**SpringBoot的学习**

（这里只记录我所疑惑的地方，具体笔记在E盘的笔记里）

可以在spring官网上选择相应的springBoot模板，然后以压缩包的形式下载，然后在IDEA打开。IDEA也可以直接创建相应的SpringBoot工程，在新建工程栏选择Spring Initializr，然后选择所需要的依赖。（一般我们就选择Lombok、web、connecter。其他需要的后来可以再加）。

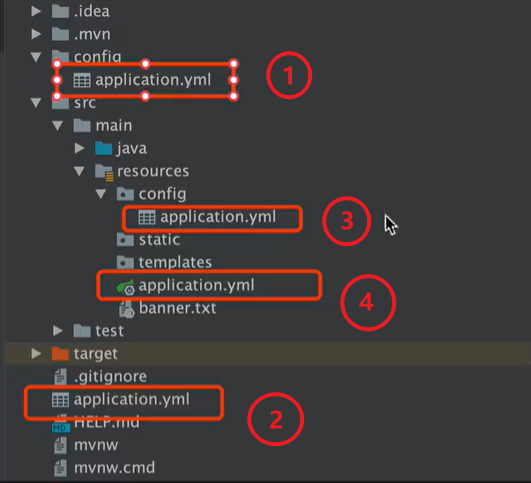
Springboot项目内嵌Tomcat，可以直接启动，然后是springboot的默认启动类：



这里的默认启动类SpringBootApplication所属的包为springboot。这里的springboot包必须覆盖所有的义务类。否则将自动装备到IOC失败。

底层是：启动类注解@SpringBootApplication，下面的@EnableAutoConfiguration、@AutoConfigurationPackage下面的@Import注解中的Registrar，其中有一个registerBeanDefinitions方法，指向拥有@SpringBootApplication注解的类的包名，即为springboot包

其中YMAL格式的配置文件的存放位置一共有四个地方可以存放，springboot可以直接读取到：



读取的优先级顺序如下所示：

1. 根路径下的config文件夹中的application.yml
2. 根路径下的配置文件
3. Resources路径下的config文件中的配置文件
4. Resources路径下的配置文件

**SpringBoot整合jsp**

创建Maven工程，选择带有webapp的组件，然后pom.xml中：

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>org.example</groupId>

<artifactId>springbootjsp</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>war</packaging>

<name>springbootjsp Maven Webapp</name>

<!-- FIXME change it to the project's website -->

<url>http://www.example.com</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<maven.compiler.source>1.7</maven.compiler.source>

<maven.compiler.target>1.7</maven.compiler.target>

</properties>

<!--首先引进它的父依赖，父依赖相当于是一个大集合，然后再从父依赖中选取需要的各种依赖（springboot也是一样的操作，只不过不需要手动来做）-->

<parent>

<artifactId>spring-boot-starter-parent</artifactId>

<groupId>org.springframework.boot</groupId>

<version>2.5.3</version>

</parent>

<dependencies>

<dependency><!--这里就不需要选版本了，因为父依赖的版本已经选好了-->

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.apache.tomcat.embed</groupId>

