

Freemarker

[快速入门](#)

[基础语法](#)

[集合指令](#)

[IF指令](#)

[运算符](#)

[空值处理](#)

[内建函数](#)

[生成Html文件](#)

快速入门

1. 引入依赖

```
XML |
1 <dependencies>
2   <dependency>
3     <groupId>org.springframework.boot</groupId>
4     <artifactId>spring-boot-starter-web</artifactId>
5   </dependency>
6   <dependency>
7     <groupId>org.springframework.boot</groupId>
8     <artifactId>spring-boot-starter-freemarker</artifactId>
9   </dependency>
10  <dependency>
11    <groupId>org.springframework.boot</groupId>
12    <artifactId>spring-boot-starter-test</artifactId>
13  </dependency>
14  <dependency>
15    <groupId>commons-io</groupId>
16    <artifactId>commons-io</artifactId>
17    <version>2.15.0</version>
18  </dependency>
19 </dependencies>
```

2. 配置

YAML

```
1  server:
2    port: 9001 #服务端口
3  spring:
4    application:
5      name: freemarker-demo #指定服务名
6    freemarker:
7      cache: false #关闭模板缓存, 方便测试
8      settings:
9        #检查模板更新延迟时间, 设置为0表示立即检查
10       #如果时间大于0会有缓存不方便进行模板测试
11       template_update_delay: 0
12    suffix: .ftl #指定Freemarker模板文件的后缀名
```

3. resource/templates 中编写页面

resources

templates

01-basic.ftl

HTML

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4    <meta charset="utf-8">
5    <title>Hello World!</title>
6  </head>
7  <body>
8    <b>普通文本 String 展示: </b><br><br>
9    Hello ${name} <br>
10   <hr>
11   <b>对象Student中的数据展示: </b><br/>
12   姓名: ${student.name}<br/>
13   年龄: ${student.age}
14   <hr>
15 </body>
16 </html>
```

4. 编写控制器

```

1  @Controller
2  public class FreemarkerController {
3
4      @GetMapping("/basic")
5      public String test(Model model){
6          model.addAttribute("name","freemarker");
7          Student student = new Student();
8          student.setAge(22);
9          student.setName("fzdkx");
10         student.setBirthday(new Date());
11         student.setMoney(1000000.0f);
12         model.addAttribute("student",student);
13         return "01-basic";
14     }
15 }

```

5. 访问测试



基础语法

1. 注释 `<#-- -->`

```
1  <#--我是一个freemarker注释-->
```

2. 插值 `${xxx}`

▼ Tcl |

```
1 ▾ Hello ${name}
```

3. 普通文本

▼ Tcl |

```
1 我是一个普通的文本
```

4. FTL指令 `<# >xxx</#>`

▼ Tcl |

```
1 <# >FTL指令</#>
```

集合指令

集合指令可以用来 获取 、 遍历 `Map` , `List` 中的数据

1. 获取数据

```
//创建Map, 存放数据
HashMap<String,Student> stuMap = new HashMap<>();
stuMap.put("stu1",stu1);
stuMap.put("stu2",stu2);
// 3.1 向model中存放Map数据
model.addAttribute( attributeName: "stuMap", stuMap);
```

▼ HTML |

```
1 <!-- 方式一 -->
2 输出stu1的学生信息: <br/>
3 姓名: ${stuMap['stu1'].name}<br/>
4 年龄: ${stuMap['stu1'].age}<br/>
5
6 <!-- 方式二 -->
7 输出stu2的学生信息: <br/>
8 姓名: ${stuMap.stu2.name}<br/>
9 年龄: ${stuMap.stu2.age}<br/>
```

2. 遍历 `Map`

```

1  <table>
2      <tr>
3          <td>序号</td>
4          <td>姓名</td>
5          <td>年龄</td>
6          <td>钱包</td>
7      </tr>
8      <#list stuMap?keys as key>
9          <tr>
10             <!-- _index获取下标, 从0开始-->
11             <td>${key_index}</td>
12             <td>${stuMap[key].name}</td>
13             <td>${stuMap[key].age}</td>
14             <td>${stuMap[key].money}</td>
15         </tr>
16     </#list>
17 </table>

```

序号 姓名 年龄 钱包

0 小红 19 200.1

1 小强 18 1,000.86

3. 遍历 List

