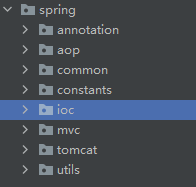
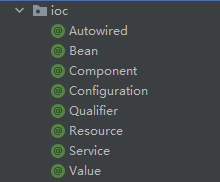
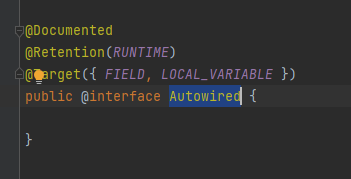
模块总览



1. Annotion - IOC

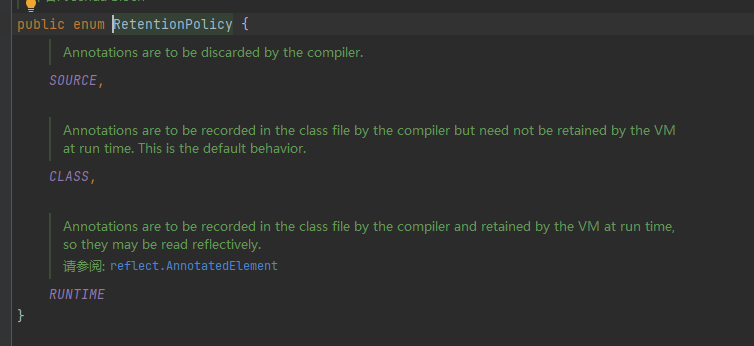


例子

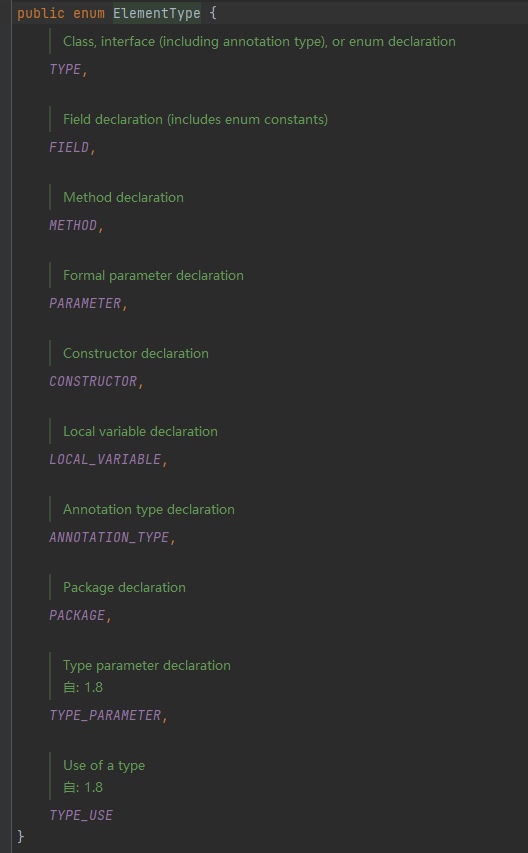


@Documented -> 给注解的注解，表明该注解会显示在文档上

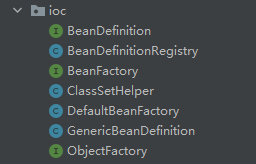
@Retention -> 注解的生命周期，分别对应.java, .class和字节码



