在 HTML 或者 XML 这样的标准表示或交换格式中，文本性数据的操作和转换是一种频繁而且通常非常单调的活动，每个开发人员都会遇到。模板引擎可以改善这个过程，它在模板中保留输出中的静态部分，而动态生成和安排变化的部分。Velocity 是一种高度实用的、开放源代码的模板引擎，可以方便地集成到其他客户端或服务器端应用程序中。

对于服务器端应用程序，如果与兼容 Servlet 2.3+ 的 Web 层容器集成，Velocity 为 JSP 技术提供了一种可行的替代方案，可以强制实施表示逻辑与应用程序业务逻辑的清晰划分。事实上，Velocity 支持的模板语言非常简单，形成的模板也十分清晰，Web 站点设计人员和样式开发人员可以学习和维护这些模板。

本文中将考察 Velocity 的简单模板语言、创建一些模板并将其用于独立的客户应用程序。然后我们将把这个模板引擎集成到 Struts MVC 框架中作为视图组件。

**基本模板引擎操作**

基本模板引擎操作非常简单。首先看一看清单 1 中的模板：

**清单 1. 基本的 Velocity 模板**

<html>

<head>

<title>A Template Based Page</title>

</head>

<body>

<p>This is a page generated by $generatedBy.</p>

<p>The customer's name is $customerName.</p>

</body>

</html>

这个模板是一个完整的 HTML 文件。您可以使用文本编辑器或者喜欢的图形化可视网页编辑器创建该文件。创建的简易性是基于模板的系统的主要好处和要求。

当模板引擎运行时，清单 1 中彩色显示的部分将被实际的数据替换。获取数据并与模板结合的过程称为 *合并*。看一看清单 2 中的脚本所表示的数据：

**清单 2. 为模板合并设置数据值**

#set ($generatedBy = "Velocity")

#set ($customerName = "John Doe")

现在，如果清单 1 中的模板与清单 2 中的数据合并，将得到清单 3 所示的结果：

**清单 3. 合并到模板中的数据**

<html>

<head>

<title>A Template Based Page</title>

</head>

<body>

<p>This is a page generated by Velocity.</p>

<p>The customer's name is John Doe.</p>

</body>

</html>

您可能发现，这种特性和字处理程序中的邮件合并功能类似。在字处理程序中，信函结构与来自邮件列表的名称和地址合并。和邮件合并一样，这种应用程序最适用于要合并的数据源非常大而且有变化的情况。

从这个单纯的意义上讲，Velocity 是一个模板引擎。Velocity 的输出格式仅受文本模板中所能放置的内容的限制。包括现在最流行的格式（HTML、XML、SQL，等等）。

**使用 Velocity 模板语言创建模板**

Velocity 模板是文本文件（HTML、XML 等等），其中包括：

* 照原样合并的静态部分
* 将被要合并的数据替代的占位符
* 脚本语言中的指示符和指令

Velocity 模板使用的脚本语言称为 Velocity 模板语言（VTL）。和其他脚本语言相比，VTL 语法相对而言不是很丰富。任何具有编程背景的人都可以非常快地学会 VTL。

**占位符与引用**

VTL 中的引用是一个命名元素，如 $customerName。引用可以在 Velocity 模板中作为占位符。在模板合并过程中，这些占位符将被替换成相应的文本值，从而形成最终的输出。比如，在 [清单 1](https://www.ibm.com/developerworks/cn/java/j-velocity/#list1)中，我们可以看到使用了两个 VTL 引用（ $generatedBy和 $customerName）已生成最终输出结果。

变量在 VTL 中是一种引用类型。您可以使用 #set()指示符为变量赋值。清单 4 给出了一些例子：

**清单 4. 变量类型的 VTL 引用**

#set( $this = "Velocity")

#set( $numericBase = 999 )

#set( $booleanCondition = true )

This page is generated using $this.

There are ($numericBase + 1) pages in total.

变量名必须从一个字母开始，因此 Velocity 很容易把变量名与模板中的货币符号分开（比如， $100不可能是一个变量名）。合并操作中所有的变量都被转化成字符串，可能造成一些有趣的现象。看一看清单 4 中用红色显示的文本。合并后的输出如清单 5 所示：

**清单 5. 合并后的模板中带有数字值的变量**

This page is generated using Velocity.

There are (999 + 1) pages in total.

