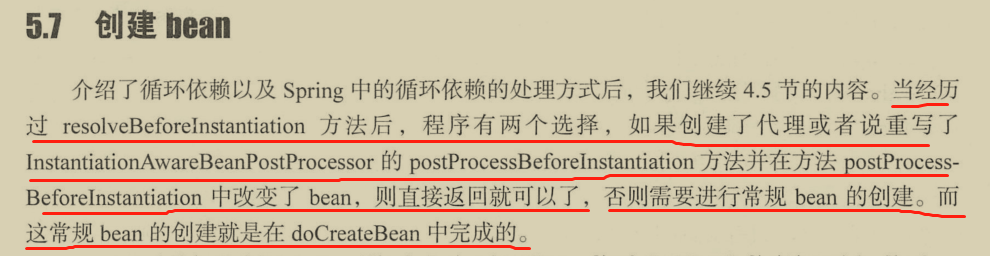
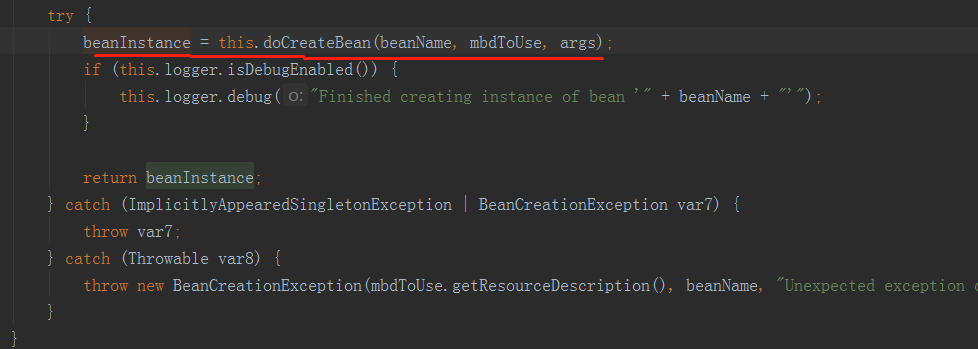
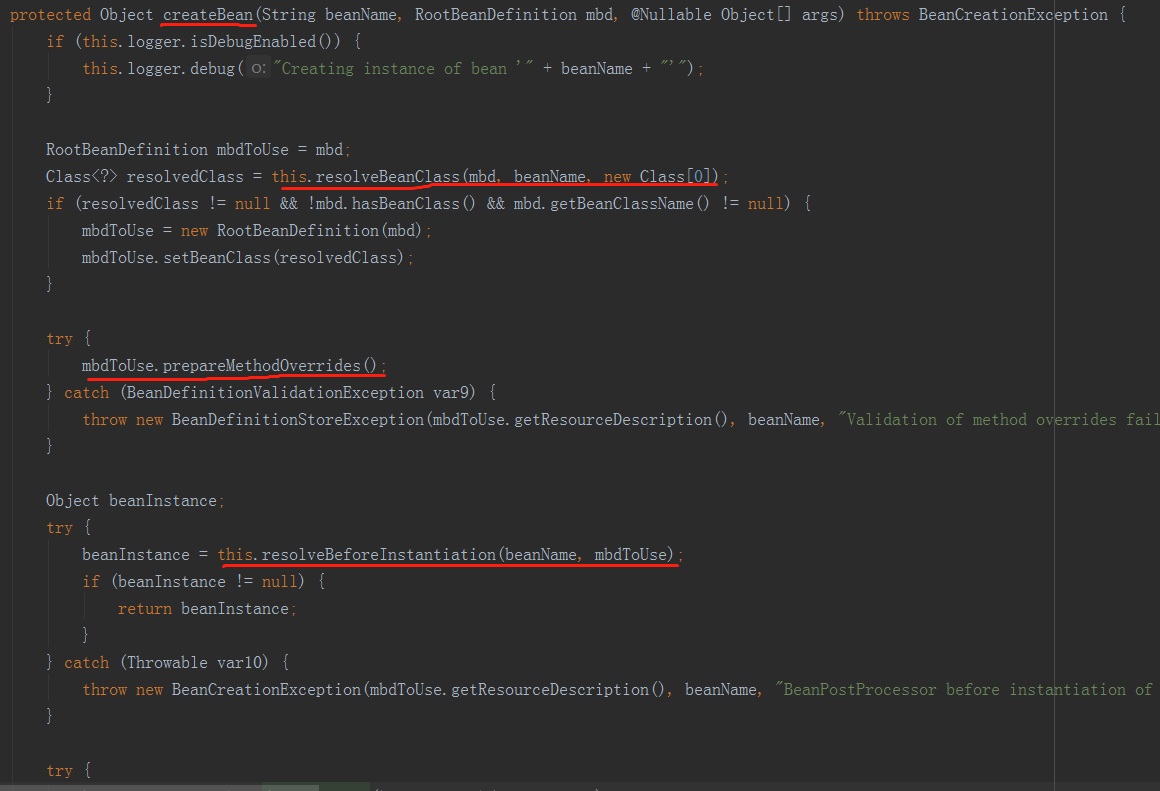
#### 五、Spring Bean加载

