第 2-5 课：Thymeleaf 页面布局

页面布局就是对前端的页面进行划分区域，每个区域有不同的职责，布局是为了更好地重复利用前端代码，避免大量重复性的劳动。在现有的前端系统中，页面布局成了前端开发最重要的工作之一，Thymeleaf 在设计之初对页面布局就有考虑，通过 Thymeleaf 的相关语法可以很容易地实现对前端页面布局。

快速入手

Spring Boot 2.0 将布局单独提取了出来，需要单独引入依赖：thymeleaf-layout-dialect。

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>
<dependency>
    <groupId>nz.net.ultraq.thymeleaf</groupId>
    <artifactId>thymeleaf-layout-dialect</artifactId>
</dependency>
```

首先定义一个代码片段 copyright.html，放到 layout 目录下：

```
<footer th:fragment="copyright">
&copy; 2018
</footer>
```

th:fragment 定义一个代码片段，创建 index.html 在页面任何地方引入：

```
<body>
    <div th:insert="layout/copyright :: copyright"></div>
    <div th:replace="layout/copyright :: copyright"></div>
</body>
```

> th:insert 和 th:replace 区别，insert 只是加载，replace 是替换。

Thymeleaf 3.0 推荐使用 th:insert 替换 2.0 的 th:replace。

layout 是文件夹地址，fileName 是文件名，语法这样写 layout/fileName:htmlhead，其中，htmlhead 是指定义的代码片段，如 th:fragment="htmlhead"。

创建一个 indexController，指向 index.html：

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

在浏览器中输入网址，http://localhost:8080/index，可以看到以下信息：

```
© 2018
© 2018
```

可以看到两条版权信息，说明两种方式都可以引入版权页面信息，这就是 Thymeleaf 最简单的页面布局了。

> Thymeleaf 的这种特性，可以让我们在开发过程中避免写很多重复的代码，如果页面有共同的头部、尾部或者其他地方均可采用这种技术。

片段表达式

Thymeleaf 3.0 引入了一种新型的表达式，作为一般 Thymeleaf 标准的语法表达式之一，它就是：**片段表达式**。

它使用类似 ~{commons::footer} 的语法，作用在 th:insert 和 th:replace 的内部。使用片段表达式支持引入片段的同时传参来构造目标页面，大大提高了代码复用的灵活度。接下来使用一个案例来介绍片段表达式的使用。

创建一个 base.html 基础页面，作为后续其他页面引入的模板。

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head th:fragment="common_header(title,links)">
    <title th:replace="${title}">comm title</title>

    <link rel="stylesheet" type="text/css" media="all" th:href="@{/css/myapp.css}"
>
    <link rel="shortcut icon" th:href="@{/images/favicon.ico}">
    <script type="text/javascript" th:src="@{/js/myapp.js}"></script>

    <th:block th:replace="${links}" />
</head>
<body>

</body>
</html>
```

使用 th:fragment 创建了一段代码片段，命名为 common_header 并且支持传入两个参数 title 和 links，并且指明了在页面中使用的位置：

- `<title th:replace="${title}">comm title</title>` 在此位置插入传入的 title；
- `<th:block th:replace="${links}" />` 在此位置插入传入的 link，th:block 作为页面的自定义使用不会展示到页面中。

接下来创建一个 fragment.html 页面，来使用上面创建的 common_header 代码片段。

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head th:replace="base :: common_header(~{::title},~{::link})">
    <title>Fragment - Page</title>
    <link rel="stylesheet" th:href="@{/css/bootstrap.min.css}">
    <link rel="stylesheet" th:href="@{/cs/fragment.css}">
</head>
<body>

</body>
</html>
```

在 fragment.html 的页面中使用 th:replace 引入了 `common_header` 代码片段，并且在页面中添加了 title 和 link 作为参数传递到 `common_header` 页面中。

在 IndexController 中添加访问入口：

```
@RequestMapping("/fragment")
public String fragment() {
    return "fragment";
}
```

重新启动项目后，在浏览器中输入网址，http://localhost:8080/fragment，右键单击查看源代码，可以看到以下信息：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Fragment - Page</title>

    <link rel="stylesheet" type="text/css" media="all" href="/css/myapp.css">
    <link rel="shortcut icon" href="/images/favicon.ico">
    <script type="text/javascript" src="/js/myapp.js"></script>

    <link rel="stylesheet" href="/css/bootstrap.min.css"><link rel="stylesheet" hr
ef="/cs/fragment.css">
</head>
<body>

</body>
</html>
```

