

2018级计算机科学与技术毕业实习实施方案

云客服背景、技术特点，项目系统架构，功能介绍

Thymeleaf模版引擎使用

[Thymeleaf](#)开发传统Java WEB工程时，我们可以使用JSP页面模板语言，但是在SpringBoot中已经不推荐使用了。SpringBoot支持如下页面模板语言

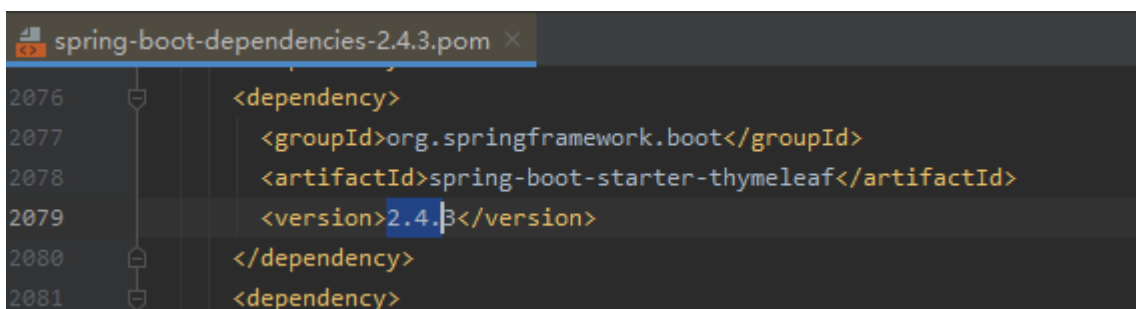
- Thymeleaf
- FreeMarker
- Velocity
- Groovy
- JSP

其中Thymeleaf是SpringBoot官方所推荐使用的，Thymeleaf是动静分离的，页面中的动态标签是需要传递有数据的时候才会渲染，不然就是原本默认的静态的样子

添加Thymeleaf依赖

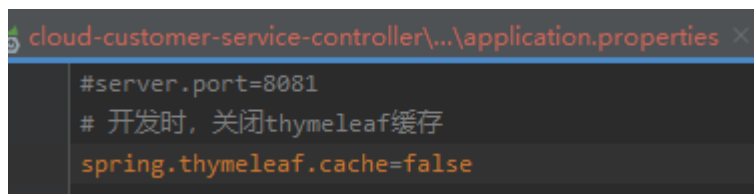
```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>
```

