

使用 Thymeleaf+ Bootstrap 完成展示页面

目录

任务目标	2
知识点	2
实现思路	2
编码步骤	3
源码参考	14
回马枪总结	14

实训邦
www.sxbang.net

任务目标

- 完成控制器设计-RegisterController 类
- 在 SpringBoot 中使用 Thymeleaf 模板
- 在 SpringBoot 中使用 Bootstrap 模板

知识点

Thymeleaf简介:

Thymeleaf 是一个跟 Velocity、FreeMarker 类似的模板引擎，它可以完全替代 JSP 。相较与其他的模板引擎，它有如下三个极吸引人的特点：

1.Thymeleaf 在有网络和无网络的环境下皆可运行，即它可以让美工在浏览器查看页面的静态效果，也可以让程序员在服务器查看带数据的动态页面效果。这是由于它支持 html 原型，然后在 html 标签里增加额外的属性来达到模板+数据的展示方式。浏览器解释 html 时会忽略未定义的标签属性，所以 thymeleaf 的模板可以静态地运行；当有数据返回到页面时，Thymeleaf 标签会动态地替换掉静态内容，使页面动态显示。

2.Thymeleaf 开箱即用的特性。它提供标准和spring标准两种方言，可以直接套用模板实现JSTL、OGNL表达式效果，避免每天套模板、改jstl、改标签的困扰。同时开发人员也可以扩展和创建自定义的方言。

3. Thymeleaf 提供spring标准方言和一个与 SpringMVC 完美集成的可选模块，可以快速的实现表单绑定、属性编辑器、国际化等功能。

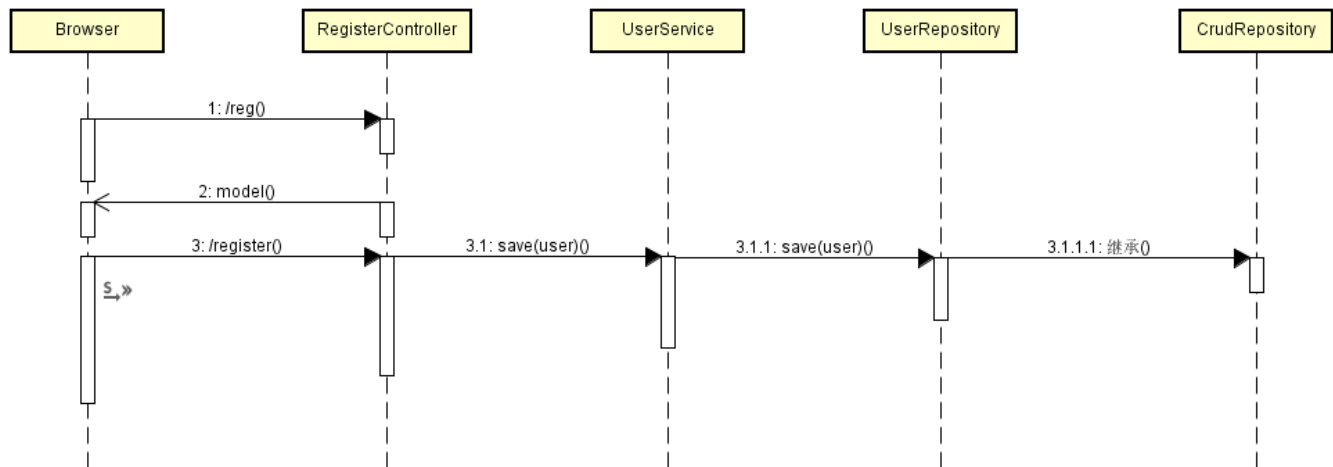
学习参考网址：<https://www.thymeleaf.org/doc/tutorials/3.0/thymeleafspring.html#creating-a-form>

Bootstrap:

关于这部分的基础学习，请参考我的“Bootstrap用户管理模块”课程。

学习参考网址：<https://v3.bootcss.com/>

实现思路



URL 请求分析：

1. get: /reg

逻辑：跳转到注册页面

参数：username, password

返回值：ModelAndView

2. post: /register

逻辑：保存提交的用户名和密码

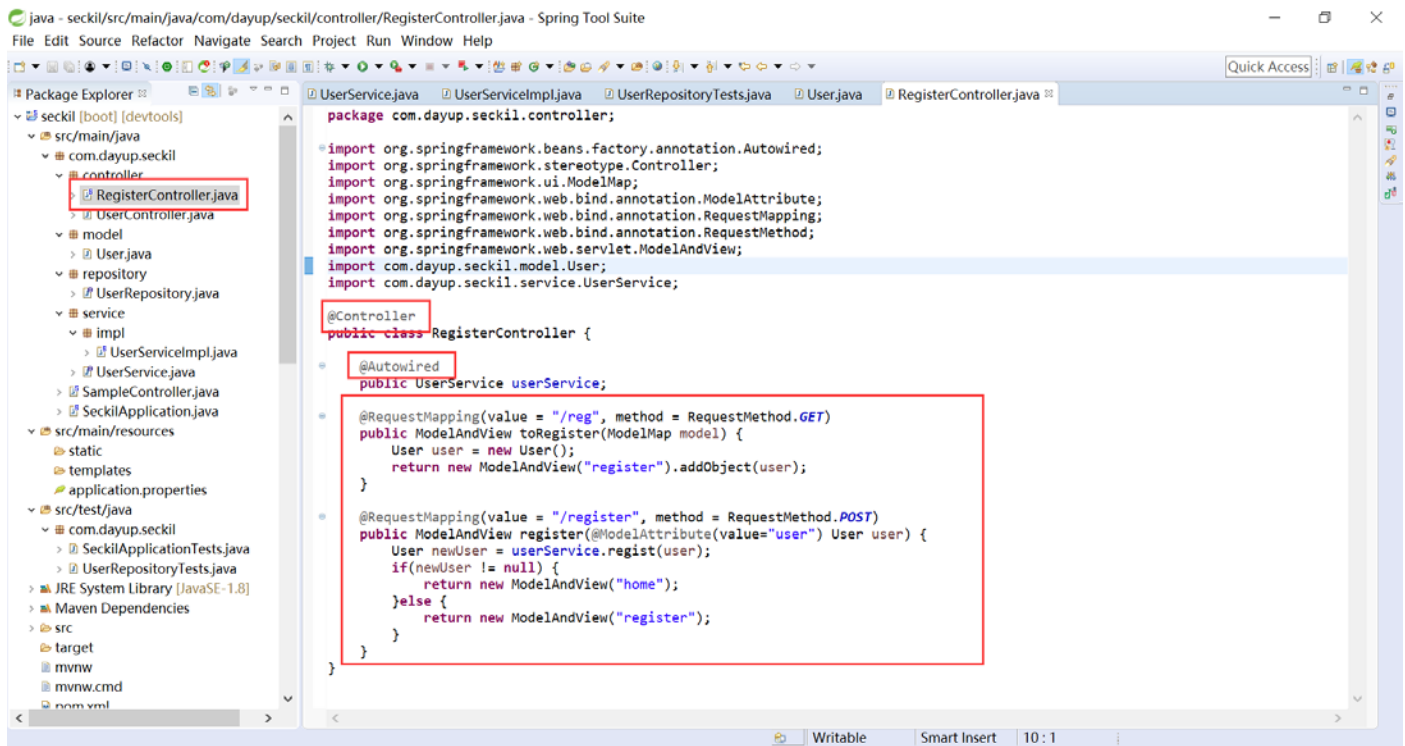
参数：

参数名	类型	是否必填	说明
username	String	N	
password	String	N	

返回值：ModelAndView

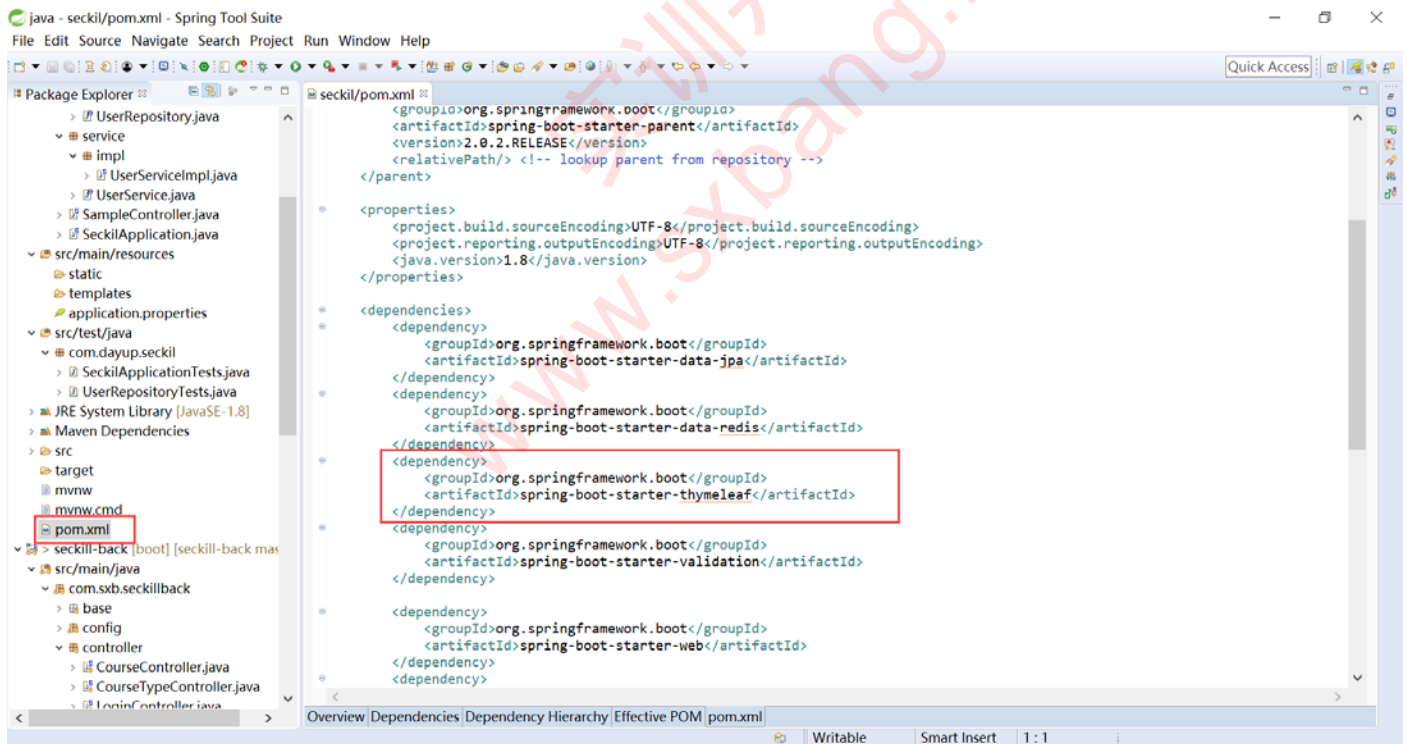
编码步骤

1. 完成控制器设计-RegisterController 类



2. 在 SpringBoot 中使用 Thymeleaf 模板

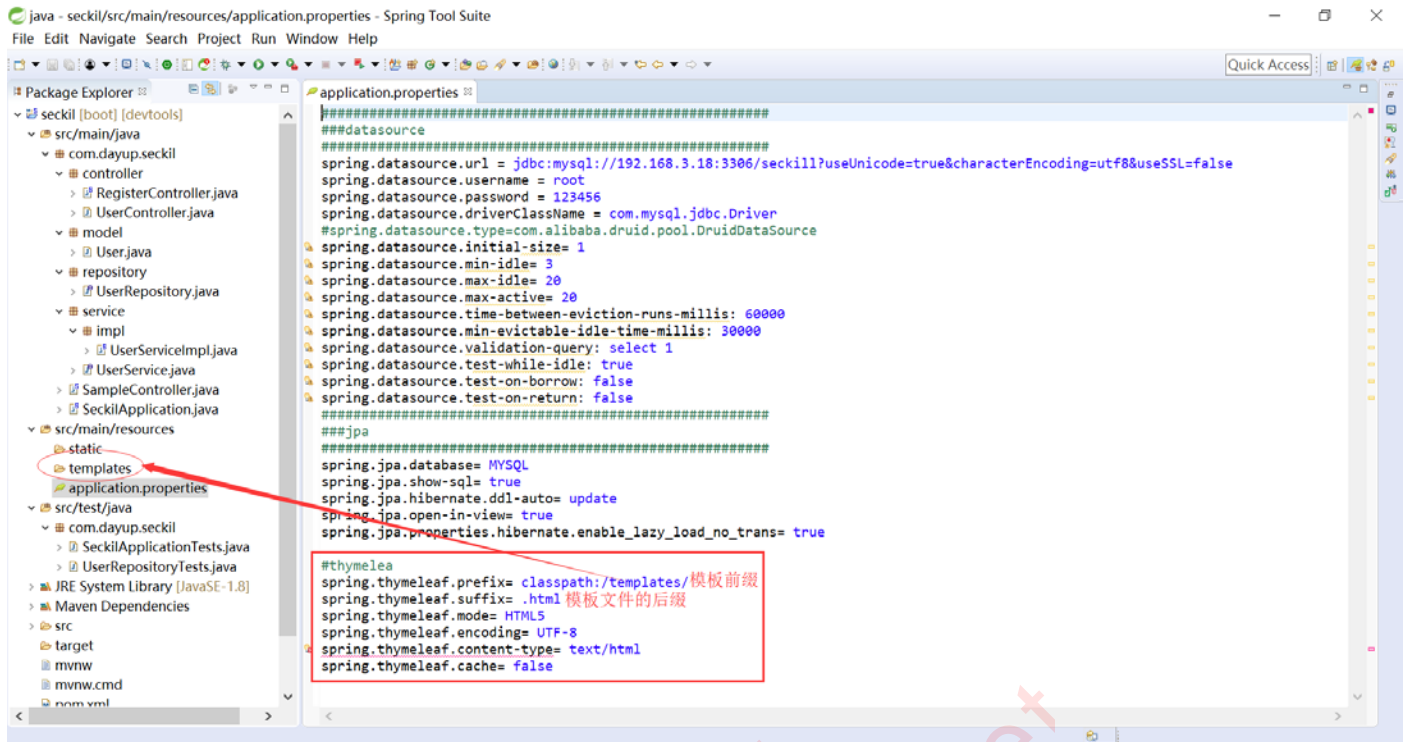
Step1.在 pom.xml 中添加依赖，引入 Thymeleaf



Step2. 配置 Thymeleaf:

