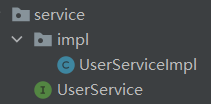
**Spring boot 三层框架dao层、service层、controller层+实体model层**

**model层**也叫**pojo层，**一般数据库的一张表对应一个pojo层，get和set两个方法，而有了lombok之后就可以不用写

**dao层**也叫**mapper层**，数据持久层，针对数据库操作的，基本用到的就是增删改查

**service层**叫**业务逻辑层**，存放业务逻辑处理，不直接对数据库进行操作，有**接口和接口实现类**，提供controller层调用的方法，创建两个文件，一个存放接口类，一个存放接口实现类。



**controller层**叫**控制器层**，负责**前后端交互**，接受前端请求，调用service层，接收service层返回的数据，最后返回具体的页面和数据到客户端。

同样，也需要两个文件，接口类和接口实现类。

基于**@Component**注解有三个扩展，分别是：**@Repository 、@Service、 @Controller**，他们只是分别具有不同的业务含义，但是被@component 、@Controller 、@Service、@Repository 注解的类，都会把这些类纳入进Spring容器中进行管理。

1、@Component：标注一个普通的POJO实例化到Spring容器中，它是一个通用泛化的注解，当某个组件不能通过业务进行归类时，可以使用这个注解进行标注。

2、@Controller： 用于标注控制层组件。

3、@Service： 用于标注业务层组件。

4、@Repository ： 用于标注持久层组件。



@GetMapping： 处理get请求，传统的RequestMapping来编写应该是

@PostMapping： 处理post请求，传统的RequestMapping来编写应该是

@**PutMapping**： 和PostMapping作用等同，都是用来向服务器提交信息。如果是添加信息，倾向于用@PostMapping，如果是**更新信息**，倾向于用@PutMapping。两者差别不是很明显。

@DeleteMapping 删除URL映射，具体没有再实践中用过，不知道好在什么地方

@**PatchMapping** 用于**局部更新**

1. java无法识别：pom中jdk和maven版本设对，我的是1.8
2. org.apache.ibatis.binding：

spring-boot-starter-parent版本降低2.3.6.RELEASE

mybatis-spring-boot-starter版本降低2.2.2

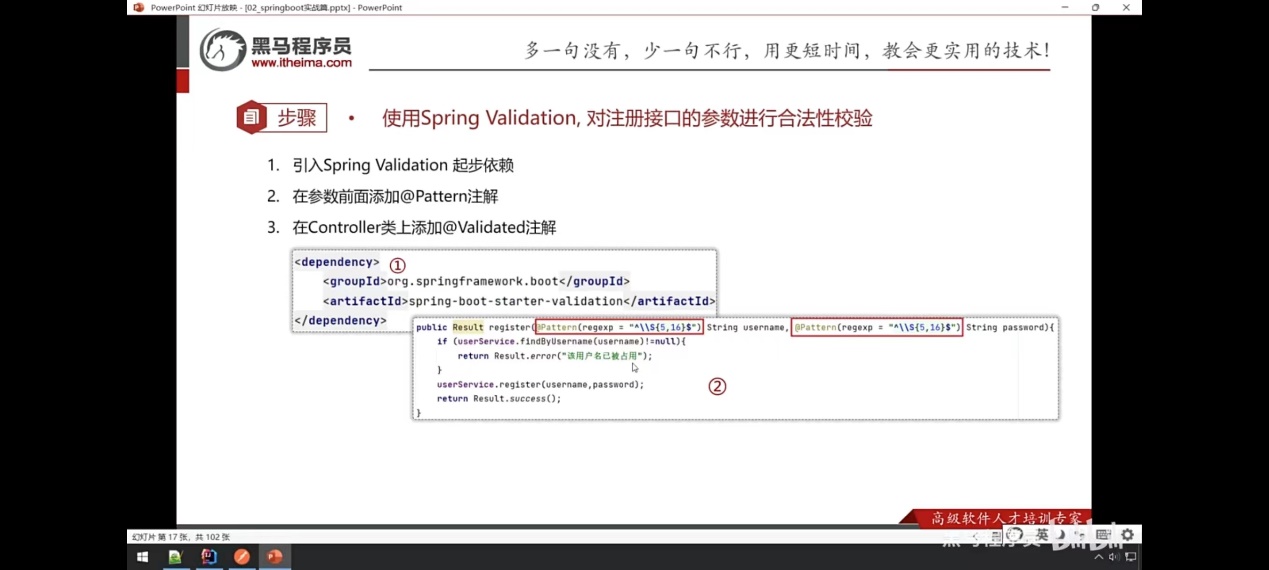
1. idea中的dataase使用要testconnect,降低Driver版本
2. Mapper无法识别：@MapperScan("com.itheima.mapper")指明包位置
3. lombok依赖后记得@Data创建get和set方法