```

1  <tr>
2      <td>序号</td>
3      <td>姓名</td>
4      <td>年龄</td>
5      <td>钱包</td>
6  </tr>
7  <#list stus as stu>
8      <tr>
9          <!--获取下标, 从0开始-->
10         <td>${stu_index}</td>
11         <td>${stu.name}</td>
12         <td>${stu.age}</td>
13         <td>${stu.money}</td>
14     </tr>
15 </#list>

```

序号 姓名 年龄 钱包

0 小强 18 1,000.86

1 小红 19 200.1

IF指令

▼ Velocity |

```
1 <#if stu.name='小红'>
2     ..... if命中逻辑
3 <#elseif stu.name= '小蓝'>
4     ..... elseif命中逻辑
5 <#else >
6     ..... else命中逻辑
7 </#if>
```

```

1  <table>
2      <tr>
3          <td>序号</td>
4          <td>姓名</td>
5          <td>年龄</td>
6          <td>钱包</td>
7      </tr>
8      <#list stus as stu >
9          <#if stu.name='小红'>
10             <tr style="color: red">
11                 <td>${stu_index}</td>
12                 <td>${stu.name}</td>
13                 <td>${stu.age}</td>
14                 <td>${stu.money}</td>
15             </tr>
16             <#elseif stu.name= '小蓝'>
17                 <tr style="color: blue">
18                     <td>${stu_index}</td>
19                     <td>${stu.name}</td>
20                     <td>${stu.age}</td>
21                     <td>${stu.money}</td>
22                 </tr>
23             <#else>
24                 <tr>
25                     <td>${stu_index}</td>
26                     <td>${stu.name}</td>
27                     <td>${stu.age}</td>
28                     <td>${stu.money}</td>
29                 </tr>
30             </#if>
31         </#list>
32     </table>

```

序号 姓名 年龄 钱包

0 小蓝 18 1,000.86

1 小红 19 200.1

2 fzdqx 22 2,000,000.125

运算符

- 算术运算符

`+` , `-` , `*` , `/` , `%`

- `+` 可以用于字符串拼接

- 比较运算符

`=` 或者 `==` : 判断两个值是否相等

- 如比较字符串, 比较时间
- 日期的比较需要通过 `?date` 函数

`!=` : 判断两个值是否不等.

`gt` : `>`

`gte` : `>=`

`lt` : `<`

`lte` : `<=`

```
Date date = new Date();
model.addAttribute(attributeName: "date1", date);
model.addAttribute(attributeName: "date2", date);
```

```
1  <#if "xiaoming" == "xiaoming">
2    字符串相等
3  </#if>
4
5  <!-- 日期的比较需要通过 ?date 将属性转为date类型才能进行比较 -->
6  <#if date1?date gte date2?date>
7    date1 > date2
8  </#if>
```

字符串相等

`date1 == date2`

- 逻辑运算符

&& , || , !

空值处理

当我们遍历的集合对象为空时，会报异常

当我们使用对象的属性为空时，会报异常

我们可以使用 `??` 来判断 集合或对象是否为空

```
▼ Velocity |
1  <#if stus??>
2    <#list stus as stu>
3
4      .....
5    </#list>
```

当对象属性为空时，我们可以赋予默认值

- 不为空使用变量值，为空使用默认值

```
▼ Velocity |
1  ${stu.name!"默认名"}
```

```

1  <table>
2      <tr>
3          <td>序号</td>
4          <td>姓名</td>
5          <td>年龄</td>
6          <td>钱包</td>
7      </tr>
8      <#if stus??>
9          <#list stus as stu >
10             <#if stu.name?? && stu.name='小红'>
11                 <tr style="color: red">
12                     <td>${stu_index}</td>
13                     <td>${stu.name}</td>
14                     <td>${stu.age}</td>
15                     <td>${stu.money}</td>
16                 </tr>
17             <#elseif stu.name?? && stu.name= '小蓝'>
18                 <tr style="color: blue">
19                     <td>${stu_index}</td>
20                     <td>${stu.name}</td>
21                     <td>${stu.age}</td>
22                     <td>${stu.money}</td>
23                 </tr>
24             <#else>
25                 <tr>
26                     <td>${stu_index}</td>
27                     <td>${stu.name!"默认名"}</td>
28                     <td>${stu.age!"默认年龄"}</td>
29                     <td>${stu.money!0.0}</td>
30                 </tr>
31             </#if>
32         </#list>
33     </#if>
34 </table>

```

序号	姓名	年龄	钱包
0	小蓝	18	1,000.86
1	小红	19	200.1
2	默认名	0	0

内建函数: `?functionName`

- 集合大小

▼ Velocity |

