

-

整理自互联网，仅供学习交流。



2018-10-30

[公司名称]

[公司地址]

目录

**未找到目录项。**

123

## 网页静态化

可以使用Freemarker实现网页静态化。

### 什么是freemarker

FreeMarker是一个用Java语言编写的模板引擎，它基于模板来生成文本输出。FreeMarker与Web容器无关，即在Web运行时，它并不知道Servlet或HTTP。它不仅可以用作表现层的实现技术，而且还可以用于生成XML，JSP或Java 等。

目前企业中:主要用Freemarker做静态页面或是页面展示

常用的java模板引擎还有哪些？

Jsp、Freemarker、Thymeleaf 、Velocity 等。

2、模板+数据模型=输出

freemarker并不关心数据的来源，只是根据模板的内容，将数据模型在模板中显示并输出文件（通常为html，也

可以生成其它格式的文本文件）

1、数据模型

数据模型在java中可以是基本类型也可以List、Map、Pojo等复杂类型。

2、来自官方的例子：（https://freemarker.apache.org/docs/dgui\_quickstart\_basics.html）

数据模型：

## 使用方法

### 演示1

freemarker作为springmvc一种视图格式，默认情况下SpringMVC支持freemarker视图格式。

#### 创建工程

需要创建Spring Boot+Freemarker工程用于测试模板。

pom.xml 如下

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <project xmlns="http://maven.apache.org/POM/4.0.0"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  <parent>  <artifactId>xc-framework-parent</artifactId>  <groupId>com.xuecheng</groupId>  <version>1.0-SNAPSHOT</version>  <relativePath>../xc-framework-parent/pom.xml</relativePath>  </parent>  <modelVersion>4.0.0</modelVersion>  <artifactId>test‐freemarker</artifactId>  <dependencies>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring‐boot‐starter‐freemarker</artifactId>  </dependency>  <dependency>  <groupId>org.projectlombok</groupId>  <artifactId>lombok</artifactId>  </dependency>  <dependency>  <groupId>com.squareup.okhttp3</groupId>  <artifactId>okhttp</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>  <dependency>  <groupId>org.apache.commons</groupId>  <artifactId>commons-io</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring‐boot‐starter‐test</artifactId>  </dependency>  </dependencies>  </project> |

#### 配置文件

配置application.yml和 logback-spring.xml，从cms工程拷贝这两个文件，进行更改， logback-spring.xml无需更

改，application.yml内容如下：

|  |
| --- |
| server:  port: 8088 #服务端口  spring:  application:  name: test‐freemarker #指定服务名  freemarker:  cache: false #关闭模板缓存，方便测试  settings:  template\_update\_delay: 0 #检查模板更新延迟时间，设置为0表示立即检查，如果时间大于0会有缓存不方便进行模板测试 |

#### 创建模型类

在freemarker的测试工程下创建模型类型用于测试

|  |
| --- |
| package com.xuecheng.test.freemarker.model;  import lombok.Data;  import lombok.ToString;  import java.util.Date;  import java.util.List;  @Data  @ToString  public class Student {      private String name;//姓名      private int age;//年龄      private Date birthday;//生日      private Float money;//钱包      private List<Student> friends;//朋友列表      private Student bestFriend;//最好的朋友  } |

#### 创建模板

在 src/main/resources下创建templates，此目录为freemarker的默认模板存放目录。

在templates下创建模板文件test1.ftl，模板中的${name}最终会被freemarker替换成具体的数据。

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf‐8">  <title>Hello World!</title>  </head>  <body>  Hello ${name}!  </body>  </html> |

#### 创建controller

创建Controller类，向Map中添加name，最后返回模板文件。

|  |
| --- |
| package com.xuecheng.test.freemarker.controller;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Controller;  import org.springframework.web.bind.annotation.RequestMapping;  import org.springframework.web.client.RestTemplate;  import java.util.Map;  /\*\*  \* @ClassName FreemarkerController  \* @Description TODO  \* @Author yaosiyuan  \* @Date 2019/4/4 21:27  \* @Version 1.0  \*\*/  @RequestMapping("/freemarker")  @Controller  public class FreemarkerController {  @Autowired  RestTemplate restTemplate;  @RequestMapping("/test1")  public String freemarker(Map<String, Object> map) {  // map就是freemarker模板所使用的数据  map.put("name", "test1");  //返回模板文件名称,基于resources/templates路径的  return "test1";  }  } |