@Target 注解的作用范围



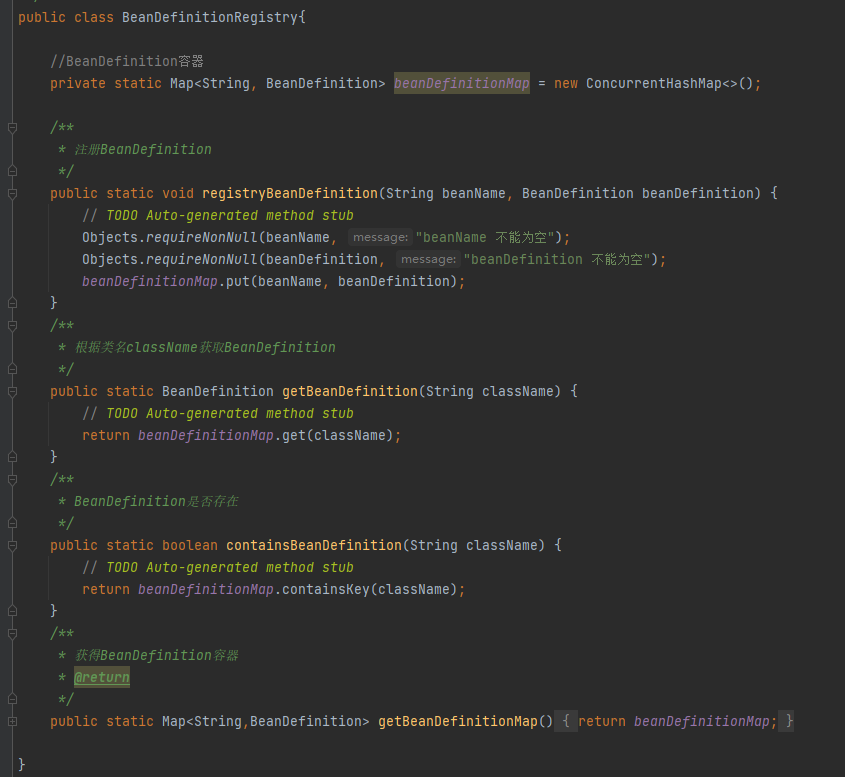
1. IOC实现



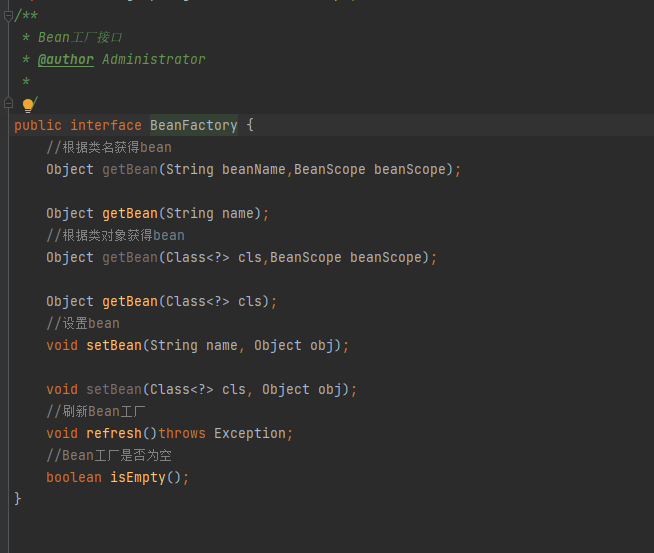
1. BeanDefinition Bean的定义（接口）



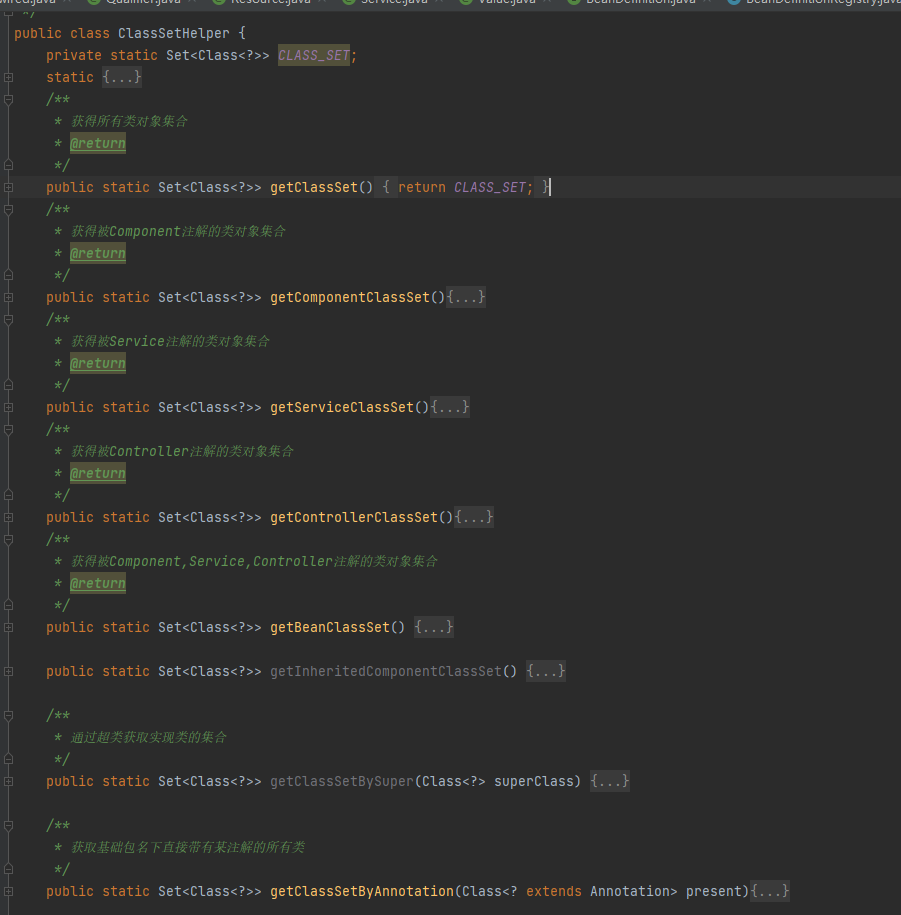
1. BeanDefinitionRegistry将bean的定义信息BeanDefinition注册到BeanDefinition容器中

