4章 Thymeleaf模板引擎

Thymeleafテンプレート

Thymeleaf模板引擎

什么是模板?

到此为止，我们已经准备好要显示为文本的内容并输出了。但是我们需要预先准备 HTML 文件等，根据需要读取并显示。然而实际上，我们一般不是这样准备网页的。因为这样网页的内容就被固定了。那么，如何才能让HTML的网页中出现变化的量呢?

模板是读取基于 HTML 编写的源代码渲染后并将其呈现以网页显示的方法。它不仅简单地“**读取并显示 HTML 代码**”，而且还具有将各种信息插入 HTML 并根据需要显示的功能。这使得**从程序中控制网页显示**成为可能 。

Thymeleaf模板引擎

Spring Boot经常使用一个名为 “Thymeleaf”的模板库。

Thymeleaf 是为 Java 创建的模板库。该程序被实现为一个 Java 类。Spring Boot 本身就有使用 Thymeleaf 的功能，所以不需要单独安装 Thymeleaf 库。

在 Java EE 中，JSP 长期以来一直用于显示网页。很多人在学校都习惯了 JSP。但是，JSP 使用类似于 HTML 的标签嵌入 Java 代码，因此不能用 HTML 可视化编辑器很好地处理。

Thymeleaf在标签中准备了一个特殊的属性“th:○○”，并使用${}等特殊符号插入值，从而在不影响 HTML 标签结构的情况下创建内容。可以描述。即使使用 HTML 可视化编辑器等，也不会损坏显示。

问题

- 登录成功之后会进入hello页面，如何在hello页面上显示当前的用户名?
- 登录失败的情况下，如何在登录界面显示登录失败信息以提醒用户再次登陆?

- 解答：需要Controller向前端传递一些讯息→前后端连调

- 大多数网站做的最基本的事情就是将数据库的内容呈现给用户

创建Controller文件

- 参考LoginThymeleafController.java

- ModelAndView类
顾名思义，这是一个将MVC “模型”和“视图” 信息一起管理的类。
通过addObject方法添加数据到模型中。
通过setViewName指定视图的html模板文件。

```
//  「/login」POST请求
 @RequestMapping(value = "/login", method = RequestMethod.POST)
 public ModelAndView login(@RequestParam String id, @RequestParam String pass, ModelAndView mav) {
   // 将POST接受到的数据报存在model中
   mav.addObject("id", "Your ID is " + id + ".");
   mav.addObject("pass", "PASS is " + pass + ".");
   // 指定所对应的view
   mav.setViewName("login/success");
   // 返回modelandview
   return mav;
 }
```

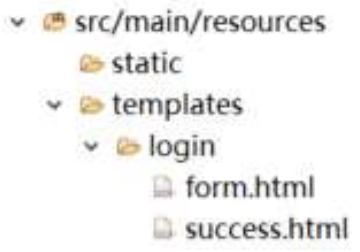
创建一个简单的登录应用程序（Thymeleaf 版本）

- 基于上一章中创建的登录项目，我们将学习如何使用 Thymeleaf 创建应用程序。
- 首先我们需要确认pom.xml的依赖中已经添加thymeleaf。

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

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>

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

创建模板文件

- 在“src/main/resources”包的“templates”文件夹内创建一个“login”文件夹
- 创建 “login”文件夹中创建“form.html”和“success.html”。
- 注意success.html中的第2行，thymeleaf模板需要声明。



form.html

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <meta charset="UTF-8">
5     <title>login form</title>
6 </head>
7
8 <body>
9     <form action="/login" method="post">
10    ID:<input type="text" name="id"><br>
11    PASSWORD:<input type="password" name="pass"><br>
12    <input type="submit" value="login">
13    </form>
14 </body>
15 </html>
```

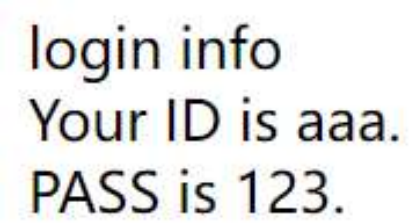
success.html