#### 创建启动类

|  |
| --- |
| package com.xuecheng.test.freemarker;  import org.springframework.boot.SpringApplication;  import org.springframework.boot.autoconfigure.SpringBootApplication;  /\*\*  \* @ClassName FreemarkerTestApplication  \* @Description 启动类  \* @Author yaosiyuan  \* @Date 2019/4/4 21:34  \* @Version 1.0  \*\*/  @SpringBootApplication  public class FreemarkerTestApplication {  public static void main(String[] args) {  SpringApplication.run(FreemarkerTestApplication.class,args);  }  } |

### 演示2

#### 创建测试工程

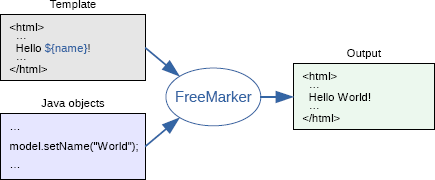
创建一个freemarker 的测试工程专门用于freemarker的功能测试与模板的测试。

把freemarker的jar包添加到工程中。

Maven工程添加依赖

|  |
| --- |
| <dependency>  <groupId>org.freemarker</groupId>  <artifactId>freemarker</artifactId>  <version>2.3.23</version>  </dependency> |

原理：



使用步骤：

第一步：创建一个Configuration对象，直接new一个对象。构造方法的参数就是freemarker对于的版本号。

第二步：设置模板文件所在的路径。

第三步：设置模板文件使用的字符集。一般就是utf-8.

第四步：加载一个模板，创建一个模板对象。

第五步：创建一个模板使用的数据集，可以是pojo也可以是map。一般是Map。

第六步：创建一个Writer对象，一般创建一FileWriter对象，指定生成的文件名。

第七步：调用模板对象的process方法输出文件。

第八步：关闭流。

常用的java模板引擎还有哪些？

Jsp、Freemarker、Thymeleaf 、Velocity 等。

2、模板+数据模型=输出

freemarker并不关心数据的来源，只是根据模板的内容，将数据模型在模板中显示并输出文件（通常为html，也

可以生成其它格式的文本文件）

1、数据模型

数据模型在java中可以是基本类型也可以List、Map、Pojo等复杂类型。

2、来自官方的例子：（https://freemarker.apache.org/docs/dgui\_quickstart\_basics.html）

数据模型：

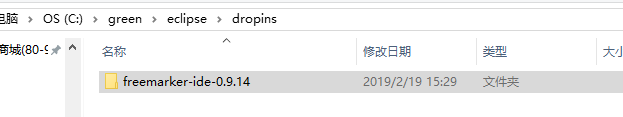
/e3-item-web/src/main/webapp/WEB-INF/ftl/hello.ftl

|  |
| --- |
| ${hello} |

/e3-item-web/src/test/java/cn/e3mall/freemarker/FreeMarkerTest.java

|  |
| --- |
| @Test  public void testFreeMarker() throws Exception {  //1、创建一个模板文件  //2、创建一个Configuration对象  Configuration configuration = new Configuration(Configuration.getVersion());  //3、设置模板文件保存的目录  configuration.setDirectoryForTemplateLoading(new File("D:/workspaces-itcast/JavaEE32/e3-item-web/src/main/webapp/WEB-INF/ftl"));  //4、模板文件的编码格式，一般就是utf-8  configuration.setDefaultEncoding("utf-8");  //5、加载一个模板文件，创建一个模板对象。  Template template = configuration.getTemplate("hello.ftl");  //6、创建一个数据集。可以是pojo也可以是map。推荐使用map  Map data = new HashMap<>();  data.put("hello", "hello freemarker!");  //7、创建一个Writer对象，指定输出文件的路径及文件名。  Writer out = new FileWriter(new File("D:/temp/JavaEE32/freemarker/hello.txt"));  //8、生成静态页面  template.process(data, out);  //9、关闭流  out.close();  } |