因为 $numericBase在合并操作中被转化成了字符串，因此不会执行算术操作。因为 VTL 专门用于模板操作而非通用的计算语言，所以只需要支持整数算术运算（尽管可以使用插件工具进行扩展）。下面的脚本说明了如何利用这种数学运算能力：

#set( $newSum = $numericBase + 1)

There are $newSum pages in total.

该模板合并后相应的输出为：

There are 1000 pages in total.

到目前我们处理的还只有标量。要创建包含多个元素的 ArrayList变量，可以使用如下的语法：

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

您可以使用 $treeList.get(1)列表中的第二个元素。

赋值以后， $treeList就是一个基于 ArrayList的变量（就像是标准 JDK 集合类中那样）。您可以直接使用符号 $treeList.get(n)访问它的每个元素，其中的 n是以 0 为基的 ArrayList索引。比如，像 [清单 6](https://www.ibm.com/developerworks/cn/java/j-velocity/#list6)种红色显示的一行所表明的那样， $treeList.get(1)用于选择ArrayList中第二项，即 oak。这种调用 ArrayList类方法的语法也可用于调用其他变量对象的方法（更多信息请参阅侧栏中的 [属性和方法参考](https://www.ibm.com/developerworks/cn/java/j-velocity/#sidebar1)）。

**属性和方法参考**

除了在模板中设置变量之外，VTL 引用也可以是对象属性或方法。这些对象是模板可以使用的 Java 类（一般通过上下文，参阅 [Velocity 上下文](https://www.ibm.com/developerworks/cn/java/j-velocity/#2)）。   
  
对象属性通过和 Javabean 类似的符号访问。比如，可以通过 VTL 引用 $customer.LastName访问 $customer对象的 LastName属性。在幕后，Velocity 使用对象的访问器方法获得属性值（即调用对象的 getLastName()方法）。   
  
您可以用和属性访问类似的符号调用对象的方法，可以带参数列表也可以不带。比如，可以通过 VTL 引用$customer.getPhone("mobile")，调用 $customer对象的 getPhone()方法获得移动电话号码。

关于占位符替换的一点说明：Velocity 把任何不能识别的引用作为普通文本打印，如清单 6 中下面突出显示的两行（蓝色和红色）所示：

**清单 6. 占位符置换**

The second item in the list is $treeList.get(1).

$notDeclared is an undeclared variable.

But $!notDeclared is invisible when not declared.

VTL 支持一种静态引用符号，以避免呈现不存在的或者 空的引用。如果使用安静引用符号，比如 $!notDeclared，那么 Velocity 将什么也不输出，而不是输出完整的引用名。注意变量名前面的“!”表示这是静态引用符号。当合并清单 6 中的模板时，两个引用都没有分配任何值，但是蓝色显示的引用将原样显示，而绿色的一个则看不到：

The second item in the list is oak.

$notDeclared is an undeclared variable.

But is invisible when not declared.

**选择性呈现和循环**

可以使用指示符 #if... #then... #else....有条件地呈现模板中特定的部分。清单 7 给出了一个例子：

**清单 7. 使用 #if、#then 和 #else 有选择地呈现**

#if $customer.GoldMember

Thank you Mr. $customer.LastName, for flying with us.

Your loyal patronage is greatly appreciated.

This flight earns you an additional 5000 miles.

#else

Thank you for flying with us.

Please consider joining our frequent flyer program.

#endif

在清单 7 的模板中，使用 $customer对象的 boolean 属性 GoldMember确定在最终输出中出现哪些信息。对于金牌顾客，最终输出中将呈现蓝色显示的消息；对于其他顾客，则在最终输出中呈现绿色显示的消息。

模板中经常要使用循环格式化表格或者列表形式的信息。显示的数据通常保存在一个 ArrayList引用中。在 Velocity 中唯一用于处理重复循环的指示符是 #foreach指示符。清单 8 中的模板通过 $treeList ArrayList变量演示了 #foreach指示符的用法。当然，也可以使用任何其他可用的集合类型的对象引用，或者返回一个集合的对象属性 / 方法引用。

**清单 8. 使用 #foreach 循环**

<table>

<tr><td>Tree Name</td></tr>

#foreach $name in $treeList

<tr><td>

$name is a big tree!

</td></tr>

#end

</table>

$treeList中包含树名的列表，清单 8 中的模板合并后的输出如清单 9 所示：

**清单 9. #foreach 循环中合并后的输出**

<table>

<tr><td>Tree Name</td></tr>

<tr><td>

pine is a big tree!