##### getBean(String name)

##### 7、创建bean

###### createBean()





###### doCreateBean()：IOC的整个过程

**createBeanInstance():**

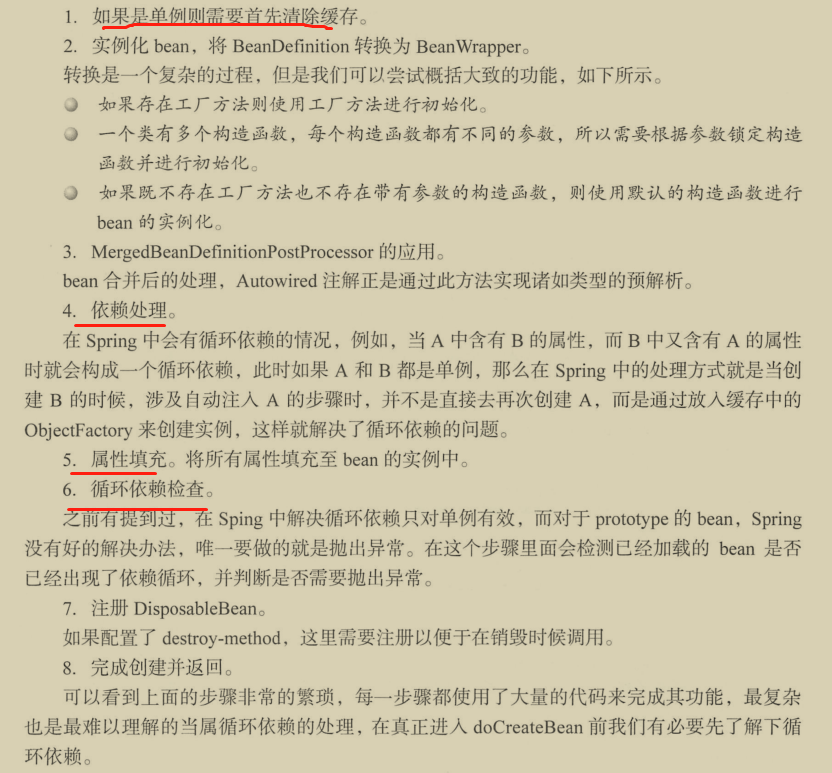
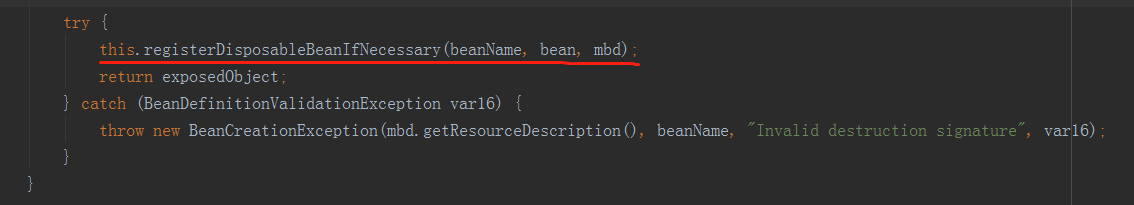
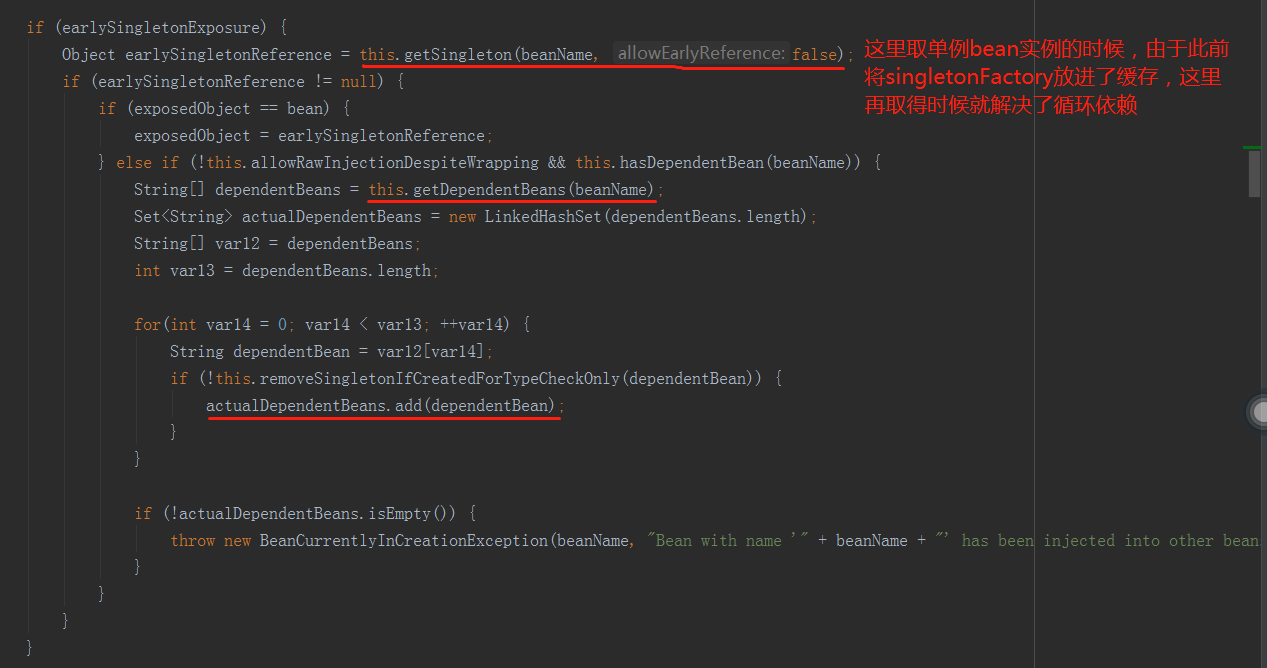