安装插件



#### Freemarker整合spring

引入jar包：

Freemarker的jar包

|  |
| --- |
| <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-context-support</artifactId>  </dependency> |

##### 创建整合spring的配置文件

|  |
| --- |
| <?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"*  xmlns:dubbo=*"http://code.alibabatech.com/schema/dubbo"* xmlns:mvc=*"http://www.springframework.org/schema/mvc"*  xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd*  *http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-4.2.xsd*  *http://code.alibabatech.com/schema/dubbo http://code.alibabatech.com/schema/dubbo/dubbo.xsd*  *http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd"*>  <bean id=*"freemarkerConfig"*  class=*"org.springframework.web.servlet.view.freemarker.FreeMarkerConfigurer"*>  <property name=*"templateLoaderPath"* value=*"/WEB-INF/ftl/"* />  <property name=*"defaultEncoding"* value=*"UTF-8"* />  </bean>  </beans> |

## FreeMarker 基础

### 核心指令

#### 数据模型

Freemarker静态化依赖数据模型和模板，下边定义数据模型：

下边方法形参map即为freemarker静态化所需要的数据模型，在map中填充数据：

|  |
| --- |
| @RequestMapping("/test1")      public String freemarker(Map<String, Object> map){          //向数据模型放数据          map.put("name","黑马程序员");          Student stu1 = new Student();          stu1.setName("小明");          stu1.setAge(18);          stu1.setMondy(1000.86f);          stu1.setBirthday(new Date());          Student stu2 = new Student();          stu2.setName("小红");          stu2.setMondy(200.1f);          stu2.setAge(19);  //        stu2.setBirthday(new Date());          List<Student> friends = new ArrayList<>();          friends.add(stu1);          stu2.setFriends(friends);          stu2.setBestFriend(stu1);          List<Student> stus = new ArrayList<>();          stus.add(stu1);          stus.add(stu2);          //向数据模型放数据          map.put("stus",stus);          //准备map数据          HashMap<String,Student> stuMap = new HashMap<>();          stuMap.put("stu1",stu1);          stuMap.put("stu2",stu2);          //向数据模型放数据          map.put("stu1",stu1);          //向数据模型放数据          map.put("stuMap",stuMap);          //返回模板文件名称          return "test1";      } |

#### List指令

本节定义freemarker模板，模板中使用freemarker的指令，关于freemarker的指令需要知道：

|  |
| --- |
| 1、注释，即<#‐‐和‐‐>，介于其之间的内容会被freemarker忽略  2、插值（Interpolation）：即${..}部分,freemarker会用真实的值代替${..}  3、FTL指令：和HTML标记类似，名字前加#予以区分，Freemarker会解析标签中的表达式或逻辑。  4、文本，仅文本信息，这些不是freemarker的注释、插值、FTL指令的内容会被freemarker忽略解析，直接输出内  容。 |

在test1.ftl模板中使用list指令遍历数据模型中的数据：

|  |
| --- |
| <table>      <tr>       <td>序号</td>          <td>姓名</td>          <td>年龄</td>          <td>钱包</td>      </tr>      <#list stus as stu>          <tr>              <td>${stu\_index + 1}</td>              <td>${stu.name}</td>              <td>${stu.age}</td>              <td>${stu.mondy}</td>          </tr>      </#list>  </table> |

3、输出：

Hello 黑马程序员! 序号 姓名 年龄 钱包 1 小明 18 1,000.86 2 小红 19 200.1

|  |
| --- |
| 说明：  \_index：得到循环的下标，使用方法是在stu后边加"\_index"，它的值是从0开始 |

#### 遍历Map数据

1、数据模型

使用map指令遍历数据模型中的stuMap。

2、模板

|  |
| --- |
| 输出stu1的学生信息：<br/>  姓名：${stuMap['stu1'].name}<br/>  年龄：${stuMap['stu1'].age}<br/>  输出stu1的学生信息：<br/>  姓名：${stuMap.stu1.name}<br/>  年龄：${stuMap.stu1.age}<br/>  遍历输出两个学生信息：<br/>  <table>      <tr>          <td>序号</td>          <td>姓名</td>          <td>年龄</td>          <td>钱包</td>      </tr>  <#list stuMap?keys as k>  <tr>      <td>${k\_index + 1}</td>      <td>${stuMap[k].name}</td>      <td>${stuMap[k].age}</td>      <td >${stuMap[k].mondy}</td>  </tr>  </#list>  </table> |