</td></tr>

<tr><td>

oak is a big tree!

</td></tr>

<tr><td>

maple is a big tree!

</td></tr>

<tr><td>

redwood is a big tree!

</td></tr>

</table>

如果从 HTML 浏览器中查看，清单 9 当然就是一个包含树名的表。

注意在 #foreach循环体内有一个内置的计数器，可以在 #foreach指示符循环体内通过 $velocityCounter引用访问它。默认情况下，这个计数器从 1 开始，每执行一次循环递增一次。

**Velocity 中的宏**

Velocity 的一个主要特性是能够很容易地定义宏，称为 *Velocimacros*。宏使您能够很容易地封装和重用模板脚本。默认情况下，宏保存在VM\_global\_library.vm文件中。比如，考虑清单 10 中名为 #showTree()的 Velocimacro：

**清单 10. 定义 Velocimacro**

#macro (showTree)

#if ($treeList )

#foreach ($e in $treeList )

$e

#end

#end

#end

您可以调用 #showTree()Velocimacro 并使用它打印 $treeListArrayList ―― 如果这个列表已经定义的话。调用的语法很简单，即#showTree()。

参数化宏也是可能的。比如，我们可以修改 #showTree()宏使其用于任何列表，如清单 11 所示：

**清单 11. 带参数的 Velocimacro**

#macro (showList $val)

#if ($val )

#foreach ($e in $val )

$e

#end

#end

#end

要使用清单 11 中的宏调用 $treeList，我们可以使用 #showList($treeList)。这两种情况的输出都一样，如清单 12 所示：

**清单 12. Velocimacro 的合并输出**

pine

oak

maple

redwood

**其他有趣的 VTL 细节**

单行注释或者行尾注释从 ##开始，多行注释则放在 #\*和 \*#之间。

在处理字符串数据时，可以使用双引号或单引号分隔。但是使用双引号允许在分隔的字符串 *内部*对 Velocity 引用、指示符甚至 Velocimacros 求值。

[**回页首**](https://www.ibm.com/developerworks/cn/java/j-velocity/#ibm-pcon)

**Velocity 上下文**

您可以把 Velocity 中的上下文看作是导入 Java 对象，以便在 Velocity 模板内部访问的一种方法。这种导入必须在 Java 编码中明确地完成。和 JSP 代码或者 JavaScript 不同，不存在“自然的”或“原生方式”使 Velocity 访问任何 Java 对象。只有明确导入的 Java 对象才能在 Velocity 模板中使用。

通过创建 org.apache.velocity.context.Context类的实例可以获得 Velocity 上下文。然后可以使用上下文的 put( *key*, *value*) 方法，把将要导入供模板使用的对象附加到上下文中。key 是一个字符串名，将在模板中作为可用的引用出现。在产品环境中，图形或者 Web 设计人员可能负责创建和维护模板，而 Java 开发人员提供可以在模板中访问的对象集。在这种情况下，设计人员和开发人员应就对象集合及其可用的属性达成一致并互相协作。在 Velocity 上下文中附加的属性将作为主要的接口机制。

**在模板中访问上下文属性**

看一看一个独立解析器中包含的示例代码（请参阅 [参考资料](https://www.ibm.com/developerworks/cn/java/j-velocity/#resources)）。可以在 \code\src目录下找到。比如，在com.ibm.dvworks.velocity.VelocityParser类中，我们已经创建并向 Velocity 上下文中添加了两个属性，如清单 13 所示：

**清单 13. 在 VelocityParser 类中创建一个 Velocity 实例**

public static void main(String[] args) {

VelocityParser velInstance = new VelocityParser(args[0]);

velInstance.addToContext( "treeFarm",

new String [] { "redwood", "maple", "oak", "pine" });

velInstance.addToContext( "title", "A Tree Farm");

velInstance.addToContext( "date", new java.util.Date());

velInstance.addToContext("fmtr",

new org.apache.velocity.app.tools.VelocityFormatter(

velInstance.getCurrentContext()));

velInstance.processTemplate();

}

属性 treeFarm是一个关于树名的 ArrayList。 title属性是一个标量字符串。一旦附加到上下文中并在合并过程中传递，这些属性在 Velocity 模板中立刻就变得没有用了。清单 14 中的模板使用了这两个属性。您可以在 \code\app\treectx.vm中找到这个例子。

**清单 14. 使用 $treeFarm 上下文属性引用**

<table>

<tr><td>$title</td></tr>

#foreach $name in $treeFarm

<tr><td>

$name is a big tree!