在 application.properties 文件中添加 Thymeleaf 的基本配置，主要的两行，一行是配置模板，也就是 html 文件

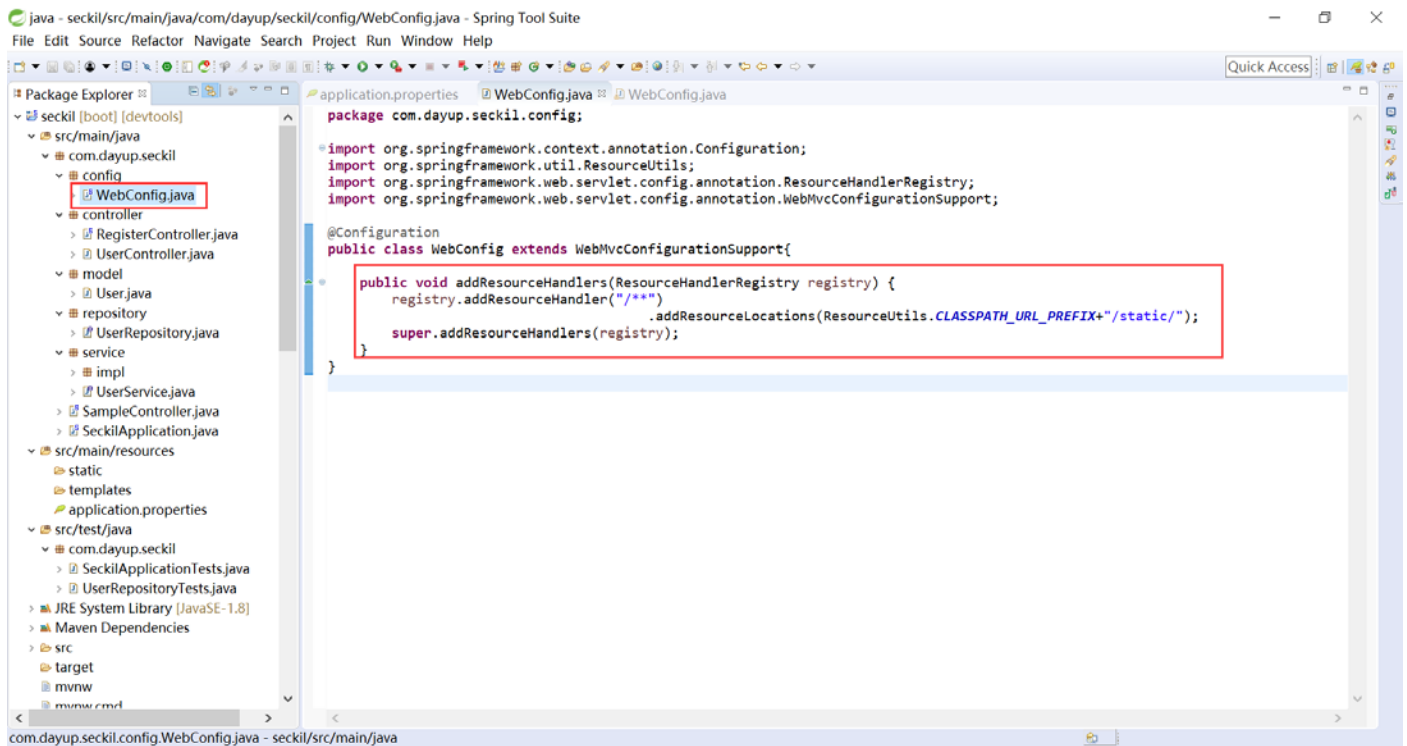
的位置；一行是配置模板文件的后缀，这里我们使用 html 文件作为模板。



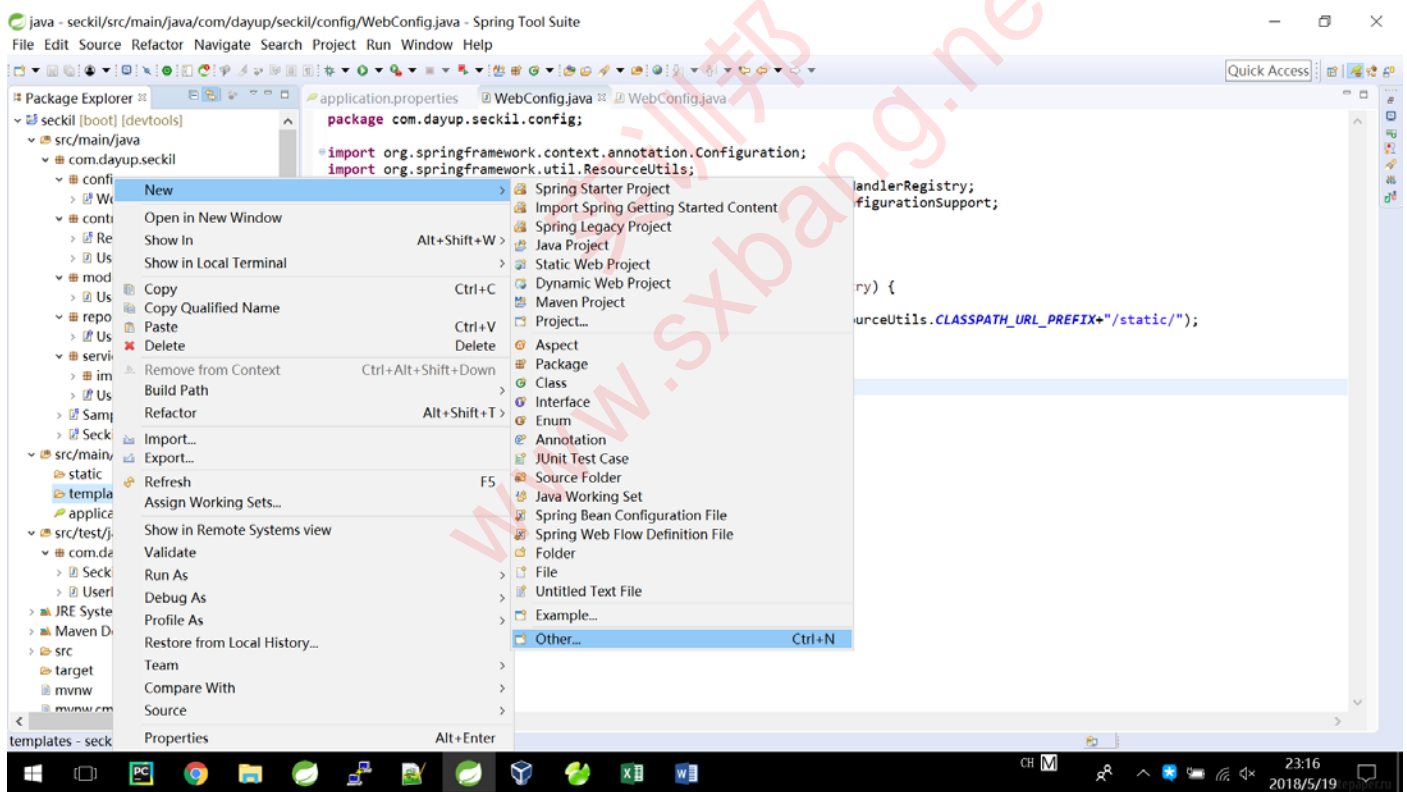
#thymelea

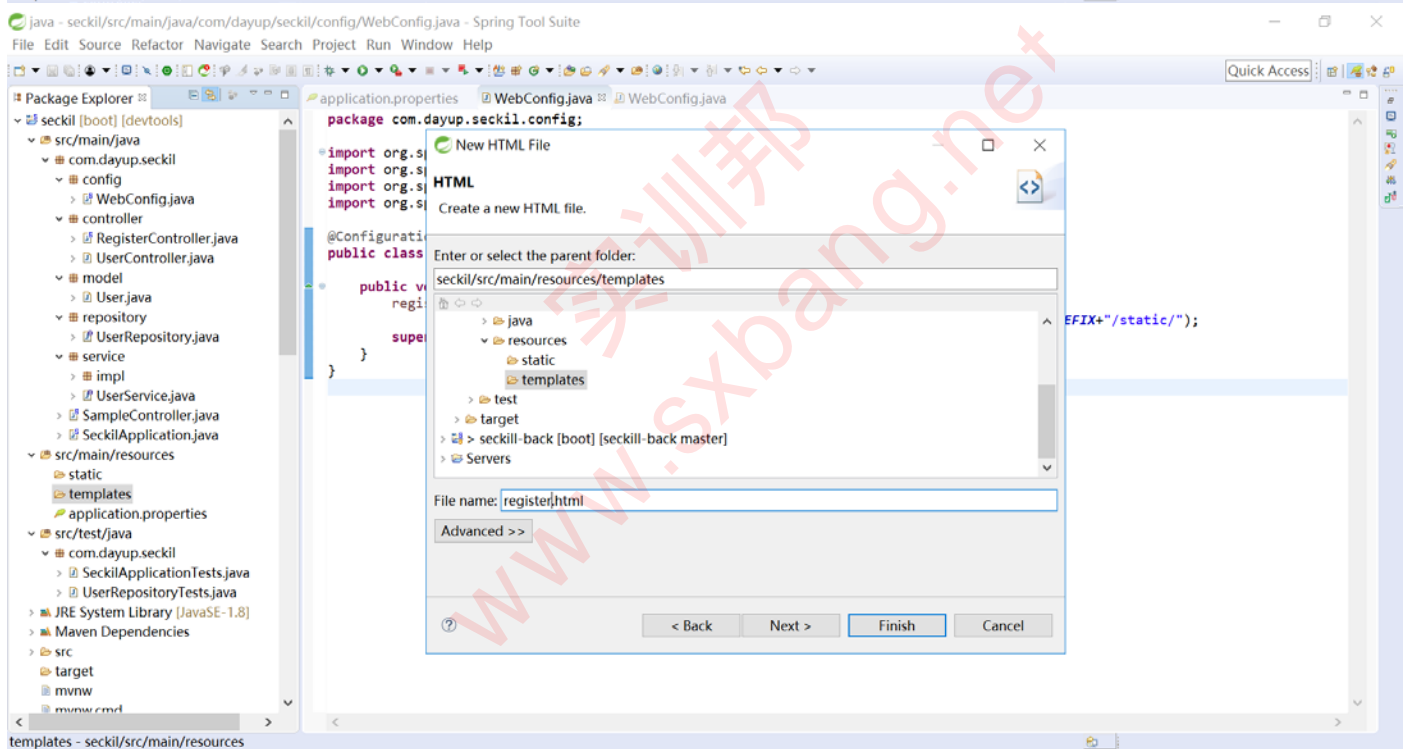
