# Velocity基础语法

## 1.1、什么是Velocity

Velocity是一个基于Java的模版引擎。它允许web页面设计者引用JAVA代码预定义的方法。Web设计者可以根据MVC模式和JAVA程序员并行工作，这意味着Web设计者可以单独专注于设计良好的站点，而程序员则可单独专注于编写底层代码。Velocity将Java代码从web页面中分离出来，使站点在长时间运行后仍然具有很好的可维护性，并提供了一个除JSP和PHP之外的可行的被选方案。

Velocity 可用来从模板产生web页面，SQL, PostScript 以及其他输出。他也可用于一个独立的程序以产生源代码和报告，或者作为其它系统的一个集成组件。这个项目完成后， Velocity 将为Turbine web应用程序框架提供模板服务。Velocity+Turbine方案提供的模板服务将允许web应用按真正的mvc模式进行开发。

Velocity使针对访问者个性的WEB页面客户化（个性化）非常容易，功能强大、灵活性强。

## 1.2、Velocity模板语言

变量赋值：#set($变量名=”值”)

单行注释：##注释内容

多行注释：#\*注释内容\*#

VTL注释块，可以用来存储诸如文档作者、版本信息等：#\*\* 注释快内容 \*\*#

形式引用符：${email}，常用在文本直接邻近的地方

安静引用符：$!{email}，当值为空时，显示空而不是${email}

转移字符：\$emai

条件：

#if( $foo < 10 )

    <strong>Go North</strong>

#elseif( $foo == 10 )

    <strong>Go East</strong>

#else

    <strong>Go West</strong>

#end

循环：

#foreach( $product in $allProducts )

<li>$product</li>

#end

包含（处于安全的原因，被包含的文件只可以放在TEMPLATE\_ROOT下）：

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

解析（就像#include指令，#parse可以使用变量而不是一个实在的模板文件）：

#parse( "me.vm" )

#stop：脚本允许模板设计员停止模板引擎的执行，并返回。这通常用作调试。

宏：



# 二、Spring渲染Velocity模板实例

## 1.1、Maven的dependency

Pom.xml配置：

<properties>

<org.springframework.version>2.5.6</org.springframework.version>

</properties>

<dependencies>

<!-- servlet api -->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>servlet-api</artifactId>

<version>2.5</version>

<scope>provided</scope>

</dependency>

<!-- Spring Framework -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>velocity</groupId>

<artifactId>velocity</artifactId>

<version>1.5</version>

</dependency>

<dependencies>

## 1.2、让Request定位到具体的Controller

web.xml配置：

<!-- 配置上下文载入器 -->

<listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

<!-- 配置DispatcherServlet -->

<servlet>

<servlet-name>demo</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>demo</servlet-name>

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

</servlet-mapping>

ContextLoaderListner默认载入配置文件appName(项目名)-servlet.xml。

appName-servlet.xml配置：

<bean name="/vmview.htm" class="com.tiemei.spring.demo.controller.VelocityController" /> <!-- for velocity view -->

<bean id="velocityConfig"

class="org.springframework.web.servlet.view.velocity.VelocityConfigurer">

<property name="resourceLoaderPath" value="/WEB-INF/templates/" />

</bean>

<bean id="vmViewResolver"

class="org.springframework.web.servlet.view.velocity.VelocityViewResolver">

<property name="cache" value="true" />

<property name="prefix" value="" />

<property name="suffix" value=".vm" />

<property name="order" value="1" /> <!-- 多ViewResovler配置 -->

</bean>

默认BeanNameUrlHandlerMapping，/vmview.htm路径的请求被分发到VelocityController。

## 1.3、VelocityController控制器的编写

public class VelocityController extends AbstractController {

@Override

protected ModelAndView handleRequestInternal(HttpServletRequest req,

HttpServletResponse resp) throws Exception {

System.out.println("------------ enter VelocityController");

List<String> list = new ArrayList<String>();

list.add("one");

list.add("two");

list.add("three");

return new ModelAndView("vmview", "list", list);

}

}

最后new MovdelAndView传入的第一个参数vmview在后续ViewSovler寻找对应view时用到。

## 1.4、定位模版，渲染输出

第2步中的appName-servlet.xml定义了寻找模版的规则。会去/WEB-INF/templates/目录下寻找vmview.vm模版（模版名由第3步确定）。

vmview.vm内容：

<ul>

#foreach( $entry in $list )

