**JSP**

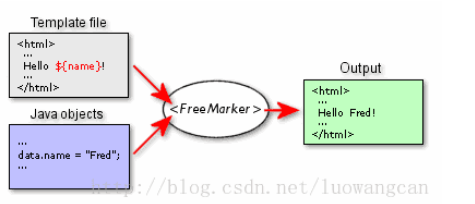
**见4ZEE\_DEEPS\_6\_Servlet&Jsp.docx**

**FreeMarker**

## 一 FreeMarker 介绍

1 FreeMarker是一个模板引擎，一个基于模板生成文本输出的通用工具，使用纯Java编写。FreeMarker被设计用来生成HTML Web页面，特别是基于MVC模式的应用程序。虽然FreeMarker具有一些编程的能力，但通常由Java程序准备要显示的数据，由FreeMarker生成页面，通过模板显示准备的数据，

简单来讲就是模板加数据模型，然后输出页面（如下图）



2 FreeMarker不是一个Web应用框架，而适合作为Web应用框架一个组件FreeMarker与容器无关，因为它并不知道HTTP或Servlet；FreeMarker同样可以应用于非Web应用程序环境

FreeMarker更适合作为Model2框架（如Struts）的视图组件，你也可以在模板中使用JSP标记库。FreeMarker是免费的。可以通过Eclipse的插件来编辑FreeMarker，经过验证，

FreeMarker 最好的 Eclipse 编辑插件是 JBoss Tools。

3 在Maven中使用它：

<dependency>

<groupId>freemarker</groupId>

<artifactId>freemarker</artifactId>

<version>2.3.9</version>

</dependency>

## 二 使用环境搭建

1 创建一个Java项目，这里名为freemarker，在根目录下创建一个名为ftl的文件夹（PS：用来存放freemarker模板），jar啥的就不说了。

2 创建一个FreeMarkerDemo类，用来生成模板以及处理模板与数据之间的关系并输出：

//第一步：实例化Freemarker的配置类

Configuration conf = new Configuration();

//第二步：给配置类设置路径

String dir = "D:\\java21\\freemarker\\ftl\\";

conf.setDirectoryForTemplateLoading(new File(dir));

Template template = conf.getTemplate("freemarker.html");

//第三步：处理模板及数据之间将数据与模板合成一个HTML

//第四步：输出html

Writer out = new FileWriter(new File(dir+"hello.html"));

3 定义一个Person对象：

public class Person {

private String id;

private String name;

public String getId() {

return id;

}

public void setId(String id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

}

4 对象输出例子

（1）定义数据

Map<String,Object> root = new HashMap<String,Object>();

Person p = new Person();

p.setId("111");

p.setName("哈哈哈");

root.put("person", p);

template.process(root, out);

out.flush();

out.close();

（2）模板页面书写

${person.id}<br /> ${person.name}

点击hello.html查看结果

5 遍历List例子

（1）定义数据

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

persons.add("阿灵罗");

persons.add("罗零");

persons.add("灵罗");

root.put("persons",persons);

template.process(root, out);

out.flush();

out.close();

（2）模板页面书写:

<#list persons as person> ${person} </#list>

<!--获取当前索引 -->

<#list persons as p> ${p\_index} </#list>

6 遍历Map

（1）定义数据

Map<String,String> mx = new HashMap<String,String>();

mx.put("alingluo", "阿灵罗");

mx.put("lling", "罗零");

root.put("mx", mx);

template.process(root, out);

out.flush();

out.close();

（2）模板页面书写

${mx.alingluo}/${mx.lling} <#list mx?keys as k> ${mx[k]} </#list>

7 ListMap遍历

（1）定义数据

List<Map<String,String>> maps = new ArrayList<Map<String,String>>();

Map<String,String> pms1 = new HashMap<String,String>();

pms1.put("id1", "张三");

pms1.put("id2", "李四");

Map<String,String> pms2 = new HashMap<String,String>();

pms2.put("id1", "张三");

pms2.put("id2", "李四");

maps.add(pms1);

maps.add(pms2);

root.put("maps", maps);

template.process(root, out);

out.flush();

out.close();

（2）模板页面书写

<#list maps as m> ${m.id1}/${m.id2} </#list> <#list maps as m> <#list

m?keys as k> ${m[k]} </#list> </#list>

8 在模板中赋值

root.put("world", "世界你好");

