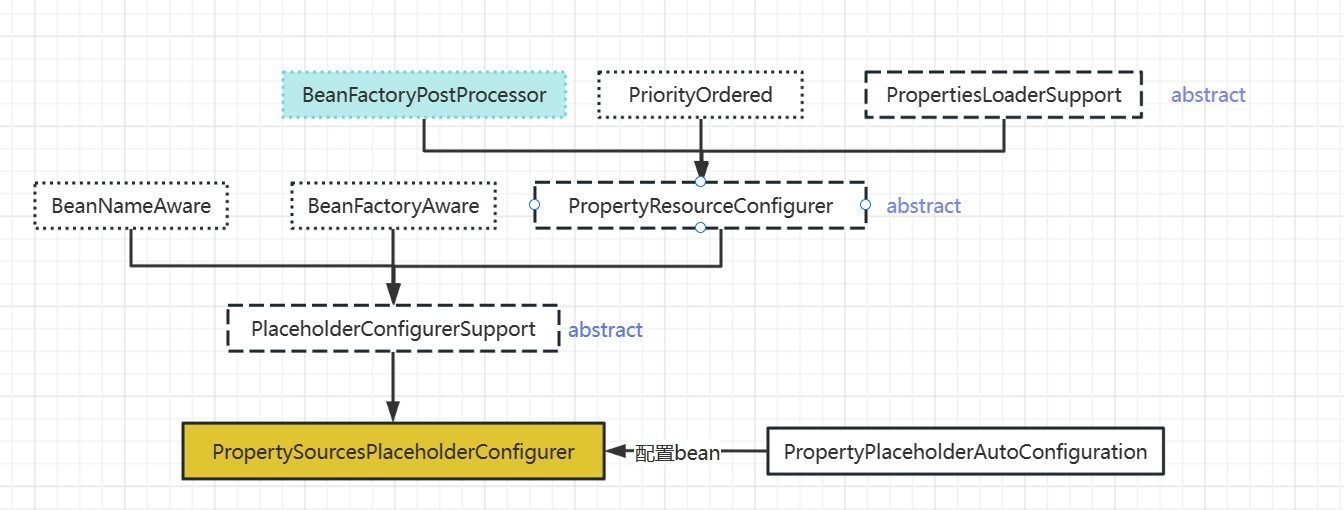
# 1、注解@Value和@Autowired



如上图，

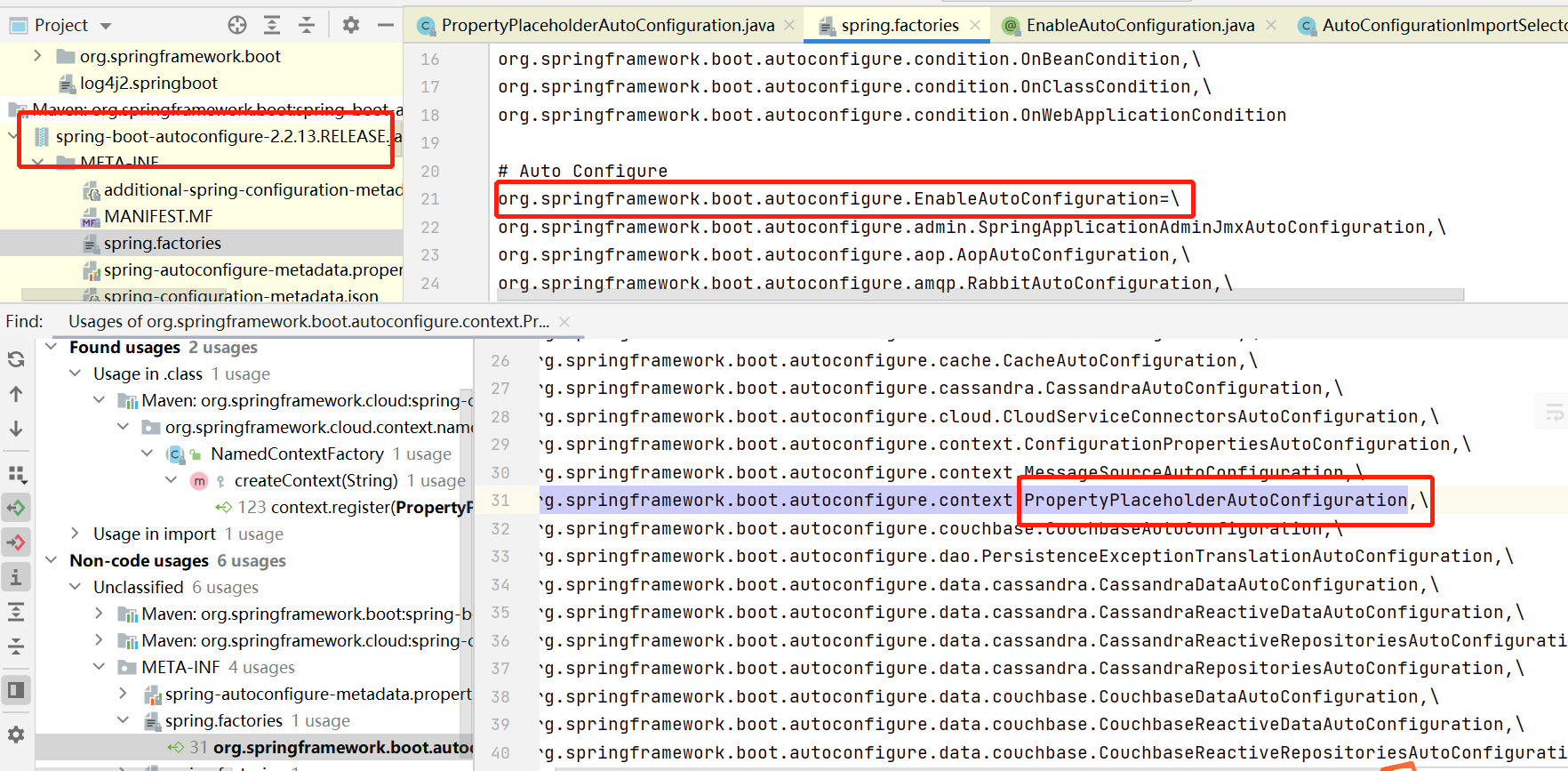
1、PropertyPlaceholderAutoConfiguration配置类里配置了