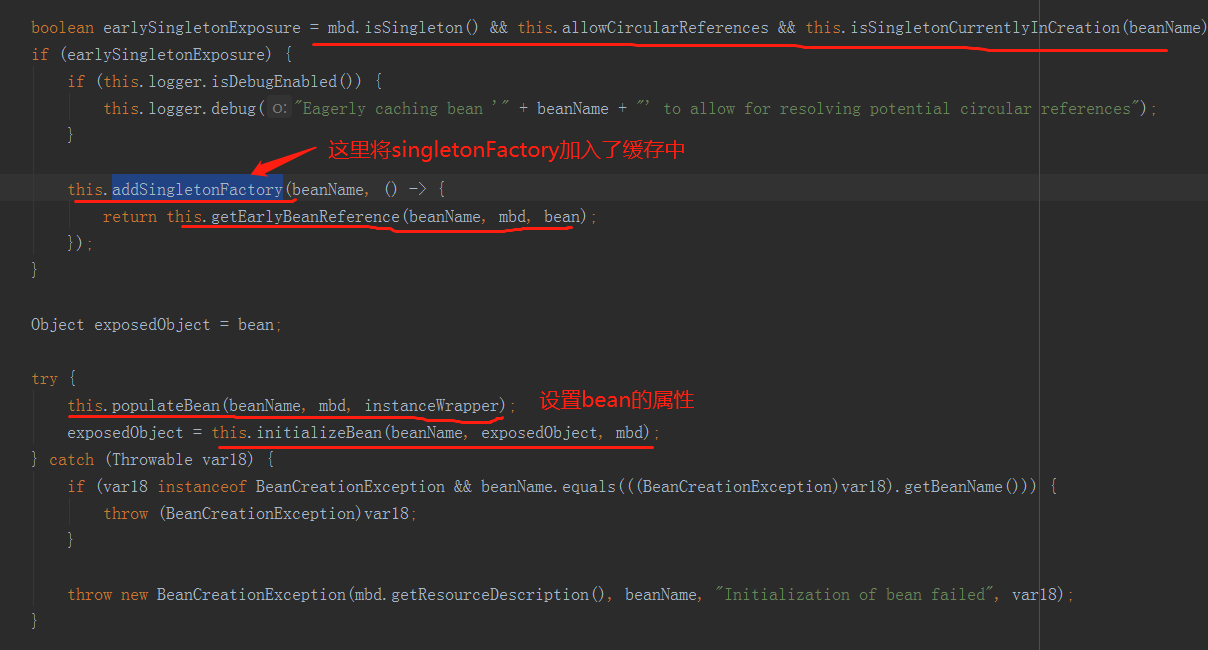
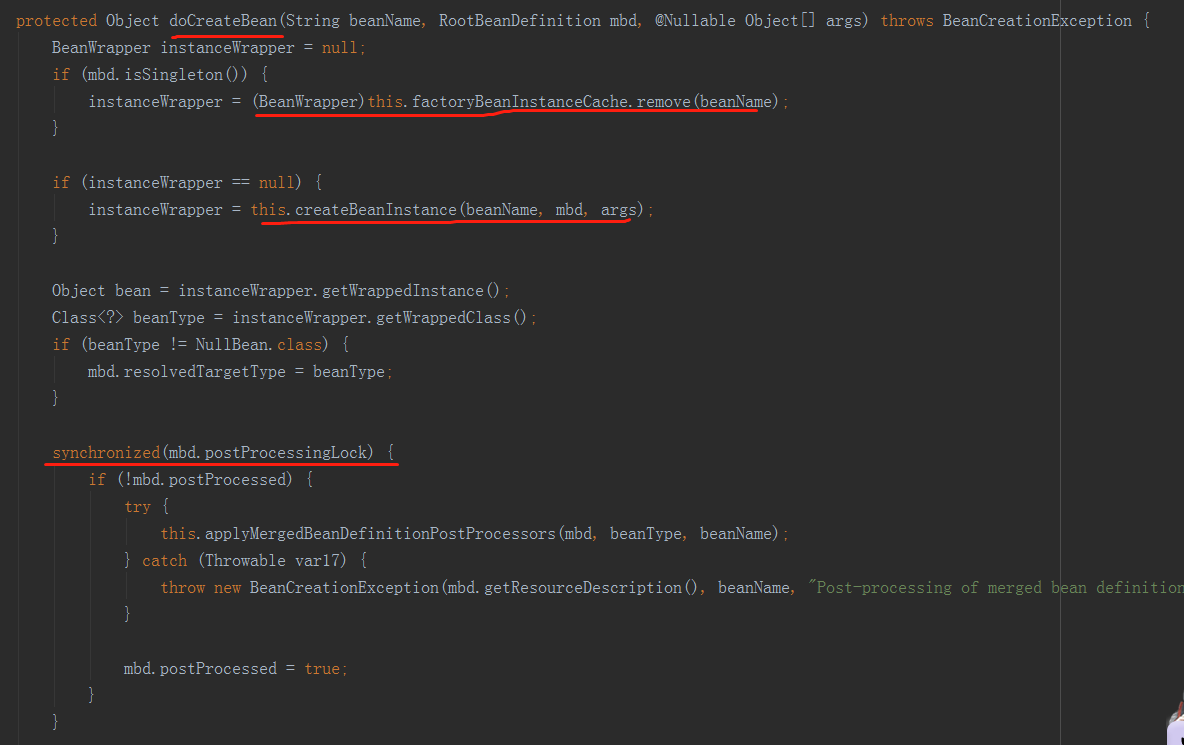
创建这个对象的空实例,相当于new Object(),然后放入BeanWrapper对象中

