# 第3章：SpringBoot整合视图层

## 本章大纲

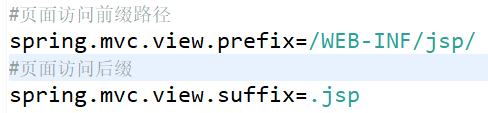
* SpringBoot整合JSP
* SpringBoot整合freemarker
* SpringBoot整合Thymeleaf

## SpringBoot整合JSP

### 在pom.xml文件添加jsp相关jar依赖

|  |
| --- |
| <!-- jstl -->  <dependency>  <groupId>javax.servlet</groupId>  <artifactId>jstl</artifactId>  </dependency>  <!-- jasper -->  <dependency>  <groupId>org.apache.tomcat.embed</groupId>  <artifactId>tomcat-embed-jasper</artifactId>  <scope>provided</scope>  </dependency> |

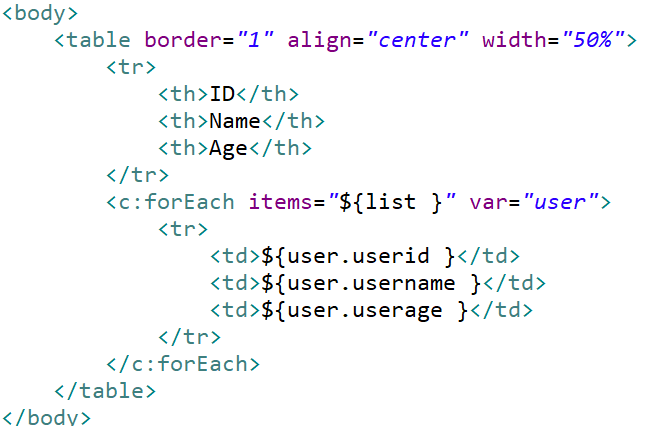
### 编写application.properties文件



### 创建Controller

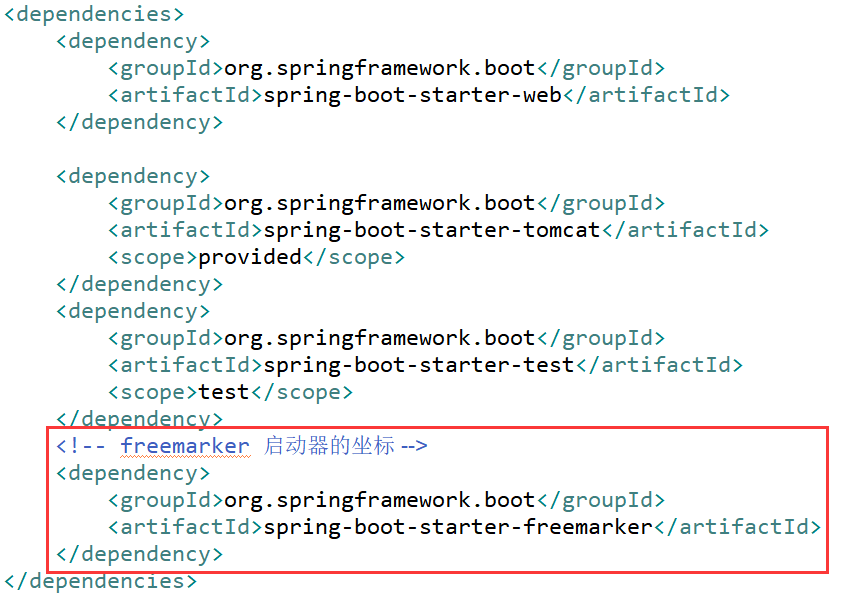


### 创建jsp页面

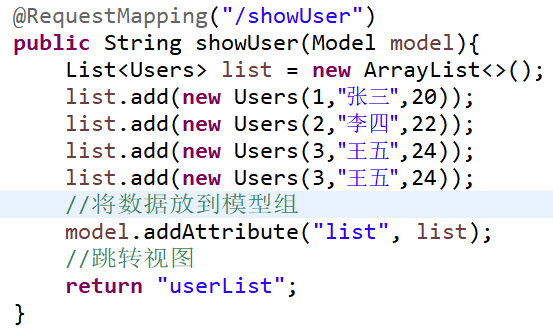


## SpringBoot整合freemarker

### 在pom.xml文件添加freemarker相关jar依赖



### 创建Controller



### 创建freemarker模板页面

注意：springBoot 要求模板形式的视图层技术的文件必须要放到 src/main/resources 目录下必