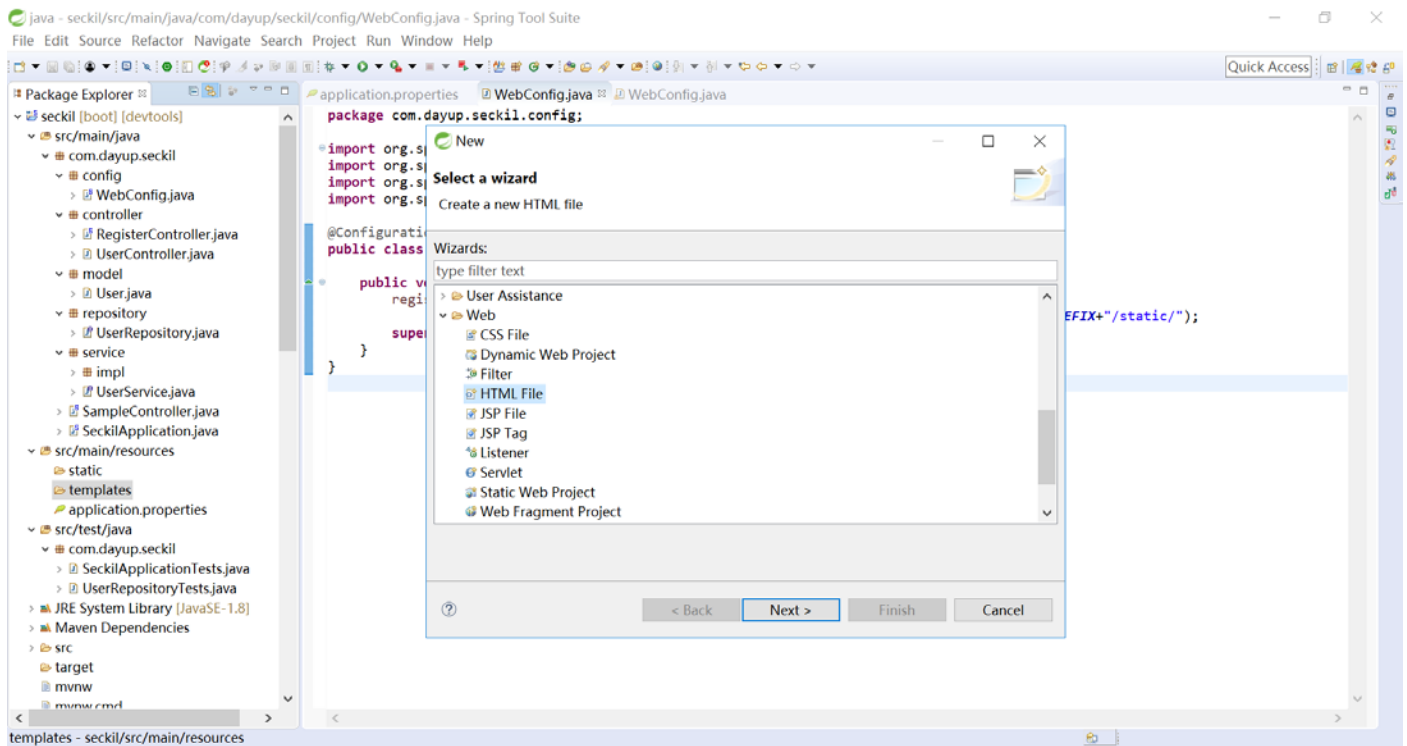
```
spring.thymeleaf.prefix= classpath:/templates/
spring.thymeleaf.suffix= .html
spring.thymeleaf.mode= HTML5
spring.thymeleaf.encoding= UTF-8
spring.thymeleaf.content-type= text/html
spring.thymeleaf.cache= false
```