3 、输出

|  |
| --- |
| 输出stu1的学生信息：  姓名：小明  年龄：18  输出stu1的学生信息：  姓名：小明  年龄：18  遍历输出两个学生信息：  序号 姓名 年龄 钱包  1 小红 19 200.1  2 小明 18 1,000.86 |

#### if指令

if 指令即判断指令，是常用的FTL指令，freemarker在解析时遇到if会进行判断，条件为真则输出if中间的内容，否

则跳过内容不再输出。

1、数据模型：

使用list指令中测试数据模型。

2、模板：

|  |
| --- |
| <table>      <tr>          <td>姓名</td>          <td>年龄</td>          <td>钱包</td>  </tr>  <#list stus as stu>          <tr>              <td <#if stu.name =='小明'>style="background:red;"</#if>>${stu.name}</td>              <td>${stu.age}</td>              <td >${stu.mondy}</td>          </tr>      </#list>  </table> |

通过阅读上边的代码，实现的功能是：如果姓名为 “小明”则背景色显示为红色。

3、输出：

通过测试发现 姓名为小明的背景色为红色。

### 其它指令

#### 运算符

1、算数运算符 FreeMarker表达式中完全支持算术运算,FreeMarker支持的算术运算符包括:+, - , \* , / , % 2、逻辑

运算符 逻辑运算符有如下几个: 逻辑与:&& 逻辑或:|| 逻辑非:! 逻辑运算符只能作用于布尔值,否则将产生错误 3、

比较运算符 表达式中支持的比较运算符有如下几个: 1 =或者==:判断两个值是否相等. 2 !=:判断两个值是否不等. 3 >

或者gt:判断左边值是否大于右边值 4 >=或者gte:判断左边值是否大于等于右边值 5 <或者lt:判断左边值是否小于右

边值 6 <=或者lte:判断左边值是否小于等于右边值

注意: =和!=可以用于字符串,数值和日期来比较是否相等,但=和!=两边必须是相同类型的值,否则会产生错误,而且

FreeMarker是精确比较,"x","x ","X"是不等的.其它的运行符可以作用于数字和日期,但不能作用于字符串,大部分的时

候,使用gt等字母运算符代替>会有更好的效果,因为 FreeMarker会把>解释成FTL标签的结束字符,当然,也可以使用括

号来避免这种情况,如:<#if (x>y)>

#### 空值处理

1、判断某变量是否存在使用 “??” 用法为:variable??,如果该变量存在,返回true,否则返回false

例：为防止stus为空报错可以加上判断如下：

|  |
| --- |
| <#if stus??>      <#list stus as stu>       ......      </#list>      </#if> |

2、缺失变量默认值使用 “!” 使用!要以指定一个默认值，当变量为空时显示默认值。

例： ${name!''}表示如果name为空显示空字符串。

如果是嵌套对象则建议使用（）括起来。

例： ${(stu.bestFriend.name)!''}表示，如果stu或bestFriend或name为空默认显示空字符串。

#### 内建函数

内建函数语法格式： 变量+?+函数名称

1、和到某个集合的大小

${集合名?size}

2、日期格式化

|  |
| --- |
| 显示年月日: ${today?date}  显示时分秒：${today?time}  显示日期+时间：${today?datetime} <br>  自定义格式化：  ${today?string("yyyy年MM月")} |

3、内建函数c

map.put("point", 102920122);

point是数字型，使用${point}会显示这个数字的值，不并每三位使用逗号分隔。

如果不想显示为每三位分隔的数字，可以使用c函数将数字型转成字符串输出

${point?c}

4、将json字符串转成对象

一个例子：

其中用到了 assign标签，assign的作用是定义一个变量。

|  |
| --- |
| <#assign text="{'bank':'工商银行','account':'10101920201920212'}" />  <#assign data=text?eval />  开户行：${data.bank}  账号：${data.account} |