须要一个名称为 templates

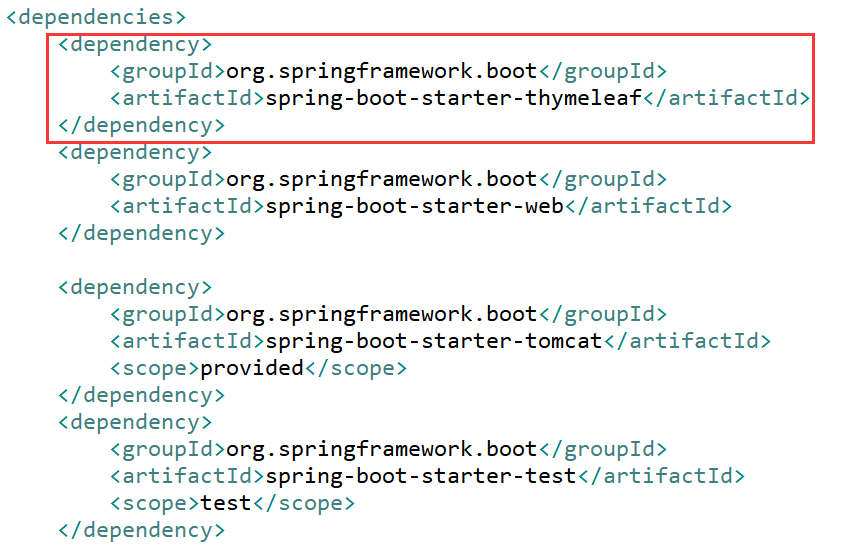
Freemarker文件后缀名是.ftl



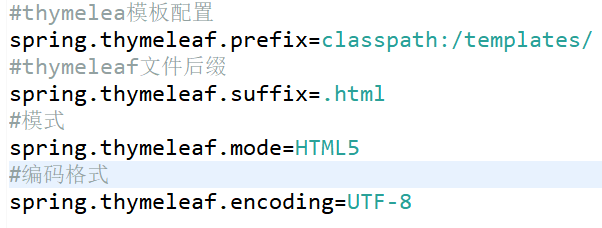
## SpringBoot整合Thymeleaf

### 创建Thymeleaf入门项目

#### 在pom.xml文件添加thymeleaf依赖



#### 编写application.properties文件



#### Thymeleaf视图存放目录

目录位置：src/main/resources/templates

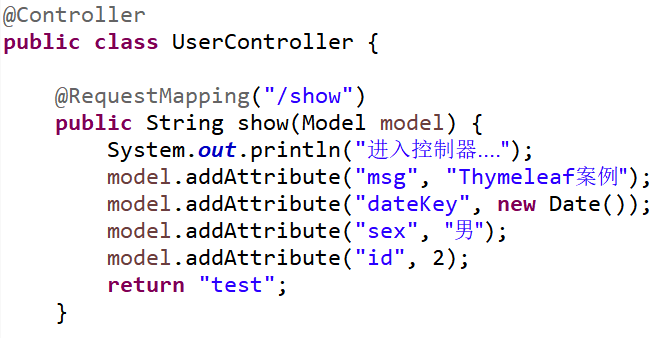
templates：该目录是安全的。意味着该目录下的内容是不允许外界直接访问的。

### Thymeleaf的基本使用

#### Thymeleaf特点

Thymelaef 是通过特定语法对 html 的标记做渲染

#### 创建Controller



#### 创建视图html



#### 注意thymeleaf版本

注意thymeleaf版本问题，如果版本是3.0以下，则需要在pom.xml文件加上以下配置：

|  |
| --- |
| <properties>  <java.version>1.8</java.version>  <thymeleaf.version>3.0.9.RELEASE</thymeleaf.version>  <thymeleaf-layout-dialect.version>2.0.4</thymeleaf-layout-dialect.version>  </properties> |

