中的值列举了出来，是不是很方便啊？仅仅只是用了#foreach（$variable in xx） 而已，如果上面的List换成Hashtable，则可以用下面的语法：

#foreach($key in $hashVariable.keySet() )   
<li> $key ‘s value: $ hashVariable.get($key) </li>   
#end

一点不觉得这些脚本很复杂。

还有不少人还会问，如果是javabean怎么办？好的，我们增加一个bean：

package com.fasttalk.velocity;   
  
public class Student {    
//注意class的属性是public的    
public String no = "";    
public String address = ""    
public Student(String \_no, String \_address) {   
no = \_no;   
address = \_address;   
}    
public String getAddress() {    
return address;   
}    
public void setAddress(String address) {    
this.address = address;   
}    
public String getNo() {    
return no;   
}    
public void setNo(String no) {    
this.no = no;   
}    
}

这个Student是实足的javabean，或者说是data bean，常见的用来装载数据的类，然后我们修改HelloVelocity.java，把：

temp.add("1");

temp.add("2");

替换成：

temp.add(new Student("123", "Guangzhou"));    
temp.add(new Student("456", "Zhuhai"));

再把hellovelocity.vm的内容改为：

#foreach ($s in $students)   
<$velocityCount> Address: $s.address    
#end

重新编译和执行运行命令，结果如下：

<1> Address: Guangzhou    
<2> Address: Zhuhai

这样把list中Student的数据打印了出来，大功告成！这里用了Velocity的内建变量$velocityCount，指的是默认的列举序号，从1开始，也可以改成0开始，但需要在Velocity.properties中更改，Velocity.properties位于velocity-1.4.jar包内的目录org/apache/velocity/runtime/defaults 下。

再复杂一些的迭代怎么处理呢？我们看看下面的模版例子就清楚了：

#foreach ($element in $list)  
 -- inner foreach --   
#foreach ($element in $list)   
This is $element.   
$velocityCount   
#end  
-- inner foreach --   
-- outer foreach --   
This is $element.   
$velocityCount   
-- outer foreach --   
#end

看出来了吧，Velocity是支持标签嵌套的，这个可是很强大的功能，这里就不深入演示了，如果有兴趣，自己试试吧。

其实，稍为深入思考刚刚我们举的例子，就已经可以看出来，Velocity的用处在哪里？即Servlet + Velocity的模式，另外，还记得我们早期Jsp开发的模式Jsp+JavaBean吗？在这里，我们更改为Servlet+JavaBean+Velocity，想想，是不是已经替代了Jsp+JavaBean，并更彻底的把Java代码去除在Jsp（vm）外，如果光使用Struts（Servlet+Jsp），那么带来的代价是Java代码总或多或少出现在Jsp上，即使可以做到不出现Java代码，但做过复杂架构系统的开发者都知道，代价也是很昂贵的，并且在可维护性、和网页设计师的集成开发上存在一定的困难，所以我们在这里能感觉到，Servlet+JavaBean+Velocity的模式较好的实现了OOD的概念。而在效率上，大家也不用担心，此种结合方式比Servlet+Jsp的方式要高效一些。

愿意了解Velocity的人应该不少，但真正实用到项目的，也许不多（还是有些项目在使用，如Jute），毕竟和Jsp比起来，Jsp更标准、更广泛使用和有不少开发工具已经支持Jsp开发。但Velocity的功能不会仅仅局限在和Jsp竞争的局面，由上可看出它在自动代码输出方面功能很强，前面提到Turbine就是采用Velocity来生成很多代码，你也可以稍加改动就可以做成代码生成器，或其他模版生成上，都是很不错的想法。

好了，我们再来看看要深入Velocity来做项目，还需要注意的一些常见问题吧，首先是国际化的问题，

Velocity本身支持模版的国际化编码转换，看看Velocity提供的方法：

Public Template getTemplate (Stirng template, String encoding)，

由此推测这样做其实不能彻底的做到国际化。

最简单的在Struts中国际化的概念，即在Jsp上使用国际化语言标签的方式来做到，而每种语言采用不同的语言标签库的方式，引申到这里，其实手工来做一样可以做到，只不过需要稍加手工处理而已。

