Spring Boot html

Thymeleaf 介绍

Thymeleaf 是面向 Web 和独立环境的服务端 Java 模版引擎，能够处理 HTML、XML、JavaScript、CSS 等文件。

Thymeleaf 提供了一个简洁的、高度可维护的创建模版的方式，为了实现这一目标，Thymeleaf 建立在自然模版的概念上，将其逻辑注入到模版文件中，不会影响模版设计原型，从而改善了设计的沟通，弥补了设计与开发团队之间的差距。

Thymeleaf 从设计之初遵循 HTML5 标准，如果需要，Thymeleaf 允许创建完全符合 H5 标准的模版。

Spring Boot 支持 Thymeleaf 模版，作为前端页面的模版。

```
<p th:text="${message}"></p>
```

Thymeleaf 的作用域在 HTML 标签内，类似于标签的一个属性来使用，这就是它的特点。

具体操作

- pom.xml 引入相关依赖

```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.0.7.RELEASE</version>
</parent>

<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

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

- 创建 application.yml

```yaml
server:
  port: 8181
spring:
  thymeleaf:
    prefix: classpath:/templates/
    suffix: .html
    mode: HTML5
    encoding: UTF-8
    #关闭 Thymeleaf 缓存，否则在开发的过程中修改页面不会立即生效，需要重启
    cache: false
```

- Handler

```java
package com.southwind.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
@RequestMapping("/hello")
public class HelloHandler {

    @GetMapping("/index")
    public String index(){
        return "index";
    }
}
```

通常情况下，访问 HTML 页面需要通过 Handler 的映射来访问，不能直接访问，可以通过设置，将所有的 HTML 资源保存在 `resources/static` 目录下，就可以直接访问 HTML 页面了。

```html
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    <h1>欢迎回来</h1>
    <table>
        <tr>
            <td>序号</td>
            <td>ID</td>
            <td>姓名</td>
        </tr>
        <tr th:each="user,stat:${list}">
```

```
            <td th:text="${stat.count}"></td>
            <td th:text="${user.id}"></td>
            <td th:text="${user.name}"></td>
        </tr>
    </table>
</body>
</html>
```

Thmeleaf 常用语法

- 赋值、拼接

```
@GetMapping("/index")
public String index(Model model){
    model.addAttribute("username","张三");
    return "index";
}
```

```
<div>
    <p th:text="${username}"></p>
    <span th:text="'欢迎回来,'+${username}+'!'"></span>
    <span th:text="|欢迎回来,${username}!|"></span>
</div>
```

- 条件判断

Thymeleaf 中使用 th:if 和 th:unless 属性进行条件判断，th:if 表示当条件成立时显示内容，th:unless 表示当条件不成立时显示内容。

```
@GetMapping("/index")
public String index(Model model){
    model.addAttribute("flag",true);
    return "index";
}
```

```
<span th:if="${flag == true}" th:text="if判断"></span>
<span th:unless="${flag != true}" th:text="unless判断"></span>
```

- 循环

```
@GetMapping("/index")
public String index(Model model){
    List<User> list = new ArrayList<>();
    list.add(new User(1L,"张三"));
    list.add(new User(2L,"李四"));
    list.add(new User(3L,"王五"));
```

```
        list.add(new User(1L,"张三"));
        list.add(new User(2L,"李四"));
        list.add(new User(3L,"王五"));
        list.add(new User(1L,"张三"));
        list.add(new User(2L,"李四"));
        list.add(new User(3L,"王五"));
        model.addAttribute("list",list);
        return "index";
    }
```

```
<table>
    <thead>
        <th>序号</th>
        <th>用户id</th>
        <th>用户姓名</th>
    </thead>
    <tbody>
        <tr th:each="user,stat:${list}" th:style="'background-
color:'+@{${stat.odd}?'#F2F3F2'}">
            <td th:text="${stat.count}"></td>
            <td th:text="${user.id}"></td>
            <td th:text="${user.name}"></td>
        </tr>
    </tbody>
</table>
```

stat 称作状态变量，属性有：

- index，当前迭代对象的 index （从0开始）；
- count，当前迭代对象的 count（从1开始）；
- size，当前迭代对象的大小；
- current，当前迭代对象；
- even / odd，布尔值，当前循环是否为 偶数 / 奇数（从0开始）；
- first，布尔值，当前迭代对象是否是第一个对象；
- last，布尔值，当前迭代对象是否是最后一个对象；

- URL

Thymeleaf 对于 URL 的处理是通过语法 `@{...}` 来处理的，如果需要 Thymeleaf 对 URL 进行渲染，必须使用 `th:href`、`th:src` 等属性。

```
@GetMapping("/index")
public String index(Model model){
    model.addAttribute("img","https://timgsa.baidu.com/timg?
image&quality=80&size=b9999_10000&sec=1551111898476&di=177aad5146fae204b254e
f1560dd7a73&imgtype=0&src=http%3A%2F%2Fask.qcloudimg.com%2Fhttp-
save%2Fdeveloper-news%2Fk4i98h3q7o.jpeg");
    return "index";
}
```

```
<div th:style="'background:url('+@{${img}}+');'">
    <br/>
    <br/>
    <br/>
</div>
```

- 三元运算

```
@GetMapping("/index")
public String index(Model model){
    model.addAttribute("age",22);
    return "index";
}
```

```
<input th:value="${age lt 30 ? '年轻人':'中年人'}"/>
```

- gt：great than（大于）
- ge：great equal（大于等于）
- eq：equal（等于）
- lt：less than（小于）
- le：less equal（小于等于）
- ne：not equal（不等于）

- switch

```
@GetMapping("/index")
public String index(Model model){
    model.addAttribute("gender","女");
    return "index";
}
```

```
<div th:switch="${gender}">
    <p th:case="女">女</p>
    <p th:case="男">男</p>
</div>
```

- 基本对象
  - `#ctx`：上下文对象
  - `#vars`：上下文变量
  - `#locale`：区域对象
  - `#request`：HttpServletRequest 对象
  - `#response`：HttpServletResponse 对象
  - `#session`：HttpSession 对象
  - `#servletContext`：ServletContext 对象

```
@GetMapping("/index")
public String index(Model model, HttpSession session){
    model.addAttribute("gender","女");
    session.setAttribute("username","张三");
    return "index";
}
```

```
<p th:text="${#request.getAttribute('gender')}"></p>
<p th:text="${#session.getAttribute('username')}"></p>
<p th:text="${#locale.country}"></p>
```

- 内嵌对象

为了模版更加好用，Thymeleaf 提供了一系列 Utility 对象（内置于 Context 中），可以通过 # 直接访问。

- dates：java.util.Date 的功能方法
- calendars：类似于 # dates，对应 java.util.Calendar
- numbers：对数字进行格式化
- strings：字符串操作
- objects：对 Object 的操作
- bools：对布尔求值的方法
- arrays：对数组进行操作
- lists：对 List 集合进行操作
- sets：对 Set 集合进行操作
- maps：对 Map 集合进行操作

```java
@GetMapping("/index")
public String index(Model model, HttpSession session){
    model.addAttribute("name","zhangsan");
    model.addAttribute("users",new ArrayList<>());
    model.addAttribute("count",22);
    model.addAttribute("date",new Date());
    return "index";
}
```

```html
<p th:text="${#dates.format(date,'yyyy-MM-dd HH:mm:sss')}"></p>
<p th:text="${#dates.createNow()}"></p>
<p th:text="${#strings.isEmpty(name)}"></p>
<p th:text="${#lists.isEmpty(users)}"></p>
<p th:text="${#strings.length(name)}"></p>
```