SpringBoot页面展示Thymeleaf


Coding小聪 (/u/3e626432615d) + 关注
1.4 2018.04.05 17:34 字数 1589 阅读 8471 评论 3 喜欢 16
(/u/3e626432615d)

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

上面并没有列举所有SpringBoot支持的页面模板技术。其中Thymeleaf是SpringBoot官方所推荐使用的，下面来谈谈Thymeleaf一些常用的语法规则。

添加Thymeleaf依赖

要想使用Thhymeleaf，首先要在pom.xml文件中单独添加Thymeleaf依赖。

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

Spring Boot默认存放模板页面的路径在 `src/main/resources/templates` 或者 `src/main/view/templates` ，这个无论是使用什么模板语言都一样，当然默认路径是可以自定义的，不过一般不推荐这样做。另外Thymeleaf默认的页面文件后缀是 `.html` 。

数据显示

在MVC的开发过程中，我们经常需要通过 `Controller` 将数据传递到页面中，让页面进行动态展示。

显示普通文本

创建一个Controller对象，在其中进行参数的传递

```
@Controller
public class ThymeleafController {

    @RequestMapping(value = "show", method = RequestMethod.GET)
    public String show(Model model){
        model.addAttribute("uid","123456789");
        model.addAttribute("name","Jerry");
        return "show";
    }
}
```

在SpringBoot默认的页面路径下创建show.html文件，内容如下

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>SpringBoot模版渲染</title>
    <meta http-equiv="Content-Type" content="text/html;charset=UTF-8"/>
</head>
<body>
    <p th:text="'用户ID：' + ${uid}"/>
    <p th:text="'用户名称：' + ${name}"/>
</body>
</html>
```

可以看到在 `p` 标签中有 `th:text` 属性，这个就是thymeleaf的语法，它表示显示一个普通的文本信息。

如果我们要显示的信息是存在资源文件中的，同样可以在页面中显示，例如资源文件中定义了内容 `welcome.msg=欢迎{0}光临！` 。可以在页面中将其显示

```
<h2 th:text="#{welcome.msg('admin')}"/>
```

另外，在 `th:utext` 中还能做一些基础的数学运算

```
<p th:text="'数学计算：1+2=' + (1 + 2)"/>
```

显示带有样式的普通文本

如果我们想要传递到的页面的信息，它本身是带有CSS样式的，这个时候如何在页面中将携带的样式信息也显示出来？此时我们的控制器方法这样写。

```
@RequestMapping(value = "showStyle", method = RequestMethod.GET)
public String showStyle(Model model){
    model.addAttribute("uid","123456789");
    model.addAttribute("name","<span style='color:red'>Jerry</span>");
    return "show_style";
}
```

此时页面中需要借助 `th:utext` 属性进行显示

```
<p th:utext="'用户名称：' + ${name}"/>
```

通过浏览器查看页面源码可以看出 th:utext 和 th:text 的区别是： th:text 会对 < 和 > 进行转义，而 th:utext 不会转义。

显示对象

我们常常需要将一个bean信息展示在前端页面当中。

1. 用于前端展示的VO类

```
public class User implements Serializable {
    private Long uid ;
    private String name ;
    private Integer age ;
    private Date birthday ;
    private Double salary ;
    //省略get/set方法
}
```

2. 控制器方法

```
@RequestMapping(value = "/message/member_show", method = RequestMethod.GET)
public String memberShow(Model model) {
    User vo = new User();
    vo.setUid(12345678L);
    vo.setName("尼古拉丁.赵四");
    vo.setAge(59);
    vo.setSalary(1000.00);
    vo.setBirthday(new Date());
    model.addAttribute("member", vo);
    return "message/member_show";
}
```

3. 页面展示

```
<div>
    <p th:text="'用户编号：' + ${member.uid}"/>
    <p th:text="'用户姓名：' + ${member.name}"/>
    <p th:text="'用户年龄：' + ${member.age}"/>
    <p th:text="'用户工资：' + ${member.salary}"/>
    <p th:text="'出生日期：' + ${member.birthday}"/>
    <p th:text="'出生日期：' + ${#dates.format(member.birthday,'yyyy-MM-dd')}"/>
</div>