好在已经有人处理了上面所说问题，做成了Velocity的tools: MessageTool，提供了变量text包含国际化标签，这样只需要简单的编写标签代码即可，如：$text.get(‘title’)，更多具体的内容还可在<http://jakarta.apache.org/velocity/tools/struts/MessageTool.html> 中了解。

好了，基于Velocity的介绍我们就说这么多，再说说其他引伸方面的内容吧。有人评论Velocity不是标准的MVC结构，没错，刚开始我们就说过Velocity只是Model和View之间的良好结合，只是个好的模版引擎，毕竟还没有形成MVC三者良好的结合。好在Apache又基于Struts和Velocity的结合，推出了VelocityStruts，这部分的陈述我们可以在后面的专题里再推出，这里简单介绍它的概念，它是在Struts的结构上，在业务逻辑处理的Action后，把业务流程转向基于Velocity的显示层，从而代替Jsp作为View层。以上我们也看到了所举的例子基本上只是基于原理和演示，没有和Web开发紧密结合起来，这方面内容我们在讲述VelocityStruts的内容时再来结合吧。

谈到Velocity，在这里要顺便提提FreeMarker，FreeMarker也是一种模版引擎，和Velocity功能基本类似，都是简单和轻量级的工具，但功能上较Velocity有不少增强的地方，这我们也在以后的文章中再来深入了解吧。

[**Velocity 语法（转）**](http://www.cnblogs.com/likwo/archive/2010/05/13/1734936.html)