PropertySourcesPlaceholderConfigurer类这个bean；

2、PropertySourcesPlaceholderConfigurer实现了BeanFactoryPostProcessor这个接口，重点是BeanFactoryPostProcessor#postProcessBeanFactory的实现类，

## 1、何时调用PropertyPlaceholderAutoConfiguration配置类

### 1.1 在 spring-boot-autoconfigure的jar里，有配置了spring.factories文件：

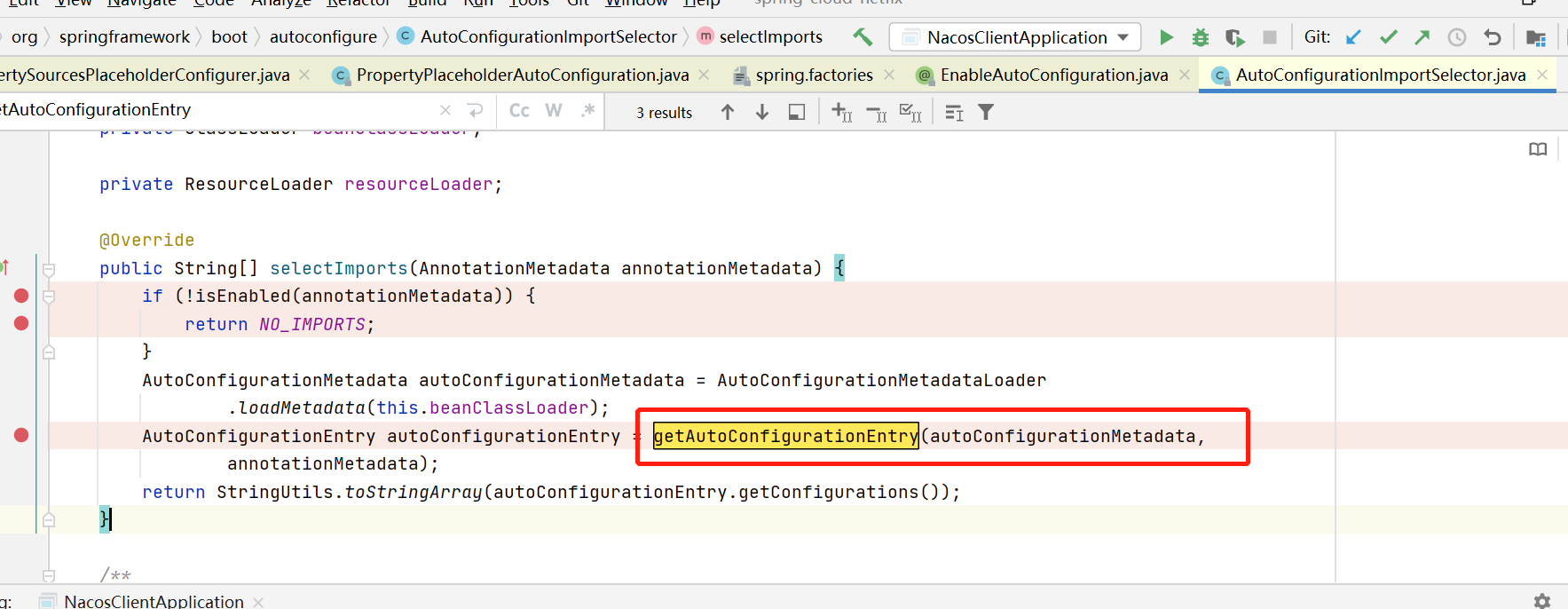


在spring-boot项目中，启动类一定会使用@SpringBootApplication注解，它里包含

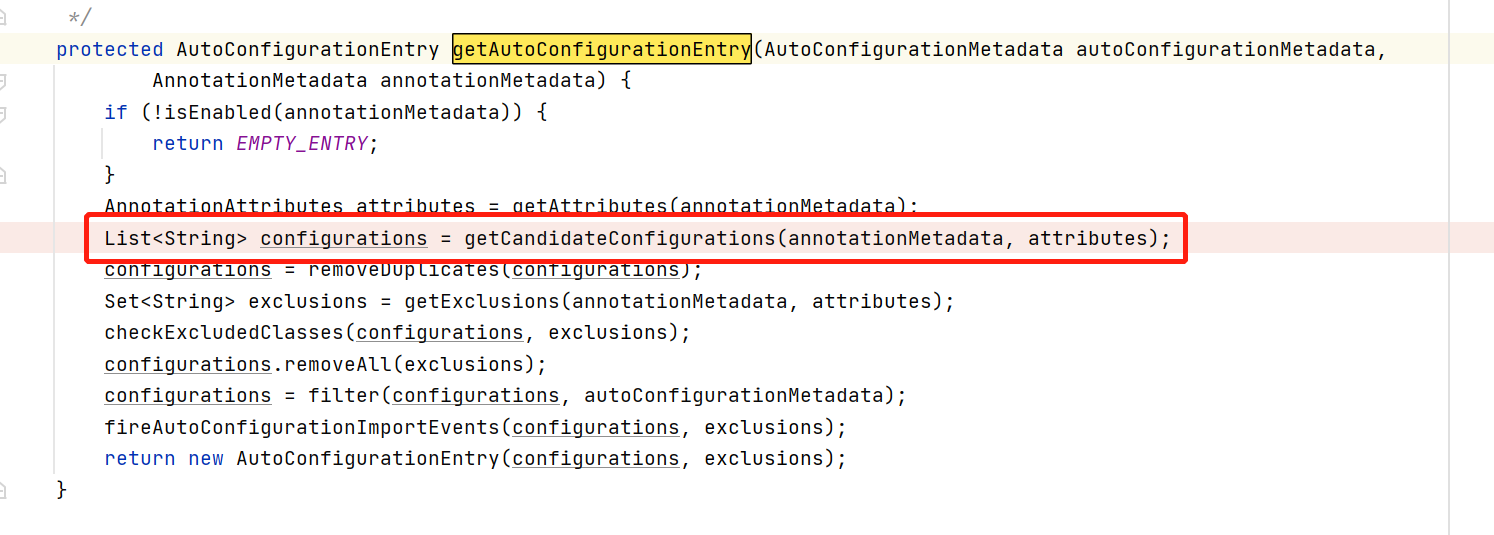
@EnableAutoConfiguration注解，而在调用ConfigurationClassPostProcessor时会解析所有对应@Import注解的类：