建议使用3.0以上版本

### Thymeleaf语法

#### 变量输出与字符串操作

##### th:text

作用：在页面中输出值



##### th:value

作用：可以将一个值放入到 input 标签的 value 中



#### 判断字符串是否为空

Thymeleaf 内置对象

注意语法：

1，调用内置对象一定要用#

2，大部分的内置对象都以 s 结尾 strings、numbers、dates

|  |
| --- |
| ${#strings.isEmpty(key)} |
| 判断字符串是否为空，如果为空返回 true，否则返回 false |
| ${#strings.contains(msg,'T')} |
| 判断字符串是否包含指定的字符串，如果包含返回 true，否则返回 false |
| ${#strings.startsWith(msg,'a')} |
| 判断当前字符串是否以字符串开头，如果是返回 true，否则返回 false |
| ${#strings.endsWith(msg,'a')} |
| 判断当前字符串是否以字符串结尾，如果是返回 true，否则返回 false |
| ${#strings.length(msg)} |
| 返回字符串的长度 |
| ${#strings.indexOf(msg,'h')} |
| 查找字符串的位置，并返回该字符串的下标，如果没找到则返回-1 |
| ${#strings.substring(msg,13)}  ${#strings.substring(msg,13,15)} |
| 截取字符串，用法与String类下 SubString() 方法相同 |
| ${#strings.toUpperCase(msg)}  ${#strings.toLowerCase(msg)} |
| 字符串转大小写。 |

#### 日期格式处理

|  |
| --- |
| ${#dates.format(key)} |
| 格式化日期，默认的以浏览器默认语言为格式化标准 |
| ${#dates.format(key,'yyy/MM/dd')} |
| 按照自定义的格式做日期转换 |
| ${#dates.year(key)}  ${#dates.month(key)}  ${#dates.day(key)} |
| year：取年  Month：取月  Day：取日 |

#### 条件判断

##### th:if

|  |
| --- |
| <span th:if="${sex} == ' 男 '">  性别：男  </span>  <span th:if="${sex} == ' 女 '">  性别：女  </span> |