1:<#assign x=0 /> ${x}

2:<#assign x="${world}" /> ${x}

3:<#assign x>世界太好了</#assign> ${x}

4:<#assign x>

<#list ["星期一", "星期二", "星期三",

"星期四", "星期五", "星期六", "星期天"] as n> ${n}

</#list>

</#assign>

${x}



9 判断

<!-- 判断 -->

<#list ["星期一", "星期二", "星期三", "星期四", "星期五", "星期六", "星期天"] as n>

<#if n != "星期一">

${n}

</#if>

</#list>

<#list ["星期一", "星期二", "星期三", "星期四", "星期五", "星期六", "星期天"] as n>

<#if n\_index != 0>

${n}

</#if>

</#list>

<#list ["星期一", "星期二", "星期三", "星期四", "星期五", "星期六", "星期天"] as n>

<#if (n\_index == 1) || (n\_index == 3)>

${n}

</#if>

</#list>

<!-- else -->

<#list ["星期一", "星期二", "星期三", "星期四", "星期五", "星期六", "星期天"] as n>

<#if (n\_index == 1) || (n\_index == 3)>

${n} --红色

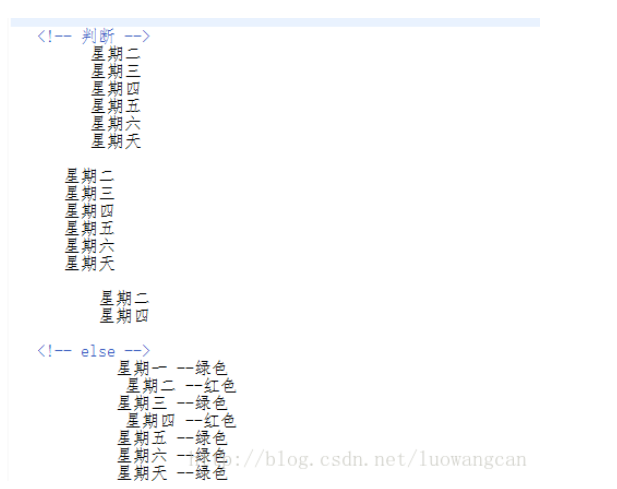
<#else>

${n} --绿色

</#if>

</#list>

生成页面效果



10 时间格式

模板页面效果如下：

1:${cur\_time?date}

2:${cur\_time?datetime}

3:${cur\_time?time}

生成页面效果如下：



11 FreeMarkerDemo全部代码：

public class FreeMarkerDemo {

public static void main(String[] args) throws IOException, TemplateException {

//第一步：实例化Freemarker的配置类

Configuration conf = new Configuration();

//第二步：给配置类设置路径

String dir = "D:\\java21\\freemarker\\ftl\\";

conf.setDirectoryForTemplateLoading(new File(dir));

Template template = conf.getTemplate("freemarker.html");

//第三步：处理模板及数据之间将数据与模板合成一个HTML

//第四步：输出html

Writer out = new FileWriter(new File(dir+"hello.html"));

//定义数据

// Map root = new HashMap();

//执行生成

//定义数据

Map<String,Object> root = new HashMap<String,Object>();

Person p = new Person();

p.setId("111");

p.setName("哈哈哈");

root.put("person", p);

root.put("world", "世界你好");

//遍历List

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

persons.add("阿灵罗");

persons.add("罗零");

persons.add("灵罗");

root.put("persons",persons);

root.put("cur\_time", new Date());

//遍历Map

Map<String,String> mx = new HashMap<String,String>();

mx.put("alingluo", "阿灵罗");

mx.put("lling", "罗零");

root.put("mx", mx);

//遍历List<Map>

List<Map<String,String>> maps = new ArrayList<Map<String,String>>();

Map<String,String> pms1 = new HashMap<String,String>();

pms1.put("id1", "张三");

pms1.put("id2", "李四");

Map<String,String> pms2 = new HashMap<String,String>();

pms2.put("id1", "张三");

pms2.put("id2", "李四");

maps.add(pms1);

maps.add(pms2);

root.put("maps", maps);

//定义数据

root.put("val", null);

template.process(root, out);

out.flush();

out.close();