<artifactId>tomcat-embed-jasper</artifactId>

</dependency>

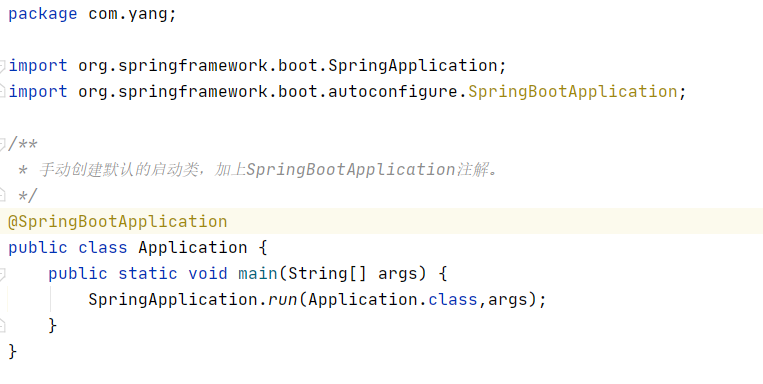
</dependencies>

然后进行测试：

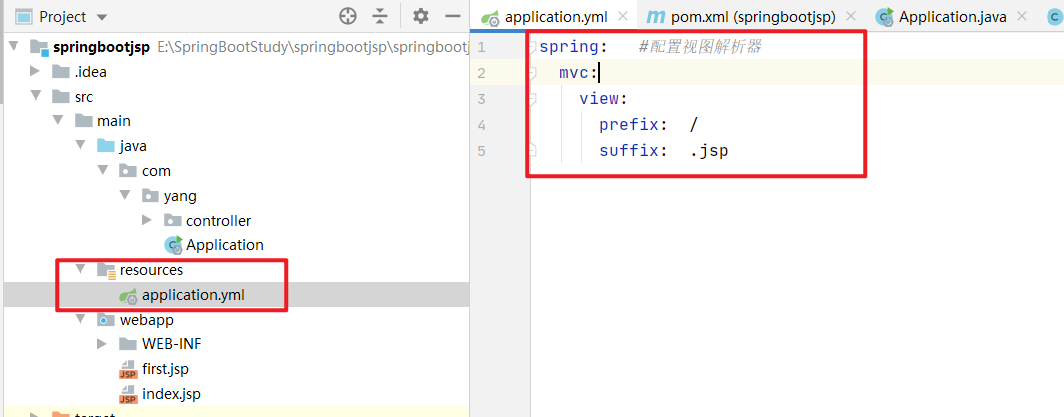
首先创建对应包名，然后创建controller包，创建HelloHandler，然后为其添加@Controller注解（注意，这里不是添加@RestController注解了，因为需要带上jsp视图），然后创建相应的测试类，返回带有view的ModelAndView模型数据，然后指向first.jsp



然后由于没有默认的启动类，所以需要手动创建启动类Application：



之后需要对返回的模型数据做视图解析，具体是创建对应的application.yml配置文件：



创建对应的resources文件夹，然后创建application.yml配置文件，在其中按照正确的格式添加解析器。与之前的xml文件一致都是spring.mvc.view.prefix/suffix：”/” “.jsp”。

注意！！！！！：这里的第一行我写过注释，第一行写注释，就会被看做第一行是空行，就会报错。。

最后创建对应的jsp文件，使用EL表达式直接从域当中取出自己设置好的值即可：



**springboot整合Thymeleaf**

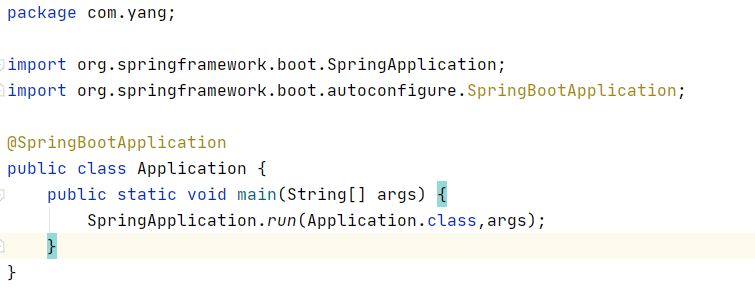


导入pom.xml依赖：

*<?*xml version="1.0" encoding="UTF-8"*?>*<project xmlns="http://maven.apache.org/POM/4.0.0"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
 <groupId>org.example</groupId>  
 <artifactId>springbootthymeleaf</artifactId>  
 <version>1.0-SNAPSHOT</version>  
 *<!--导入父节点-->* <parent>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <groupId>org.springframework.boot</groupId>  
 <version>2.5.3</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>  
</project>

然后在resources文件夹下创建application.yml配置文件，为thymeleaf配置好相应的视图解析器。和一些基本的servlet相关的配置。**注意**！：这里的解析器前缀一般是也放在resources文件夹下的templates文件夹当中，一般都这么写，还有它的suffix是：**.html**