##### th:switch

|  |
| --- |
|  |

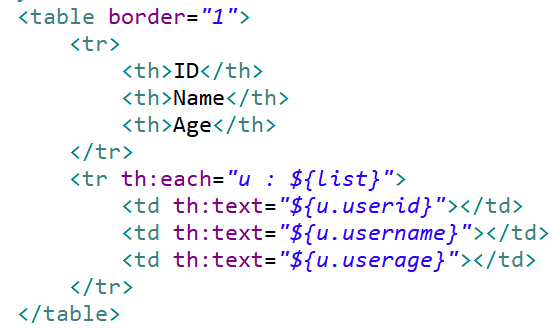
#### 迭代循环

##### th:each

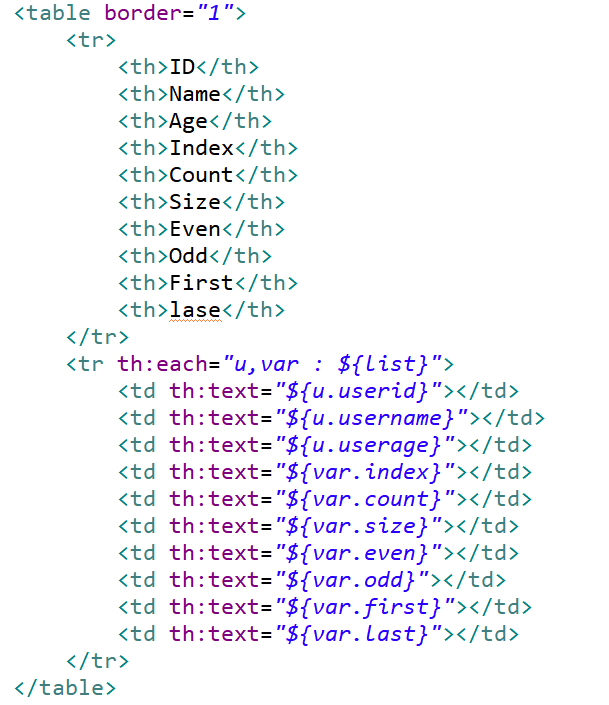
###### 控制器



###### 页面



##### th:each状态变量



状态变量属性

1,index:当前迭代器的索引 从 0 开始

2,count:当前迭代对象的计数 从 1 开始

3,size:被迭代对象的长度

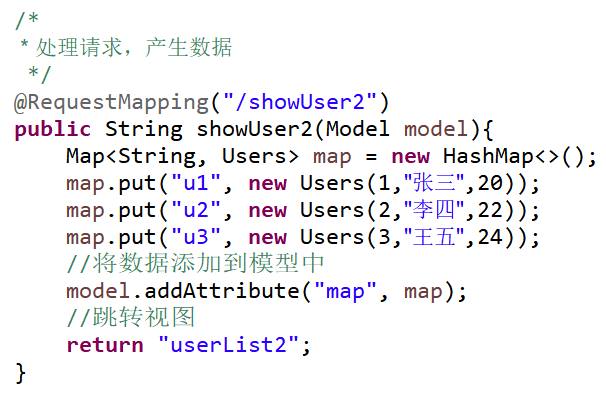
4,even/odd:布尔值，当前循环是否是偶数/奇数 从 0 开始

5,first:布尔值，当前循环的是否是第一条，如果是返回 true 否则返回 false

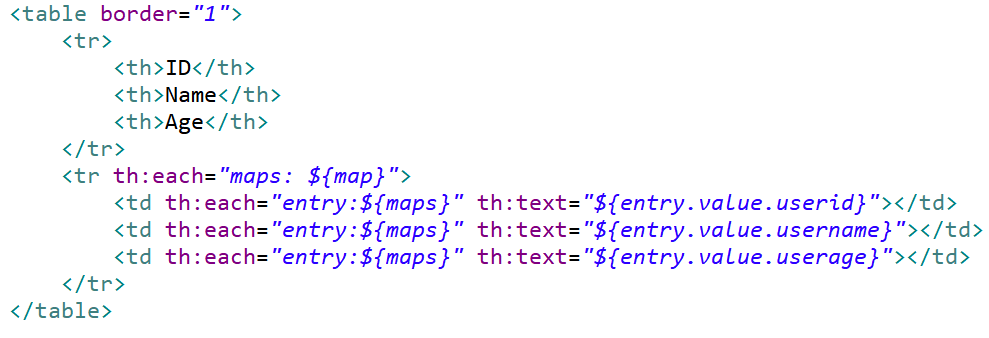
6,last:布尔值，当前循环的是否是最后一条，如果是则返回 true 否则返回 false

##### th:each迭代map

###### 控制器



###### 页面



#### 域对象操作

##### HttpServletRequest

###### 控制器



###### 页面



##### HttpSession

###### 控制器



###### 页面



##### ServletContext

###### 控制器



###### 页面



#### URL表达式

th:href

th:src

##### url表达式语法

基本语法：@{}

##### url类型

###### 绝对路径

|  |
| --- |
| <a th:href=*"@{http://www.baidu.com}"*>绝对路径</a> |

###### 相对路径

1)相对于当前项目的根

相对于项目的上下文的相对路径

|  |
| --- |
| <a th:href=*"@{/attr}"*>相对路径</a> |

2) 相对于服务器路径的根

|  |
| --- |
| <a th:href=*"@{~/project2/resourcename}"*>相对于服务器的根</a> |

#### 在url中实现参数传递

|  |
| --- |
| <a th:href=*"@{/attr(id=1,name=zhagnsan)}"*>相对路径-传参</a> |

#### 在url中通过restful风格进行参数传递

|  |
| --- |
| <a th:href=*"@{/path/{id}/show(id=1,name=zhagnsan)}"*> 相对路径 - 传参-restful</a> |