|  |  |
| --- | --- |
| |  | | --- | | **一、基本语法** | |
| **1、"#"用来标识Velocity的脚本语句**，包括#set、#if 、#else、#end、#foreach、#end、#iinclude、#parse、#macro等； 如: #if($info.imgs) <img src="$info.imgs" border=0> #else <img src="noPhoto.jpg"> #end  **2、"$"用来标识一个对象(或理解为变量)；**如 如：$i、$msg、$TagUtil.options(...)等。  **3、"{}"用来明确标识Velocity变量；** 比如在页面中，页面中有一个$someonename，此时，Velocity将把someonename作为变量名，若我们程序是想在someone这 个变量的后面紧接着显示name字符，则上面的标签应该改成${someone}name。  **4、"!"用来强制把不存在的变量显示为空白。** 如当页面中包含$msg，如果msg对象有值，将显示msg的值，如果不存在msg对象同，则在页面中将显示$msg字符。这是我们不希望的，为了把不存 在的变量或变量值为null的对象显示为空白，则只需要在变量名前加一个“!”号即可。 如：$!msg |
|  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | |  | | --- | | **二、在EasyJWeb中的最佳实践** | | | 理论上你可以在EasyjWeb模板使用所有Velocity的脚本及功能，但我们不推荐你在界面模板中使用过多过复杂的脚本表达方式，在万不得已的情况下，不要在界面模板中加入任何复杂的逻辑，更不要在界面模板中加入变量声明、逻辑运算符等等。  　　在EasyJWeb中，我们提供了五条基本的模板脚本语句，基本上就能满足所有应用模板的要求。这四条模板语句很简单，可以直接由界面设计人员来添加。在当前很多EasyJWeb的应用实践中，我们看到，所有界面模板中归纳起来只有下面四种简单模板脚本语句即可实现： **1、$!obj 　直接返回对象结果。** 如：在html标签中显示java对象msg的值。<p>$!msg</p> 在html标签中显示经过HtmlUtil对象处理过后的msg对象的值　　<p>$!HtmlUtil.doSomething($!msg)</p>  **2、#if($!obj) #else #end 判断语句** 如：在EasyJWeb各种开源应用中，我们经常看到的用于弹出提示信息msg的例子。 #if($msg) <script> alert('$!msg'); </script> #end 上面的脚本表示当对象msg对象存在时，输出<script>等后面的内容。  **3、#foreach( $info in $list) $info.someList #end**　　**循环读取集合list中的对象，并作相应的处理。** 如：EasyJF开源论坛系统中论(0.3)坛首页显示热门主题的html界面模板脚本： #foreach( $info in $hotList1)  <a href="/bbsdoc.ejf?easyJWebCommand=show&&cid=$!info.cid" target="\_blank">$!info.title</a><br> #end  上面的脚本表示循环遍历hotList1集合中的对象，并输出对象的相关内容。  **4、#macro(macroName)#end 脚本函数(宏)调用，不推荐在界面模板中大量使用。** 如：在使用EasyJWeb Tools快速生成的添删改查示例中，可以点击列表的标题栏进行升降排序显示，这是我们在EasyJWeb应用中经常看到的一个排序状态显示的模板内容。 函数(宏)定义，一般放在最前面 #macro(orderPic $type) #if ($orderField.equals($type))  <img src="http://images.cnblogs.com/ico/${orderType}.gif">  #end #end 具体的调用如：<font color="#FFFFFF">头衔#orderPic("title")</font>  经过测试，宏不支持方法重载    **5、包含文件#inclue("模板文件名")或#parse("模板文件名")** 主要用于处理具有相同内容的页面，比如每个网站的顶部或尾部内容。 使用方法，可以参考EasyJF开源Blog及EasyJF开源论坛中的应用！ 如：#parse("/blog/top.html")或#include("/blog/top.html") parse与include的区别在于，若包含的文件中有Velocity脚本标签，将会进一步解析，而include将原样显示。 | |  | |
| |  |  | | --- | --- | | |  | | --- | | **三、关于#set的使用** | | | 在万不得已的时候，不要在页面视图自己声明Velocity脚本变量，也就是尽量少使用#set。有时候我们需要在页面中显示序号，而程序对象中又没有包 含这个序号属性同，可以自己定义。如在一个循环体系中，如下所示： #set ($i=0) #foreach($info in $list) 序号:$i #set($i=$i+1) #end | |  | |
| |  |  | | --- | --- | | |  | | --- | | **四、Velocity脚本语法摘要** | | | **1、声明:#set ($var=XXX)** 左边可以是以下的内容 Variable reference  String literal  Property reference  Method reference  Number literal #set ($i=1)  ArrayList #set ($arr=["yt1","t2"]) 算术运算符  **2、注释:** 单行## XXX 多行#\* xxx xxxx xxxxxxxxxxxx\*#  References 引用的类型 **3、变量 Variables**  以 "$" 开头，第一个字符必须为字母。character followed by a VTL Identifier. (a .. z or A .. Z). 变量可以包含的字符有以下内容： alphabetic (a .. z, A .. Z)  numeric (0 .. 9)  hyphen ("-")  underscore ("\_")   **4、Properties**  $Identifier.Identifier $user.name hashtable user中的的name值.类似：user.get("name")  **5、Methods** object user.getName() = $user.getName()  **6、Formal Reference Notation**  用{}把变量名跟字符串分开   如 #set ($user="csy"} ${user}name  返回csyname  $username $!username $与$!的区别 当找不到username的时候，$username返回字符串"$username"，而$!username返回空字符串""   **7、双引号 与 引号**  #set ($var="helo") test"$var" 返回testhello test'$var' 返回test'$var' 可以通过设置 stringliterals.interpolate=false改变默认处理方式  **8、条件语句** #if( $foo )  <strong>Velocity!</strong> #end #if($foo) #elseif() #else #end 当$foo为null或为Boolean对象的false值执行.  **9、逻辑运算符:== && || !