之后配置Handler，创建controller包，在父包下创建默认的启动类，application：（固定写法）



Handler：

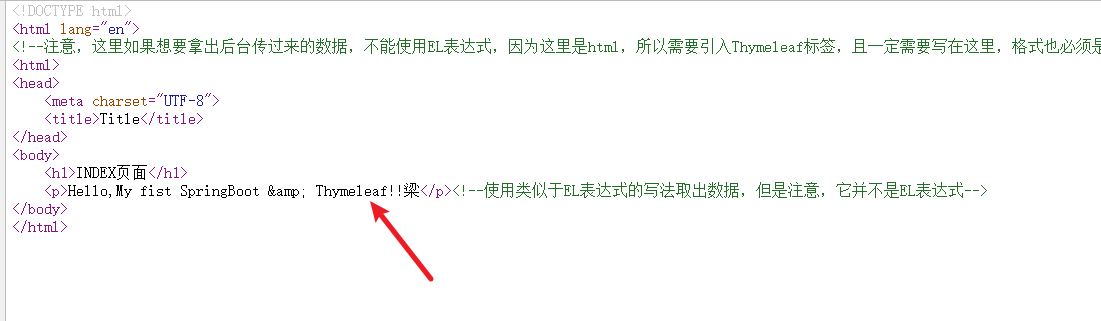


Resources文件夹下：



然后动态访问的时候（使用内嵌的tomcat打开）：

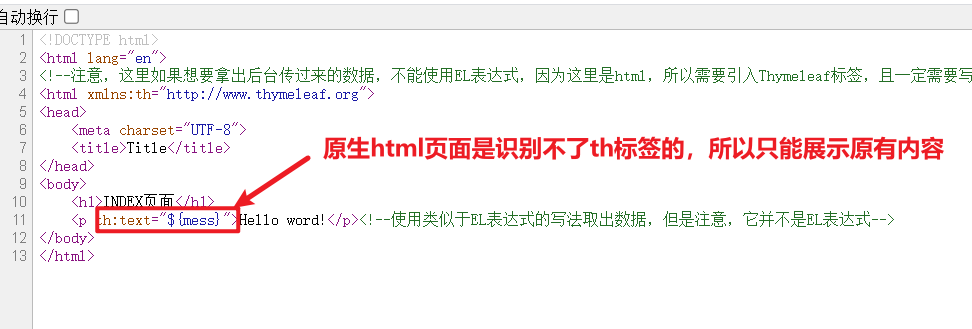




显示的就是我们使用Thymeleaf标签添加的内容

使用静态访问（直接将其在浏览器上访问）：



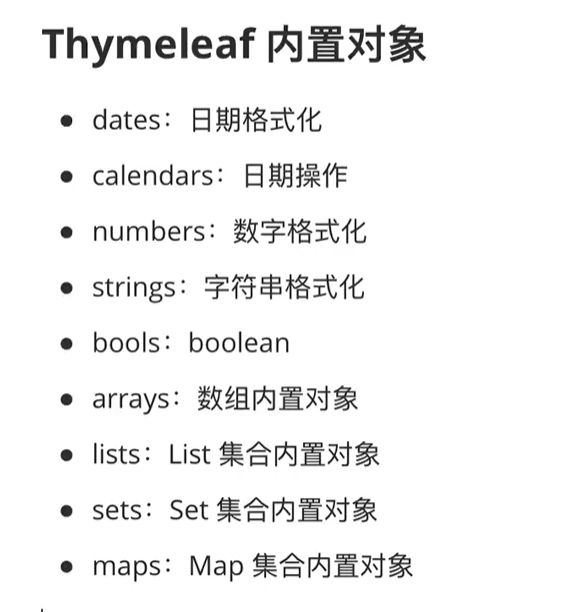


包括th:标签里面的 if、switch，case、action等的一些用法在IDEA中都有具体的实现。

**注意：**在客户端访问资源的时候，springboot是无法直接访问resources资源文件夹里的静态资源的（在网页里直接输入.html文件路径），**必须**通过controller内部访问。

但是，也是有办法可以在客户端输入uri，访问到静态资源的，需要在controller中专门写一个redirect资源定向的方法，然后获取到用户输入的uri，并作为返回值