</td></tr>

#end

</table>

合并后的输出如清单 15 所示：

**清单 15. 模板的合并输出**

<table>

<tr><td>A Tree Farm</td></tr>

<tr><td>

redwood is a big tree!

</td></tr>

<tr><td>

maple is a big tree!

</td></tr>

<tr><td>

oak is a big tree!

</td></tr>

<tr><td>

pine is a big tree!

</td></tr>

</table>

要注意，使用 $treeFarm上下文属性引用的方法和前面分析的 $treeList变量引用一致。

**初始化模板引擎**

分析 VelocityParser类中列出的 main()方法（参见清单 13）。 VelocityParser构造函数创建解析器并加载模板，然后增加模板引擎要使用的属性，最后调用 processTemplate()合并数据和模板。我们将按照顺序依次分析这些方法。

您可以在 org.apache.velocity.app.Velocity类中使用静态方法初始化 Velocity 并加载一个模板文件。要使用的方法分别为 init()和getTemplate()。对 init()方法的调用出现在 VelocityParser类的构造函数中，如清单 16 所示：

**清单 16. 在 VelocityParser 类的构造函数中初始化模板引擎**

public VelocityParser(String templateFile) {

try {

Velocity.init("velocity.properties");

mainTemplate = Velocity.getTemplate(templateFile);

}

catch( Exception ex ) {

System.out.println("Error processing template file: " + templateFile );

}

}

在清单 16 中，对 init()的调用创建了一个 Velocity 引擎实例。如果应用程序需要创建和管理多个 Velocity 模板引擎实例，则应使用org.apache.velocity.app.VelocityEngine类。

**上下文链**

只要调用 org.apache.velocity.VelocityContext类的普通构造函数就可以创建直接可用的 Velocity 上下文。

Velocity 上下文可以 *进行链接*（包装在另一个上下文内部）。如果需要在模板中控制特定对象引用的可见性和可用性，这样做非常有用。

Velocity 将在对象引用的所有链接上下文中搜索属性。如果遇到重复的名称，则使用最外层的属性，而内部的同名属性永远不会被访问。

为了链接 Velocity 上下文，要链接的上下文应该作为参数传递给一个新上下文的构造函数。清单 17 中 VelocityParser类的重载方法addToContext()说明了这一点：

**清单 17. 使用 addToContext() 方法增加上下文属性或者上下文链接**

public void addToContext(String key, Object value) {

if (mainContext == null)

mainContext = new VelocityContext();

mainContext.put(key, value);

}

public void addToContext(VelocityContext chainCtx) {

mainContext = new VelocityContext(chainCtx);

}

在 processTemplate()方法中，调用模板的 merge()方法结合上下文信息和模板生成输出流，如清单 18 所示：

**清单 18. 在 processTemplate() 方法中合并模板**

public void processTemplate() {

try {

BufferedWriter writer = writer = new BufferedWriter(

new OutputStreamWriter(System.out));

if ( mainTemplate != null)

mainTemplate.merge(mainContext, writer);

writer.flush();

writer.close();

}

catch( Exception ex ) {

ex.printStackTrace();

}

}

[**回页首**](https://www.ibm.com/developerworks/cn/java/j-velocity/#ibm-pcon)

**Velocity 作为独立的解析器**

要编译上述示例独立解析器，请使用安装目录下 \code\app中的 compile.bat文件。要试验该解析器，请使用 process.bat批处理文件，其中包括：

set VEL\_HOME=\jdk1.4\vel14rc1

java -classpath ..\classes;%VEL\_HOME%\velocity-dep-1.4-rc1.jar

com.ibm.dvworks.velocity.VelocityParser %1 %2 %3

注意，必须同时在 compile.bat和 process.bat中把环境变量 VEL\_HOME设置成安装 Velocity 的目录。在 Velocity 发行包中包含两类不同的 JAR 文件： velocity-dep---?.jar（其中的 *--?*是版本号信息）和 velocity---?.jar。 velocity-dep---?.jar文件包括所有的外部依赖（Jakarta common-collections、Avalon Logkit 和 ORO 正则表达式库），可以直接使用。如果您的 classpath 中已经有一些这样的库，您可能希望使用 velocity---?.jar文件来代替。如果这些 JAR 组成都不能满足您的需要，可以很容易地按照需要的方式重新建立 Velocity。Velocity 发行包中包括一个 ant 脚本，可以为不同的应用场景建立 7 种不同的 JAR 配置。