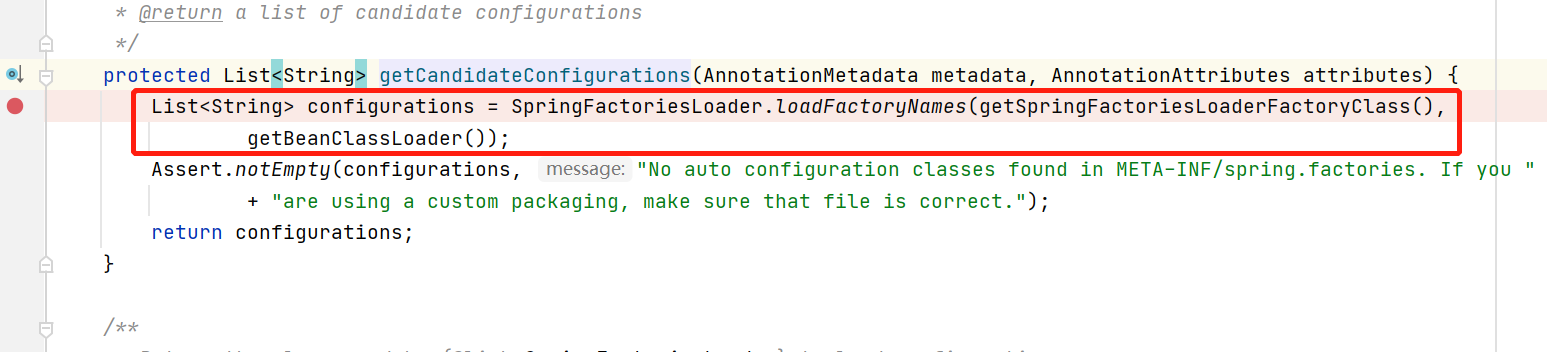
这个类AutoConfigurationImportSelector# selectImports方法就会被调用到：



接着调用AutoConfigurationImportSelector#getAutoConfigurationEntry方法：



接着就是加载key= EnableAutoConfiguration类的所有spring.factories文件中的配置：

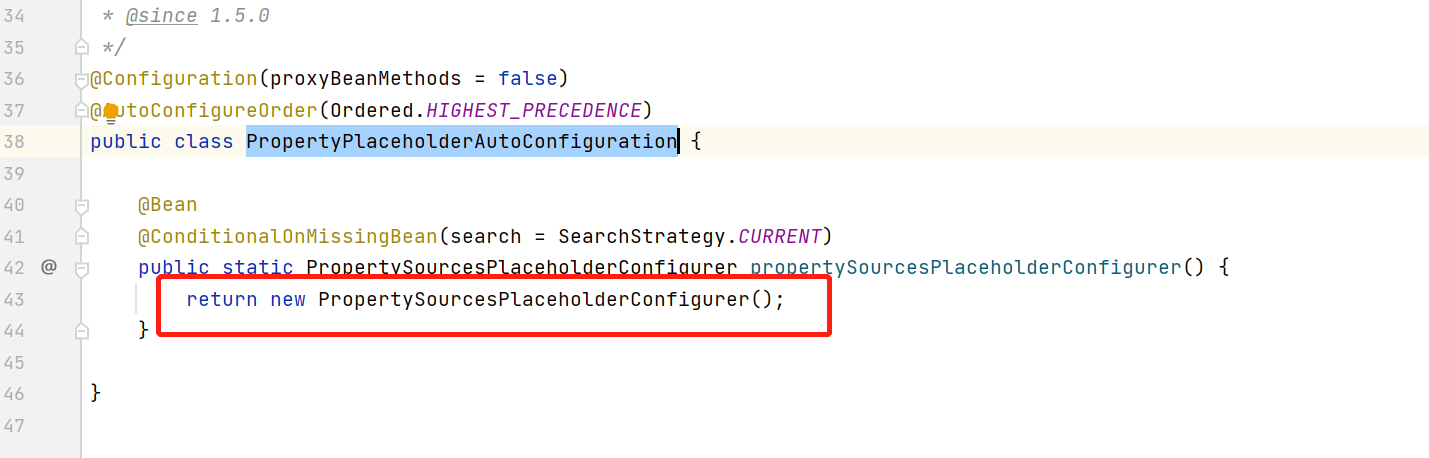


最开始已经讲了在 spring-boot-autoconfigure的jar里，有配置了spring.factories文件：

org.springframework.boot.autoconfigure.EnableAutoConfiguration=\

org.springframework.boot.autoconfigure.context.PropertyPlaceholderAutoConfiguration

所以这里会加载到这个PropertyPlaceholderAutoConfiguration配置类：



从而把PropertySourcesPlaceholderConfigurer加载到spring容器中；

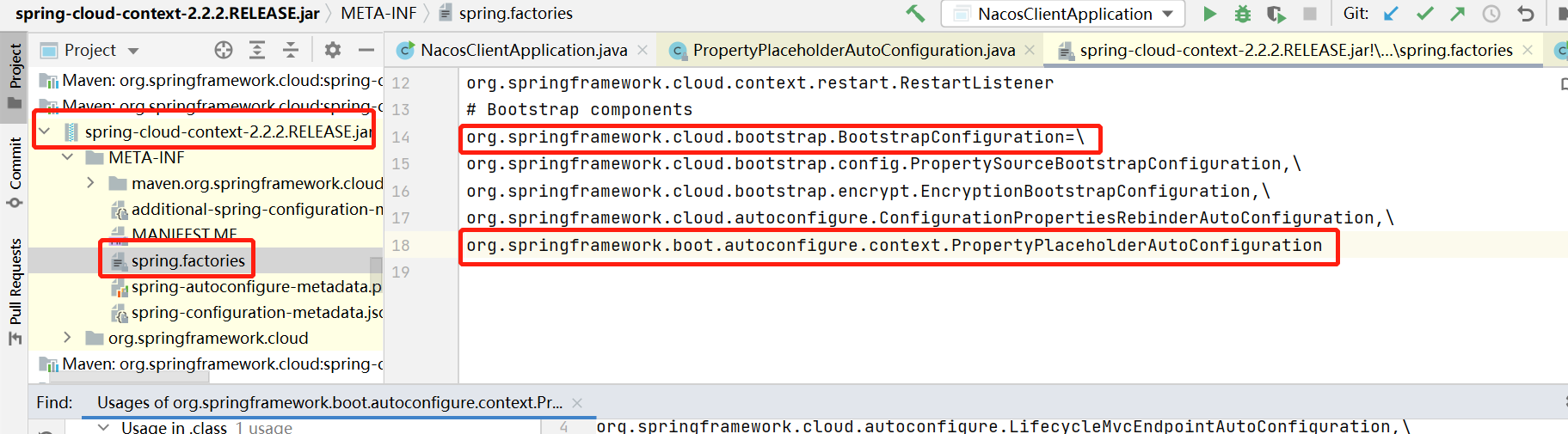
而PropertySourcesPlaceholderConfigurer是BeanFactoryPostProcessor的实现类，