<hr/>
<div th:object="${member}">
    <p th:text="'用户编号：' + *{uid}"/>
    <p th:text="'用户姓名：' + *{name}"/>
    <p th:text="'用户年龄：' + *{age}"/>
    <p th:text="'用户工资：' + *{salary}"/>
    <p th:text="'出生日期：' + *{birthday}"/>
    <p th:text="'出生日期：' + *{#dates.format(birthday,'yyyy-MM-dd')}"/>
</div>
```

上面给出了两种展现方式，一种是通过${属性}，另外一种是通过*{属性}*。

关于"${属性}"和"*{属性}"的区别?

*$访问完整信息，而*访问指定对象中的属性内容， 如果访问的只是普通的内容两者没有区别；

数据处理

在 thymeleaf 之中提供有相应的集合的处理方法，例如：在使用 List 集合的时候可以考虑采用 get()方法获取指定索引的数据，那么在使用 Set 集合的时候会考虑使用 contains()来判断某个数据是否存在，使用 Map 集合的时候也希望可以使用 containsKey()判断某个 key 是否存在，以及使用get()根据 key 获取对应的 value，而这些功能在之前并不具备，下面来观察如何在页面中使用此类操作

1. 控制器方法向页面传递一些数据，以供操作

```
@RequestMapping(value = "/user/set", method = RequestMethod.GET)
public String set(Model model) {
    Set<String> allNames = new HashSet<String>() ;
    List<Integer> allIds = new ArrayList<Integer>() ;
    for (int x = 0 ; x < 5 ; x ++) {
        allNames.add("boot-" + x) ;
        allIds.add(x) ;
    }
    model.addAttribute("names", allNames) ;
    model.addAttribute("ids", allIds) ;
    model.addAttribute("mydate", new java.util.Date()) ;
    return "user_set" ;
}
```


2. 数据处理

```
<body>
    <p th:text="${#dates.format(mydate,'yyyy-MM-dd')}"/>
    <p th:text="${#dates.format(mydate,'yyyy-MM-dd HH:mm:ss.SSS')}"/>
    <hr/>
    <p th:text="${#strings.replace('www.baidu.cn','.','$')}"/>
    <p th:text="${#strings.toUpperCase('www.baidu.cn')}"/>
    <p th:text="${#strings.trim('www.baidu.cn')}"/>
    <hr/>
    <p th:text="${#sets.contains(names,'boot-0')}"/>
    <p th:text="${#sets.contains(names,'boot-9')}"/>
    <p th:text="${#sets.size(names)}"/>
    <hr/>
    <p th:text="${#sets.contains(ids,0)}"/>
    <p th:text="${ids[1]}"/>
    <p th:text="${names[1]}"/>
</body>
```

路径处理

在传统WEB工程开发时，路径的处理操作是有点麻烦的。SpringBoot中为我们简化了路径的处理。

1. 在"src/main/view/static/js"中创建一个js文件

js文件路径

2. 然后在页面中可以通过“@{路径}”来引用。

```
<script type="text/javascript" th:src="@{/js/main.js}"></script>
```

页面之间的跳转也能通过@{}来实现

```
<a th:href="@{/show}">访问controller方法</a>
<a th:href="@{/static_index.html}">访问静态页面</a>
```

操作内置对象

虽然在这种模版开发框架里面是不提倡使用内置对象的，但是很多情况下依然需要使用内置对象进行处理，所以下面来看下如何在页面中使用JSP内置对象。

1. 在控制器里面增加一个方法，这个方法将采用内置对象的形式传递属性。

```
@RequestMapping(value = "/inner", method = RequestMethod.GET)
public String inner(HttpServletRequest request, Model model) {
    request.setAttribute("requestMessage", "springboot-request");
    request.getSession().setAttribute("sessionMessage", "springboot-session");
    request.getServletContext().setAttribute("applicationMessage",
            "springboot-application");
    model.addAttribute("url", "www.baidu.cn");
    request.setAttribute("url2",
            "<span style='color:red'>www.baidu.cn</span>");
    return "show_inner";
}
```

2. 在页面之中如果要想访问不同属性范围中的内容，则可以采用如下的做法完成：

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>SpringBoot模版渲染</title>
    <script type="text/javascript" th:src="@{/js/main.js}"></script>
    <meta http-equiv="Content-Type" content="text/html;charset=UTF-8"/>
</head>
<body>
    <p th:text="${#httpServletRequest.getRemoteAddr()}"/>
    <p th:text="${#httpServletRequest.getAttribute('requestMessage')}"/>
    <p th:text="${#httpSession.getId()}"/>
    <p th:text="${#httpServletRequest.getServletContext().getRealPath('/')}"/>
    <hr/>
    <p th:text="'requestMessage = ' + ${requestMessage}"/>
    <p th:text="'sessionMessage = ' + ${session.sessionMessage}"/>
    <p th:text="'applicationMessage = ' + ${application.applicationMessage}"/>
</body>
</html>
```

thymeleaf 考虑到了实际的开发情况，因为 request 传递属性是最为常用的，但是 session 也有可能使用，例如：用户登录之后需要显示用户 id，那么就一定要使用到 session，所以现在必须增加属性范围的形式后才能够正常使用。在 thymeleaf 里面也支持有 JSP 内置对象的获取操作，不过一般很少这样使用。