**populateBean():**

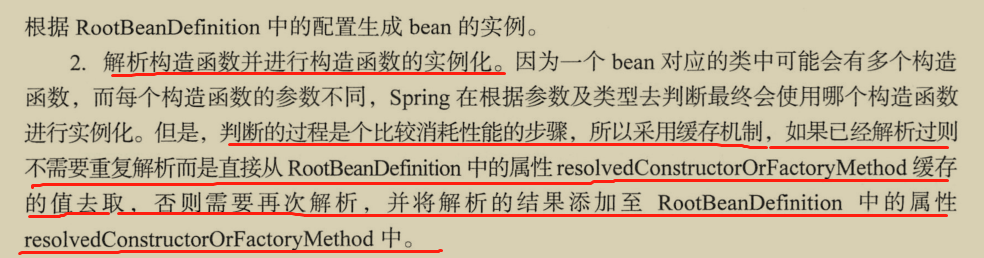
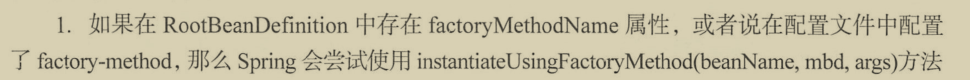
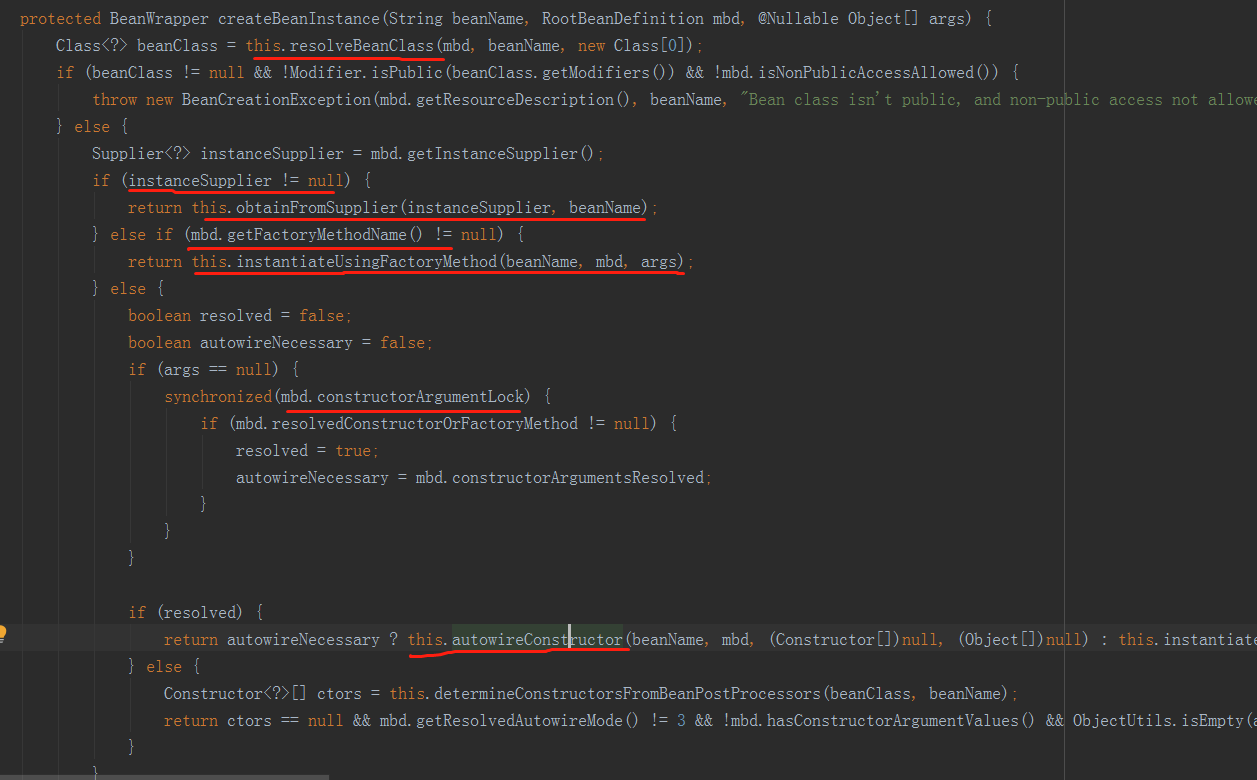
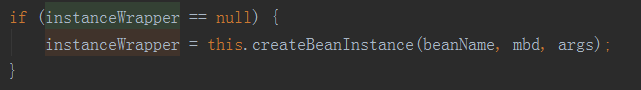
依赖注入的主要过程。相当于调用setter方法。