为了便于上手，Velocity 预设了一些默认配置属性，对于多数应用而言，这都是合理的和可以接受的。这就避免了开发人员从一开始就忙于复杂的配置选项，让他们能马上体验到这种模板引擎。

[**回页首**](https://www.ibm.com/developerworks/cn/java/j-velocity/#ibm-pcon)

**服务器上的 Velocity 与 JSP 技术**

在服务器端可以使用 Velocity 处理模板和生成的动态内容（HTML、XML 等）。这和 JSP 技术的目标非常接近。但是，JSP 模型可以毫无阻碍地访问底层的 Servlet API 和 Java 编程语言。事实上，为了避免访问这些固有的特性，您在编码中必须严格约束（只是使用 EL、标签库和类似的特性）。它基本上是一种在很大程度上开放的访问模型。

拿 Velocity 与之比较。作为一种完全自包含的模板引擎和脚本解释器，Velocity 拥有完全封闭的模型。任何针对系统和 / 或 Java 编程语言的访问都必须明确地启用。默认情况，Velocity 模板中不能访问 Java 编程语言的任何方面。这种封闭的模型使 Velocity 能够提供分离的模板表示层，与任何应用程序业务逻辑或者数据管理代码清晰地划分开。

现在让我们把这种模板引擎与 Tomcat 5 的最新版本集成在一起，看一看 Velocity 在服务器端的应用。

[**回页首**](https://www.ibm.com/developerworks/cn/java/j-velocity/#ibm-pcon)

**与 Tomcat 5 一起部署 Velocity**

Velocity 发行包带有一个 org.apache.velocity.servletVelocityServlet库类，扩展它可以很快地创建一个模板处理 servlet。作为独立的客户机应用程序测试的任何模板都可以使用 VelocityServlet部署在服务器上。把独立的 Velocity 模板转移到 Web 应用程序中相对比较简单。只需要以下几个步骤：

**Velocity 中的工具**

工具是在模板中可以通过 Velocity 上下文使用的实用 Java 对象。虽然可以手工把这些对象附加到上下文中，但 VelocityViewServlet工具箱管理器可通过更加灵活和结构化的方式完成。在模板中一般是通过调用工具的方法来使用它们。 VelocityViewServlet与VelocityStruts都提供了非常有价值的经过测试的工具，在作为一种视图技术部署 Velocity 时极其有用。

1. 从 org.apache.velocity.servlet.VelocityServlet类派生一个 Servlet 类。
2. 重写并实现其中的一个 handleRequest()方法。
3. 在 handleRequest()的实现中，添加希望在模板中作为上下文属性使用的数据或工具（请参阅侧栏 [Velocity 中的工具](https://www.ibm.com/developerworks/cn/java/j-velocity/#sidebar2)）。
4. 在 handleRequest()的实现中，从文件或资源（如 JAR 文件）中取得模板并返回它。

在示例代码包中， com.ibm.dvworks.velocity.VelTestServlet就是按照上述步骤创建的一个 servlet。您可以查看 webapps\vservlet\WEB-INF\src目录下的代码。如果改变了这些代码，一定要使用 compile.bat 批处理文件重新编译它。

部署描述符（web.xml 文件）定义了该 servlet 并把它映射到 /vServletURL 模式中，如清单 19 所示：

**清单 19. 自定义基于 Velocity 的 servlet 的 Tomcat 部署描述符**

<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE web-app

PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"

"http://java.sun.com/dtd/web-app\_2\_3.dtd">

<web-app>

<servlet>

<servlet-name>vServlet</servlet-name>

<servlet-class>com.ibm.dvworks.velocity.VelTestServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>vServlet</servlet-name>

<url-pattern>/vServlet</url-pattern>

</servlet-mapping>

</web-app>

加载并处理的模板放在 webapps\vServlet目录中。在这个例子中，模板文件称为 variables.vm。测试之前一定要保证 velocity-dep---?.jar文件已经放在 webapps\vServlet\WEB-INF\lib目录下，然后启动 Tomcat 5 并访问http://localhost:8080/vservlet/ServletURL。