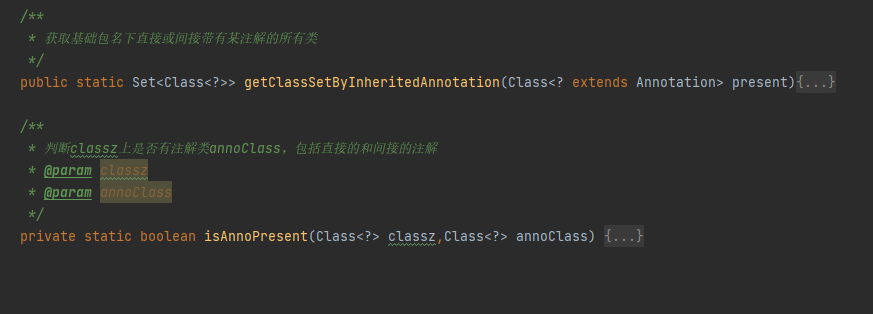


1. BeanFactory

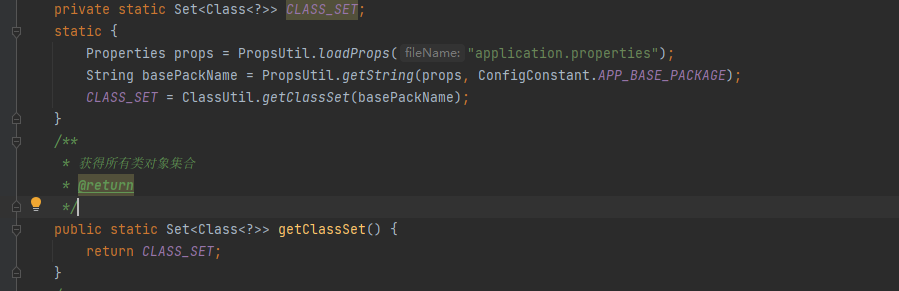


1. ClassSetHelper 类集合助手，可扫描配置文件中的包路径，获得指定类型或被指定注解的类对象集合





1. 获取所有类（单例构造）



1. 获取被@Component注解的类



1. 获取被@Service注解的类



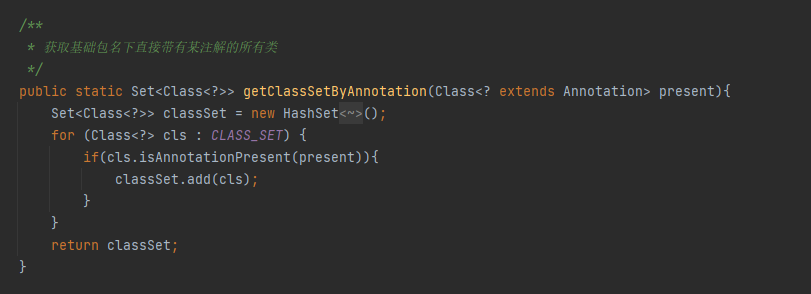
1. 获取被@Controller注解的类



