**模板+数据模型=输出 FTL = FreeMarker Tag Language**

数据模型的结构类似于一棵单根树，使用hash和下标来查找。访问的时候使用级联语法，如 animals.monkey[1].eye

**01、语法**

${…} interpolations插值，用于描述被替换的内容；

<#..></#..> 标签，是freemarker的指令，不会直接输出。用户自定义的标签用@代替#

<#-- --> 注释，和<!-- -->作用相同

常用指令：

<#if><#else></#if> 注意else嵌套在里面

<#list collection as var>…</#list>

<#include “…”> 引入其他页面

! 指定默认值 如 ${name!”sherlokk”}表示如果name不存在或者为null，输出sherlokk

?? 判断变量是否存在和是否为null 如<#if var??>哈哈</#if> 如果var不存在 则不输出

注意：对于多级变量，如果要判断每级是否存在，则应该将其用括号括起来。比较符和javascript类似，但严格区分字符串和数值。

**02、数据类型**

标量：字符串、数字、布尔值、日期

容器：哈希表、序列、集

子程序：方法和函数、用户自定义指令

其他：节点

方法和函数：方法来自于数据模型，反射Java对象的方法；函数是定义在模板内的。

序列表示方法：[“1”,2,”3”,animal] 类型可以不同，也可以使用 2..5 这种

哈希表：JSON

特殊变量(内置变量)： .varName

内建函数： ?functionName 如3.5?int输出3，”cap”?upper\_case输出CAP

宏指令 macro 用于自定义指令 如<#macro greet>…</#macro> 调用 <@greet/>，可以定义参数，指定参数默认值。

模板中的自定义变量：

1、简单变量 能从模板中的任意位置访问，或者从使用include指令引入的文件访问。使用assign或者macro指令来创建和替换。

2、局部变量 只能在宏的定义体内声明和使用。使用local来创建和替换。

3、循环变量 在<#list>中使用。

命名空间：

通过include指令引入的文件，和当前文件使用同一个命名空间，可能会有变量冲突。使用import指令来解决。如<#import “/lib/stock.ftl” as stock> 调用 <@stock.user> 假设stock中定义了user宏指令。

此时，如果要替换stock中的变量，需要这样写： <#assign mail=”..” in stock>

**03、与Java的配合**

所有数据模型都会被包装为freemarker.template.TemplateModel类型。这个借口有很多子接口，一般命名都是Template[Type]Model，Type是数据类型，如TemplateHashModel。

page 155