#### 完整的模板

上边测试的模板内容如下，可自行进行对照测试。

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>      <meta charset="utf‐8">      <title>Hello World!</title>  </head>  <body>  Hello ${name}!  <br/>  <table>      <tr>          <td>序号</td>          <td>姓名</td>          <td>年龄</td>          <td>钱包</td>      </tr>      <#list stus as stu>          <tr>              <td>${stu\_index + 1}</td>              <td <#if stu.name =='小明'>style="background:red;"</#if>>${stu.name}</td>              <td>${stu.age}</td>              <td >${stu.mondy}</td>          </tr>      </#list>  </table>  <br/><br/>  输出stu1的学生信息：<br/>  姓名：${stuMap['stu1'].name}<br/>  年龄：${stuMap['stu1'].age}<br/>  输出stu1的学生信息：<br/>  姓名：${stu1.name}<br/>  年龄：${stu1.age}<br/>  遍历输出两个学生信息：<br/>  <table>      <tr>          <td>序号</td>          <td>姓名</td>          <td>年龄</td>          <td>钱包</td>      </tr>  <#list stuMap?keys as k>  <tr>      <td>${k\_index + 1}</td>      <td>${stuMap[k].name}</td>      <td>${stuMap[k].age}</td>      <td >${stuMap[k].mondy}</td>  </tr>  </#list>  </table>  </br>  <table>      <tr>          <td>姓名</td>          <td>年龄</td>          <td>出生日期</td>          <td>钱包</td>          <td>最好的朋友</td>          <td>朋友个数</td>          <td>朋友列表</td>      </tr>      <#if stus??>  <#list stus as stu>          <tr>              <td>${stu.name!''}</td>              <td>${stu.age}</td>              <td>${(stu.birthday?date)!''}</td>              <td>${stu.mondy}</td>              <td>${(stu.bestFriend.name)!''}</td>              <td>${(stu.friends?size)!0}</td>              <td>                  <#if stu.friends??>                  <#list stu.friends as firend>                      ${firend.name!''}<br/>                  </#list>                  </#if>              </td>          </tr>      </#list>      </#if>  </table>  <br/>  <#assign text="{'bank':'工商银行','account':'10101920201920212'}" />  <#assign data=text?eval />  开户行：${data.bank}  账号：${data.account}  </body>  </html> |