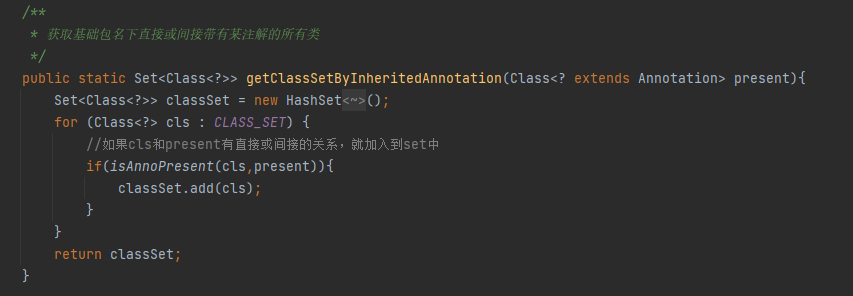
1. 通过超类获取类的集合



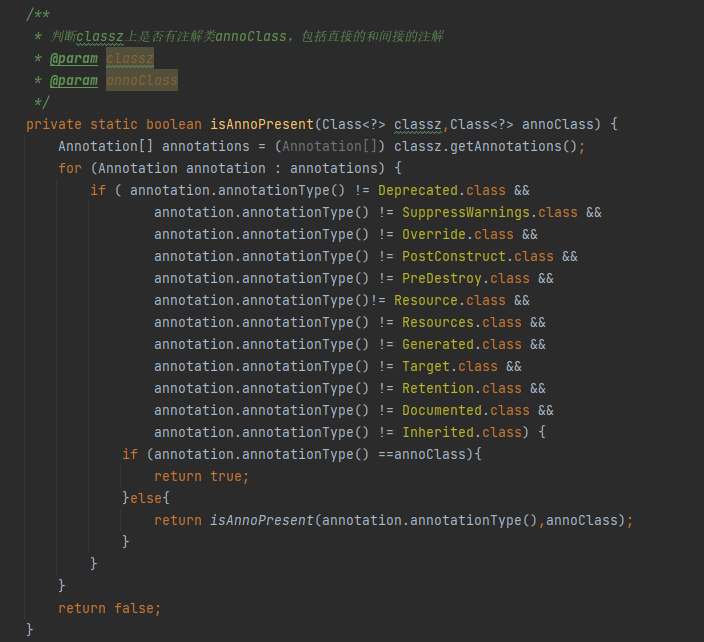
1. 获取基础包下直接带有某注解的所有类



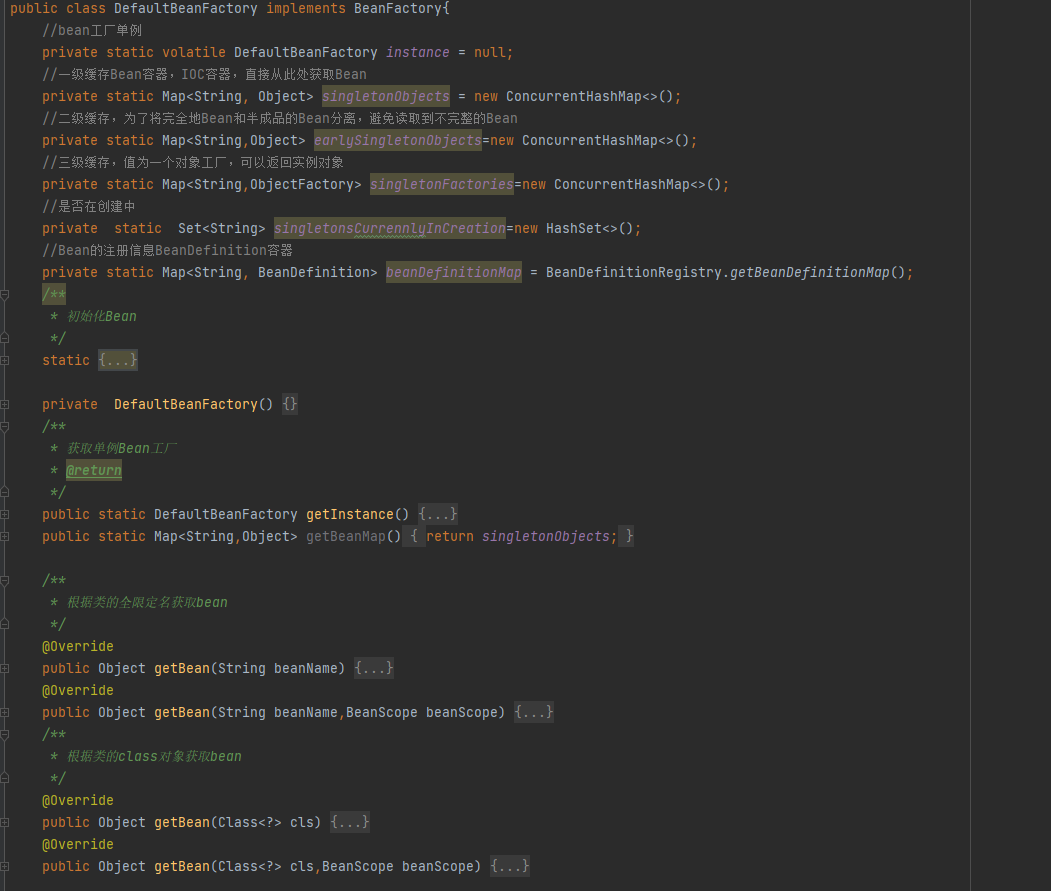
1. 获取基础包下间接带有某注解的所有类

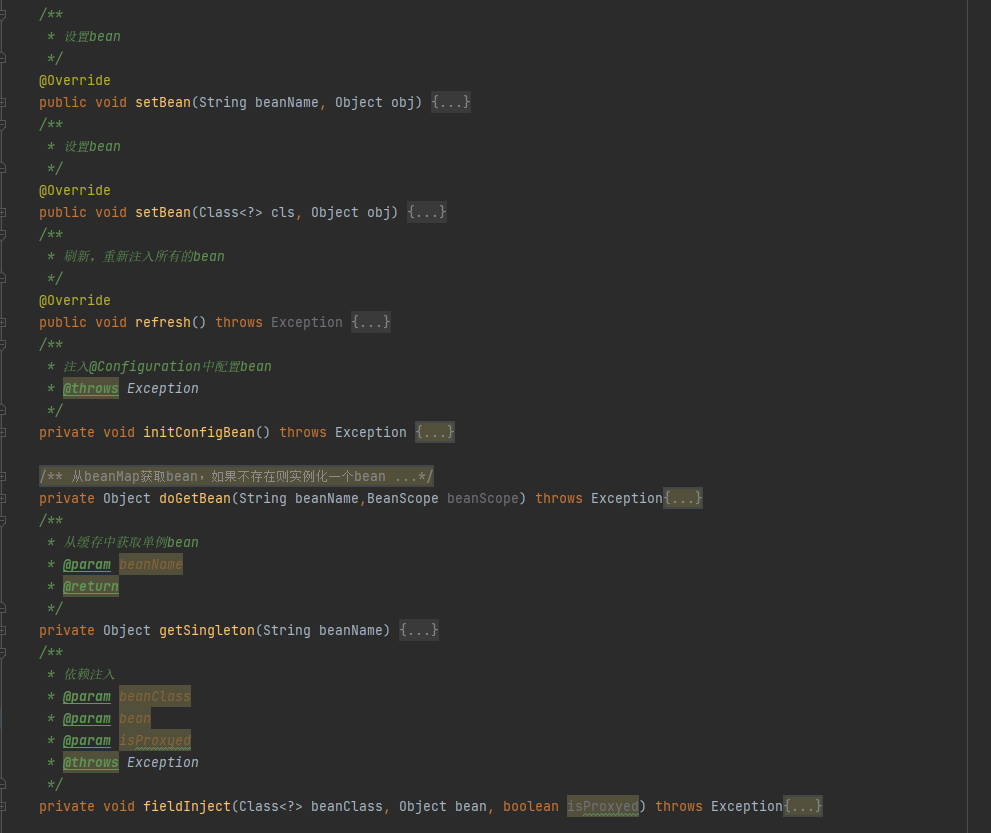


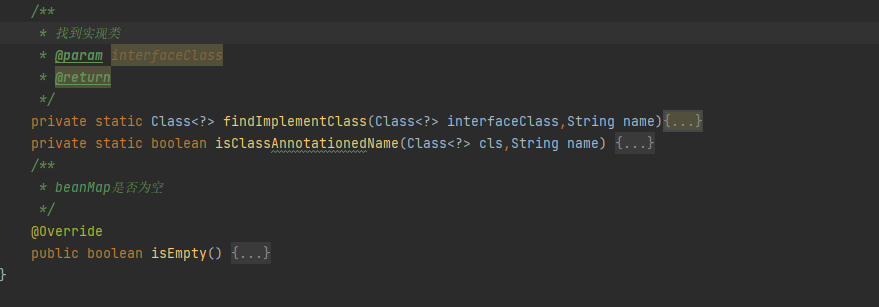
1. 判断classz是否有注解annoClass，包括直接和间接的注解