无需指定版本号，spring-boot-dependencies.pom中已指定



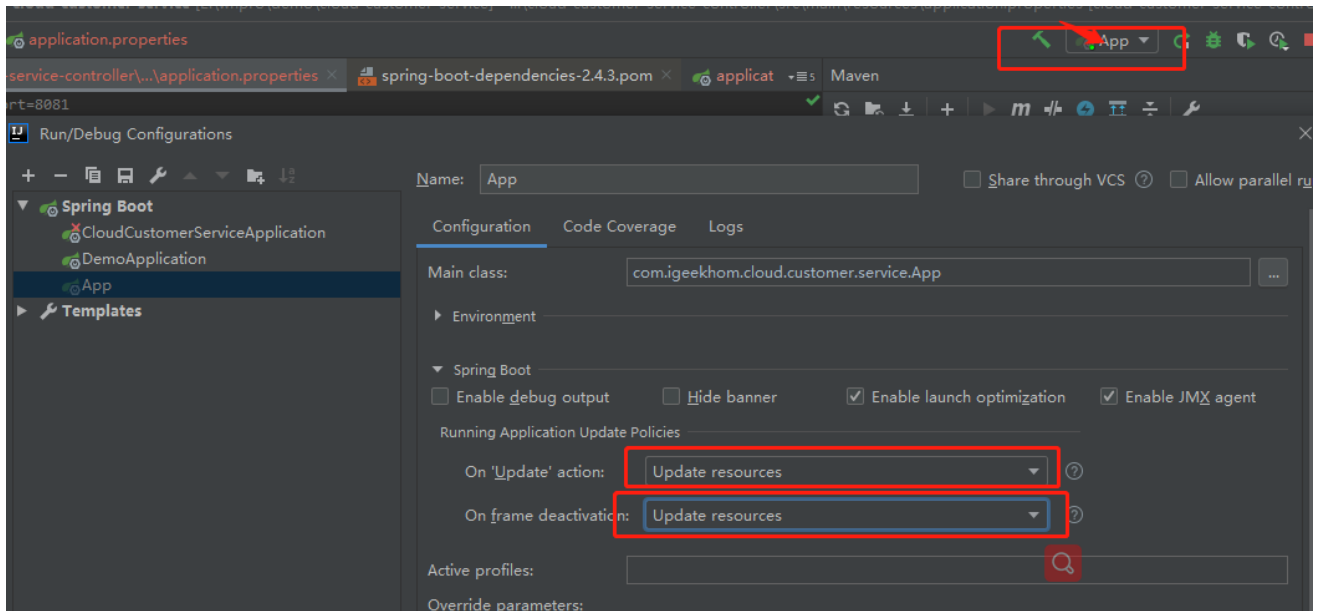
```
spring-boot-dependencies-2.4.3.pom
2076 <dependency>
2077   <groupId>org.springframework.boot</groupId>
2078   <artifactId>spring-boot-starter-thymeleaf</artifactId>
2079   <version>2.4.3</version>
2080 </dependency>
2081 <dependency>
```

开发时，关闭缓存

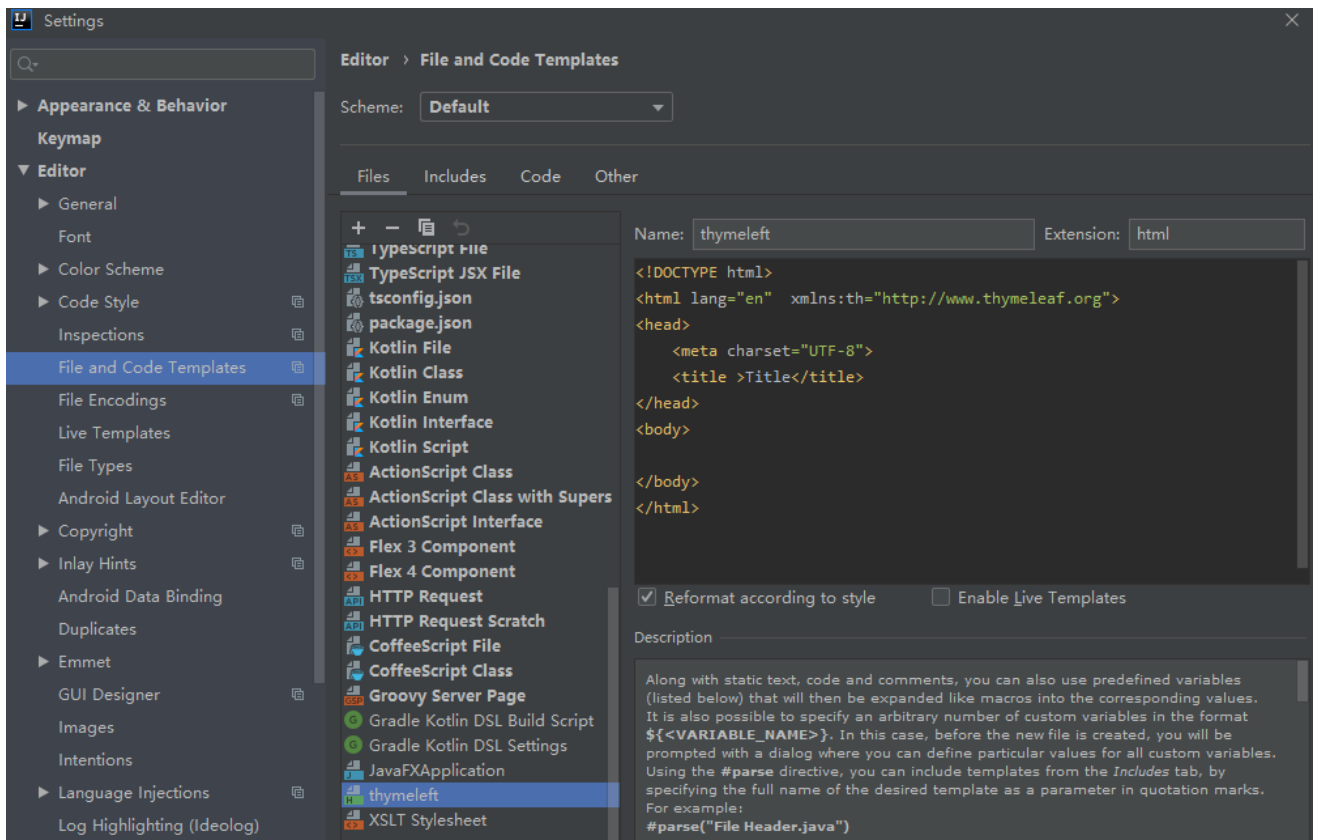


```
cloud-customer-service-controller\...\application.properties
#server.port=8081
# 开发时，关闭thymeleaf缓存
spring.thymeleaf.cache=false
```

