Atitit java字符串模板渲染总结

目录

[1. 总结：指标 1](#_Toc1959)

[1.1. 支持中文变量 提升可读性 1](#_Toc31585)

[1.2. 变量placeholder简单性，，velo可以直接￥前导简单。。Free的必须全包 1](#_Toc28516)

[1.3. 支持位置索引，命名索引 1](#_Toc10737)

[2. 选项 1](#_Toc27128)

[2.1. 对于简单的格式化或字符串组装，使用MessageFormat.format。格式化处理更丰富要使用String.format方法 2](#_Toc138)

[2.2. Table of Contents 2](#_Toc2000)

[3. Ati总结 2](#_Toc25660)

[4. Oth 3](#_Toc16045)

[4.1. 1. MessageFormat  3](#_Toc19697)

[4.2. 3。 Velocity  4](#_Toc17235)

# ****总结：指标****

## ****支持中文变量 提升可读性****

****只有freemark支持，velo貌似不支持****

## ****变量placeholder简单性，，velo可以直接￥前导简单。。Free的必须全包****

## 支持位置索引，命名索引

Velo，和free默认不支持，可以自己diy实习基于他们。。使用obj数组参数模式。

# ****选项****

## ****Diy模板引擎atiTemplt，直接支持以上所有指标****

## 对于简单的格式化或字符串组装，使用MessageFormat.format。格式化处理更丰富要使用String.format方法

## Table of Contents

* [1. ++](https://github.com/venusdrogon/feilong-core/wiki/%E5%AD%97%E7%AC%A6%E4%B8%B2%E6%A0%BC%E5%BC%8F%E5%8C%96%E6%B1%87%E6%80%BB)
* [2. StringBuffer / StringBuilder](https://github.com/venusdrogon/feilong-core/wiki/%E5%AD%97%E7%AC%A6%E4%B8%B2%E6%A0%BC%E5%BC%8F%E5%8C%96%E6%B1%87%E6%80%BB" \l "stringbuffer-stringbuilder)
* [3. StringUtil.format(String, Object…​)](https://github.com/venusdrogon/feilong-core/wiki/%E5%AD%97%E7%AC%A6%E4%B8%B2%E6%A0%BC%E5%BC%8F%E5%8C%96%E6%B1%87%E6%80%BB" \l "stringutil-format-string-object)
* [4. MessageFormatUtil.format(String, Object…​)](https://github.com/venusdrogon/feilong-core/wiki/%E5%AD%97%E7%AC%A6%E4%B8%B2%E6%A0%BC%E5%BC%8F%E5%8C%96%E6%B1%87%E6%80%BB" \l "messageformatutil-format-string-object)
* [5. Slf4jUtil.format(String, Object…​)](https://github.com/venusdrogon/feilong-core/wiki/%E5%AD%97%E7%AC%A6%E4%B8%B2%E6%A0%BC%E5%BC%8F%E5%8C%96%E6%B1%87%E6%80%BB" \l "slf4jutil-format-string-object)
* [6. StringUtil.replace(CharSequence, Map<String, V>)](https://github.com/venusdrogon/feilong-core/wiki/%E5%AD%97%E7%AC%A6%E4%B8%B2%E6%A0%BC%E5%BC%8F%E5%8C%96%E6%B1%87%E6%80%BB" \l "stringutil-replace-charsequence-map-string-v)
* [7. VelocityUtil.parseString(String, Map<String, ?>)](https://github.com/venusdrogon/feilong-core/wiki/%E5%AD%97%E7%AC%A6%E4%B8%B2%E6%A0%BC%E5%BC%8F%E5%8C%96%E6%B1%87%E6%80%BB" \l "velocityutil-parsestring-string-map-string)
* [8. VelocityUtil.parseTemplateWithClasspathResourceLoader(String, Map<String, ?>)](https://github.com/venusdrogon/feilong-core/wiki/%E5%AD%97%E7%AC%A6%E4%B8%B2%E6%A0%BC%E5%BC%8F%E5%8C%96%E6%B1%87%E6%80%BB" \l "velocityutil-parsetemplatewithclasspathresourceloader-string-map-string)
* [9. 性能对比](https://github.com/venusdrogon/feilong-core/wiki/%E5%AD%97%E7%AC%A6%E4%B8%B2%E6%A0%BC%E5%BC%8F%E5%8C%96%E6%B1%87%E6%80%BB" \l "%E6%80%A7%E8%83%BD%E5%AF%B9%E6%AF%94)
* [10. 参考](https://github.com/venusdrogon/feilong-core/wiki/%E5%AD%97%E7%AC%A6%E4%B8%B2%E6%A0%BC%E5%BC%8F%E5%8C%96%E6%B1%87%E6%80%BB" \l "%E5%8F%82%E8%80%83)

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

# Ati总结

优先使用****MessageFormat  ，三个变量内。。因为都是索引位置变量但是简单。。****

****其次使用velocity模板，因为变量定义简单。。。****

****Freemark的必须全包含麻烦些****

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。

五、引用

   引用语句就是对引擎上下文对象中的属性进行操作。语法方面分为常规语法（ $属性 ）和正规语法（ ${属性} ）。在普通模式下上述两种写法，当引擎上下文对象中没有对应的属性时，最终结果会直接输出 $属性 或 ${属性} ，若要不输出则需要改写为 $!属性 和 $!{属性} 。

# Oth

## ****1. MessageFormat****

优点：不需要映入第三方类库，门槛低   
缺点：使用序号来和后面参数约定，耦合性比较大，维护成本高，可重用性不高   
      对于所有信息都放到bean中，需要后期将对象一个个的get属性，开发代码比较多

**Java代码  [IMG_256](https://xiaoqiangcn.iteye.com/blog/javascript:void())**

1. System.out.println(MessageFormat.format("我是{0},我来自{1},今年{2}岁", "中国人", "北京", "22"));

****2. freemarker****  
  
优点：重用性高，只要传入待替换string及数据对象，可以完成所有替换   
      可维护性高，模板修改，替换代码不需要变更   
缺点：bean属性删除的时候替换代码不会报错，导致原值直接输出 需要映入第三方类库

**Java代码  [IMG_257](https://xiaoqiangcn.iteye.com/blog/javascript:void())**

1. **try** {
2. Configuration cfg = **new** Configuration();
3. StringTemplateLoader stl =  **new** StringTemplateLoader();
4. stl.putTemplate("", "hello：${name}");
5. cfg.setTemplateLoader(stl);
6. Template template = cfg.getTemplate("");
8. Bean b = **new** Bean();
9. b.setName("aaa");
11. StringWriter writer = **new** StringWriter();
12. template.process(b, writer);
13. System.out.println(writer.toString());
14. } **catch** (Exception e) {
15. // TODO Auto-generated catch block
16. e.printStackTrace();
17. }

## ****3。 Velocity****

优点：键值对的形式，由于MessageFormat不需要维护序号   
缺点：重用性不高;需要映入第三方类库

**Java代码  [IMG_258](https://xiaoqiangcn.iteye.com/blog/javascript:void())**

1. Context context  = **new** VelocityContext();
2. context.put("name", "aaa");
3. StringWriter sw = **new** StringWriter();
4. **try** {
5. Velocity.evaluate(context, sw, "velocity", "hello：${name}");
6. } **catch** (Exception e) {
7. // TODO Auto-generated catch block
8. e.printStackTrace();
9. }
10. System.out.println(sw.toString());

Velocity基本常用语法 - @ 小浩 - 博客园.mhtml

Velocity魔法堂系列二：VTL语法详解 - ^\_^肥仔John - 博客园.mhtml