**部署 VelocityViewServlet**

要把模板功能扩展到 Web 应用程序中，应该使用 Velocity 工具集中的 VelocityViewServlet。Velocity 工具是 Velocity 的一个子项目（请参阅[参考资料](https://www.ibm.com/developerworks/cn/java/j-velocity/#resources)找到这个 URL 并下载最新的版本）。该 Servlet 为 Velocity 用作一种视图层技术提供了更复杂的支持，既可以与 JSP 技术联合使用也可以代替后者。使用 VelocityViewServlet可以减少许多冗余代码，因为它提供了：

* 对请求对象和属性、会话对象和属性以及 servlet 上下文和属性的直接模板访问
* 正式的、可外部配置的“工具箱”，可以增加在模板中使用的自定义工具（这里讲的工具只是具有公共方法的已编译的类）
* 一个通用的、经过测试的、随时可用的工具库

要把 VelocityViewServlet集成到 Web 应用程序中，可以看一看示例 velview Web 应用程序（在 webapps\velview目录中）。该应用程序包括本文中所讨论的那些模板。此外，它还显示了请求、会话以及 servlet 上下文对象的属性。集成的步骤如下：

首先要保证 velocity-tools-view.jar文件在应用程序的 lib目录中。当然，这个 velocity JAR 文件也应该在那儿。

在部署描述符 web.xml 文件中，包括 VelocityViewServlet。初始化参数是一个工具箱描述 XML 文件。该 servlet 映射为处理所有扩展名为 .vm 的文件，如清单 20 所示：

**清单 20. VelocityViewServlet 的 Tomcat 部署描述符（web.xml）**

<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE web-app

PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"

"http://java.sun.com/dtd/web-app\_2\_3.dtd">

<web-app>

<servlet>

<servlet-name>velocityView</servlet-name>

<servlet-class>

org.apache.velocity.tools.view.servlet.VelocityViewServlet

</servlet-class>

<init-param>

<param-name>org.apache.velocity.toolbox</param-name>

<param-value>/WEB-INF/toolbox.xml</param-value>

</init-param>

</servlet>

<servlet-mapping>

<servlet-name>velocityView</servlet-name>

<url-pattern>\*.vm</url-pattern>

</servlet-mapping>

</web-app>

该例子的工具箱描述符（toolbox.xml）文件中，包含了两个来自 Velocity 工具库的通用工具可以在模板 DateTool和 MathTool中访问。这两个工具使我们能够格式化日期和时间信息，并在模板中执行浮点运算，如清单 21 所示：

**清单 21. 包括 DateTool 和 MathTool 的工具箱描述符**

<?xml version="1.0"?>

<toolbox>

<tool>

<key>date</key>

<scope>application</scope>

<class>org.apache.velocity.tools.generic.DateTool</class>

</tool>

<tool>

<key>math</key>

<scope>application</scope>

<class>org.apache.velocity.tools.generic.MathTool</class>

</tool>

...

在 VelocityViewServlet中有一组常用的标准工具，如表 1 所示：

**表 1. VelocityViewServlet 中的标准工具**

| **工具名** | **描述** |
| --- | --- |
| LinkTool | 处理 URI。该工具经常会用到，如果在模板中创建可点击的链接就要用到该工具，可以生成依赖于上下文的 URI 部分。 |
| CookieTool | 使模板能够创建或访问浏览器缓冲的 cookie。 |
| ParameterParser | 简化后面收到的请求参数的解析。 |

还有两个高度专门化的、不那么常用的工具，如表 2 所示：

**表 2. 专门的 VelocityViewServlet 工具**

| **工具名** | **描述** |
| --- | --- |
| ViewRenderTool | 使模板能够解析包含 VTL 的字符串。 |
| AbstractSearchTool | 提供了一种骨架工具（必须使用自定义的 Java 代码来扩展），以便实现在线搜索和搜索结果分页。 |

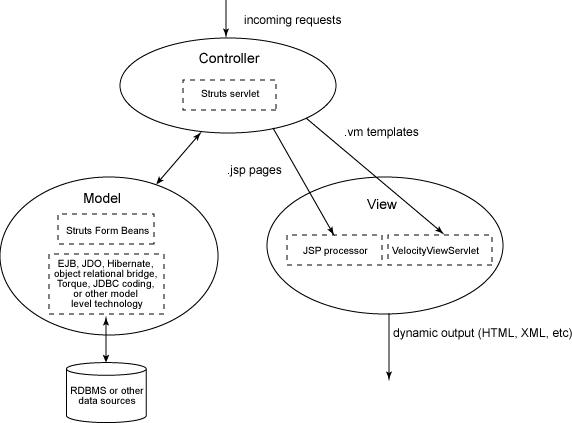
您可以使用 http://localhost:8080/velview/variables.vmURL 测试 velview 应用程序。您应该打开模板源代码看一看所用的 Velocity 引擎、 LinkTool和 CookieTool。