逻辑处理

所有的页面模版都存在各种基础逻辑处理，例如：判断、循环处理操作。在 Thymeleaf 之中对于逻辑可以使用如下的一些运算符来完成，例如：and、or、关系比较（>、<、>=、<=、==、!=、lt、gt、le、ge、eq、ne）。

通过控制器传递一些属性内容到页面之中：

```
<span th:if="${member.age lt 18}">
未成年人!
</span>
<span th:if="${member.name eq '啊三'}">
欢迎小三来访问!
</span>
```

不满足条件的判断

```
<span th:unless="${member.age gt 18}">
你还不满18岁，不能够看电影!
</span>
```

通过swith进行分支判断

```
<span th:switch="${member.uid}">
<p th:case="100">uid为101的员工来了</p>
<p th:case="99">uid为102的员工来了</p>
<p th:case="*">没有匹配成功的数据！</p>
</span>
```

数据遍历

在实际开发过程中常常需要对数据进行遍历展示，一般会将数据封装成list或map传递到页面进行遍历操作。

1. 定义控制器类，向页面传递List数据和Map数据

```
@Controller
public class UserController {
    @RequestMapping(value = "/user/map", method = RequestMethod.GET)
    public String map(Model model) {
        Map<String,User> allMembers = new HashMap<String,User>();
        for (int x = 0; x < 10; x++) {
            User vo = new User();
            vo.setUid(101L + x);
            vo.setName("赵四 - " + x);
            vo.setAge(9);
            vo.setSalary(99999.99);
            vo.setBirthday(new Date());
            allMembers.put("mldn-" + x, vo);
        }
        model.addAttribute("allUsers", allMembers);
        return "user_map";
    }

    @RequestMapping(value = "/user/list", method = RequestMethod.GET)
    public String list(Model model) {
        List<User> allMembers = new ArrayList<User>();
        for (int x = 0; x < 10; x++) {
            User vo = new User();
            vo.setUid(101L + x);
            vo.setName("赵四 - " + x);
            vo.setAge(9);
            vo.setSalary(99999.99);
            vo.setBirthday(new Date());
            allMembers.add(vo) ;
        }
        model.addAttribute("allUsers", allMembers);
        return "user_list";
    }
}
```

2. list类型数据遍历

```
<body>
    <table>
        <tr><td>No.</td><td>UID</td><td>姓名</td><td>年龄</td><td>偶数</td><td>奇数</t
        <tr th:each="user,memberStat:${allUsers}">
            <td th:text="${memberStat.index + 1}"/>
            <td th:text="${user.uid}"/>
            <td th:text="${user.name}"/>
            <td th:text="${user.age}"/>
            <td th:text="${memberStat.even}"/>
            <td th:text="${memberStat.odd}"/>
        </tr>
    </table>
</body>
```

3. map类型数据遍历

```
<body>
    <table>
        <tr><td>No.</td><td>KEY</td><td>UID</td><td>姓名</td><td>年龄</td><td>偶数</td><td
        <tr th:each="memberEntry,memberStat:${allUsers}">
            <td th:text="${memberStat.index + 1}"/>
            <td th:text="${memberEntry.key}"/>
            <td th:text="${memberEntry.value.uid}"/>
            <td th:text="${memberEntry.value.name}"/>
            <td th:text="${memberEntry.value.age}"/>
            <td th:text="${memberStat.even}"/>
            <td th:text="${memberStat.odd}"/>
        </tr>
    </table>
</body>
```

页面引入

我们常常需要在一个页面当中引入另一个页面，例如，公用的导航栏以及页脚页面。thymeleaf中提供了两种方式进行页面引入。

- th:replace
- th:include

1. 新建需要被引入的页面文件，路径为"src/main/view/templates/commons/footer.html"

```
<meta http-equiv="Content-Type" content="text/html;charset=UTF-8"/>
<footer th:fragment="companyInfo">
    <p>设为首页 ©2018 SpringBoot 使用<span th:text="${projectName}"/>前必读
        意见反馈 京ICP证030173号 </p>
</footer>
```

可以看到页面当中还存在一个变量projectName，这个变量的值可以在引入页面中通过 `th:with="projectName=百度"` 传过来。

2. 引入页面中只需要添加如下代码即可

```
<div th:include="@{/commons/footer} :: companyInfo" th:with="projectName=百度"/>
```



Coding小聪 (/u/3e626432615d) ♂
写了 53145 字，被 58 人关注，获得了 137 个喜欢
(/u/3e626432615d)
+关注