在AbstractApplicationContext#refresh方法里会调用：

AbstractApplicationContext#invokeBeanFactoryPostProcessors方法，从而调用

BeanFactoryPostProcessor#postProcessBeanFactory接口的实现类：

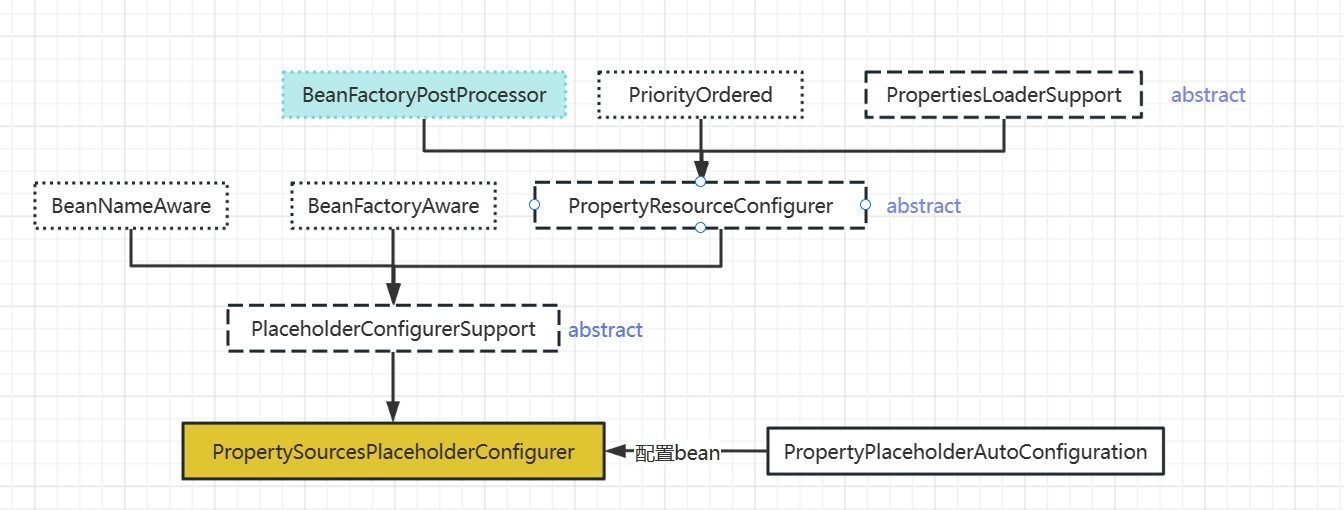
### 1.2 在spring-cloud-context的jar包里有spring.factories文件里