**  **10、循环语句#foreach($var in $arrays )**// 集合包含下面三种Vector, a Hashtable or an Array #end #foreach( $product in $allProducts ) <li>$product</li> #end  #foreach( $key in $allProducts.keySet() ) <li>Key: $key -> Value: $allProducts.get($key)</li> #end  #foreach( $customer in $customerList ) <tr><td>$velocityCount</td><td>$customer.Name</td></tr> #end  **11、velocityCount变量在配置文件中定义** # Default name of the loop counter # variable reference. directive.foreach.counter.name = velocityCount # Default starting value of the loop # counter variable reference. directive.foreach.counter.initial.value = 1  **12、包含文件**  #include( "one.gif","two.txt","three.htm" )  **13、Parse导入脚本** #parse("me.vm" )  **14、#stop 停止执行并返回**   **15、定义宏Velocimacros ,相当于函数 支持包含功能** #macro( d ) <tr><td></td></tr> #end 调用  #d()  **16、带参数的宏** #macro( tablerows $color $somelist ) #foreach( $something in $somelist ) <tr><td bgcolor=$color>$something</td></tr> #end #end  **17、Range Operator**  #foreach( $foo in [1..5] ) | |  | |
| |  |  | | --- | --- | | |  | | --- | | **附：《浅析MVC框架中View层的优雅设计及实例》** | | | 作者：EasyJF开源团队 大峡  在基于B/S的应用程序开发中，从基本的技术分工上来说就是两大块，一是软件显示界面，另一个是程序逻辑。在N年前的脚本语言时代，无论是asp、 php还是jsp，我们基本是都是把这两者柔和在一起的。尽管我们想方设法做好很多函数或者包含文件来努力达到软件的复用，但仍然无法满足多变的用户需 求,这主要是因为以前的纯脚本编码方式无法很好支持及应用面向对象(OO)领域中的强大功能。  在常见的B/S软件项目中，界面的设计包括html界面、Wap界面及其它由文本字符协议为基本表示的界面等。以我们接触最多的html页面为例子，在 做这些界面的时候往往需要美工先使用photoshop或fireworks等图形界面设计工具进行全局设计，然后再使用进行Dreamweaver等 html页面制作工具进行加工制作。而强大的程序逻辑及后台处理都是由服务器端程序完成，这些程序具有较高的稳定性，其开发工具如JBuilder、 Eclipse等对View层的界面无法很好的支持(当然那些用记事本写界面的应用除外)，这就使得很多MVC框架的设计都无法两全齐美。  作位一个比较友好的MVC的框架，在简化服务器应用开发的同时，还需要在View这一层设计上不要过多的影响到界面人员的工作，最基本的要求就是不要过 多的加入一些设计软件不支持的标签等元素(如Struts的很多标签在Dreamweaver中都不支持)。这里我们以EasyJWeb为例，探讨在 View层如何实现比较合理的设计。  　　EasyJWeb作为一个快速Java Web MVC框架，其设计目标不尽是要简化软件开发人员的代码书写工作，更是要能方便界面设计人员的工作。  　　当然，要使界面能跟后台程序逻辑能融合，显示界面及程序逻辑之间需要一定的对话协议在所难免。EasyJWeb作为一个MVC框架，同样也存在着这样的对话协议，这就是EasyJWeb中的界面模板标识语言。  　　在当前发布的版本中，EasyJWeb界面模板标识语言使用的是Apache开源的Velocity模板引擎(template engine)，当然以后会根据际需要加入更多的模板引擎以供框架应用者选择。  Velocity是一个基于java的模板引擎（template engine），它允许任何人仅仅简单的使用模板语言（template language）来引用由java代码定义的对象。作为一个比较完善的模板引擎，Velocity的功能是比较强大的，但强大的同时也增加了应用复杂 性。  　　理论上你可以在EasyjWeb模板使用所有Velocity的脚本及功能，但我们不推荐你在界面模板中使用过多过复杂的脚本表达方式，在万不得已的情况下，不要在界面模板中加入任何复杂的逻辑，更不要在界面模板中加入变量声明、逻辑运算符等等。  　　在EasyJWeb中，我们提供了四条基本的模板脚本语句，基本上就能满足所有应用模板的要求。这四条模板语句很简单，可以直接由界面设计人员来添加。在当前很多EasyJWeb的应用实践中，我们看到，所有界面模板中归纳起来只有下面四种简单模板脚本语句即可实现：  　　1、$!obj 　直接返回对象结果。  如：在html标签中显示java对象msg的值。<p>$!msg</p> 在html标签中显示经过HtmlUtil对象处理过后的msg对象的值.<p>$!HtmlUtil.doSomething($!msg)</p>  　　2、#if($!obj) #else #end 判断语句  　　如：在EasyJWeb各种开源应用中，我们经常看到的用于弹出提示信息msg的例子。 #if($msg) <script> alert('$!msg'); </script> #end 上面的脚本表示当对象msg对象存在时，输出<script>等后面的内容。  　　3、#foreach( $info in $list) $info.something #end　　循环读取集合list中的对象，并作相应的处理。  如：EasyJF开源论坛系统中论坛首页显示热门主题的html界面模板脚本  　　#foreach( $info in $hotList1)  <a href="/bbsdoc.ejf?easyJWebCommand=show&&cid=$!info.cid" target="\_blank">$!info.title</a><br> #end  上面的脚本表示循环遍历hotList1集合中的对象，并输出对象的相关内容。  4、#macro(macroName)#end 脚本函数(宏)调用，不推荐在界面模板中大量使用。  　　如：在EasyJF简易订销管等系统中经常看到的一个排序状态显示的模板内容。 函数(宏)定义，一般放在最前面 #macro(orderPic $type) #if ($orderField.equals($type))  <img src="http://images.cnblogs.com/ico/${orderType}.gif">  #end #end  具体的调用如：<font color="#FFFFFF">头衔#orderPic("title")</font>    总结：　 当然，在实际应用项目中，为了实现界面的更加友好、人性化，会出现很多复、易变的需求。如根据对象的不同状态，显示不同的提示颜色、提示语音等功能。在这 种时候，仍然要慎用太多的模板脚本功能，大多数需求都可以通过变通的方式解决，有些信息可以直接在对象中增加逻辑转化信息，有些界面要求可以通过在界面中 使用与特定界面有关的表达方式来实现，如html页面中使用javascript、css，Wap页面中使用WMLScript等等。只有，这样才能确保 证你的系统核心不受界面的的影响及控制，才能更好的扩展及维护。  　　我们希望你能成为真正的程序高手，而不只是精通某个处于表层的脚本语言，因此也不希望广大Java开发人员在View层消耗太多的时间。  　　由于水平有限，本文所谈的内容有N多不合理或者需要改进的地方，恳请大家指正！  附：EasyJWeb简介  EasyJWeb是基于java技术，应用于WEB应用程序快速开发的MVC框架，框架设计构思来源于国内众多项目实践，充分借签了当前主流的开源 Web框架（Struts、JSF、Tapestry 、Webwork等），吸取了其优点及精华，利用Velocity作为模板页面引擎，是一个实现了页面及代码完全分离的MVC开发框架，是一个旨在于为中 小型Web应用系统提供快速开发实践的简易Web框架。  　　EasyJF开源团队于2006年初才开始建设，因此当前整个开发团队组建以及所发布的作品，都显得极不成熟。EasyJWeb仍然处于测试阶段，请广大的Java爱好者多多批评及建议,同时也非常欢迎您能加入到我们的国产开源队伍中。  　　EasyJWeb官方网址：www.easyjf.com EasyJF团队官方网址：www.easyjf.com    http://www.cnblogs.com/siye1982/archive/2007/11/14/959678.html | |