```
1 <!DOCTYPE html>
2 <html xmlns:th="https://thymeleaf.org">
3 <head>
4     <meta charset="UTF-8">
5     <title>login result</title>
6 </head>
7
8 <body>
9     login info<br>
10    <span th:text="${id}"></span><br>
11    <span th:text="${pass}"></span>
12 </body>
13 </html>
```

启动Spring项目

- 启动项目
- 打开浏览器http://localhost:8080/loginForm

- 输入账户密码，提交后跳转至http://localhost:8080/login



- 尝试输入不同的值，可以发现跳转结果的内容是不同的。

Thymeleaf动态绑定

- 让我们看看success.html代码，了解thymeleaf是如何实现html内容的动态绑定的。动态绑定的内容如下两行代码所示。

```
<span th:text="${id}"></span><br>
<span th:text="${pass}"></span>
```

- 在这里，添加了一个名为“th: text”的属性。这是 Thymeleaf 特有的属性。
- 当页面内容被 Thymeleaf 渲染时，唯一属性“th:XX”的值将被替换为标签值并渲染。由于是唯一属性，所以除非渲染，否则对显示没有影响。（可以尝试添加th:text=“${test}”进行确认）

Thymeleaf 的多种使用方式

Thymeleaf 的用途还有很多，让我们一一尝试基本功能。在此之前，首先创建基础模板文件和类文件。

在“src/main/resources”包的“templates”文件夹内创建一个“thymeleaf”文件夹，并创建“howto.html”。定义一个空的页面。

```
1 <!DOCTYPE html>
2 <html xmlns:th="https://thymeleaf.org">
3 <head>
4     <meta charset="UTF-8">
5     <title>thymeleaf use case</title>
6 </head>
7
8 <body>
9     <div class = "contents">
10    <!-- write here -->
11    </div>
12 </body>
13 </html>
```

Thymeleaf 的多种使用方式

然后准备一个Controller，ThymeleafController.java

```java
package com.example.login;


import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.servlet.ModelAndView;

@Controller
public class ThymeleafController {
  // 链接
  @RequestMapping("/thymeleaf")
  public ModelAndView thymeleaf(ModelAndView mav) {
    // 指定view
    mav.setViewName("thymeleaf/howto");
    // 返回model和view
    return mav;
  }

}

```

Thymeleaf 的多种使用方式

然后准备一个Controller，ThymeleafController.java

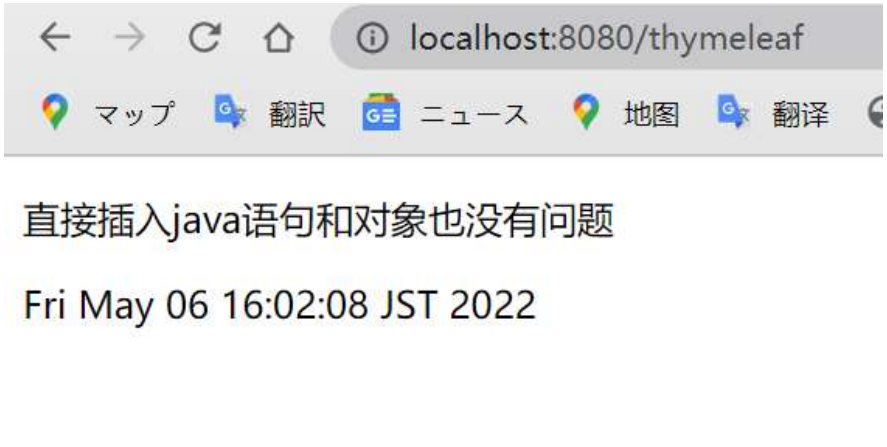
```java
package com.example.login;


import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.servlet.ModelAndView;