@NoArgsConstructor//无参构造方法自动创建

@AllArgsConstructor//全参构造方法自动创建

1. **参数校验：validation**

**注意两步骤**



**实体类中**

@NotEmpty：不能不传，不能为空

@NotNull：不能不传

@Email

@Pattern(regexp = "^\\S{1,10}$")

格式化输出

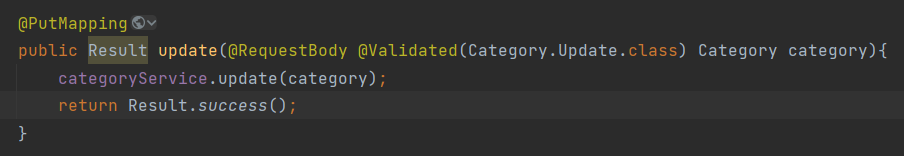
@JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss")

**分组校验**

**//如果某个校验没有指定分组，默认属于Default分组**

**//分组之间可以继承**

**指定分组：**

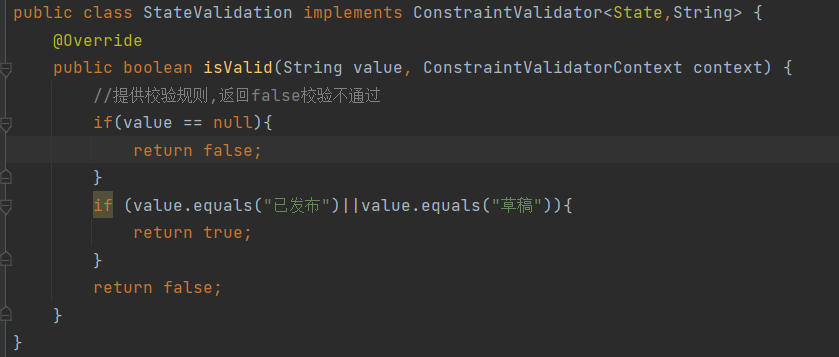


****

**自定义校验：**

**模仿官方给的（去官方那复制）**





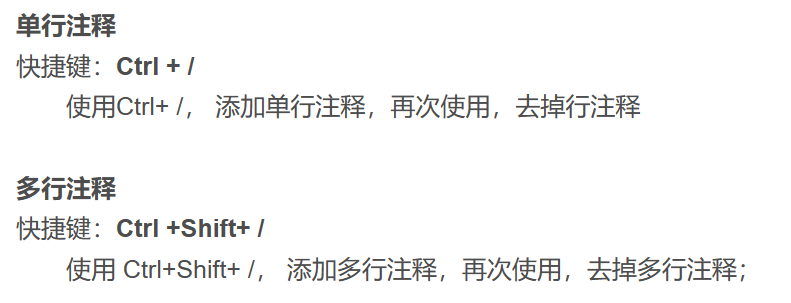


1. jwt令牌

为了支持非对称的Map版本注意（这次使用了4.4.0）

按住ctrl + alt + t快捷try/catch

快捷注释



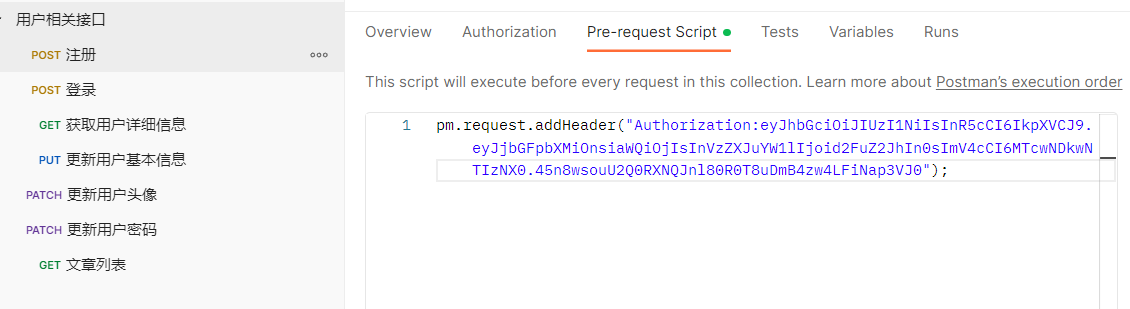
1. 拦截器

借助ThreadLocal将用户数据隔离共享



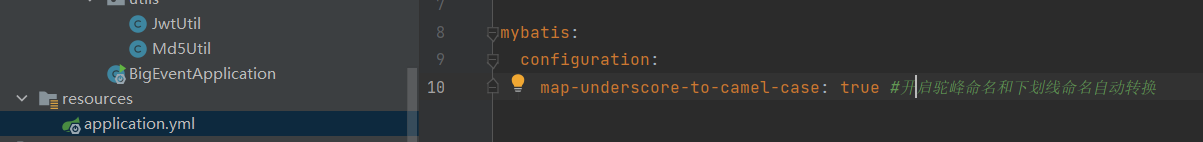
使用redis解决：修改密码后旧令牌依旧有效，比对redis中和浏览器携带的令牌