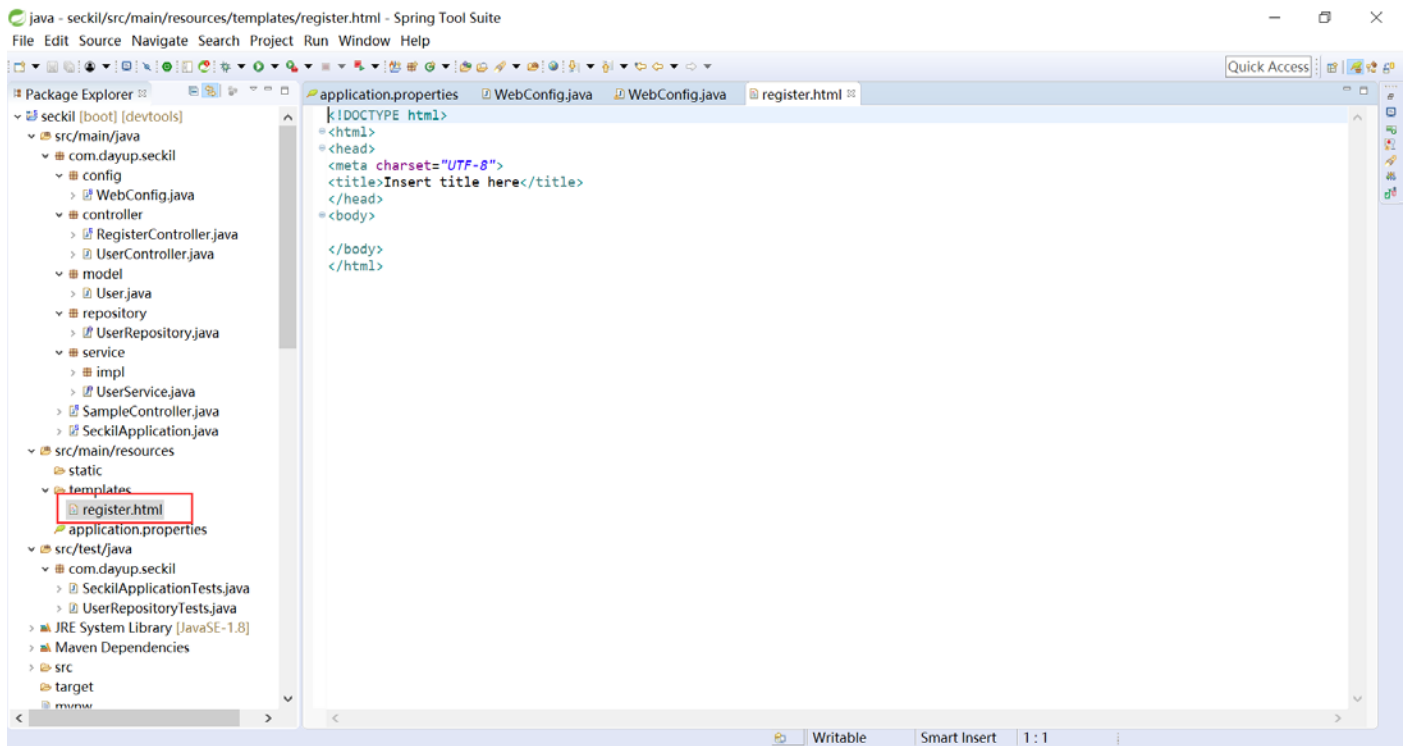
Step3.如果你使用 java 的配置方式，另外配置了 WebConfig 的话，为了避免静态资源文件无法引入，最好在你的 WebConfig 文件加上下面红框中的方法



Step4.接着在 templates 文件夹下新建一个 register.html 文件

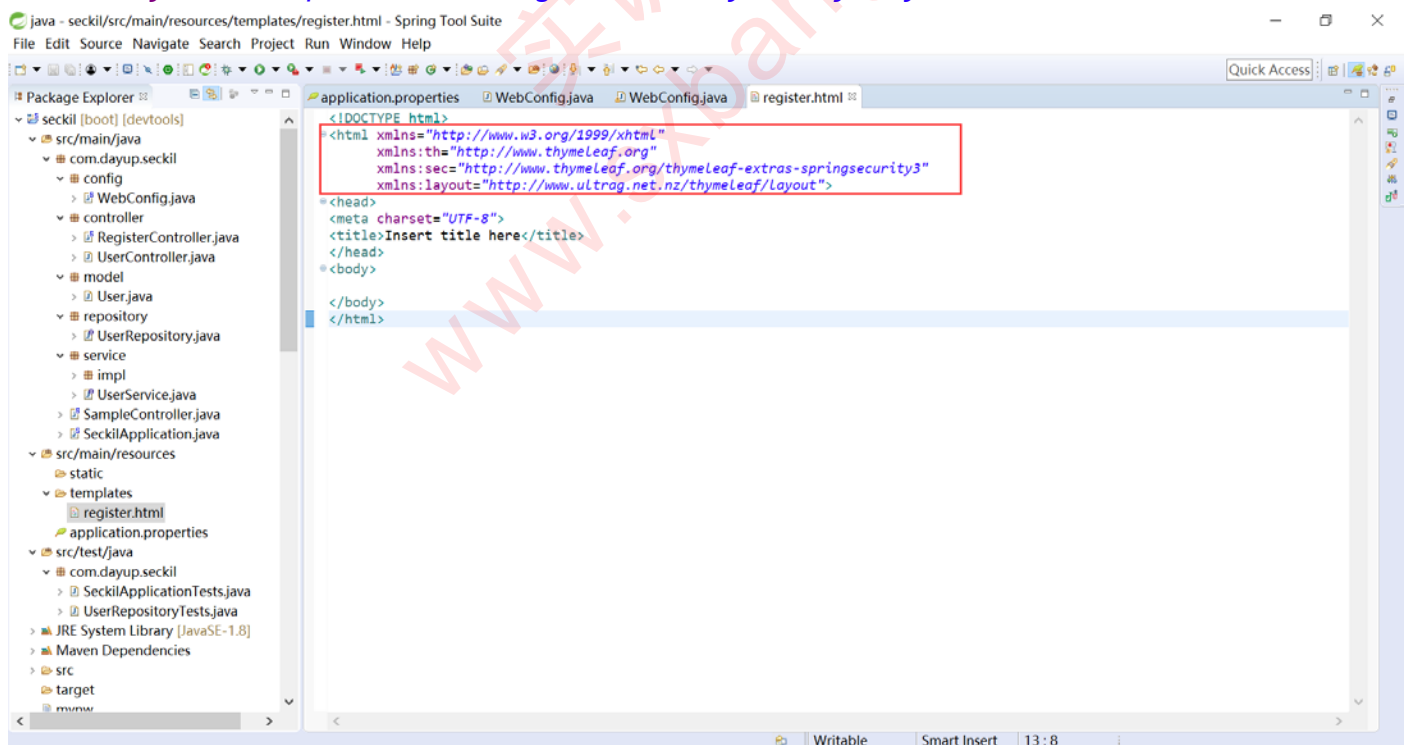






Step5.在<html>标签中加入:

```
xmlns="http://www.w3.org/1999/xhtml"
xmlns:th="http://www.thymeleaf.org"
xmlns:sec="http://www.thymeleaf.org/thymeleaf-extras-springsecurity3"
xmlns:layout="http://www.ultrag.net.nz/thymeleaf/layout"
```



Step6. 接着引入 Bootstrap 的 css 样式。

首先下载 bootstrap,网址: <https://v3.bootcss.com/getting-started/#download>, 建议下载生产环境的。

下载

Bootstrap (当前版本 v3.3.7) 提供以下几种方式帮你快速上手, 每一种方式针对具有不同技能等级的开发者和不同的使用场景。继续阅读下面的内容, 看看哪种方式适合你的需求吧。

用于生产环境的 Bootstrap

编译并压缩后的 CSS、JavaScript 和字体文件。不包含文档和源码文件。

下载 Bootstrap

Bootstrap 源码

Less、JavaScript 和字体文件的源码, 并且带有文档。需要 Less 编译器和一些设置工作。

下载源码

Sass

这是 Bootstrap 从 Less 到 Sass 的源码移植项目, 用于快速地在 Rails、Compass 或只针对 Sass 的项目中引入。

下载 Sass 项目

使用 BootCDN 提供的免费 CDN 加速服务 (同时支持 http 和 https 协议)

Bootstrap 中文网 为 Bootstrap 专门构建了免费的 CDN 加速服务, 访问速度更快、加速效果更明显、没有速度和带宽限制、永久免费。BootCDN 还对大量的前端开源工具库提供了 CDN 加速服务, 请进入[BootCDN 主页](#)查看更多可用的工具库。

```
<!-- 最新版本的 Bootstrap 核心 CSS 文件 -->
<link rel="stylesheet" href="https://cdn.bootcss.com/bootstrap/3.3.7/css/bootstrap.min.css">
```