}

}

12 freemarker.html全部代码：

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>Insert title here</title>

</head>

<body>

${person.id}

<br /> ${person.name} <#list persons as person> ${person} </#list>

<!--Map写法 -->

${mx.alingluo}/${mx.lling} <#list mx?keys as k> ${mx[k]} </#list>

<!--List<Map>写法 -->

<#list maps as m> ${m.id1}/${m.id2} </#list> <#list maps as m> <#list

m?keys as k> ${m[k]} </#list> </#list>

<!--获取当前索引 -->

<#list persons as p> ${p\_index} </#list>

<!-- 在模板中赋值 -->

1:<#assign x=0 /> ${x}

2:<#assign x="${world}" /> ${x}

3:<#assign x>世界太好了</#assign> ${x}

4:<#assign x>

<#list ["星期一", "星期二", "星期三",

"星期四", "星期五", "星期六", "星期天"] as n> ${n}

</#list>

</#assign>

${x}

<!-- 判断 -->

<#list ["星期一", "星期二", "星期三", "星期四", "星期五", "星期六", "星期天"] as n>

<#if n != "星期一">

${n}

</#if>

</#list>

<#list ["星期一", "星期二", "星期三", "星期四", "星期五", "星期六", "星期天"] as n>

<#if n\_index != 0>

${n}

</#if>

</#list>

<#list ["星期一", "星期二", "星期三", "星期四", "星期五", "星期六", "星期天"] as n>

<#if (n\_index == 1) || (n\_index == 3)>

${n}

</#if>

</#list>

<!-- else -->

<#list ["星期一", "星期二", "星期三", "星期四", "星期五", "星期六", "星期天"] as n>

<#if (n\_index == 1) || (n\_index == 3)>

${n} --红色

<#else>

${n} --绿色

</#if>

</#list>

<!--时间格式 -->

1:${cur\_time?date}

2:${cur\_time?datetime}

3:${cur\_time?time}

<!-- null -->

${val!}

<!-- 宏定义 -->

1:

<#macro table u>

${u}

</#macro>

<@table u=8 />

2:

<#macro table u>

${u}

<#nested/>

</#macro>

<@table u=8 >这是8</@table>

</body>

</html>

13 生成的HTML页面内容如下：

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>Insert title here</title>

</head>

<body>

111

<br /> 哈哈哈 阿灵罗 罗零 灵罗

<!--Map写法 -->

阿灵罗/罗零 阿灵罗 罗零

<!--List<Map>写法 -->

张三/李四 张三/李四 李四 张三 李四 张三

<!--获取当前索引 -->

0 1 2

<!-- 在模板中赋值 -->

1: 0

2: 世界你好

3: 世界太好了

4:

星期一

星期二

星期三

星期四

星期五

星期六

星期天

<!-- 判断 -->

星期二

星期三

星期四

星期五

星期六

星期天

星期二

星期三

星期四

星期五

星期六

星期天

星期二

星期四

<!-- else -->

星期一 --绿色

星期二 --红色

星期三 --绿色

星期四 --红色

星期五 --绿色

星期六 --绿色

星期天 --绿色

<!--时间格式 -->

1:2017-5-8

2:2017-5-8 10:56:56

3:10:56:56

<!-- null -->

<!-- 宏定义 -->

1:

8

2:

8

这是8

</body>

</html>

**Velocity**

## 一、语法及概述

**velocity与freemaker、jstl并称为java web开发三大标签技术，而且velocity在codeplex上还有.net的移植版本NVelocity**，(注：castle团队在github上也维护了一个版本)对于使用异构技术的团队（即要搞.NET又要搞JAVA），总是希望找一种通用的技术，兼容所有技术平台，以便降低学习成本，无疑velocity是一种值得考虑的选择。Velocity是一个基于java的模板引擎。它允许任何人仅仅简单的使用模板语言来引用由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。