1. 测试

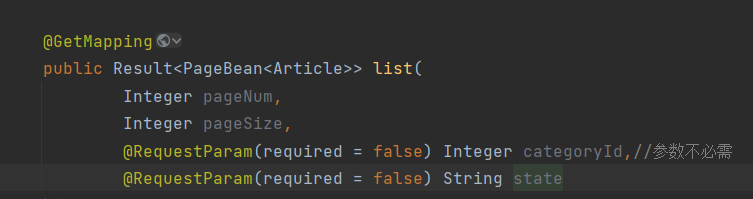


postman统一添加请求头，记得保存

yml中开启驼峰下划线自动转换



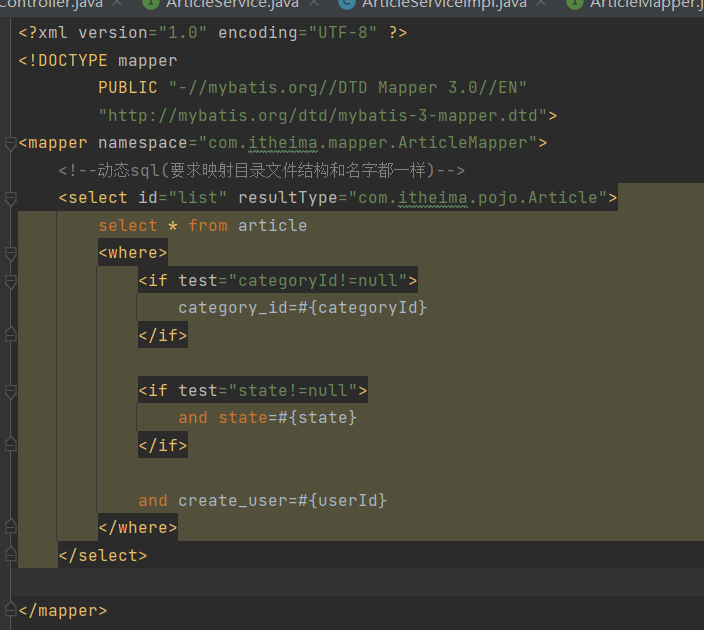
1. 参数



设置参数不必须

1. 映射配置文件处理动态sql语句

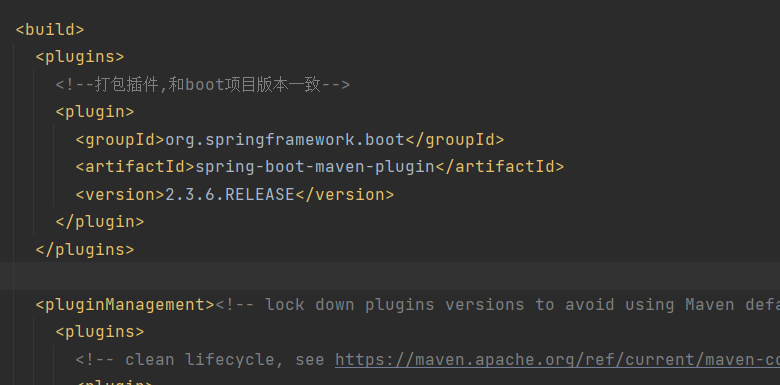