开发运行时，更新资源



添加thymeleaf模板



Thymeleaf常用语法

- 简单表达式：
 - 变量表达式: `${...}`
 - 选择变量表达式: `*{...}`
 - 消息表达: `#{...}`
 - 链接URL表达式: `@{...}`
 - 片段表达式: `~{...}`
- 文字
 - 文本文字: `'one text', 'Another one!', ...`
 - 号码文字: `0, 34, 3.0, 12.3, ...`
 - 布尔文字: `true, false`
 - 空文字: `null`
 - 文字标记: `one, sometext, main, ...`
- 文字操作：
 - 字符串串联: `+`
 - 文字替换: `|The name is ${name}|`
- 算术运算：
 - 二元运算符: `+, -, *, /, %`
 - 减号 (一元运算符): `-`
- 布尔运算：
 - 二元运算符: `and, or`
 - 布尔否定 (一元运算符): `!, not`
- 比较和平等：
 - 比较: `>, <, >=, <= (gt, lt, ge, le)`
 - 等号运算符: `==, != (eq, ne)`
- 条件运算符：
 - 如果-则: `(if) ? (then)`
 - 如果-则-否则: `(if) ? (then) : (else)`
 - 默认: `(value) ?: (defaultvalue)`

`<p>[[${abc+'123'}]]</p>` 将控制器传入的数据显示在`<p></p>`中

使用 `th:content`, `th:text` 标签来渲染页面描述和页面关键字及标题

```
<head>
  <meta charset="UTF-8">
  <title th:text="${title}">默认的标题</title>
  <meta name="description" th:content="${description}">
  <meta name="keywords" th:content="${keywords}">
</head>
```

`${title}` 可以使用 `||` 来包裹

```
<title th:text="|wx-${title}|">默认的标题</title>
```

渲染User这个对象的信息我们可以这样

```
<div>
  <h2 th:text="${user.getUsername()}"></h2>
  <p th:text="${user.getAge()}"></p>
</div>
```

也可以将User定义为临时变量，接着使用 `*{xxx}` 就能取到值了

```
<div th:object="${user}">
  <h2 th:text="*{username}"></h2>
  <p th:text="*{age}"></p>
</div>
```

还可以不使用get的方式，直接使用属性名

```
<h2 th:text="${user.username}" ></h2>
```

th:if

`th:if` 通过布尔值决定这个元素是否渲染

比如：

```
<p th:if="${user.isVIP}">会员</p>
```

th:each

`th:each` 可以迭代循环出数据，前面我们User对象里面的tags是一个数组，我们来渲染一下

```
<ul>
  <li th:each="tag:${user.getTags()}"
      th:text="${tag}"></li>
</ul>
```