# [velocity介绍及语法](http://blog.csdn.net/naruto_ahu/article/details/10490283)

标签： [velocity](http://www.csdn.net/tag/velocity)[模板引擎](http://www.csdn.net/tag/%e6%a8%a1%e6%9d%bf%e5%bc%95%e6%93%8e)[web开发](http://www.csdn.net/tag/web%e5%bc%80%e5%8f%91)

2013-08-29 09:30 6989人阅读 [评论](http://blog.csdn.net/naruto_ahu/article/details/10490283#comments)(0) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/naruto_ahu/article/details/10490283#report)

http://static.blog.csdn.net/images/category_icon.jpg 分类：

web（9） http://static.blog.csdn.net/images/arrow_triangle%20_down.jpg

版权声明：本文为博主原创文章，未经博主允许不得转载。

目录[(?)[+]](http://blog.csdn.net/naruto_ahu/article/details/10490283)

    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。

    Velocity的能力远不止web站点开发这个领域，例如，它可以从模板（template）产生SQL和PostScript、XML，它也可以被当 作一个独立工具来产生源代码和报告，或者作为其他系统的集成组件使用。Velocity也可以为Turbine web开发架构提供模板服务（template service）。Velocity+Turbine提供一个模板服务的方式允许一个web应用以一个真正的MVC模型进行开发。