我们在讲解spring-boot的启动过程时，讲过这样的配置会在父容器的时候就先执行了，

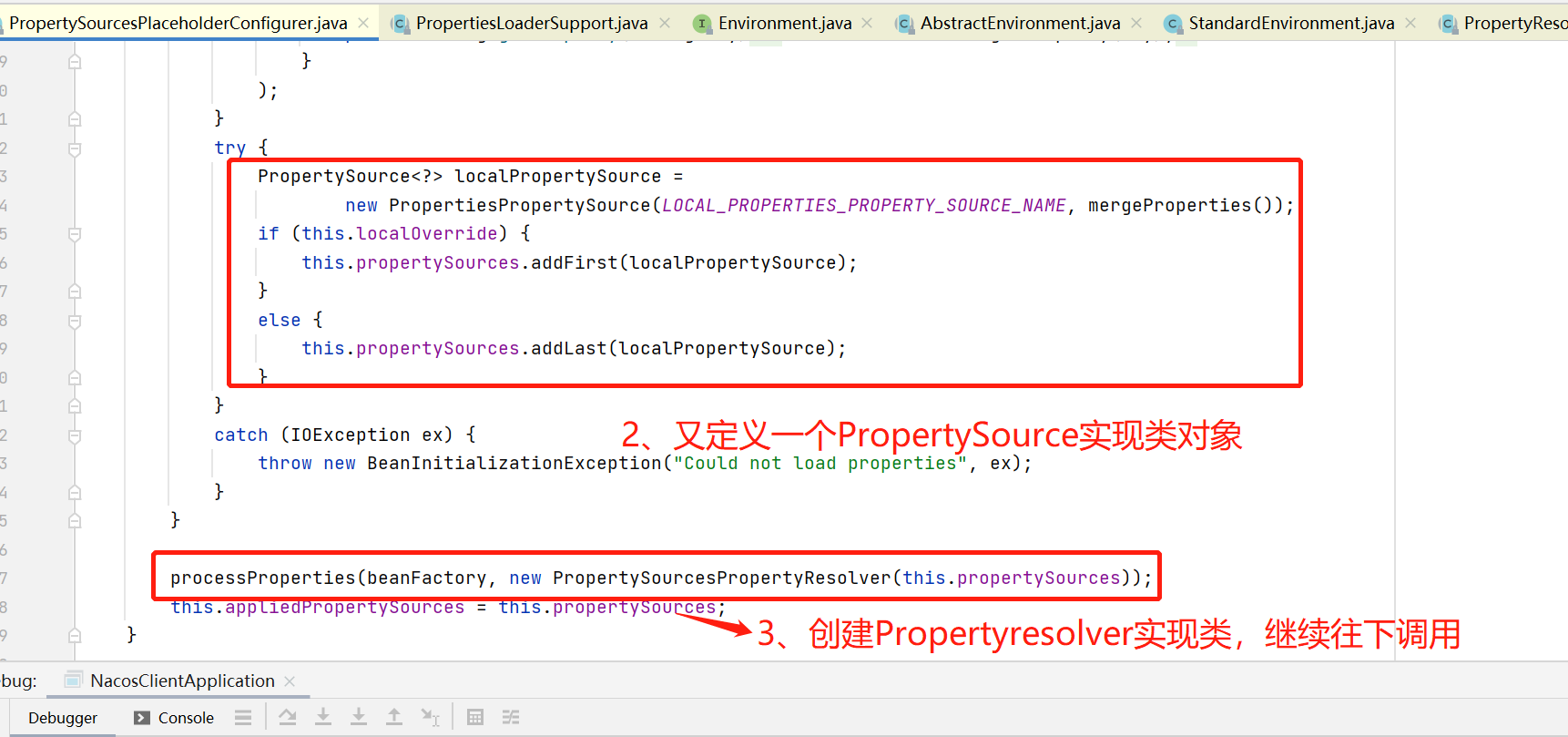
所以也会把这个配置类提前加载到spring容器中去。

## 2、PropertySourcesPlaceholderConfigurer里是如何实现#postProcessBeanFactory方法的：



下面看下PropertySourcesPlaceholderConfigurer#postProcessBeanFactory方法：





继续看PropertySourcesPlaceholderConfigurer#processProperties方法：



继续看PlaceholderConfigurerSupport#doProcessProperties方法：

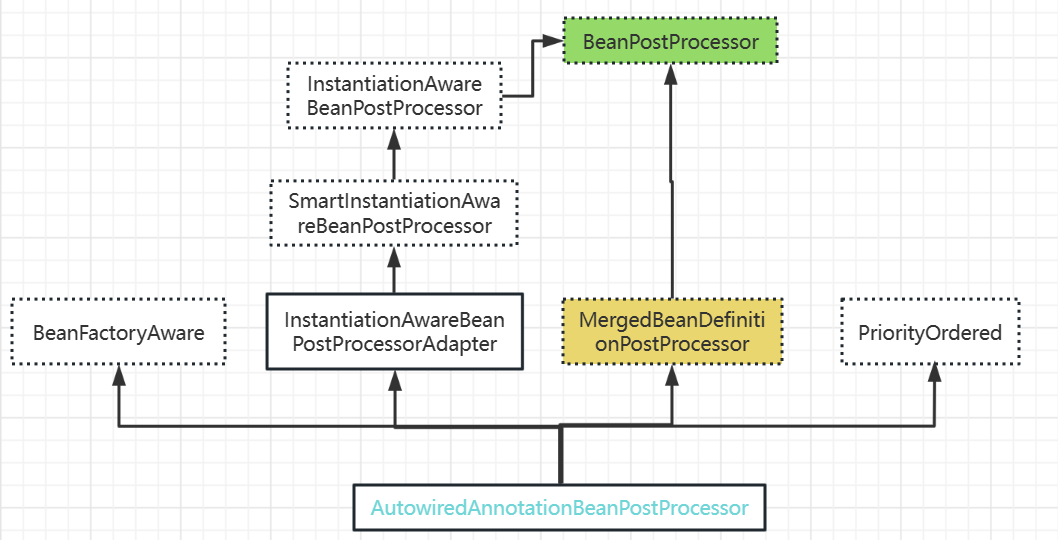


这里核心是把StringValueResolver对象（实际是走的PropertySourcesPropertyResolver对象的具体实现类来具体实现的）传给了ConfigurableListableBeanFactory对象，后续在给对象实例化赋值的时候就会使用了。

这样，ConfigurableListableBeanFactory的具体实现类DefaultListableBeanFactory里的

List<StringValueResolver> embeddedValueResolvers（在AbstractBeanFactory类定义）就有了具体的值了。

## 3、 AutowiredAnnotationBeanPostProcessor



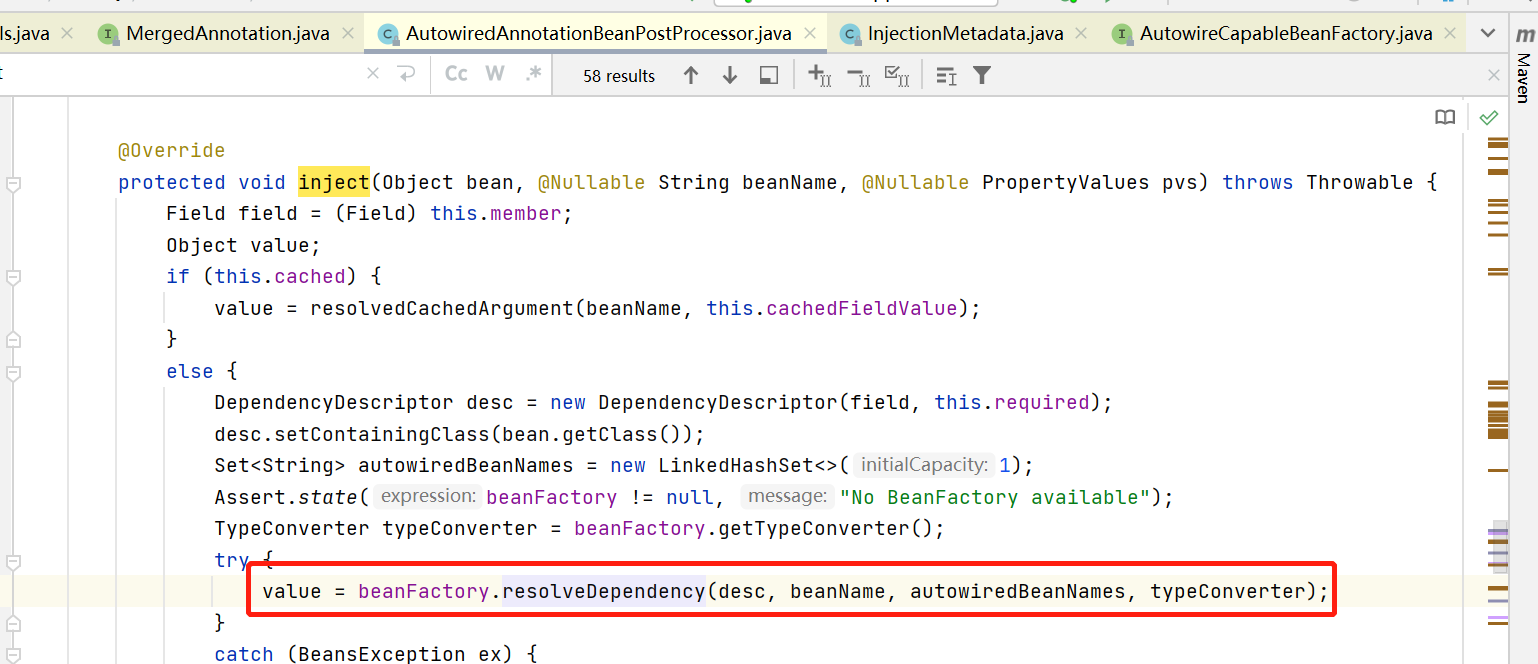
我们前面已经说过，AutowiredAnnotationBeanPostProcessor是解决注解@Value及@Autowired的核心类，它又是如何做的呢？