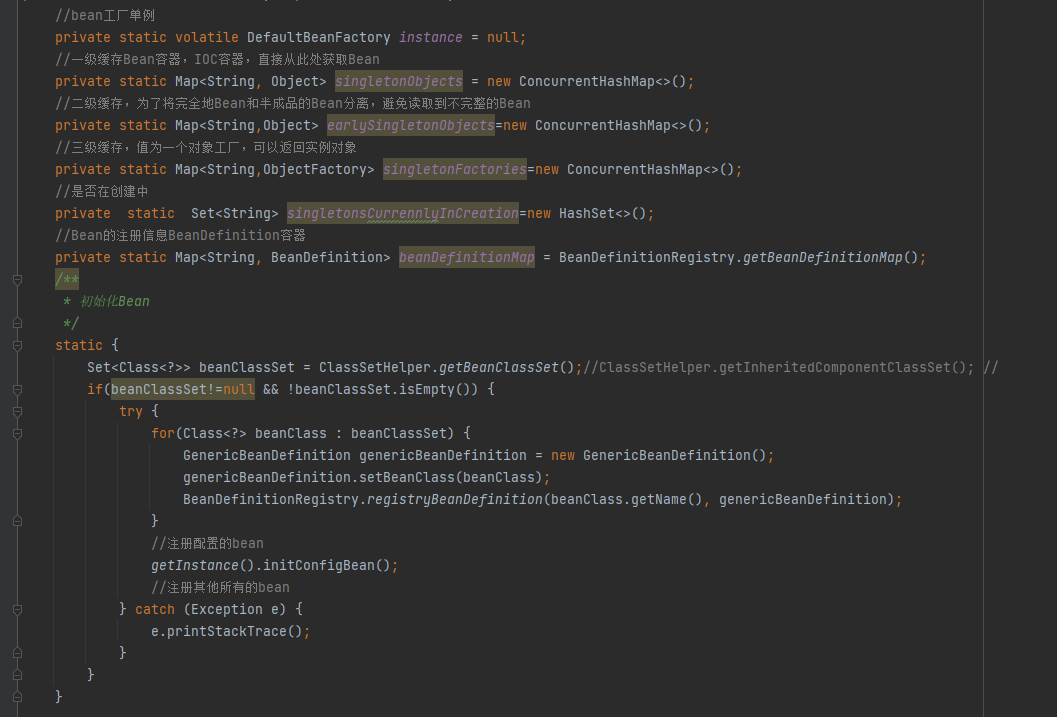
1. DefaultBeanFactory 默认BeanFactory实现类





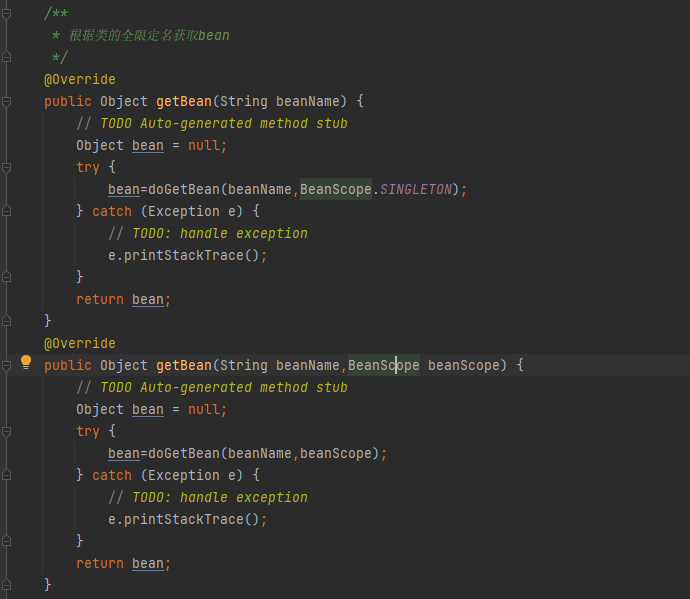


1. 单例构造BeanFactory实例（双重校验懒加载， 三重缓存解决循环依赖

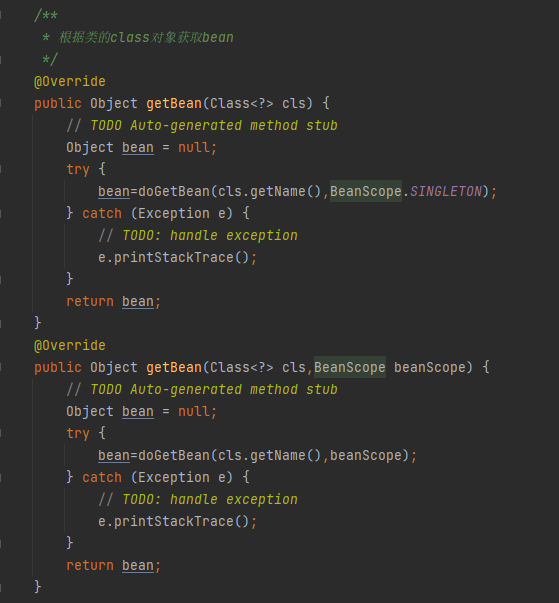




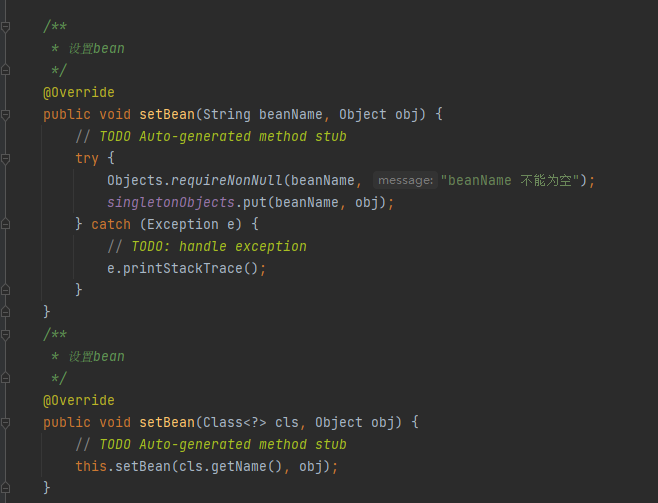
1. 根据Bean名称获取Bean对象



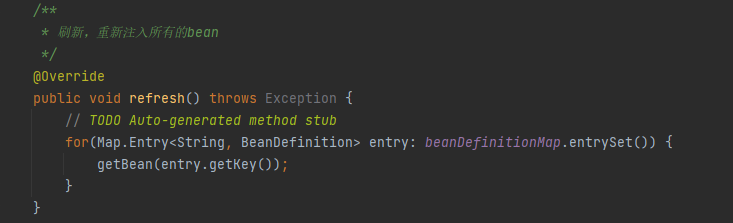