具体整合举例方法，IDEA有详细代码

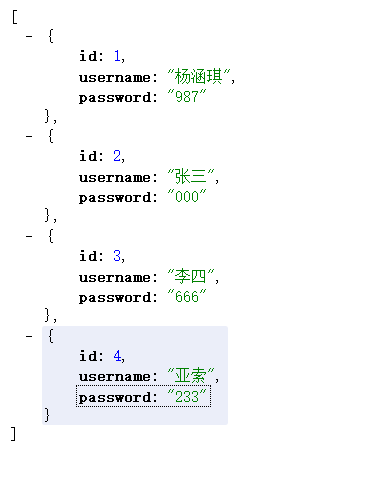
**Springboot整合jdbcTemplate**

创建maven工程，导入依赖，然后在resources文件夹下创建yml配置文件。然后创建数据库表格对应的实体类，然后在dao（repository）包中创建对应的接口和具体的实现类，最后创建controller。由于创建的maven工程没有选择选择web，所以需要手动实现默认的启动类。

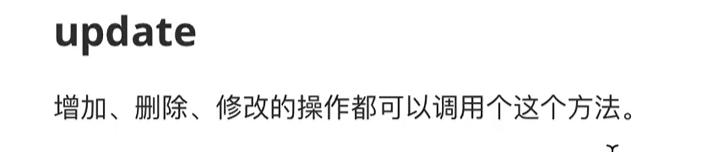
注意：由于服务器上传至客户端的是json数据格式，要想清晰的观察json格式，可以在Chrome浏览器安装json-view插件。方法：

**https://blog.csdn.net/qq\_35771266/article/details/99419757**

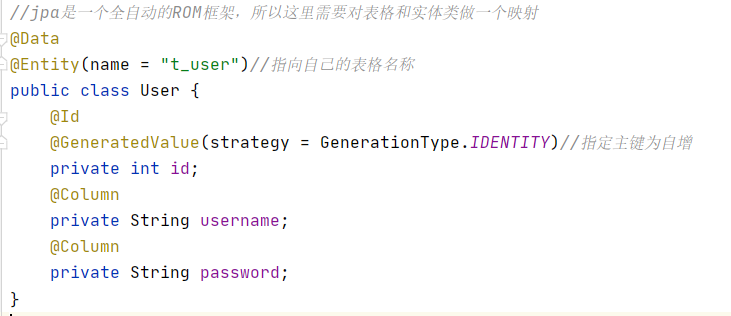
效果图：



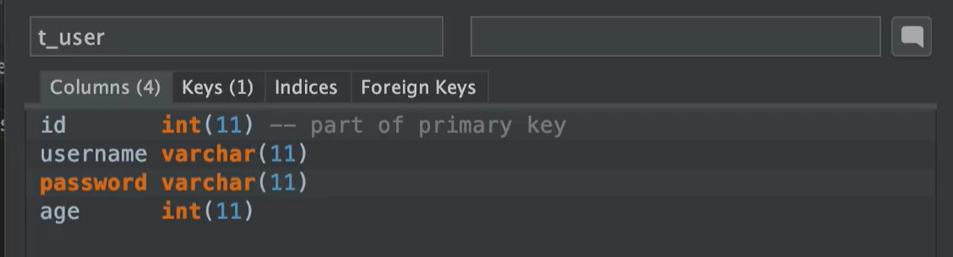
JdbcTemplate是springboot自带的一个组件，所以不需要再单独手写这个类，关于jdbcTemplate的使用，使用restful风格分别针对不同的请求使用PostMan进行测试.