这里采用倒推法：

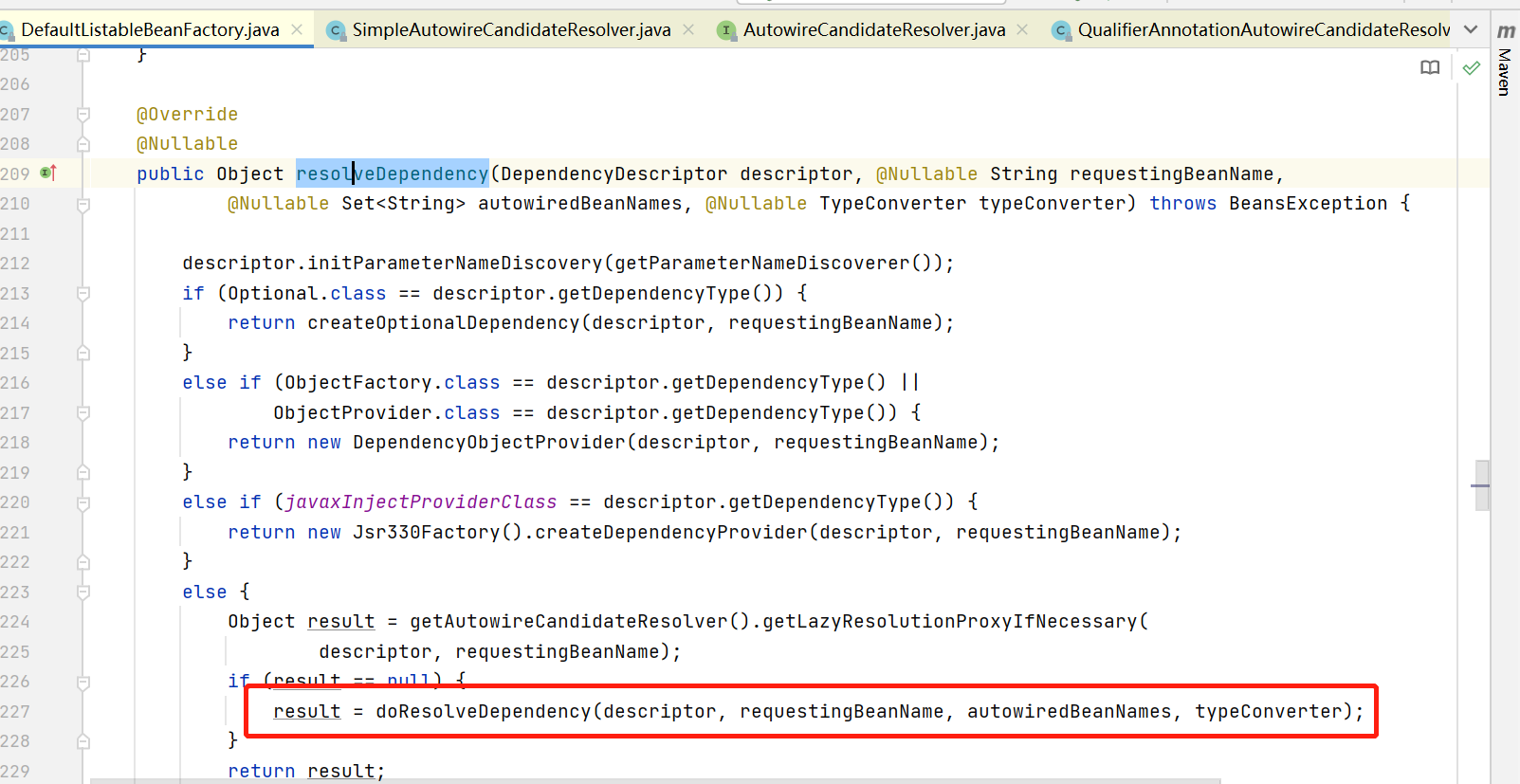
AutowiredAnnotationBeanPostProcessor.AutowiredFieldElement#inject 方法：