<li>$entry</li>

#end

</ul>

$list对应第三步中传入的变量名。

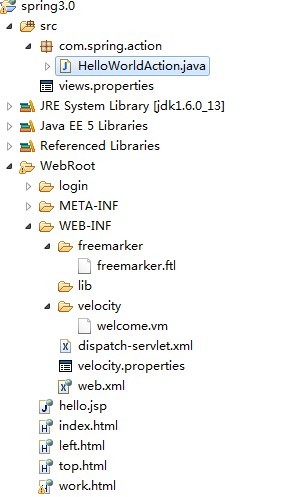
最终浏览器中输出：

one

two

three

# 三、SpringMvc配置(Jsp、Velocity、freemarker)



## 2.1、Spring配置文件

<?xml version="1.0"encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:p="http://www.springframework.org/schema/p"

xmlns:context="http://www.springframework.org/schema/context"

xsi:schemaLocation="

http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-3.0.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context-3.0.xsd">

<context:component-scan

base-package="com.spring.action" />

<!—

org.springframework.web.servlet.view.ResourceBundleViewResolver 用于多个视图

集成时, ResourceBundleViewResolver是通过解析资源文件来解析请求输出文件的。

<property name="basename"value="views"></property>,即表示在/WEB-INF/classes路径下有一个views.properties文件,本例中views.properties的内容为

welcome.(class)=org.springframework.web.servlet.view.velocity.VelocityView

welcome.url=welcome.vm

freemarker.(class)=org.springframework.web.servlet.view.freemarker.FreeMarkerView

freemarker.url=freemarker.ftl

-->

<bean class="org.springframework.web.servlet.view.ResourceBundleViewResolver">

<property name="basename" value="views"></property>

<!--

<property name="order"value="0"></property>

-->

</bean>

<!-- jsp视图解析器 -->

<bean id="jspViewResolver"

class="org.springframework.web.servlet.view.InternalResourceViewResolver">

<property name="viewClass" value="org.springframework.web.servlet.view.JstlView"/>

<property name="prefix" value="/"/>

<property name="suffix" value=".jsp"/>

</bean>

<!-- velocity视图解析器 -->

<bean id="velocityViewResolver"

class="org.springframework.web.servlet.view.velocity.VelocityViewResolver">

<property name="cache" value="true"/>

<property name="prefix" value="/"/>

<property name="suffix" value=".vm"/>

</bean>

<!-- velocity环境配置 -->

<bean id="velocityConfig"

class="org.springframework.web.servlet.view.velocity.VelocityConfigurer">

<!-- velocity配置文件路径 -->

<property name="configLocation" value="/WEB-INF/velocity.properties"/>

<!-- velocity模板路径 -->

<property name="resourceLoaderPath" value="/WEB-INF/velocity/"/>

<!-- velocity模板属性值 -->

<property name= "velocityProperties">

<props>

<prop key= "input.encoding">utf-8</prop>

<prop key= "output.encoding">utf-8</prop>

</props>

</property>

</bean>

<!-- FreeMarker环境配置 -->

<bean id="freemarkerConfig"

class="org.springframework.web.servlet.view.freemarker.FreeMarkerConfigurer">

<!-- freemarker模板位置 -->

<property name="templateLoaderPath" value="/WEB-INF/freemarker/"/>

</bean>

<!-- FreeMarker视图解析 -->

<bean id="freeMarkerViewResolver"

class="org.springframework.web.servlet.view.freemarker.FreeMarkerViewResolver">

<property name="cache" value="true"/>

<property name="prefix" value="/"/>

<property name="suffix" value=".ftl" />

</bean>

</beans>

## 2.2、views.properties文件

#welcome为modelAndView.setViewName(" welcome " ) ;中的welcome .(class)固定写法

welcome.(class)=org.springframework.web.servlet.view.velocity.VelocityView

#welcome.url路径welcome.vm模板名称

welcome.url=welcome.vm

#freemarker为modelAndView.setViewName( " freemarker " ) ; 中的freemarker .(class)固定写法

freemarker.(class)=org.springframework.web.servlet.view.freemarker.FreeMarkerView

#freemarker.url 路径 freemarker.ftl模板名称

freemarker.url= freemarker.ftl

# 四、Velocity中文乱码