    Velocity的**Eclipse插件**的Update地址是：http://propsorter.sourceforge.net/veloeclipse

    Velocity **的 JavaDoc:**<http://www.ostools.net/apidocs/apidoc?api=velocity-1.7>

## 一、基本语法

1、"#"用来标识Velocity的脚本语句，包括#set、#if 、#else、#end、#foreach、#end、#iinclude、#parse、#macro等；  
如:  
#if($info.imgs)  
<img src="$info.imgs" border=0>  
#else  
<img src="noPhoto.jpg">  
#end  
  
2、"$"用来标识一个对象(或理解为变量)；如  
如：$i、$msg、$TagUtil.options(...)等。  
  
3、"{}"用来明确标识Velocity变量；  
比如在页面中，页面中有一个$someonename，此时，Velocity将把someonename作为变量名，若我们程序是想在someone这个变量的后面紧接着显示name字符，则上面的标签应该改成${someone}name。  
  
4、"!"用来强制把不存在的变量显示为空白。  
如当页面中包含$msg，如果msg对象有值，将显示msg的值，如果不存在msg对象同，则在页面中将显示$msg字符。这是我们不希望的，为了把不存在的变量或变量值为null的对象显示为空白，则只需要在变量名前加一个“!”号即可。  
如：$!msg

## 二，关于#set的使用

在万不得已的时候，不要在页面视图自己声明Velocity脚本变量，也就是尽量少使用#set。有时候我们需要在页面中显示序号，而程序对象中又没有包含这个序号属性同，可以自己定义。如在一个循环体系中，如下所示：  
#set ($i=0)  
#foreach($info in $list)  
序号:$i  
#set($i=$i+1)  
#end

## 三，Velocity脚本语法摘要

### 1、声明:#set ($var=XXX)

左边可以是以下的内容  
Variable reference  
String literal  
Property reference  
Method reference  
Number literal #set ($i=1)  
ArrayList #set ($arr=["yt1","t2"])  
算术运算符

### 2、注释:

单行## XXX  
多行#\* xxx  
xxxx  
xxxxxxxxxxxx\*#  
  
References 引用的类型

### 3、变量 Variables

以 "$" 开头，第一个字符必须为字母。character followed by a VTL Identifier. (a .. z or A .. Z).  
变量可以包含的字符有以下内容：  
alphabetic (a .. z, A .. Z)  
numeric (0 .. 9)  
hyphen ("-")  
underscore ("\_")

### 4、Properties

$Identifier.Identifier  
$user.name  
hashtable user中的的name值.类似：user.get("name")

### 5、Methods

object user.getName() = $user.getName()

### 6、Formal Reference Notation

用{}把变量名跟字符串分开  
  
如  
#set ($user="csy"}  
${user}name  
返回csyname  
  
$username  
$!username  
$与$!的区别  
当找不到username的时候，$username返回字符串"$username"，而$!username返回空字符串""

### 7、双引号 与 引号

#set ($var="helo")  
test"$var" 返回testhello  
test'$var' 返回test'$var'  
可以通过设置 stringliterals.interpolate=false改变默认处理方式

### 8、条件语句

#if( $foo )  
<strong>Velocity!</strong>  
#end  
#if($foo)  
#elseif()  
#else  
#end  
当$foo为null或为Boolean对象的false值执行.

### 9、逻辑运算符

== && || !

### 10、循环语句

#foreach($var in $arrays ) // 集合包含下面三种Vector, a Hashtable or an Array  
#end  
#foreach( $product in $allProducts )  
<li>$product</li>  
#end  
  
#foreach( $key in $allProducts.keySet() )  
<li>Key: $key -> Value: $allProducts.get($key)</li>  
#end  
  
#foreach( $customer in $customerList )  
<tr><td>$velocityCount</td><td>$customer.Name</td></tr>  
#end

### 11、velocityCount变量在配置文件中定义

# Default name of the loop counter  
# variable reference.  
directive.foreach.counter.name = velocityCount  
# Default starting value of the loop  
# counter variable reference.  
directive.foreach.counter.initial.value = 1

### 12、包含文件

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

### 13、Parse导入脚本

#parse("me.vm" )

### 14、#stop 停止执行并返回

15、定义宏Velocimacros ,相当于函数 支持包含功能  
#macro( d )  
<tr><td></td></tr>  
#end  
调用  
#d()

### 16、带参数的宏

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

### 17、Range Operator

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