WEB前端 免费公开课

2007-2018
资深讲师 陈冠雄

下载

- 包含的内容
- 编译 CSS 和 JavaScript 文件
- 基本模板
- 实例精选
- 工具
- 社区
- 禁止响应式布局
- 从 2.x 版本升级到 3.0 版本
- 对浏览器和设备的支持情况
- 对第三方组件的支持
- 可访问性
- 许可证 FAQ

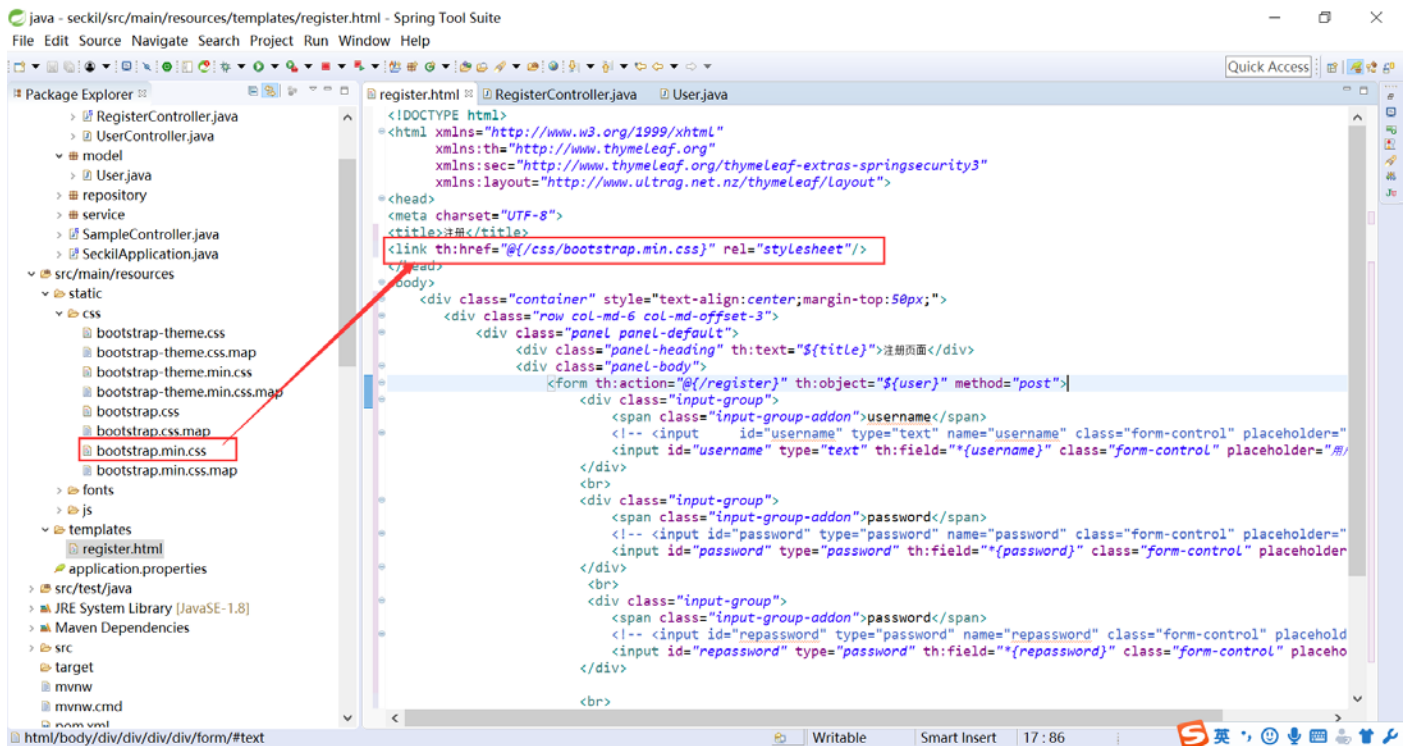
返回顶部

引入 bootstrap 文件: 括号里的路径是你 src/main/resources 下 static 文件夹里的文件的路径; 例如我这个是 src/main/resources -> static -> css/bootstrap.min.css。

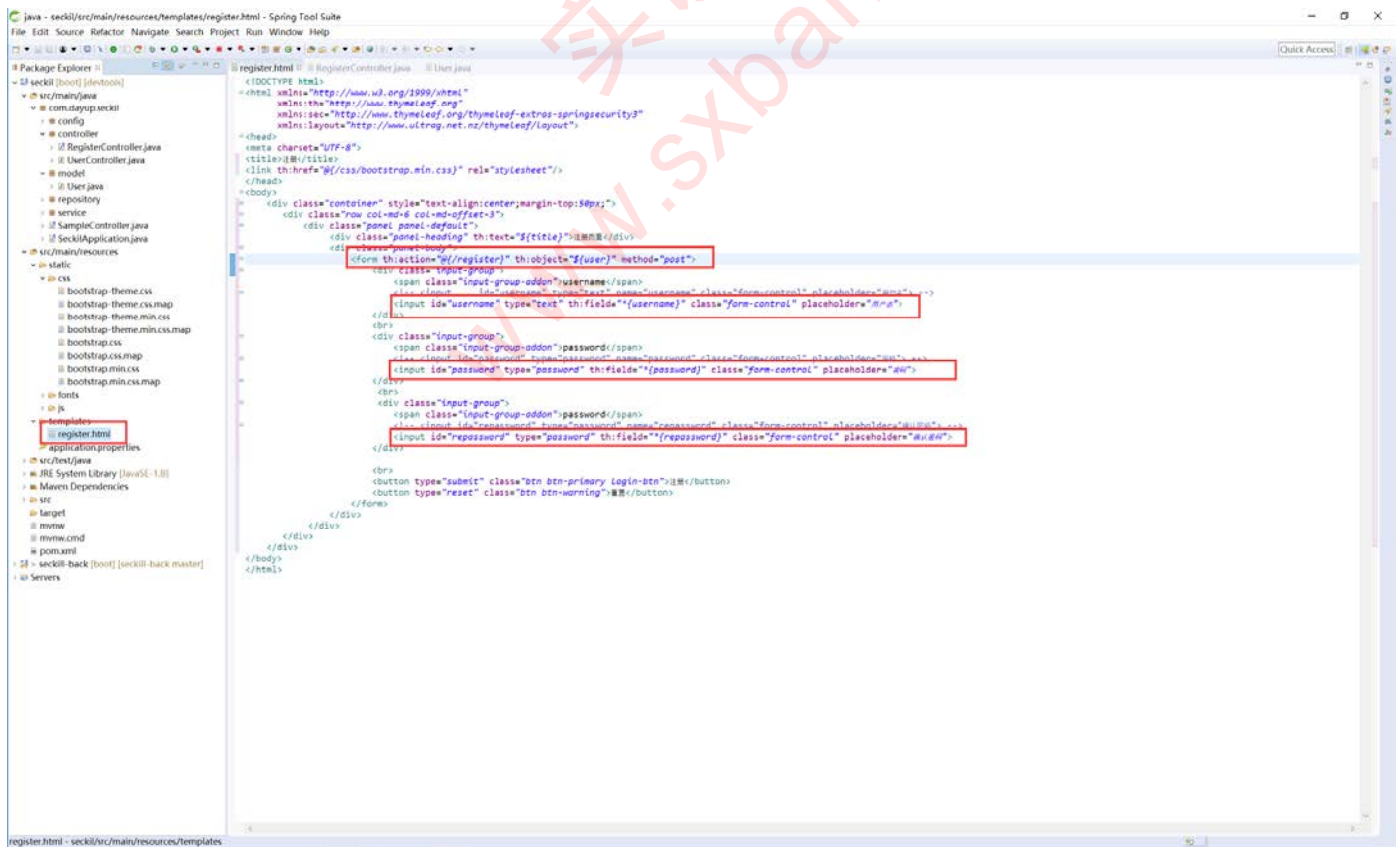
The screenshot shows an IDE window with the following components:

- File Explorer:** Displays the project structure. The 'src/main/resources' directory is expanded, showing a 'static' folder. Inside 'static', there are 'css', 'fonts', and 'js' folders. A red box highlights the 'css' folder, and a red arrow points to it with the word '复制' (Copy).
- Package Explorer:** Shows the project's package structure, including 'src/main/java', 'src/main/resources', 'static', 'templates', and 'application.properties'.
- Editor:** Displays the 'register.html' file, which contains HTML code. The code includes a link to the Bootstrap CSS file: `<link rel="stylesheet" href="https://cdn.bootcss.com/bootstrap/3.3.7/css/bootstrap.min.css">`.