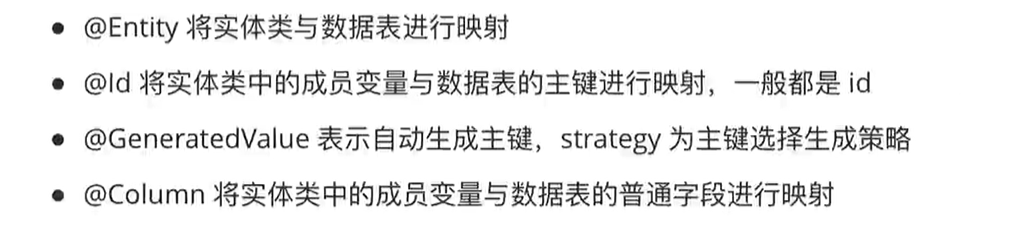
**Springboot 整合spring Data JPA**



表格：



注解解释：



接口：

接口只需要继承JPA自己封装好的接口即可，其中支持CRUD操作，所以如果只是执行一般的一些操作，完全不需要自己手动来写。

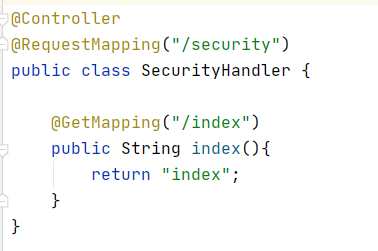
**注意：**当常规的业务方法无法满足需求的时候，就需要在这个JpaUserRepository接口中进行扩充 注意，扩充的方法只需要写出方法名即可，不需要具体实现。它会自动帮助实现（有够智能）

**Springboot整合spring security**

1. 首先导入相关依赖：

<parent>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <groupId>org.springframework.boot</groupId>  
 <version>2.5.3</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>  
  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-security</artifactId>  
 </dependency>  
</dependencies>