@Controller
public class ThymeleafController {
  // 链接
  @RequestMapping("/thymeleaf")
  public ModelAndView thymeleaf(ModelAndView mav) {
    // 指定view
    mav.setViewName("thymeleaf/howto");
    // 返回model和view
    return mav;
  }

}

```

Thymeleaf 的多种使用方式

- Thymeleaf 的基础是**“输出（显示）值”**。这以**$ {○○}**的形式编写。这种编写 ${} 的方式称为**“变量表达式”**。
  变量表达式中描述的是**“OGNL”**（对象图形导航语言）表达式，一种用于访问 Java 值的表达式语言。它不仅用于 Thymeleaf，还用于各种库和框架。

- OGNL 的编写类似于 Java 的简化版本，因此对于 Java 程序员来说并不太难。基本上，你可以认为如果你用 Java 写一个表达式，如果它简单的话，它通常会是一个 OGNL 表达式。
  打开之前创建的 howto.html 并像下面这样重写 标签部分。

```
1 <!DOCTYPE html>
2 <html xmlns:th="https://thymeleaf.org">
3 <head>
4     <meta charset="UTF-8">
5     <title>thymeleaf use case</title>
6 </head>
7
8 <body>
9     <div class = "contents">
10    <!-- write here -->
11    <p> 直接插入java语句和对象也没有问题</p>
12    <p th:text="${new java.util.Date().toString()}"></p>
13    </div>
14 </body>
15 </html>
```



- 值得一提的是，th:text语句不是写在标签之间，而是写在标签内部

```
<p th:text="${new java.util.Date().toString()}"></p>   √
<p> th:text="${new java.util.Date().toString()}"</p>   ×
```

Thymeleaf 的多种使用方式

- Thymeleaf有一些内部处理数据的方法

```
1 <!DOCTYPE html>
2 <html xmlns:th="https://thymeleaf.org">
3 <head>
4     <meta charset="UTF-8">
5     <title>thymeleaf use case</title>
6 </head>
7
8 <body>
9     <div class = "contents">
10    <!-- write here -->
11    <p> thymeleaf的一些内部方法</p>
12    <p th:text="${#dates.format(new java.util.Date(),'dd/MM/yyyy HH:mm')}"></p>
13    <p th:text="${#numbers.formatInteger(1234,7)}"></p>
14    <p th:text="${#strings.toUpperCase('welcome')}"></p>
15    </div>
16 </body>
17 </html>
```

thymeleaf的一些内部方法

06/05/2022 16:29

0001234

WELCOME

具体可以参考官方文档
https://www.thymeleaf.org/doc/tutorials/3.0/usingthymeleaf_ja.html

Thymeleaf 的多种使用方式

- Web 应用程序经常发送带有参数的SQL查询字符串。
  我们已经学习了控制器如何处理这些参数。还有方法可以直接在模板中使用参数而不通过控制器。

- 在这种情况下使用一个名为**“param”的变量。**这是一个可以直接在变量表达式中使用的变量。可以通过指定参数名称从变量中检索值。例如，输入 ${param.id}，将收到以 id = ○○ 形式发送的值。
  不过这种方式得到的值通常都是数组的形式，所以我们会从这里提取更多的值。

- 注意第12行的写法
- `<p th:text="'id is'+${param.id[0]}"></p>`

```
1  <!DOCTYPE html>
2  <html xmlns:th="https://thymeleaf.org">
3  <head>
4      <meta charset="UTF-8">
5      <title>thymeleaf use case</title>
6  </head>
7
8  <body>
9      <div class = "contents">
10     <!-- write here -->
11     <p> thymeleaf的一些方法</p>
12     <p th:text="'id is'+${param.id[0]}"></p>
13     </div>
14 </body>
15 </html>
```

```
<p th:text="'one'+'two'+'three' "></p>
```

通过以下链接访问 http://localhost:8080/thymeleaf?id=123

Thymeleaf 的多种使用方式

- 拼接字符串