1. 根据Bean类型获取Bean对象



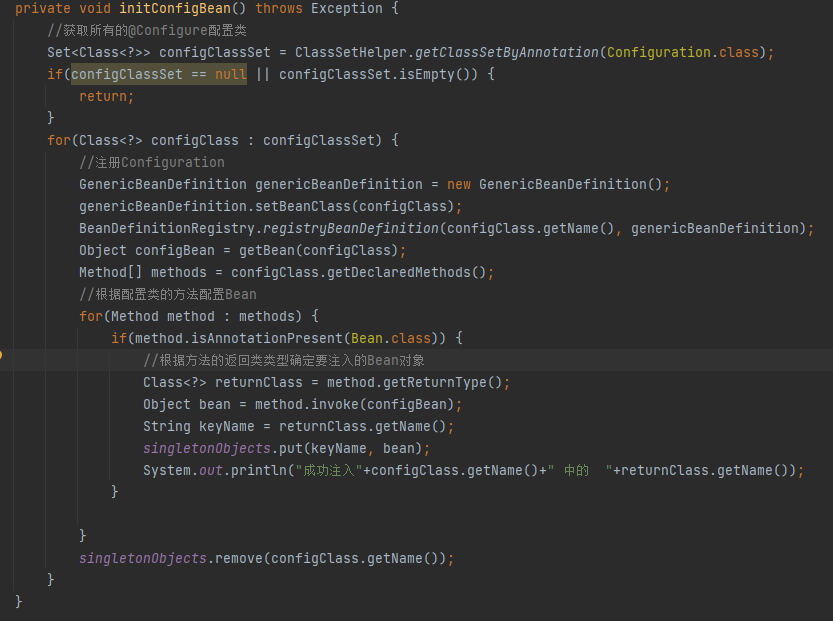
1. 设置Bean



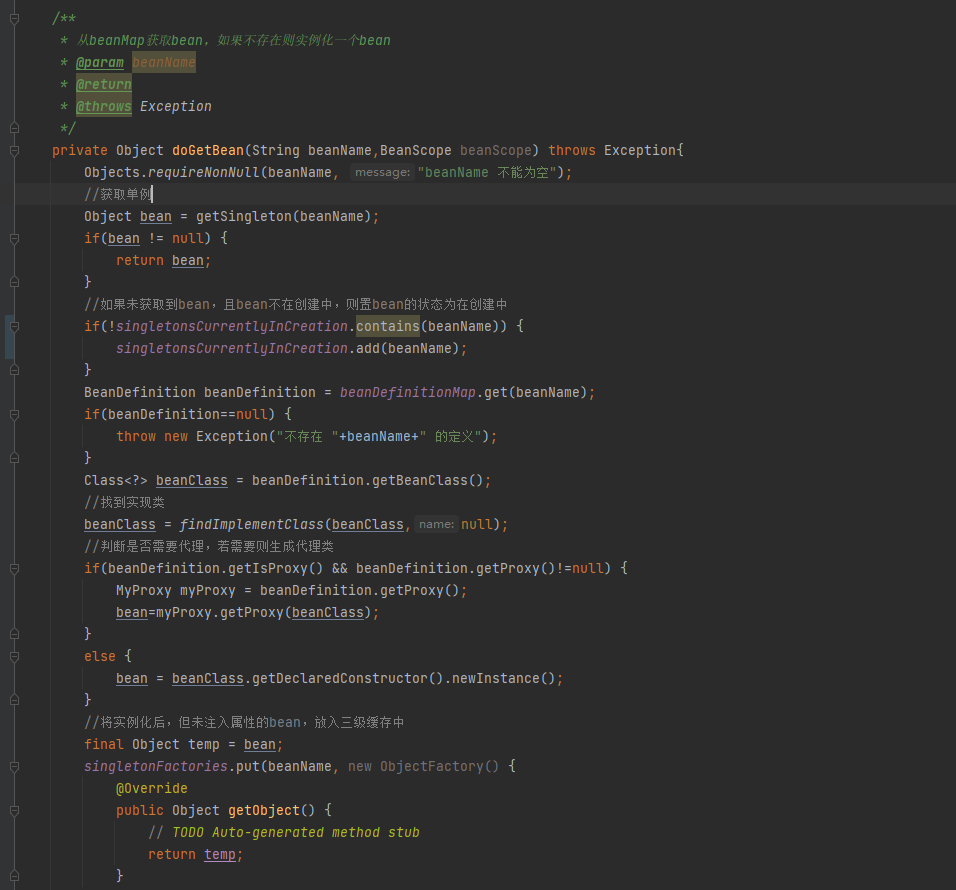
1. 重新注入所有bean



1. 利用configure配置Bean



1. 获取bean，如果不存在就创建Bean

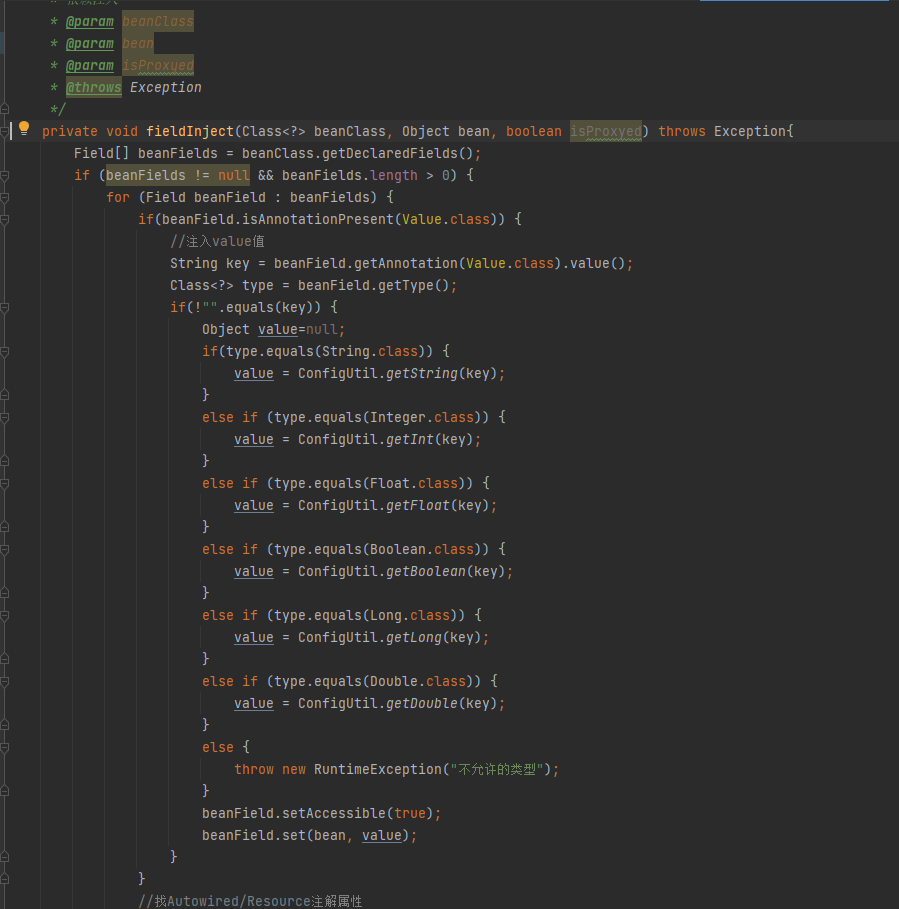


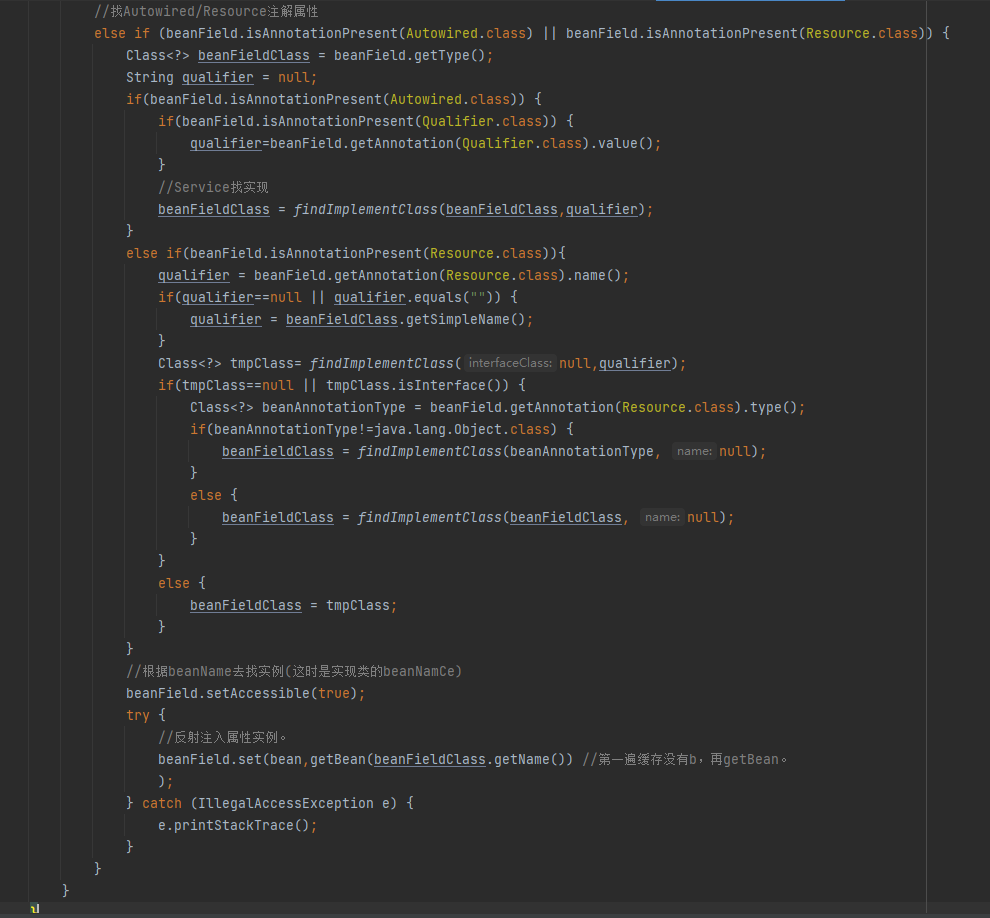


1. 三级缓存避免循环依赖