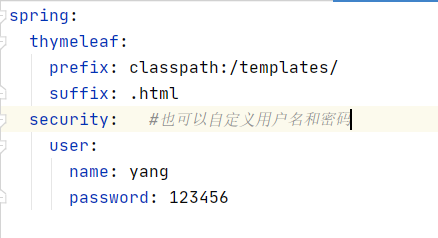
1. 创建handler：



3、创建相关html资源（这里使用thymeleaf静态模块资源）

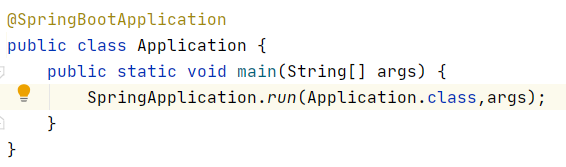


1. 创建application.yml配置文件，添加相关配置：



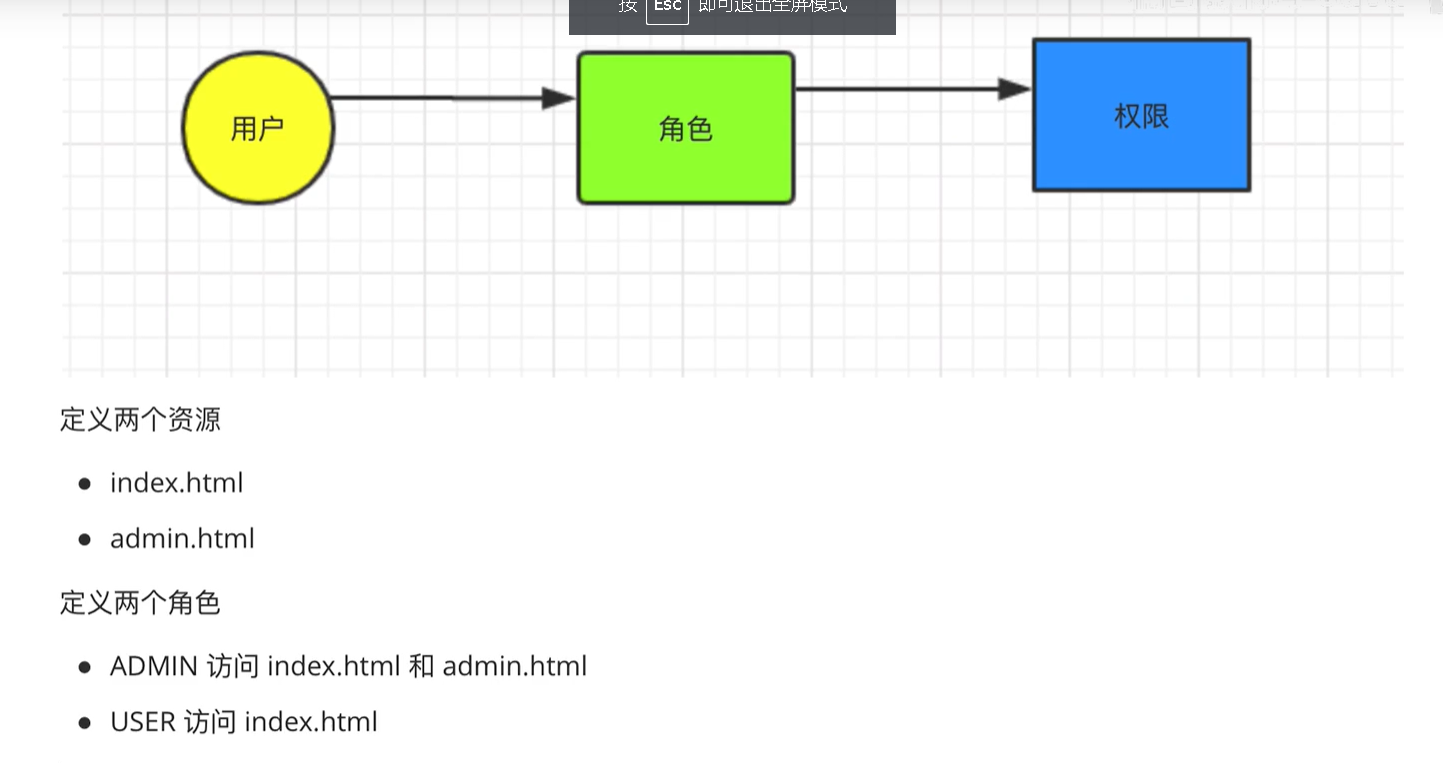
Security模块会自动再控制台为每一次访问动态创建一个字符串作为密码，但是用户名默认的是user。也可以自己指定它的用户名和密码。

1. 创建Application启动类：

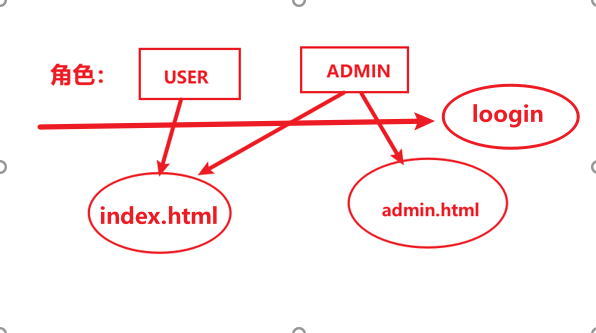


最后在浏览器上访问即可。

**注意：**这里的权限并不是直接赋给用户的，而是通过创建的角色，把用户指向拥有不同权限的角色，然后用户才间接获得的权限。



Security模块的具体应用：



关系就是当我们想要访问resources文件夹下的templeates的thymeleaf静态资源的时候，配置文件创造了两个角色，分别对应了不同的权限，其中：USER、ADMIN都可以访问index.html，但是admin.html只有ADMIN有权限访问。Spring security模块在对其操作的时候，全称都需要进行权限验证，所以在用户访问后台资源之前，就需要通过login，通过输入的账号密码来进行身份角色权限的验证。

具体的代码实现（**这里只记录大致实现思想，详见IDEA编译器里的springbootsecurity工程**）：

它需要一个config文件夹，其中包括MyPasswordEncoder和一个SecurityConfig配置文件（**配置文件东西很多很多，多看**），具体的逻辑关系就是，访问spring security的资源的时候，需要进行权限验证，然后我们手写了权限验证login的页面，然后为其拟定了两个角色，分别对应不同的权限，当我们访问index.html的时候，拥有USER和ADMIN这两个角色的用户都可以访问，但是当我们访问admin.html的时候，只有ADMIN角色对应的用户才可以访问，在访问admin.html的时候，如果输入的是USER信息，那么就会自动跳转至index.html对应的页面。