不写注解



1. 打包

Maven中的package执行打包（clean可以删除）

(被坑死了，别放到自动生成的标签pluginManagement下面啊)



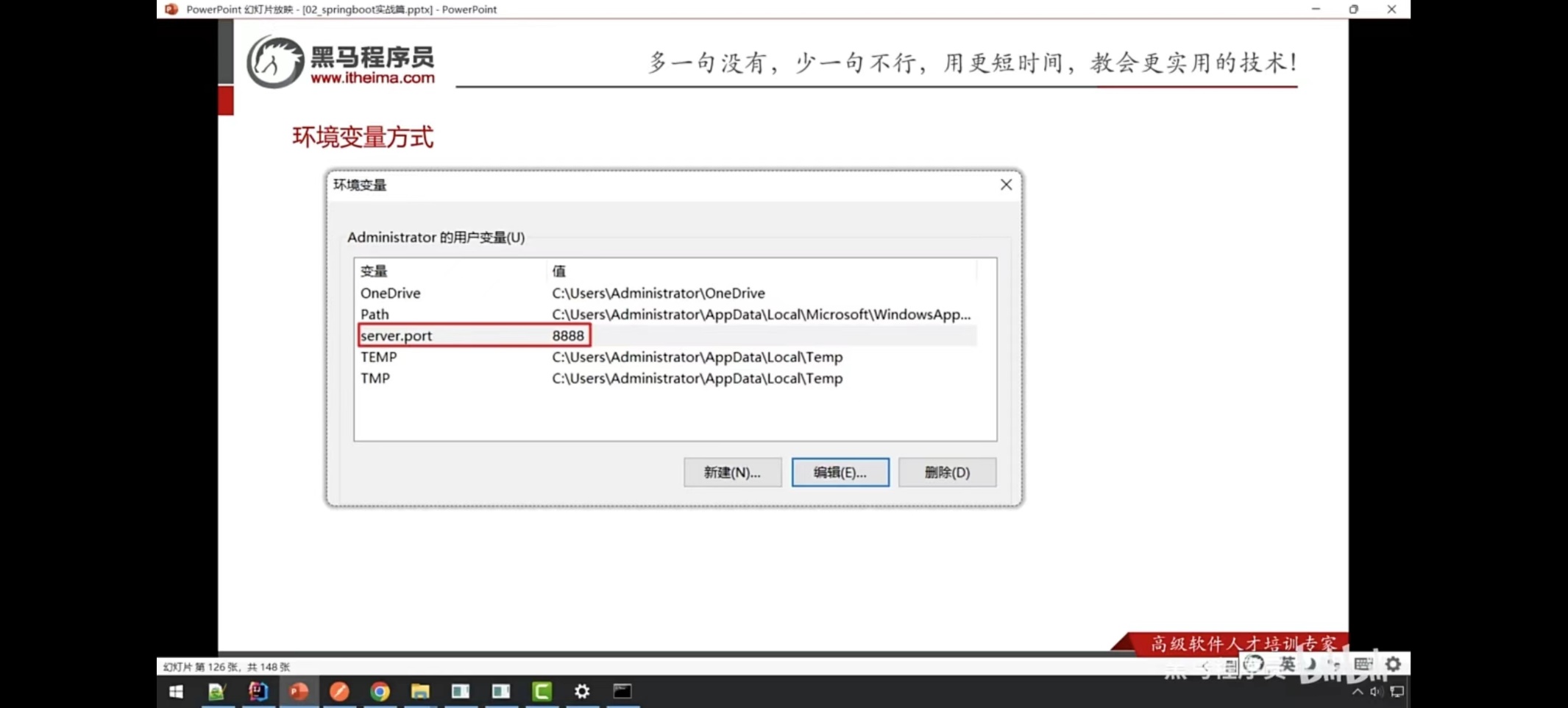
**传参方式修改端口**

Java -jar “xxx.jar”执行

(小tip在cmd界面运行可以使用tab补全文件名)