9 / 16



3. 然后在 body 中加入页面元素, 并使用 Bootstrap 的样式进行美化; 然后在元素中使用 th:text 属性, 通过 {@(属性名)} 接受后台传过来的对象, 并利用 th:text 替换元素中的内容;



代码展示：

```

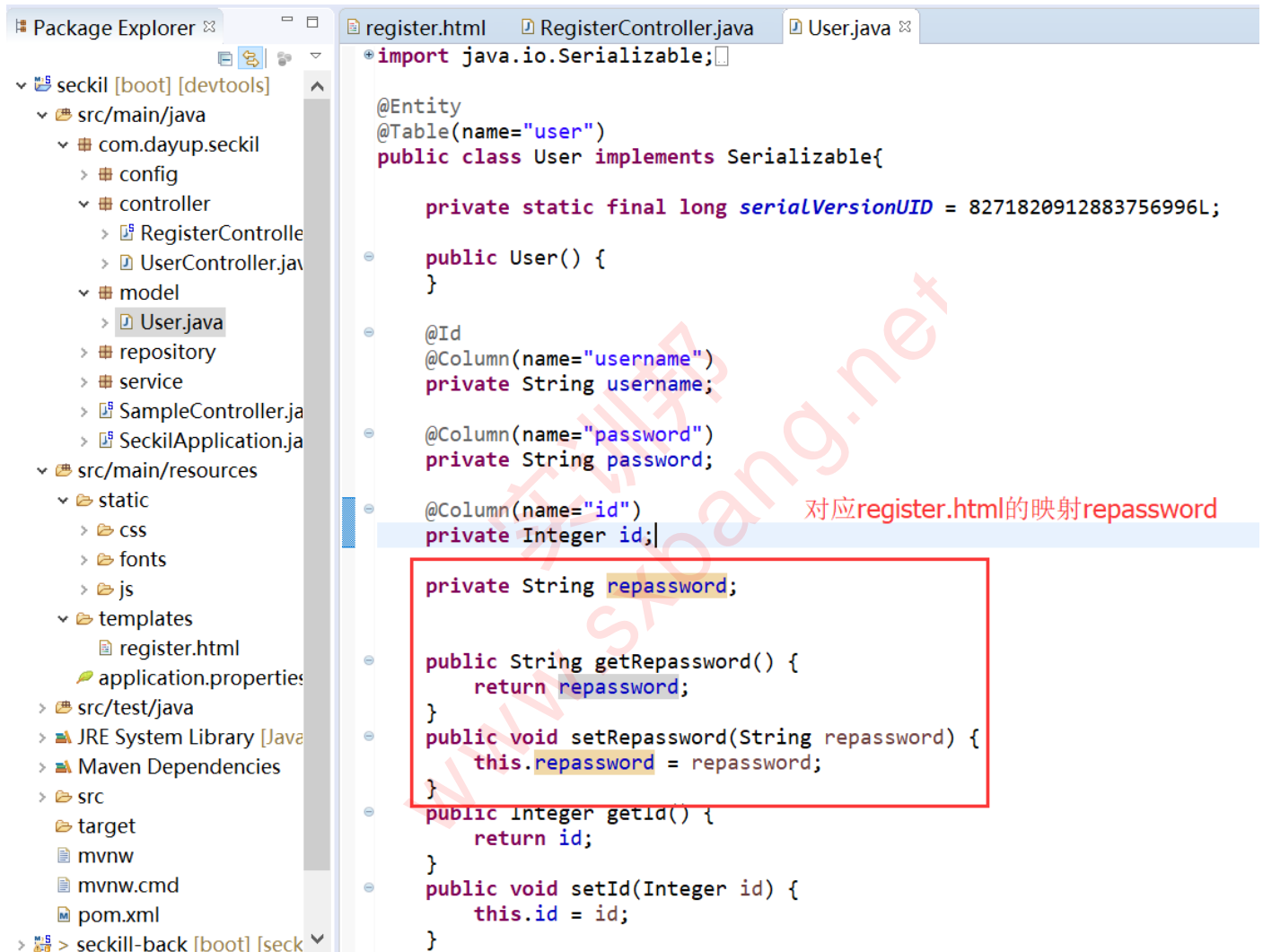
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:th="http://www.thymeleaf.org"
      xmlns:sec="http://www.thymeleaf.org/thymeleaf-extras-springsecurity3"
      xmlns:layout="http://www.ultrag.net.nz/thymeleaf/layout">
<head>
<meta charset="UTF-8">
<title>注册</title>
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet"/>
</head>
<body>
    <div class="container" style="text-align:center;margin-top:50px;">
        <div class="row col-md-6 col-md-offset-3">
            <div class="panel panel-default">
                <div class="panel-heading" th:text="${title}">注册页面</div>
                <div class="panel-body">
                    <form id="registerForm" th:action="@{/register}" th:object="${user}" method="post">
                        <div class="input-group">
                            <span class="input-group-addon">username</span>
                            <!-- <input id="username" type="text" name="username" class="form-control"
placeholder="用户名"> -->
                            <input id="username" type="text" th:field="*{username}" class="form-control"
placeholder="用户名">
                        </div>
                        <div><span th:if="${#fields.hasErrors('username')}" th:errors="*{username}"
style="color:red;"></span></div>
                        <br>
                        <div class="input-group">
                            <span class="input-group-addon">password</span>
                            <!-- <input id="password" type="password" name="password" class="form-
control" placeholder="密码"> -->
                            <!-- <input id="password" type="password" th:field="*{password}" class="form-
control" placeholder="密码" >-->
                            <input id="password" type="password" name="password" class="form-control"
placeholder="密码" >
                        </div>
                        <div><span th:if="${#fields.hasErrors('password')}" th:errors="*{password}"
style="color:red;"></span></div>
                        <br>
                        <div class="input-group">
                            <span class="input-group-addon">re-password</span>
                            <!-- <input id="repassword" type="password" name="repassword" class="form-
control" placeholder="确认密码"> -->
                            <input id="repassword" type="password" th:field="*{repassword}" class="form-
control" placeholder="确认密码" >
                        </div>
                    </form>
                </div>
            </div>
        </div>
    </div>