AutowiredAnnotationBeanPostProcessor.AutowiredMethodElement#inject 方法：

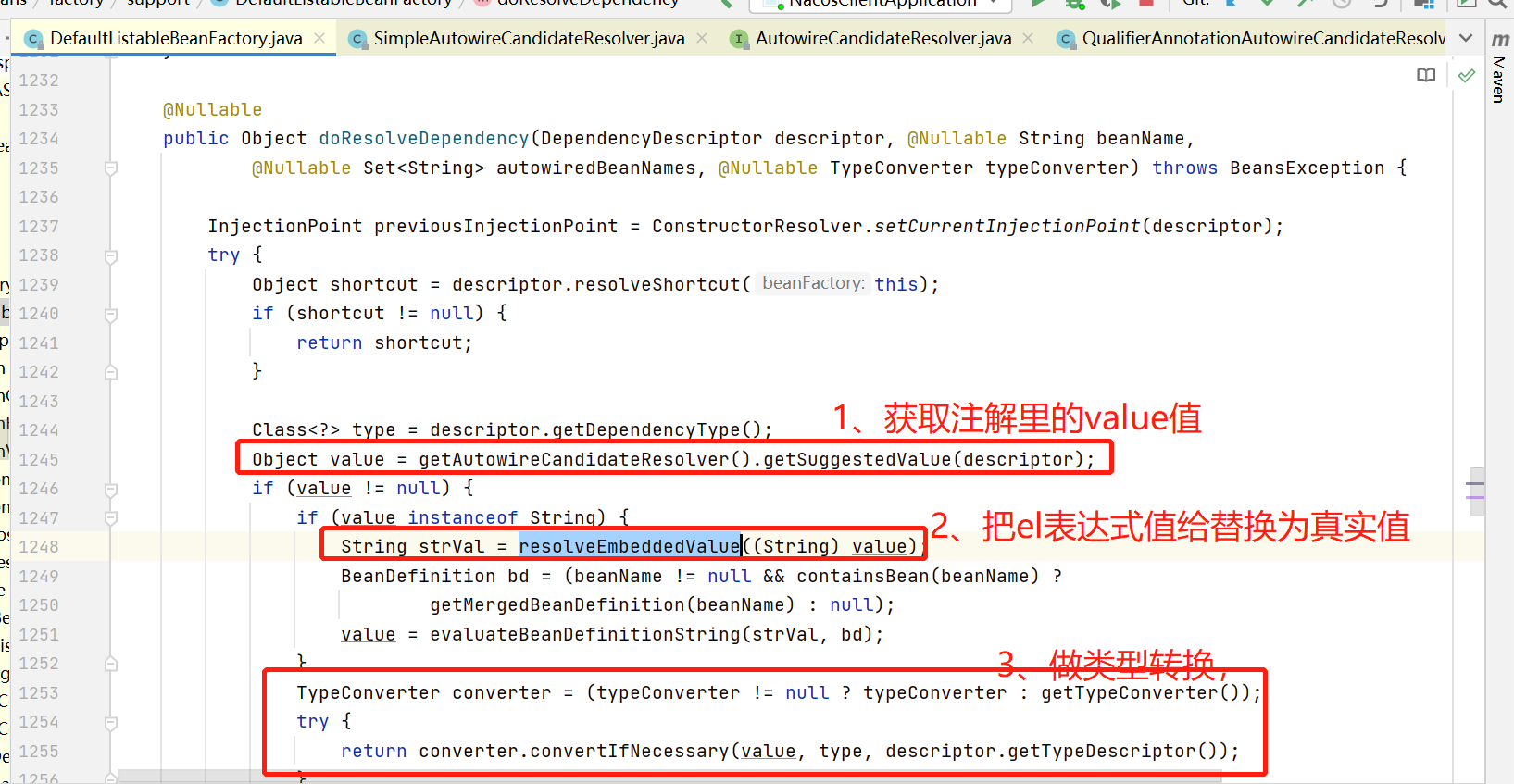
他俩都调用DefaultListableBeanFactory#resolveDependency方法:



而DefaultListableBeanFactory#resolveDependency方法:



继续看DefaultListableBeanFactory#doResolveDependency：



这里我们重点看下第2步：调用的是AbstractBeanFactory#resolveEmbeddedValue方法：



后续我们只要推导出

AutowiredAnnotationBeanPostProcessor.AutowiredFieldElement#inject 方法：

和 AutowiredAnnotationBeanPostProcessor.AutowiredMethodElement#inject 方法：何时被调用即可。

## 4、AbstractAutowireCapableBeanFactory

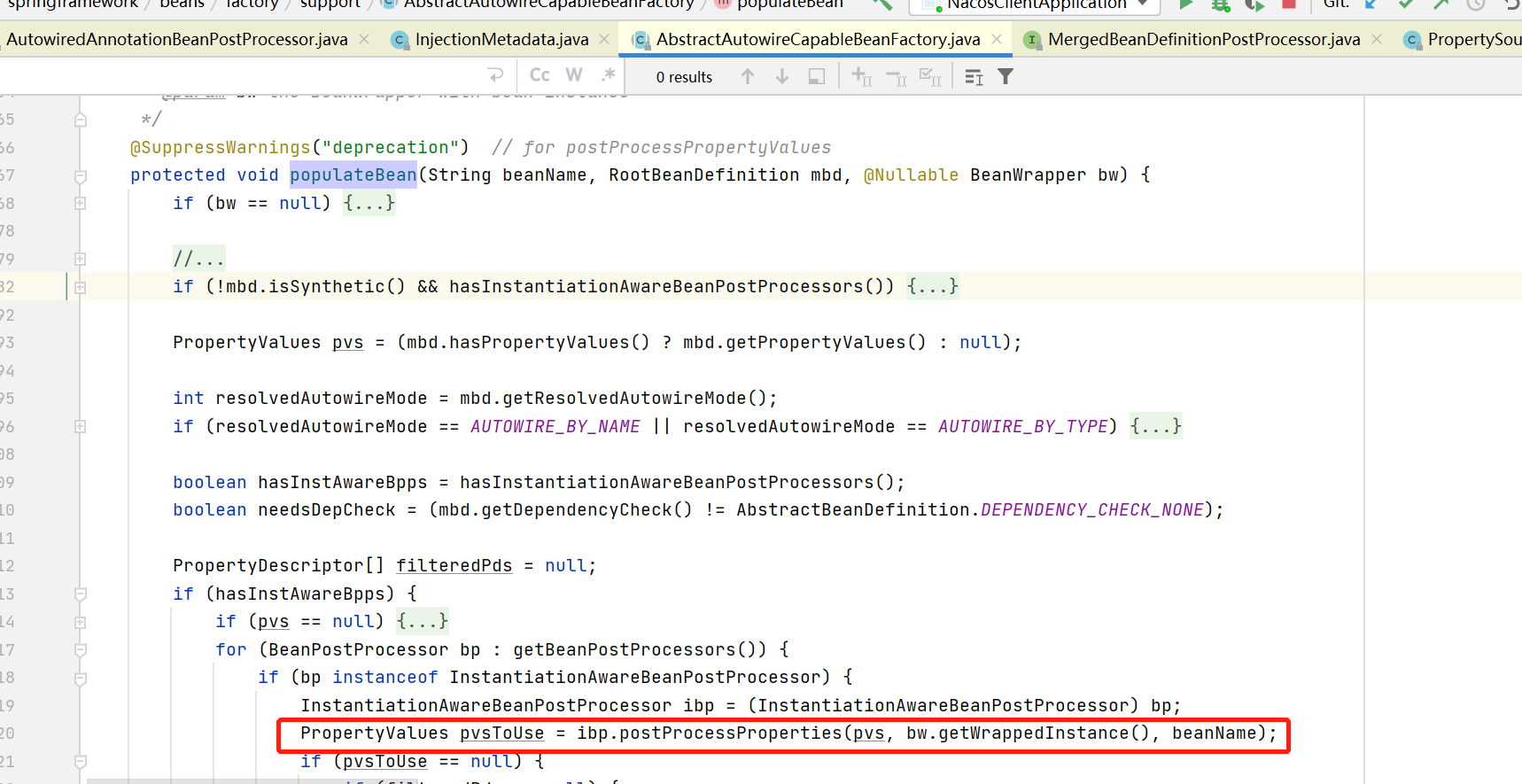
我们在使用spring容器创建对象的时候，最后会走到

AbstractAutowireCapableBeanFactory#doCreateBean方法，完成具体对象的创建、填充值、以及初始化工作：

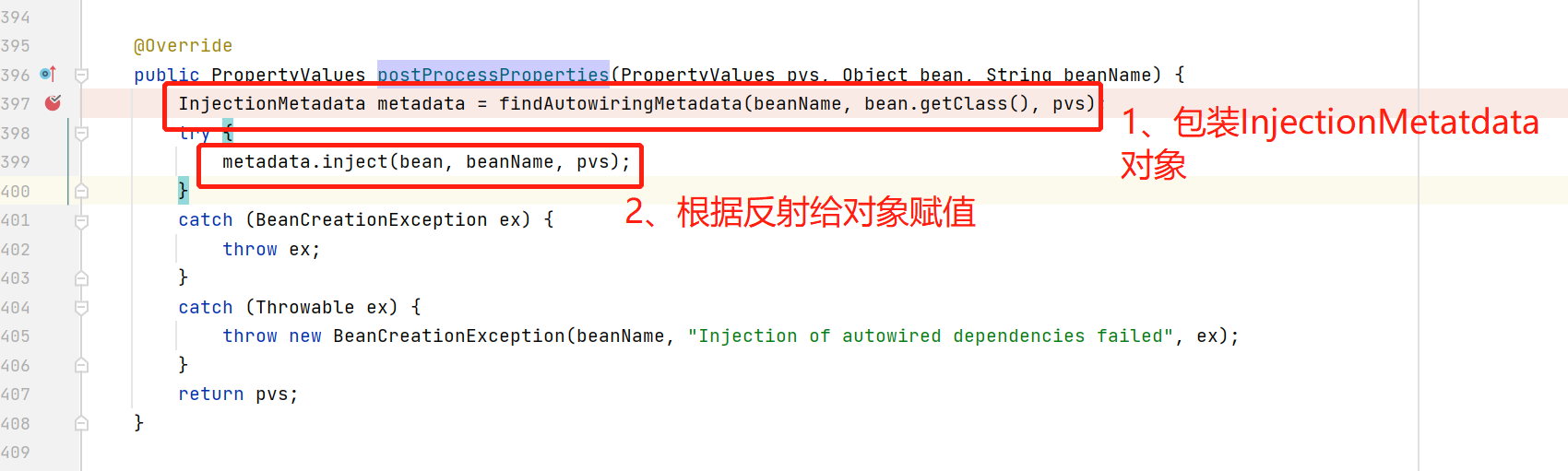




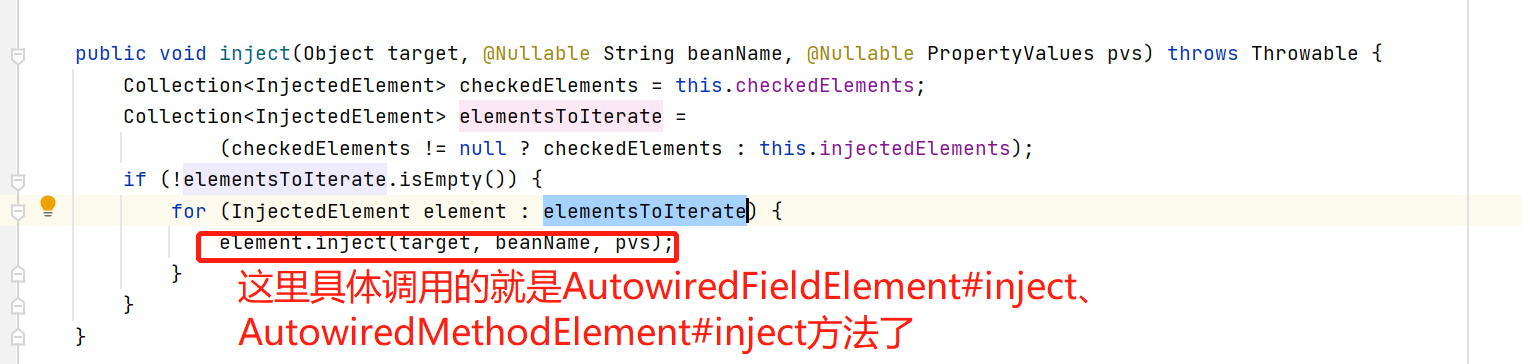
我们这里重点看下第2个AbstractAutowireCapableBeanFactory#populateBean方法，如何给创建完的实例去填充值的：



具体调用的是AutowiredAnnotationBeanPostProcessor#postProcessProperties方法：



看下2：



## 总结

至此，整体介绍完毕，在

1. AnnotationConfigApplicationContext构造方法里创建AnnotatedBeanDefinitionReader对象，
2. AnnotatedBeanDefinitionReader构造方法里会把“AutowiredAnnotationBeanPostProcessor” 注入到spring容器；
3. AutowiredAnnotationBeanPostProcessor是BeanPostProcessor的具体实现类，在构建每一个bean的时候都会被调用到；

以上就确定了创建每个bean，都会走到AutowiredAnnotationBeanPostProcessor里这么一个基础。

4、PropertyPlaceholderAutoConfiguration配置类配置了PropertySourcesPlaceholderConfigurer这个bean。而PropertySourcesPlaceholderConfigurer是BeanFactoryPostProcessor的实现类，在AbstractApplicationContext#refresh方法里会调用：

AbstractApplicationContext#invokeBeanFactoryPostProcessors方法，从而调用PropertySourcesPlaceholderConfigurer#postProcessBeanFactory接口的实现类；

5、PropertySourcesPlaceholderConfigurer#postProcessBeanFactory方法确定el表达式的解析器：StringValueResolver对象；

6、在AbstractAutowireCapableBeanFactory#doCreateBean 创建每个具体的bean时，第一步是创建实体，第二步是填充值，第三步是初始化，

在进行第二步的填充值的时候，就会对使用@Value、@Autowired注解属性进行解析，通过反射给属性填充值。