```
<p th:text="'one'+'two'+'three' "></p>
```

- 判断if

```
<p th:if="2>1">yes</p>
<p th:if="2<1">no</p>
```

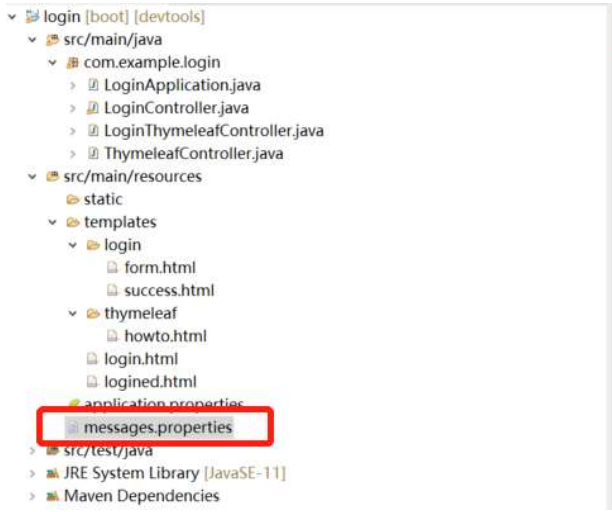
以上例子只显示yes

- 链接href

```
<p><a th:text="'visit website'" th:href="@{/url}"></a></p>
```

Thymeleaf 的多种使用方式

- 除了变量表达式 ${}，Thymeleaf 中还有其他可用的东西。它是一种“信息表达”。
- 它从项目中准备的属性文件中检索值并显示它。在 Java 中，通常会预先将文本放在一个属性文件中并读取它以供使用。可以说消息表达式可以直接从模板中获得。
- 消息表达式描述方法：#{}
- 让我们创建messgae.properties在resources文件夹下，然后修改howto.html来调用它的内容吧。




```
howto.title=How to Thymeleaf
howto.message=This is sample message from properties
```

```
<!DOCTYPE html>
<html xmlns:th="https://thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>thymeleaf use case</title>
</head>

<body>
    <div class = "contents">
    <!-- write here -->
    <p> thymeleaf的一些方法</p>
    <p th:text="#{howto.title}"></p>
    <p th:text="#{howto.message}"></p>
    </div>
</body>
</html>
```

- 最终显示

thymeleaf的一些方法

How to Thymeleaf

This is sample message from properties

Thymeleaf 的多种使用方式

- 变量表达式基本上只是按原样输出控制器端准备的值。然而，这应该是一个简单的值，例如数值或文本，但是当涉及到使用对象时，它变得难以处理。
- 当然可以通过指定对象的属性或者方法来写是肯定的，比如${object.name}，但是如果对象里面的值很多的话，一个个都写就麻烦了，就是更改对象名称时，很难再次重写所有名称。
- 在这种情况下，会提供一个专用的变量表达式来指定对象并检索该选定对象中的值。
- 此变量表达式用于处理对象的变量表达式内部。

```
<body>
<div class = "contents">
<!-- write here -->
<p> thymeleaf的一些方法</p>

<table th:object="${user}">
      <tr>
            <th>name</th>
            <td th:text="*{name}"></td>
      </tr>
      <tr>
            <th>age</th>
            <td th:text="*{age}"></td>
      </tr>
</div>

</body>
```

thymeleaf的一些方法

name anna
age 25

本章小结

使用模板引擎的好处

1. 通过缩短描述可以预期开发效率会提高。
2. 提高 HTML 文件的可读性。

使用模板引擎的缺点

1. 由学习成本，需要学习新的语法。
2. 大多数描述性文本基本上较短，但有些则不然。
3. 如果不了解模板引擎，会很难修改一些文件。

总结

虽然从零开始学习如何使用模板引擎比较困难，但使用它并没有什么坏处，所以让我们以使用模板引擎来提高开发效率为目标。