### 静态化测试

在cms中使用freemarker将页面生成html文件，本节测试html文件生成的方法：

1、使用模板文件静态化

定义模板文件，使用freemarker静态化程序生成html文件。

2、使用模板字符串静态化

定义模板字符串，使用freemarker静态化程序生成html文件。

#### 使用模板文件静态化

在test下创建测试类，并且将main下的resource/templates拷贝到test下，本次测试使用之前我们在main下创建的模板文件。

|  |
| --- |
| package com.xuecheng.test.freemarker;  import com.xuecheng.test.freemarker.model.Student;  import freemarker.cache.StringTemplateLoader;  import freemarker.template.Configuration;  import freemarker.template.Template;  import freemarker.template.TemplateException;  import org.apache.commons.io.IOUtils;  import org.junit.Test;  import org.junit.runner.RunWith;  import org.springframework.boot.test.context.SpringBootTest;  import org.springframework.test.context.junit4.SpringRunner;  import org.springframework.ui.freemarker.FreeMarkerTemplateUtils;  import java.io.File;  import java.io.FileOutputStream;  import java.io.IOException;  import java.io.InputStream;  import java.util.\*;  /\*\*  \* @ClassName FreemarkerTest  \* @Description TODO  \* @Author yaosiyuan  \* @Date 2019/4/5 9:25  \* @Version 1.0  \*\*/  @SpringBootTest  @RunWith(SpringRunner.class)  public class FreemarkerTest {  //测试静态化，基于ftl模板生成html文件  @Test  public void testGenerateHtml() throws IOException, TemplateException {  //定义配置类  Configuration configuration = new Configuration(Configuration.getVersion());  //得到classpath的路径  String classpath = this.getClass().getResource("/").getPath();  //定义模板路径  configuration.setDirectoryForTemplateLoading(new File(classpath+"/templates/"));  //获取模板文件的内容  Template template = configuration.getTemplate("test1.ftl");  //定义数据模型  Map map = getMap();  //静态化  String string = FreeMarkerTemplateUtils.processTemplateIntoString(template, map);  InputStream inputStream = IOUtils.toInputStream(string);  FileOutputStream outputStream = new FileOutputStream(new File("C:/Users/yaosiyuan/Desktop/java"));  //输出文件  IOUtils.copy(inputStream,outputStream);  inputStream.close();  outputStream.close();  }  //数据模型  private Map getMap(){  Map<String, Object> map = new HashMap<>();  //向数据模型放数据  map.put("name","黑马程序员");  Student stu1 = new Student();  stu1.setName("小明");  stu1.setAge(18);  stu1.setMoney(1000.86f);  stu1.setBirthday(new Date());  Student stu2 = new Student();  stu2.setName("小红");  stu2.setMoney(200.1f);  stu2.setAge(19);  // stu2.setBirthday(new Date());  List<Student> friends = new ArrayList<>();  friends.add(stu1);  stu2.setFriends(friends);  stu2.setBestFriend(stu1);  List<Student> stus = new ArrayList<>();  stus.add(stu1);  stus.add(stu2);  //向数据模型放数据  map.put("stus",stus);  //准备map数据  HashMap<String,Student> stuMap = new HashMap<>();  stuMap.put("stu1",stu1);  stuMap.put("stu2",stu2);  //向数据模型放数据  map.put("stu1",stu1);  //向数据模型放数据  map.put("stuMap",stuMap);  return map;  }  //基于模板字符串生成静态化文件  @Test  public void testGenerateHtmlByString() throws IOException, TemplateException {  //创建配置类  Configuration configuration=new Configuration(Configuration.getVersion());  //获取模板内容  //模板内容，这里测试时使用简单的字符串作为模板  String templateString="" +  "<html>\n" +  " <head></head>\n" +  " <body>\n" +  " 名称：${name}\n" +  " </body>\n" +  "</html>";  //加载模板  //模板加载器  StringTemplateLoader stringTemplateLoader = new StringTemplateLoader();  stringTemplateLoader.putTemplate("template",templateString);  configuration.setTemplateLoader(stringTemplateLoader);  Template template = configuration.getTemplate("template","utf-8");  //数据模型  Map map = getMap();  //静态化  String content = FreeMarkerTemplateUtils.processTemplateIntoString(template, map);  //静态化内容  System.out.println(content);  InputStream inputStream = IOUtils.toInputStream(content);  //输出文件  FileOutputStream fileOutputStream = new FileOutputStream(new File("C:/Users/yaosiyuan/Desktop/java/1.html"));  IOUtils.copy(inputStream, fileOutputStream);  }  } |