可以跟参数--server.port=9999

**环境变量方式修改端口**



**Yml文件配置端口（项目中和jar包目录下都可以）**

**server:**

**port: 8081**

**优先级：命令行参数>环境变量>jar包下yml>项目中yml**

1. 多环境开发

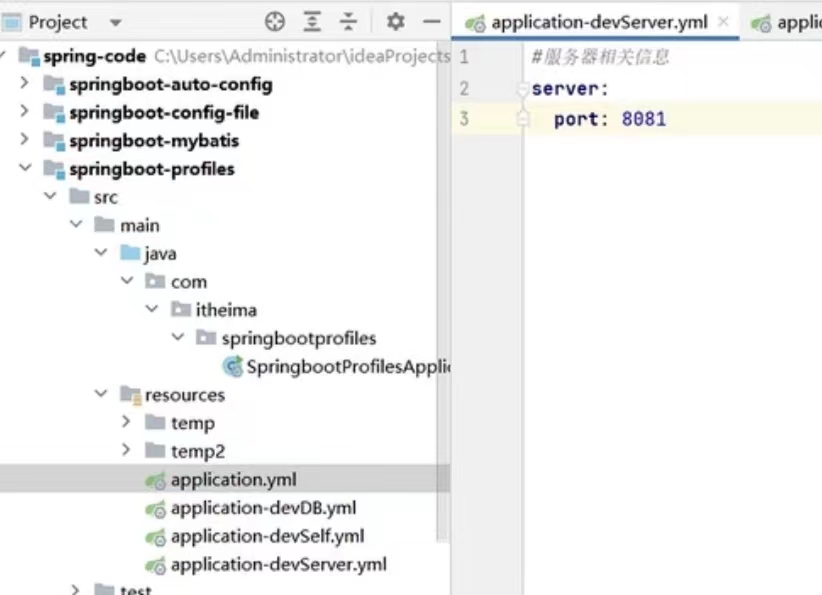
拆分成4个yml文件或写在一个yml文件---分割

---  
*#spring支持多环境开发，用“---”分割  
#通用信息，指定生效的环境,多环境下共性的属性  
#如果特定和通用配置冲突，特定环境生效*spring:  
 profiles:  
 active: dev  
server:  
 servlet:  
 context-path: /aaa  
---  
*#开发环境*spring:  
 config:  
 activate:  
 on-profile: dev  
server:  
 port: 8080

servlet:  
 context-path: /bbb  
---  
*#测试环境*spring:  
 config:  
 activate:  
 on-profile: test  
server:  
 port: 8081  
---  
*#生产环境*spring:  
 config:  
 activate:  
 on-profile: pro  
server:  
 port: 8082  
---

1. 分组环境
2. ---  
   *#spring支持分组配置*spring:  
    profiles:  
    group:  
    "dev": devServer,devDB,devSelf  
    "test": testServer,testDB,testSelf  
    active: dev

Application-“名字”