```
1 1. ${集合名?size}
```

- 日期格式化

▼ Velocity |

```
1 1. 显示年月日      ${today?date}
2 2. 显示时分秒      ${today?time}
3 3. 显示日期+时间    ${today?datetime}
4 4. 自定义格式化    ${today?string("yyyy年MM月")}
```

- `c` 函数

数字类型, 默认 3个数字 隔一个顿号

如果想要去掉顿号, 可以使用 `c` 函数

▼ Velocity |

```
1 1. ## 原始number 显示为 123,321,456,654
2
3 3. ${number?c}
4 4. ## 使用c函数后 显示为 123321456654
```

- `json` 串转对象: `eval`

▼ Velocity |

```
1 1. ## assign标签作用 定义一个变量, 可以在 其他地方/标签中 使用该变量
2 2. <#assign text="{ 'bank': '工商银行', 'account': '10101920201920212' }" />
```

▼ Velocity |

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

生成Html文件

1. 添加模板存放的配置信息

YAML |

```
1  server:
2    port: 9001 #服务端口
3  spring:
4    application:
5      name: freemarker-demo #指定服务名
6    freemarker:
7      cache: false #关闭模板缓存，方便测试
8      settings:
9        #检查模板更新延迟时间，
10       # 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
11       template_update_delay: 0
12     suffix: .ftl #指定Freemarker模板文件的后缀名
13     template-loader-path: classpath:templates # 模板存放位置，默认
```

2. 调用 API 生成 HTML 文件

1. 获取模板
2. 构建数据
3. 填充模板，指定 html 生成位置

```
1  @SpringBootTest(classes = FreemarkerApplication.class)
2  @RunWith(SpringRunner.class)
3  public class freemarkerTest {
4      @Autowired
5      private Configuration configuration;
6
7      @Test
8      public void test() throws IOException, TemplateException {
9          // 从模板路径下获取模板
10         Template template = configuration.getTemplate("02-list.ftl");
11
12         // 构建数据
13         Map<String, Object> data = getData();
14
15         // 填充模板, 指定html生成位置
16         template.process(data, new FileWriter("d:/list.html"));
17     }
18
19     public Map<String, Object> getData(){
20         HashMap<String, Object> map = new HashMap<>();
21
22         Student stu1 = new Student();
23         stu1.setName("小蓝");
24         stu1.setAge(18);
25         stu1.setMoney(1000.86f);
26         stu1.setBirthday(new Date());
27
28         //小红对象模型数据
29         Student stu2 = new Student();
30         stu2.setName("小红");
31         stu2.setMoney(200.1f);
32         stu2.setAge(19);
33
34         //小红对象模型数据
35         Student stu3 = new Student();
36         // stu3属性为null
37
38         //将两个对象模型数据存放到List集合中
39         List<Student> stus = new ArrayList<>();
40         stus.add(stu1);
41         stus.add(stu2);
42         stus.add(stu3);
43
44         //向model中存放List集合数据
45         map.put("stus", stus);
```

```
46
47 //-----
48
49 //创建Map, 存放数据
50 HashMap<String,Student> stuMap = new HashMap<>();
51 stuMap.put("stu1",stu1);
52 stuMap.put("stu2",stu2);
53 // 3.1 向model中存放Map数据
54 map.put("stuMap", stuMap);
55
56 Date date = new Date();
57 map.put("date1",date);
58 map.put("date2",date);
59
60 return map;
61 }
62 }
```

此电脑 > data (D:) >

名称

修改日期

 list.html

2024/2/5 10:53

- 直接打开 `html` 文件

展示list中的stu数据:

序号 姓名 年龄 钱包

map数据的展示:

方式一: 通过map['keyname'].property.

输出stu1的学生信息:

姓名: 小蓝

年龄: 18

方式二: 通过map.keyname.property.

输出stu2的学生信息:

姓名: 小红

年龄: 19

遍历map中两个学生信息:

序号 姓名 年龄 钱包

0 小红 19 200.1

1 小蓝 18 1,000.86

序号 姓名 年龄 钱包

0 小蓝 18 1,000.86

1 小红 19 200.1

2 默认名 0 0

字符串相等

date1 == date2