**1. "#"用来标识Velocity的脚本语句。**

如：#set、#if 、#else、#foreach、#end、#include、#parse、#macro等。

**2. "$"用来标识一个对象(或理解为PHP的变量)**

如：i、

user等。

**3. "{}"用来明确标识Velocity变量，和普通模版字符串区分开来；**

如：${user}‘s age 可以显示为 riqi's age。

**4. "!"强制把不存在的变量显示为空白。**

如：!msg，假如msg对象为空，则模版中不显示该变量；如果缺少"!"，则显示

msg字符串，这是我们不想要的结果。

**5. 变量的定义和赋值。不需要指定变量的类型，类似弱类型语言PHP可以随意指定，在赋值后自动判定变量的类型**，如：

#set($username="riqi") ##设置用户名

#set($age=26) ##设置年龄

**6. 数组循环：**

#foreach ($user in $users)

$!{user} $!{velocityCount} <br />

#end

users可以是Vector、Hashtable或者Array，Velocity提供了得到循环次数的值：

velocityCount。

**7. 语句注释：**

单行注释：## 单行注释代码

多行注释：#\* 多行注释代码 \*#

**8. 模版支持关系和逻辑操作符运算，如：&&、||、!**

**9. 宏定义：#macro ，类似PHP声明一个函数，其中有函数名称和参数列表。先定义再调用。**

**10. 终止命令：#stop，类似PHP的exit(); 停止执行模板引擎并返回。**

**11. 引入公共模版文件：#include与#parse，它们的差异是：**

(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。

**12. 转义字符'\'.**

这个和其它语言没有差异，假如：user="riqi";那么，$user表示输出

user字符串，\$user表示输出\riqi。

1. **Velocity内置了一部分java对象**

如：request、response、$session等，在vm模版里可以直接调用。

## 二、与strtus2的集成

1 <dependency>

2 <groupId>org.apache.velocity</groupId>

3 <artifactId>velocity</artifactId>

4 <version>1.7</version>

5 </dependency>

6

7 <dependency>

8 <groupId>org.apache.velocity</groupId>

9 <artifactId>velocity-tools</artifactId>

10 <version>2.0</version>

11 </dependency>

pom.xml中加入这二项即可，其它不用刻意配置。struts2同时支持jstl(.jsp)、velocity(.vm)、freemaker(.ftl)三种模板。

## 三、定义变量

1 #set($awbpre='112')

2 #set($awbno='89089011')

3 #set($airwayBillNo=$awbpre+' - '+$awbno)

4 $awbpre - $awbno <br/>

5 $airwayBillNo

velocity的语法符号大概分二类，一类用#开头，代表控制符号，#set表示定义变量，另一类用$开头，通常用于显示变量，上面的示例定义了三个变量：

awbpre 值为'112'，awbno值为'89089011'，airwayBillNo值为 '112 - 89089011'

第4,5二行输出内容

## 四、遍历数组

1 #set($list = ["CTU", "SHA", "LAX"])

2 #foreach ($item in $list)

3 $velocityCount . $item <br/>

4 #end

解释：定义了一个数组，然后遍历输出，其中velocityCount为索引变量

五、遍历HashTabl**e**

1 #foreach($key in $table.keySet())

2 $key -> $table.get($key)<br/>

3 #end

## 六、判断是否为空

1 #if($null.isNull($orderList.orders) || $orderList.orders.size()==0)

2 订单列表为空

3 #else

4 订单列表：<br/>

5 #foreach ($order in $orderList.orders)

6 $velocityCount: $order.id / $order.clientName / $order.amount / $order.createTime<br/>

7 #end

8 #end

上面是判断集合是否为空的，如果判断单个对象是否为空，参考下面这样：

1 #if($(orderDto))

2 订单对象有值

3 #else

4 订单对象为空

5 #end

6

7 #if(!$(orderDto))

8 订单对象为空

9 #else

10 订单对象有值

11 #end

## 七、宏示例

宏可以理解为“函数”，定义一个宏即相当于定义一个子函数，调用宏，即为调用子函数

1 #macro(renderOrderList $orders)

2 <table border="1">

3 <tr>

4 <th>Id</th>

5 <th>ClientName</th>

6 <th>Amount</th>

7 <th>CreateTime</th>

8 </tr>

9 #foreach($o in $orders)

10 <tr><td>$o.id</td><td>$o.clientName</td><td>$o.amount</td><td>$o.createTime</td></tr>

11 #end

12 </table>

13 #end

14

15 #renderOrderList($orderList.orders)