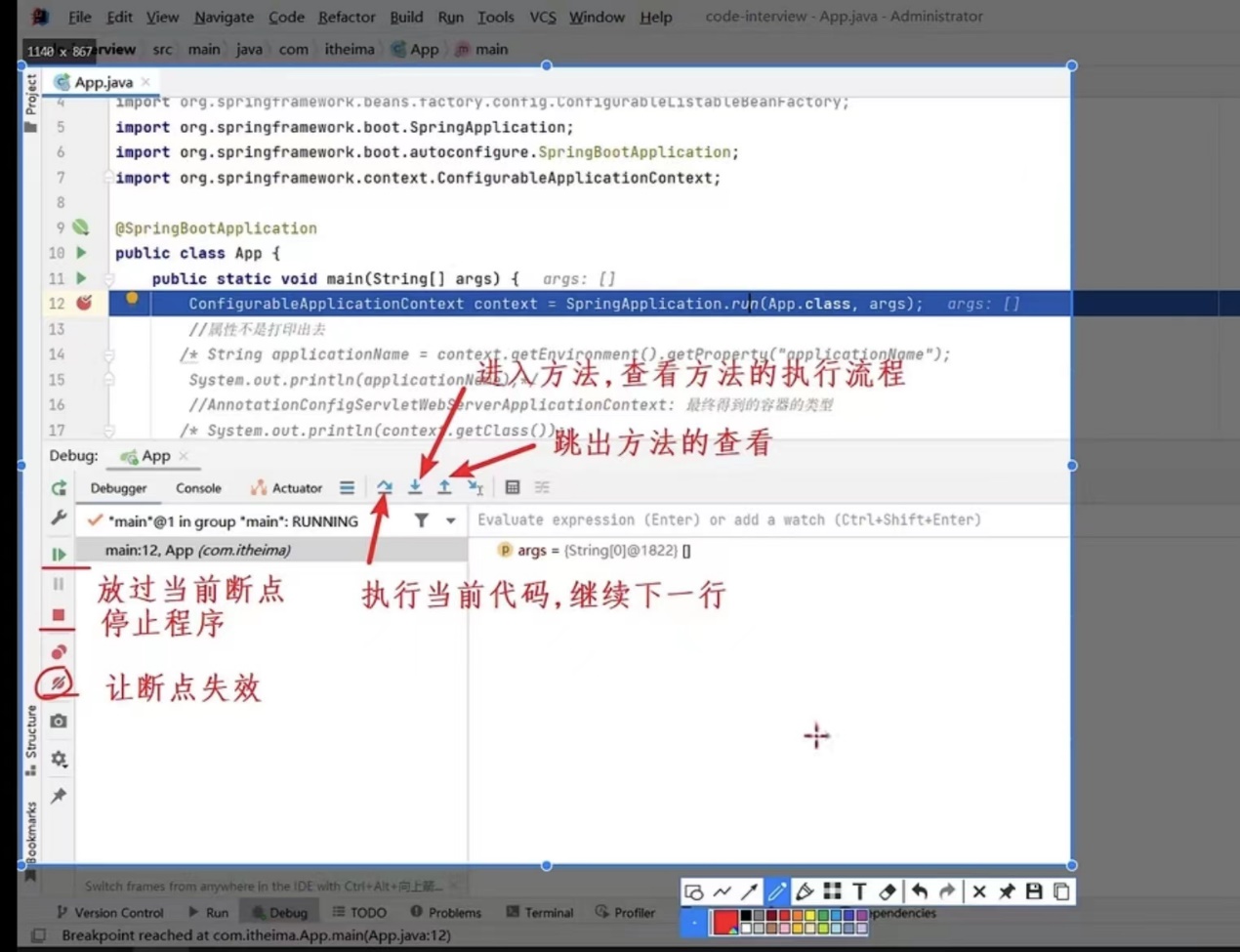
底层代码：





Bean容器（IOC容器）

@Component  
public class MyBeanFactoryPostProcesser implements BeanFactoryPostProcessor {  
 @Override  
 public void postProcessBeanFactory(ConfigurableListableBeanFactory beanFactory) throws BeansException {  
 GenericBeanDefinition gbdf = new GenericBeanDefinition();  
 gbdf.setBeanClass(Teacher.class);  
  
 //向下强转  
 DefaultListableBeanFactory dbf = (DefaultListableBeanFactory) beanFactory;  
 dbf.registerBeanDefinition("teacher",gbdf);  
   
 }  
}



Ctrl+f12搜索

在修改密码模块前端报了missing required prop，然后我就去后端看了看错误是can not desierlize String out of token,我以为是token不对，就前端info输出一下和Redis里面比对，发现是一样的，后来才发现是jeson的前后端key名字不对