### 语法

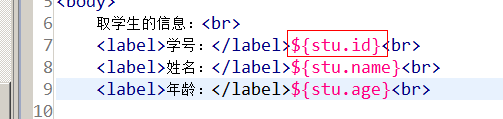
#### 访问map中的key

${key}

#### 访问pojo中的属性

Student对象。学号、姓名、年龄

${key.property}

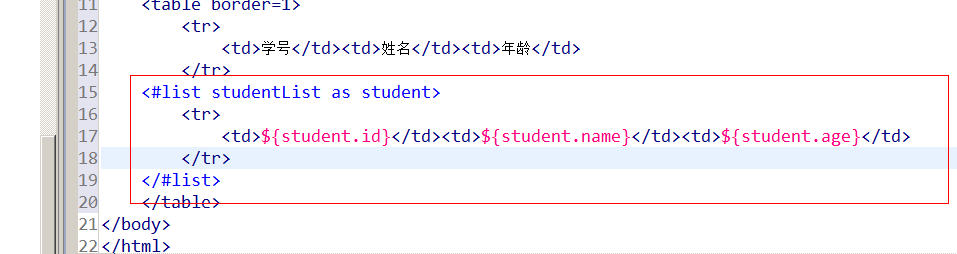


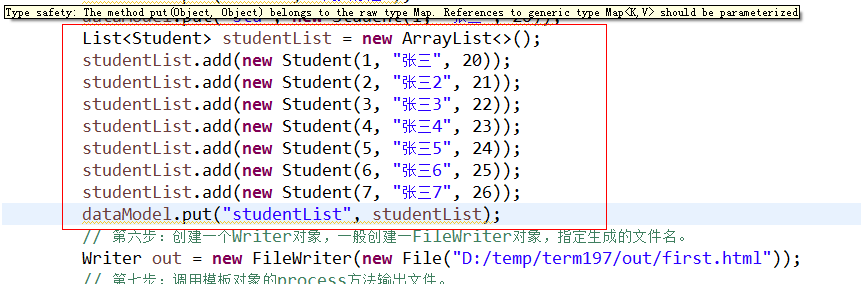
#### 取集合中的数据

<#list studentList as student>

${student.id}/${studnet.name}

</#list>



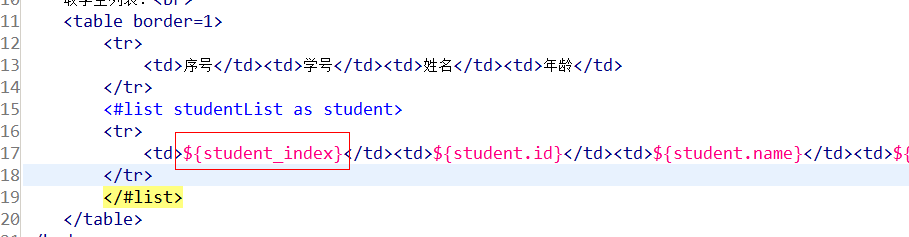


#### 取循环中的下标

<#list studentList as student>

${student\_index}

</#list>

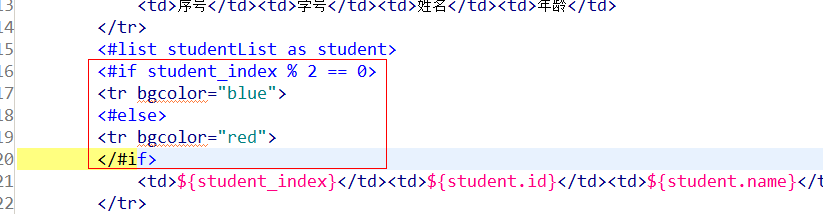


#### 判断

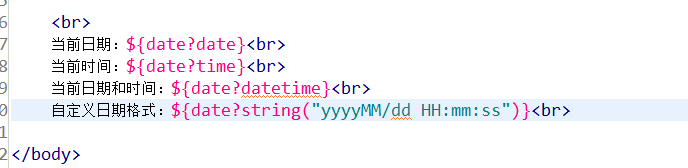
<#if student\_index % 2 == 0>

<#else>

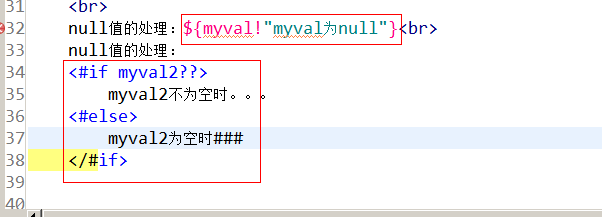
</#if>



#### 日期类型格式化

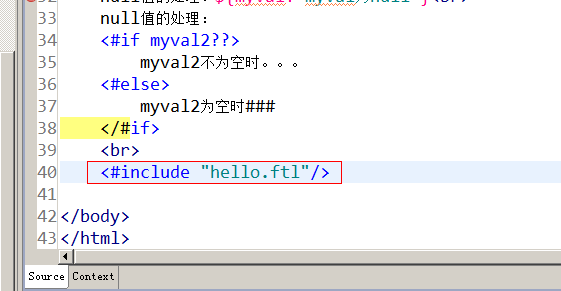


#### Null值的处理



#### Include标签

<#include “模板名称”>



/e3-item-web/src/main/webapp/WEB-INF/ftl/student.ftl

|  |
| --- |
| <html>  <head>  <title>student</title>  </head>  <body>  学生信息：<br>  学号：${student.id}&nbsp;&nbsp;&nbsp;&nbsp;  姓名：${student.name}&nbsp;&nbsp;&nbsp;&nbsp;  年龄：${student.age}&nbsp;&nbsp;&nbsp;&nbsp;  家庭住址：${student.address}<br>  学生列表：  <table border="1">  <tr>  <th>序号</th>  <th>学号</th>  <th>姓名</th>  <th>年龄</th>  <th>家庭住址</th>  </tr>  <#list stuList as stu>  <#if stu\_index % 2 == 0>  <tr bgcolor="red">  <#else>  <tr bgcolor="green">  </#if>  <td>${stu\_index}</td>  <td>${stu.id}</td>  <td>${stu.name}</td>  <td>${stu.age}</td>  <td>${stu.address}</td>  </tr>  </#list>  </table>  <br>  <!-- 可以使用?date,?time,?datetime,?string(parten)-->    当前日期：${date?string("yyyy/MM/dd HH:mm:ss")}<br>    null值的处理：${val!"val的值为null"}<br>  判断val的值是否为null：<br>  <#if val??>  val中有内容  <#else>  val的值为null  </#if>  引用模板测试：<br>  <#include "hello.ftl">  </body>  </html> |

/e3-item-web/src/test/java/cn/e3mall/freemarker/FreeMarkerTest.java

|  |
| --- |
| package cn.e3mall.freemarker;  import java.io.File;  import java.io.FileWriter;  import java.io.Writer;  import java.util.ArrayList;  import java.util.Date;  import java.util.HashMap;  import java.util.List;  import java.util.Map;  import org.junit.Test;  import freemarker.template.Configuration;  import freemarker.template.Template;  public class FreeMarkerTest {  @Test  public void testFreeMarker() throws Exception {  // 创建一个模板文件  // 创建一个Configuration对象  