说明 fragment.html 页面已经引入了 base.html 页面中的代码片段，fragment.html 页面中传入的 title 和 link 已经成功地插入到了 base.html 页面的代码片段中。

这就是片段表达式最常用的使用方式之一，也是版本 3.0 针对页面布局推出的最新语法。因为这一点，许多布局（或页面）技术在 Thymeleaf 3.0 中变得更容易使用。

页面布局

按照上面这个思路很容易做页面布局，按照常用的框架模式，将页面分为头部、左侧菜单栏、尾部和中间的展示区来做个示例：

在 templates/layout 目录下新建 footer.html、header.html、left.html 页面，内容如下。

footer.html：

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8"></meta>
    <title>footer</title>
</head>
<body>
<footer th:fragment="footer">
    <h1>我是 尾部</h1>
</footer>
</body>
</html>
```

header.html：

```
<!DOCTYPE html>
<html lang="en"  xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8"></meta>
    <title>header</title>
</head>
<body>
<header th:fragment="header">
    <h1>我是 头部</h1>
</header>
</body>
</html>
```

left.html：

```
<!DOCTYPE html>
<html lang="en"  xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8"></meta>
    <title>left</title>
</head>
<body>
<left th:fragment="left">
    <h1>我是 左侧</h1>
</left>
</body>
</html>
```

templates 目录下新建 layout.html 页面内容如下：

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org" xmlns:layout="http://www.ultra
q.net.nz/web/thymeleaf/layout">
<head>
    <meta charset="UTF-8"></meta>
    <title>Layout</title>
</head>
<body>
    <div >
        <div th:replace="layout/header :: header"></div>
        <div th:replace="layout/left :: left"></div>
        <div layout:fragment="content" > content</div>
        <div th:replace="layout/footer :: footer"></div>
    </div>
</body>
</html>
```

引入布局的语法命名空间：xmlns:layout="http://www.ultraq.net.nz/web/thymeleaf/layout"，使用 th:replace 的语法将网站的头部、尾部、左侧引入到页面中，同时定义了一个代码片段 content，可以作为后期页面正文的替换内容。

后端添加访问入口：

```
@RequestMapping("/layout")
public String layout() {
    return "layout";
}
```

启动后访问地址：http://localhost:8080/layout，可以看到页面展示如下：

```
我是 头部
我是 左侧
content
我是 尾部
```

可以看出 layout.html 页面已经成功地引入了页面的头部、尾部、左侧。接下来以 layout.html 作为一个页面模板，任何页面想使用此布局时，只需要替换中间的 content 模块即可，新建 home.html 来测试：

```
<html xmlns:th="http://www.thymeleaf.org"  xmlns:layout="http://www.ultraq.net.nz/
web/thymeleaf/layout" layout:decorate="layout">
<head>
    <meta charset="UTF-8"></meta>
    <title>Home</title>
</head>
<body>
    <div layout:fragment="content" >
        <h2>个性化的内容</h2>
    </div>
</body>
</html>
```

- `<html>` 标签中添加 layout:decorate="layout" 使用 layout.html 页面的布局。
- `<div layout:fragment="content" >` 替换 layout.html 页面中的 content 代码片段。
- **layout:decorator 标签在 3.0 过期，推荐使用新的标签 layout:decorate 进行页面布局。**

Controller 添加访问：

```
@RequestMapping("/home")
public String home() {
    return "home";
}
```

重启后，在浏览器中输入网址 http://localhost:8080/home，发现 home.html 页面已经采用了 layout.html 的页

面布局，content 部分的内容被替换为：“个性化的内容”，展示内容如下：

```
我是 头部
我是 左侧
个性化的内容
我是 尾部
```

这样其他页面如果都是类似的结构都可以采用这样的方式来使用。

采用页面模板布局的时候有两个关键设置：

- 在模板页面定义需要替换的部分 layout:fragment="content"；
- 在需要引入模板的页面头部写 layout:decorate="layout""，再修改 layout:fragment="content" 中的内容。

总结

通过本课的学习发现 Thymeleaf 对代码片段重复利用、页面布局都有很好的支持，Thymeleaf 3.0 引入代码片段语法更加方便布局技术的使用。Thymeleaf 3.0 的页面布局技术，降低了后端开发人员学习页面布局的成本，使用页面布局技术后会高效地复用前端页面，减少了开发量、稳定前端页面结构。

> 点击这里下载源码。