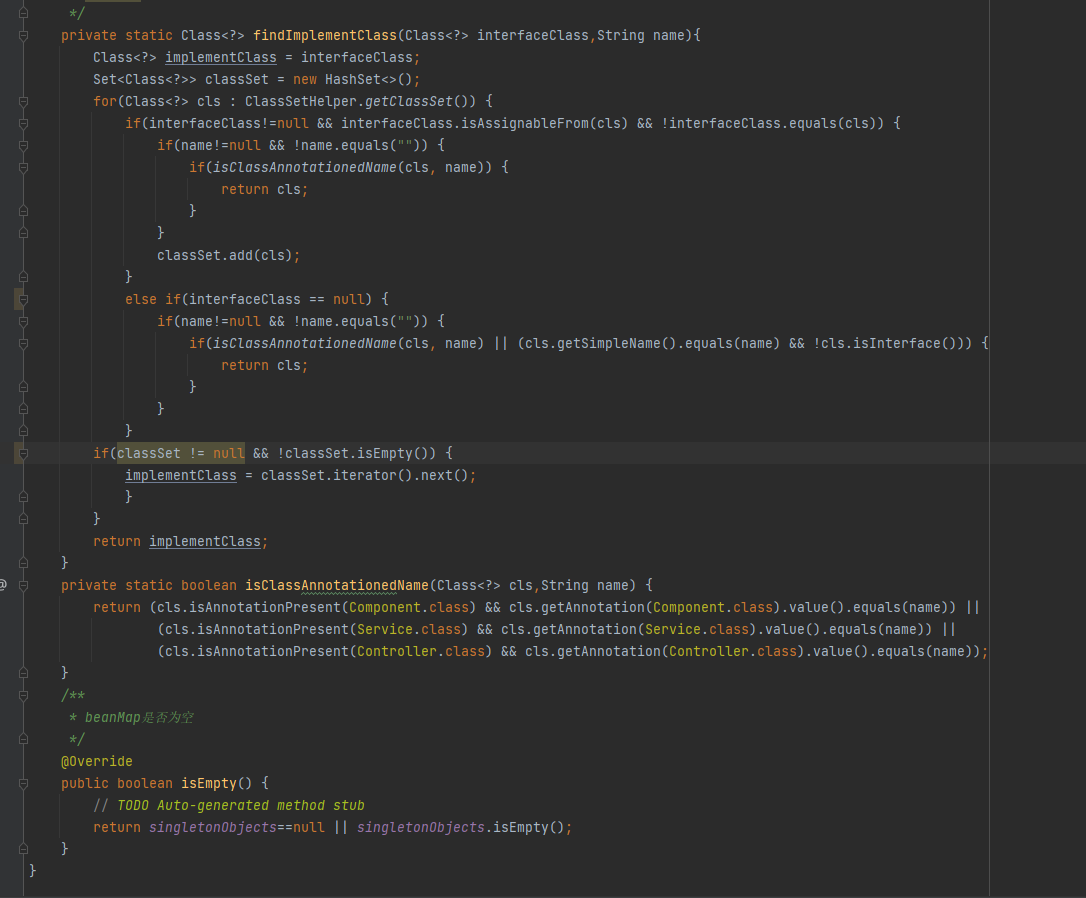


1. 依赖注入（@Value，@Autowired和@Resource）

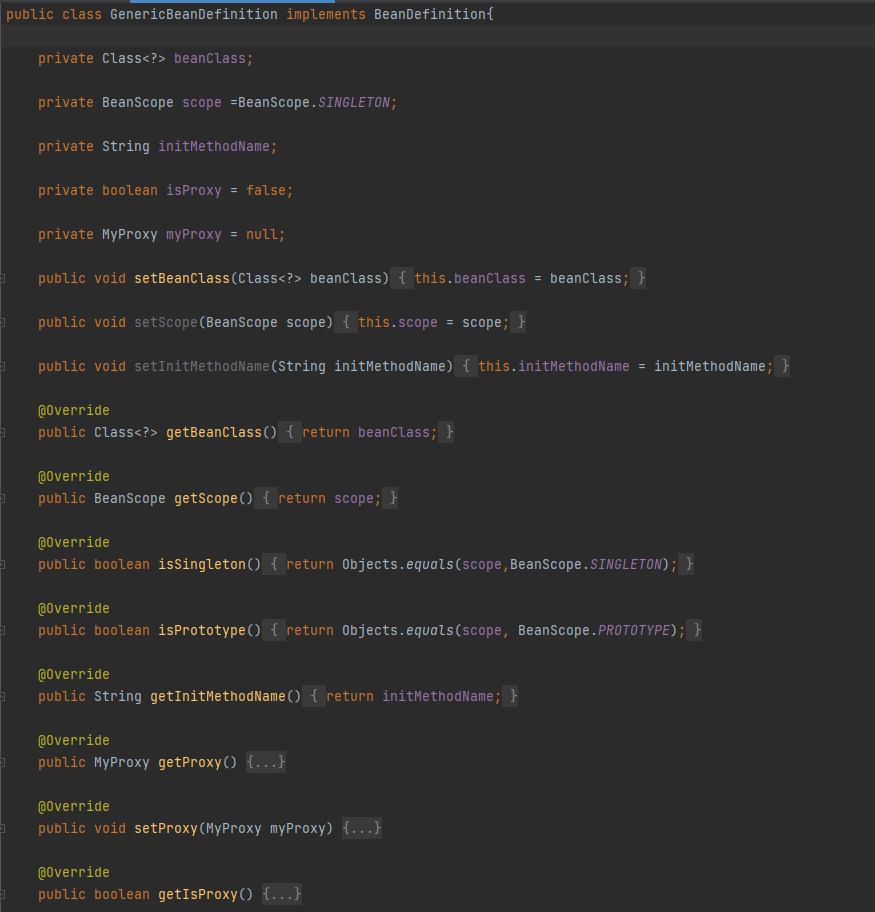




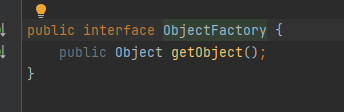
1. 工具类，根据类型或名称找到对应的类



1. GenericBeanDefinition (BeanDefinition的实现类)



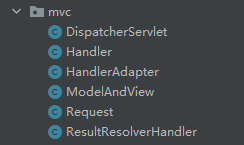
1. ObjectFactory (三级缓存的工厂)