[**回页首**](https://www.ibm.com/developerworks/cn/java/j-velocity/#ibm-pcon)

**与 Struts 框架的互操作**

Struts 是一种构造基于 MVC 模型的框架的流行 Web 应用程序。Struts 默认的视图组件技术是 JSP 技术。但是，可以很容易把 Velocity 集成进来作为视图组件。图 1 说明了 Velocity 的这种具体应用：

**图 1. Velocity 与 Struts MVC 框架集成**



重要的是要看到，在这种结合中 Velocity 并没有代替 JSP 技术。相反，JSP 技术和 Velocity 模板可以协同工作。集成 Velocity 需要配置VelocityViewServlet以便处理 .vm 模板，就像 [部署 VelocityViewServlet](https://www.ibm.com/developerworks/cn/java/j-velocity/#velviewserv) 部分所讲的那样。这意味着 .jsp 文件将继续由容器（即 Tomcat 5 中的 Jasper）处理，而任何 .vm 模板则传递给 Velocity。

Velocity Tools 子项目中的 VelocityStruts组件（请参阅 [参考资料](https://www.ibm.com/developerworks/cn/java/j-velocity/#resources)）包含集成 Velocity 与 Struts 的所有功能。 VelocityStruts提供了一组专用的 Velocity 工具，用于访问 Struts 专有的资源和 Velocity 模板中的信息。表 3 列出了最常用的工具：

**表 3. 用于 VelocityStruts 集成的工具**

| **工具名** | **描述** |
| --- | --- |
| StrutsLinkTool | 针对 Struts 的 LinkTool 专用版本，提供了 setAction()和 setForward()访问预先配置的活动映射。 |
| FormTool | 访问 Struts 的表单 beans。 |
| ErrorsTool | 处理 Struts 错误消息，包括对国际化的支持。 |
| MessageTool | 提供对 Struts 国际化支持的访问，尤为特别的是依赖于语言的消息资源。 |

还有一组工具专用于 Struts 1.1 中的新特性，如表 4 所示：

**表 4. 专用的 Struts 1.1 访问工具**

| **工具名** | **描述** |
| --- | --- |
| SecureLinkTool | 用于 Struts 1.1 的安全链接（SSL）扩展。 |
| ActionMessagesTool | 提供对 Struts 1.1 新对象 ActionMessages的访问。 |
| TilesTool | 提供对 Struts 1.1 Tiles 扩展支持的访问。 |
| ValidatorTool | 提供对 Struts 1.1 Validator 扩展的访问，生成代码验证表单输入字段。 |

在 webapps\struts-example目录中可以找到一个例子，使用 Struts 而非 JSP 技术创建 Struts 页面。本例中我们使用 Struts 取代了实例 Web 应用程序所发布的第一个标题页，您可以试着改变其他的页面。下面列出了操作的步骤。

1. 把 Velocity 库复制到 Struts 示例应用程序下的 WEB-INF\lib目录中。要使用 Tomcat 5（5.0.16 是撰写本文时的最新版本）和 Struts 1.1，需要把以下 JAR 文件复制到 webapps\struts-example\WEB-INF\lib目录中：
   * velocity-tools-1.1-beta1.jar
   * velocity-1.4-rc1.jar
2. 然后在 Struts 配置文件（ WEB-INF\struts-config.xml），把 Struts 动作映射设置为转向 index.vm 文件而不是 index.jsp 文件，如清单 22 所示：   
   **清单 22. 把 Struts 动作转向 index.vm**
3. <action path="/logoff"
4. type="org.apache.struts.webapp.example.LogoffAction">
5. <forward name="success" path="/index.vm"/>