```

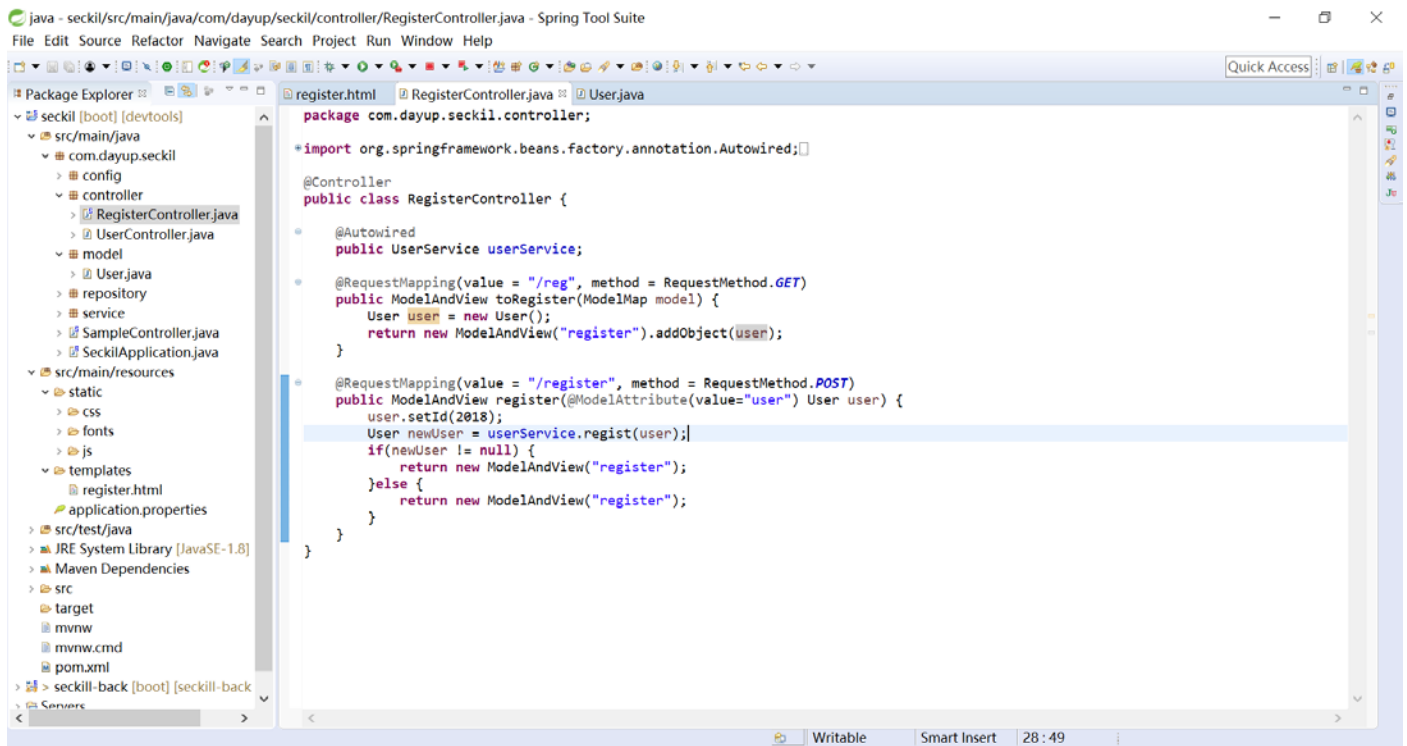
```

<br>
<button type="submit" class="btn btn-primary register-btn">注册</button>
<button type="reset" class="btn btn-warning">重置</button>
</form>
</div>
</div>
</div>
</body>
</html>

```



4. 页面编写完后，在 controller 包下新建 RegisterController 文件；在类名上添加需要用到的注解；定义一个 mapping 和方法，来进行页面的跳转；方法的返回值应该 ModelAndView 对象，返回时赋予一个字符串对象，值为我们要跳转的目标文件的文件名，这里我们要跳转到 register.html，所以字符串的值，就是 “register” 。



5. Controller 编写完后，重启 Spring boot app，重启完成后，在浏览器中使用 你的 ip: 端口 “/reg”（我的 是：localhost:8080/reg）来访问注册页面；可以看到，我的页面出现了一个表单，标题是 Register，说明接 受到后台的值并且替换了；然后样式也出现了，说明 Bootstrap 的样式也引入成功了。那么 Thymeleaf 的整合 就完成了。



源码参考

请从我们提供的码云上获取。

回马枪总结

Thymeleaf 总结:

✓ 数据访问模式:

1. `${...}`, 变量引用模式,

比如`${myBean.property}`, 如果用 `springDialect`, 则使用的是 `spring EL`, 如果不用 `spring`, 则用的 `ognl`。

2. `*{...}`, 选择表达式:

一般是 `th:object` 之后, 直接取 `object` 中的属性。当没有选取对象时, 其功能等同`${...}`,

`*{firstName}`也等同于`${#object.firstName}`, `#object` 代表当前选择的对象。

3. `@{...}`链接 url 的表达式:

`th:href="@{/xxx/aa.do(id=${o.id})}"`, 会自动进行 `url-encoding` 的处理。`@{...}`内部可以是需要计算的表达式,

比如: `th:href="@{/details/' + ${user.login}(orderId=${o.id})}"`

4. `#{...}`, `i18n`, 国际化。

需要注意的:

`#${welcomeMsgKey}(${session.userName})`: `i18n message` 支持占位。

✓ 表达式基本对象:

`#ctx`: context object

`#root` 或者 `#vars`

`#locale`

#HttpServletRequest

#httpSession

✓ 表达式功能对象：

#dates: java.util.Date 的功能方法类。

#calendars:类似#dates, 面向 java.util.Calendar

#numbers:格式化数字的功能方法类。

#strings: 字符串对象的功能类, contains,startWiths,prepending/appendig 等等。

#objects:对 objects 的功能类操作。

#bools:对布尔值求值的功能方法。

#arrays: 对数组的功能类方法。

#lists:对 lists 功能类方法

#sets

#maps

#aggregates:对数组或者集合创建聚合的功能方法,

th:text="\${#aggregates.sum(o.orderLines.{purchasePrice * amount})}"

#messages:在变量表达式中获取外部信息的功能类方法。

#ids: 处理可能重复的 id 属性的功能类方法。

✓ 条件操作：

(if)?(then):满足条件, 执行 then。

(if)?(then):(else)

(value)?:(defalutValue)

✓ 常用标签：

th:action, 定义后台控制器路径, 类似<form>标签的 action 属性。

例如: <form id="login-form" th:action="@{/login}">...</form>

th:each, 对象遍历, 功能类似 jstl 中的<c:forEach>标签。

th:field, 常用于表单字段绑定。通常与 th:object 一起使用。 属性绑定、集合绑定。

例如:

```
<form id="login-form" th:action="@{/login}" th:object="${loginBean}">...
```

```
<input type="text" value="" th:field="*{username}"></input>
```

```
<input type="text" value="" th:field="*{user[0].username}"></input>
```

```
</form>
```

th:href, 定义超链接, 类似<a>标签的 href 属性。value 形式为@{/logout}

例如: <a th:href="@{/logout}" class="signOut">

th:id, div id 声明, 类似 html 标签中的 id 属性。

例如: <div class="student" th:id="stu+(\${rowStat.index}+1)"></div>

th:if, 条件判断。

例如: <div th:if="\${rowStat.index} == 0">... do something ...</div>

th:src, 用于外部资源引入, 类似于<script>标签的 src 属性, 常与@{}一起使用。

例如: <script th:src="@{/resources/js/jquery/jquery.json-2.4.min.js}"

th:text, 文本显示。

例如: <td class="text" th:text="\${username}"></td>

th:value, 用于标签复制, 类似<option>标签的 value 属性。

例如:

```
<option th:value="Adult">Adult</option>
```

```
<input id="msg" type="hidden" th:value="${msg}" />
```