Configuration configuration = new Configuration(Configuration.getVersion());  // 设置模板文件的保存目录  configuration.setDirectoryForTemplateLoading(  new File("C:/Users/yaosiyuan/workspace/e3-item-web/src/main/webapp/WEB-INF/ftl"));  // 模板文件的编码格式，一般是utf8  configuration.setDefaultEncoding("utf-8");  // 加载一个模板文件，创建一个模板对象  // Template template = configuration.getTemplate("hello.ftl");  Template template = configuration.getTemplate("student.ftl");  // 创建一个数据集，可以是pojo，也可以是map，推荐使用map  Map data = new HashMap<>();  data.put("hello", "hello freemarker!");  // 创建一个Pojo对象  Student student = new Student(1, "小明", 15, "漯河");  data.put("student", student);  // 添加一个list  List<Student> stuList = new ArrayList<>();  stuList.add(new Student(1, "小明", 11, "漯河"));  stuList.add(new Student(1, "小明", 11, "漯河"));  stuList.add(new Student(1, "小明", 11, "漯河"));  stuList.add(new Student(1, "小明", 11, "漯河"));  stuList.add(new Student(1, "小明", 11, "漯河"));  stuList.add(new Student(1, "小明", 11, "漯河"));  stuList.add(new Student(1, "小明", 11, "漯河"));  stuList.add(new Student(1, "小明", 11, "漯河"));  stuList.add(new Student(1, "小明", 11, "漯河"));  data.put("stuList", stuList);  // 添加日期类型  data.put("date", new Date());  // 创建一个Writer对象，指定输出文件的路径和文件名  // Writer out = new FileWriter(new File("C:/temp/code/hello.txt"));  Writer out = new FileWriter(new File("C:/temp/code/student.html"));  // 生成静态页面  template.process(data, out);  // 关闭流  out.close();  }  } |

/e3-item-web/src/test/java/cn/e3mall/freemarker/Student.java

|  |
| --- |
| package cn.e3mall.freemarker;  public class Student {  private int id;  private String name;  private int age;  private String address;  public int getId() {  return id;  }  public void setId(int id) {  this.id = id;  }  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  public int getAge() {  return age;  }  public void setAge(int age) {  this.age = age;  }  public String getAddress() {  return address;  }  public void setAddress(String address) {  this.address = address;  }  public Student(int id, String name, int age, String address) {  super();  this.id = id;  this.name = name;  this.age = age;  this.address = address;  }  public Student(){}  } |