## 八、数值、日期格式化

1 $order.createTime<br/>

2 $date.year - $date.month - $date.day <br/>

3 $date.format('yyyy-MM-dd HH:mm:ss',$order.createTime,$locale)<br/>

4 $date.format('MMMdd',$order.createTime,$locale)<br/>

5 $convert.toLocale("en\_US") <br/>

6 $date.format('MMM,dd',$order.createTime,$convert.toLocale("en\_US"))<br/>

7 $date.format('yyyy-MM-dd',$order.createTime,$locale)<br/>

8 $order.amount<br/>

9 $number.format('0.00',$order.amount)<br/>

NumberTool中还有货币格式化的功能：$number.format("currency", $agentBillDto.feeTotal)

要使用格式化功能，需要加一点配置，struts.xml文件中加一行

<constant name="struts.velocity.toolboxlocation" value="WEB-INF/classes/toolbox.xml" />

然后在toolbox.xml中，参考下面的内容：

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

2

3 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"

4 "http://www.w3.org/2002/xmlspec/dtd/2.10/xmlspec.dtd">

5 <toolbox>

6 <tool>

7 <key>number</key>

8 <scope>application</scope>

9 <class>org.apache.velocity.tools.generic.NumberTool</class>

10 </tool>

11 <tool>

12 <key>date</key>

13 <scope>application</scope>

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

15 </tool>

16 <tool>

17 <key>text</key>

18 <scope>request</scope>

19 <class>org.apache.velocity.tools.struts.MessageTool</class>

20 </tool>

21 <tool>

22 <key>convert</key>

23 <scope>application</scope>

24 <class>org.apache.velocity.tools.generic.ConversionTool</class>

25 </tool>

26 </toolbox>

这些XXXTool其实是一个很好的例子，因为velocity的vm文件里不能直接写java代码，如果我们想扩展一些常用方法，可以将一些常用方法写成XXXTool工具类，然后在toolbox中注册即可。

## 九、国际化

1 当前语言环境：$locale <br/>

2 #stext("name=%{getText('appName')}")

虽然Velocity-Tools 2.0中提供了MessageTool，但是我一直没尝试成功，只能借助struts2本身的标签来处理了。struts2中首先得定义国际化资源文件的BaseName

1 <constant name="struts.custom.i18n.resources" value="message"></constant>

然后在classPath目录下，放二个文件message\_zh\_CN.properties、message\_en\_US.properties，里面放一个appName=XXX的内容，用#stext就能取到国际化的内容了

## 十、使用struts2标签

虽然有了velocity基本上可以告别struts2的那一堆tags，但是如果怀念struts2里的标签，也可以继续使用，方法：以“#s”开头就行了，参考下面的示例：

1 #stextarea ("label=Biography" "name=bio" "cols=20" "rows=3") <br/>

2 #sselect("label=Favourite Color" "list={'Red', 'Blue', 'Green'}" "name=favouriteColor" "emptyOption=true" "headerKey=None" "headerValue=None") <br/>

## 十一、内建对象

1 $request<br/>

2 name = $request.getParameter("name")<br/>

3 $session<br/>

Velocity可以直接使用struts2的很多内置对象，比如Request、Session、Response，上面的示例演示了如何获取 url请求参数

## 十二、include、parse实现布局模块化

每个页面，通常会有一些公用的头、尾，可以用include或parse来包括其它vm文件（或txt/html文件），这二个的区别在于include只是简单的把其它文件导入进来，不会执行任何vm语法的解析。而parse导入其它vm文件时，如果其它vm文件里有一些指令，比如定义变量，定义宏之类，parse会解析执行。

1 #parse("template/header.vm")

2 #include("template/footer.vm")

关于加载的路径，这里要重点解释一下，官方文档上也讲得不清不楚，Velocity支持二种路径加载机制，按classPath或按filePath，默认是按classPath路径加载，即：只要被包含的.vm文件在/WEB-INF/classes目录下即可。上面的示例，将在/WEB-INF/classes/template目录下，搜索header.vm、footer.vm这二个文件，如果找到就加载，否则出错。最后谈下IDE以.vm的可视化支持问题，目前最新的eclipse上，暂无好用的插件(googlecode上的插件大多已经没人维护了，与最新的eclipse不兼容)，建议使用IntelliJ Idea，它对vm的可视化支持程度较好。