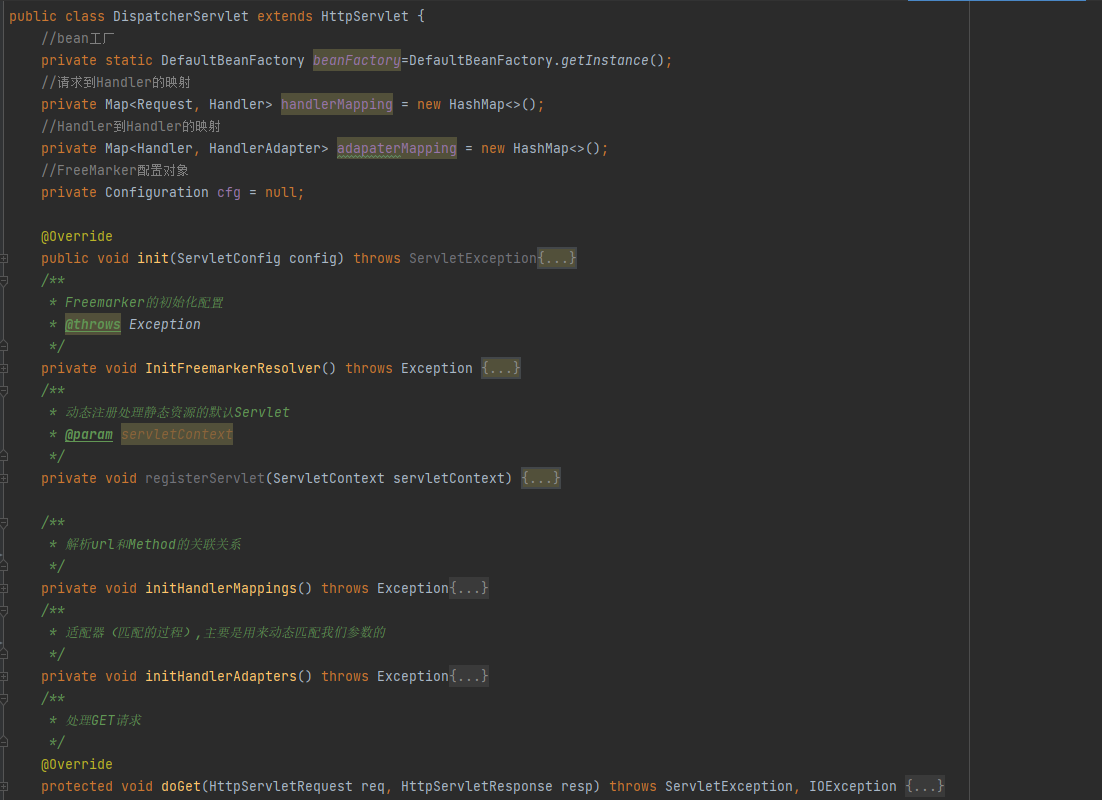
1. Annotion-MVC

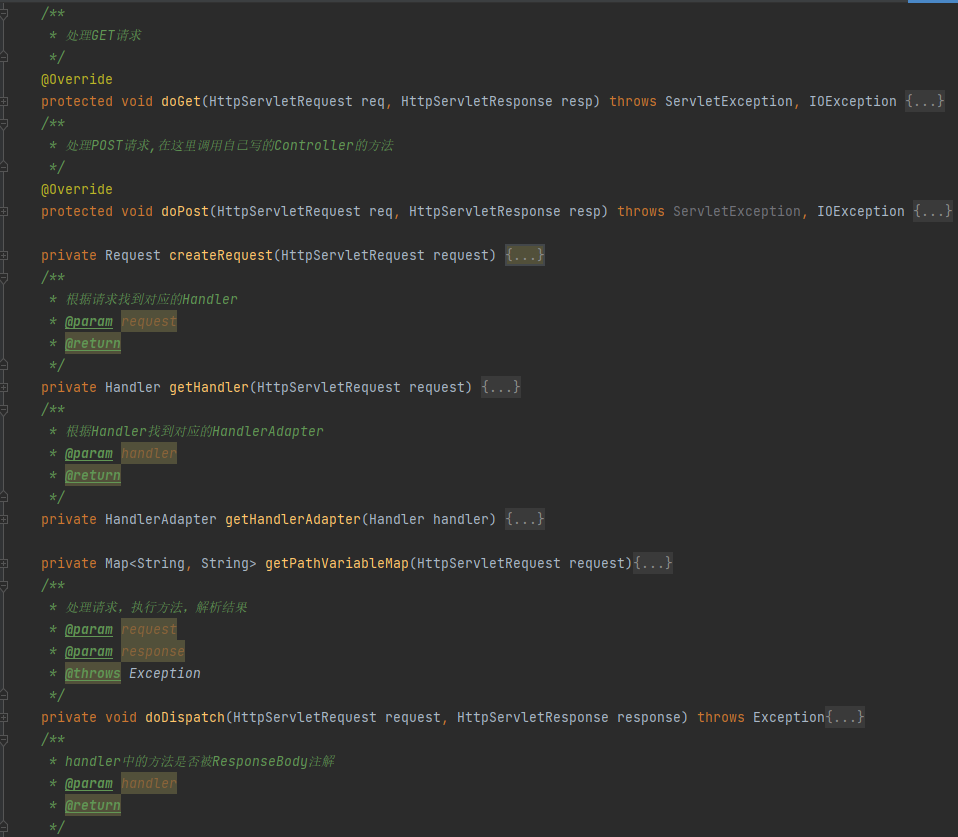


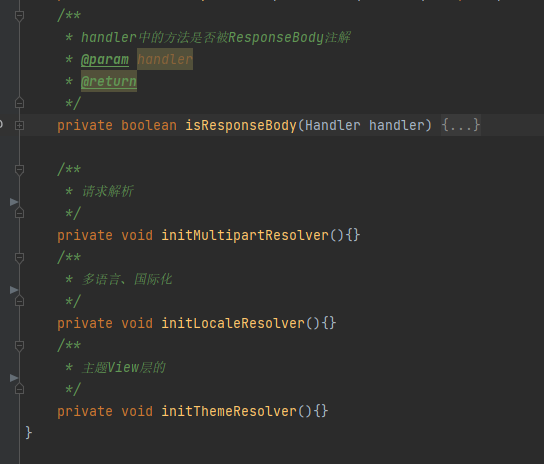
1. MVC实现



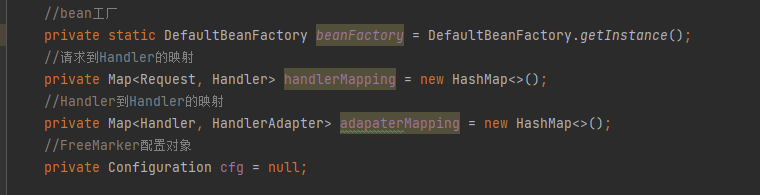
1. DispatcherServlet







1. 全局变量，除工厂类共享，其他均为Servlet对象私有



1. Init 根据config配置serlet对象



1. Freemaker的初始化