状态变量在 `th:each` 属性中定义，并且包含以下数据：

- 当前的迭代索引，从0开始。这是 `index` 属性。
- 从1开始的当前迭代索引。这是 `count` 属性。
- 迭代变量中元素的总数。这是 `size` 属性。
- 每次迭代的 *iter* 变量。这是 `current` 属性。
- 当前迭代是偶数还是奇数。这些是 `even/odd` 布尔属性。
- 当前迭代是否是第一个。这是 `first` 布尔属性。
- 当前迭代是否为最后一次。这是 `last` 布尔属性。

th:switch

`th:switch` 选择语句

```
<div th:switch="${user.getSex()}">
  <p th:case="'1'">男</p>
  <p th:case="'2'">女</p>
  <p th:case="*">默认</p>
</div>
```

url

如果在springboot中需要引入static目录下的静态资源可以使用 `@{xxx}` 的方式

```
<link th:href="@{/app.css}" rel="stylesheet">
```

JavaScript动态渲染

```
<script th:inline="javascript">
  const user = /*[[${customer}]]*/ {};
  console.log(user);
</script>
```

同理css也是可以的

```
<style th:inline="css">
  .isSelect {
    color: /*[[${selectColor}]]*/red ;
  }
</style>
```

碎片（组件）

日常开发中呢我们经常将有些可以复用的部分抽离出来

新建一个fragment.html，一个文件里面可以写多个碎片，使用 `th:fragment`` 来定义

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">

<footer th:fragment="com1">
    this is com1
</footer>

<footer th:fragment="com2">
    this is com2
</footer>
```

在其它页面中引用

```
<!--replace    fragment表示fragment.html文件-->
<div th:replace="{fragment::com1}"></div>
<!--insert    fragment表示fragment.html文件-->
<div th:insert="{fragment::com2}"></div>
```

这两种方式的区别就是，replace会将新标签完全替换原本的标签，也就是说原本写 `th:replace` 属性的标签就不会渲染出来，`insert` 是往这个地方插入标签

直接通过选择器使用

对于碎片，甚至可以不定义，我们再次添加一个 碎片

```
<footer id="com2">
    this is com2
</footer>
```

然后使用它

```
<div th:insert="{fragment::#com2}"></div>
```

注释类型

在碎片里面，我们是可以使用控制传递的数据的，比如上面的User对象，但是开发工具在 `component.html` 页面中可能不能识别到User对象，我们可以打一个注释

```
<footer th:fragment="com1">
    this is com1    <!--/*@thymesVar id="customer"
type="com.igeekhom.cloud.customer.pojo.Customer"*/-->
    <p th:text="{customer.name}">p</p>
</footer>
```

组件传递参数

组件也是可以传递数据的

```
<div th:fragment="com3(message)">
  <p th:text="${message}"></p>
</div>
```

使用的时候

```
<div th:insert=~{ fragment::com3('传递数据')}></div>
```

基本对象

#ctx：上下文对象

```
${#ctx.request}
${#ctx.response}
${#ctx.session}
${#ctx.servletContext}
```

请求/会话属性

```
${session.xxx}
${application.xxx}
${#request.getAttribute('xxx')}
```

工具类

在thymeleaf里面是可以直接使用一些java的函数的，并且你可以通过传递参数的方式把一些自己写的方法传递给页面，在里面调用也是可以的

一些可以直接的使用函数

- #dates
- #calendars
- #strings
- #numbers
- #objects
- #bools
- #arrays
- #lists
- #sets
- #maps
- #aggregates

以日期格式化来举例

```
<!--日期格式化-->
```

```
<p th:text="${#dates.format(user.createTime, 'yyyy-MM-dd HH:mm')}"></p>
```