</action>

1. 在部署描述符 WEB-INF\web.xml文件中配置 VelocityViewServlet 处理 .vm文件。同样把欢迎文件设为 index.vm 而非 index.jsp，如清单 23 所示：   
   **清单 23. 改变 struts 示例 Web 应用程序的部署描述符**
2. <!-- Action Servlet Configuration -->
3. <servlet>
4. <servlet-name>action</servlet-name>
5. <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
6. <init-param>
7. <param-name>config</param-name>
8. <param-value>/WEB-INF/struts-config.xml,
9. /WEB-INF/struts-config-registration.xml</param-value>
10. </init-param>
11. <load-on-startup>1</load-on-startup>
12. </servlet>
13. <servlet>
14. <servlet-name>velocity</servlet-name>
15. <servlet-class>org.apache.velocity.tools.view.servlet.VelocityViewServlet
16. </servlet-class>
17. <init-param>
18. <param-name>org.apache.velocity.toolbox</param-name>
19. <param-value>/WEB-INF/toolbox.xml</param-value>
20. </init-param>
21. <init-param>
22. <param-name>org.apache.velocity.properties</param-name>
23. <param-value>/WEB-INF/velocity.properties</param-value>
24. </init-param>
25. </servlet>
26. <!-- Action Servlet Mapping -->
27. <servlet-mapping>
28. <servlet-name>action</servlet-name>
29. <url-pattern>\*.do</url-pattern>
30. </servlet-mapping>
31. <servlet-mapping>
32. <servlet-name>velocity</servlet-name>
33. <url-pattern>\*.vm</url-pattern>
34. </servlet-mapping>
35. <!-- The Welcome File List -->
36. <welcome-file-list>
37. <welcome-file>index.vm</welcome-file>

</welcome-file-list>

1. 最后，把 toolbox.xml 和 velocity.properties 文件从本文的源代码下载中（请参阅 [参考资料](https://www.ibm.com/developerworks/cn/java/j-velocity/#resources)）移动到 WEB-INF目录下。

新的 index.vm 文件如清单 24 所示，可以把它与原来的 index.jsp 文件比较。

**清单 24. 通过使用 index.vm Velocity 模板与 Struts 互操作**

<html>

<head>

<title>$msg.get("index.title")</title>

</head>

<body bgcolor="white">

#if ( !$application.database)

<font color="red">

ERROR: User database not loaded -- check servlet container logs

for error messages.

</font>

<hr>

#end

<h3>$msg.get("index.heading")</h3>

<ul>

<li>

<a href="$link.setURI("editRegistration.do").addQueryData("action","Create")">

$msg.get("index.registration")

</a>

</li>

<li>

<a href="$link.setURI("logon.jsp")">

$msg.get("index.logon")

</a>

</li>

</ul>

<p>&nbsp;</p>

<a href="$link.setURI("tour.do")">

<font size="-1">$msg.get("index.tour")</font>

</a>

<p>&nbsp;</p>

<img src="$link.setURI("powered-by-logo.gif")" alt="Powered by Velocity"/>

</body>

</html>

在 index.vm 中，整个模板都使用 $msg内的消息工具访问 Struts 的地域有关的国际化资源。通过对包含国际化字符串的资源包的本地化更改，这种方法避免了模板中的多数硬编码字符串。

您可以使用 VTL 的条件指示符 #if直接检查在 servlet 上下文中是否存在数据库属性。 $application引用可用于访问 servlet 上下文中的任何属性（ $request、 $response和 $session也可用于访问其他 Servlet API 对象的属性）。

LinkTool的 setURI()方法用于生成服务器端到 Struts 动作和“Powered by Velocity”标志图片的 URI 链接。注意，这里使用 LinkTool的addQueryData() 方法向结果 URI 种增加附加的动作信息。

要测试该 Velocity 页面，您可以启动 Tomcat 5 并访问 http://localhost:8080/struts-example/URL。注意它的结果与原来的 JSP 版本完全一致。

[**回页首**](https://www.ibm.com/developerworks/cn/java/j-velocity/#ibm-pcon)

**结束语**

Velocity 模板处理程序可以直接集成到 Java 语言应用程序中，立即提供报告生成或者模板处理的功能。

将模板引擎扩展到 Web 应用程序，可以使用 VelocityServlet处理动态生成 HTML 输出的 Velocity 模板。Velocity 工具项目对使用VelocityViewServlet组件化 Web 层应用程序开发提供了更多的支持。 VelocityViewServlet以模板为基础为基于 Web 的 UI 构造提供了方便的视图层。

在使用 MVC 模型框架设计复杂的 Web 应用程序时，Velocity 作为一种视图 / 模板化技术——以 VelocityViewServlet的形式——可以很方便地插入到框架中。对于流行的 Jakarta Struts MVC 框架，Velocity 可以与基于 JSP 的视图技术协作，也可以和选择的任何模型技术进行交互。