**initializeBean():**

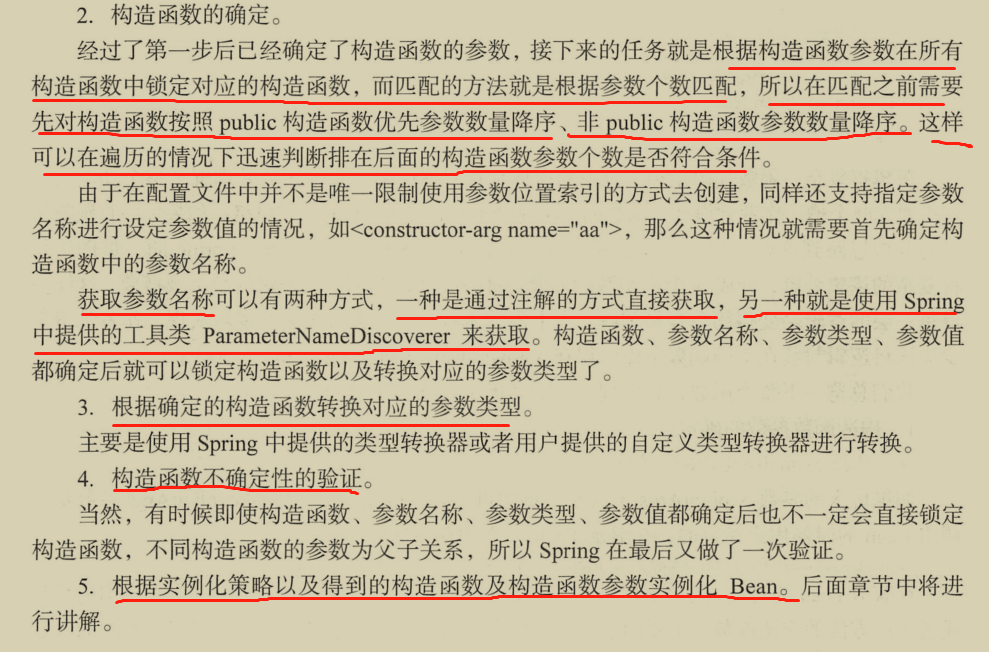
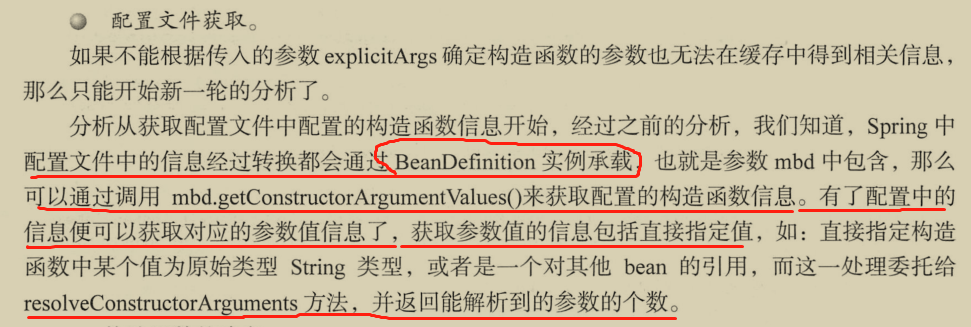
调用init方法,Spring中的init-method。



###### createBeanInstance()：创建bean的实例

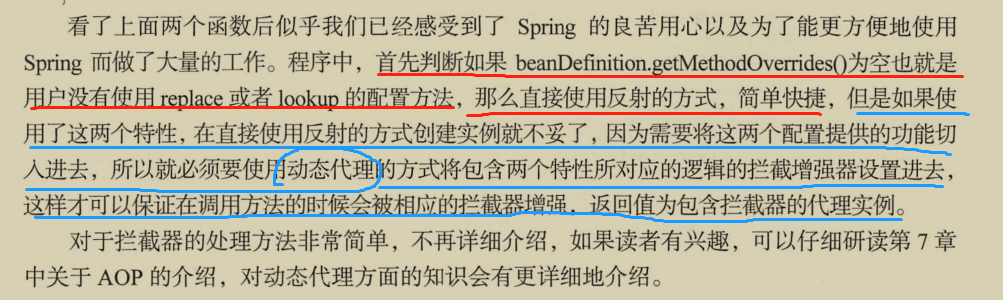
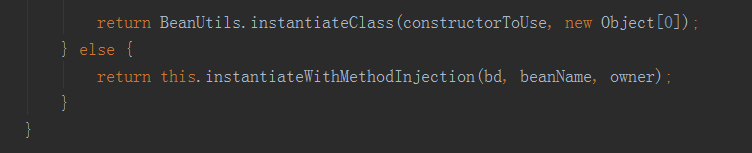
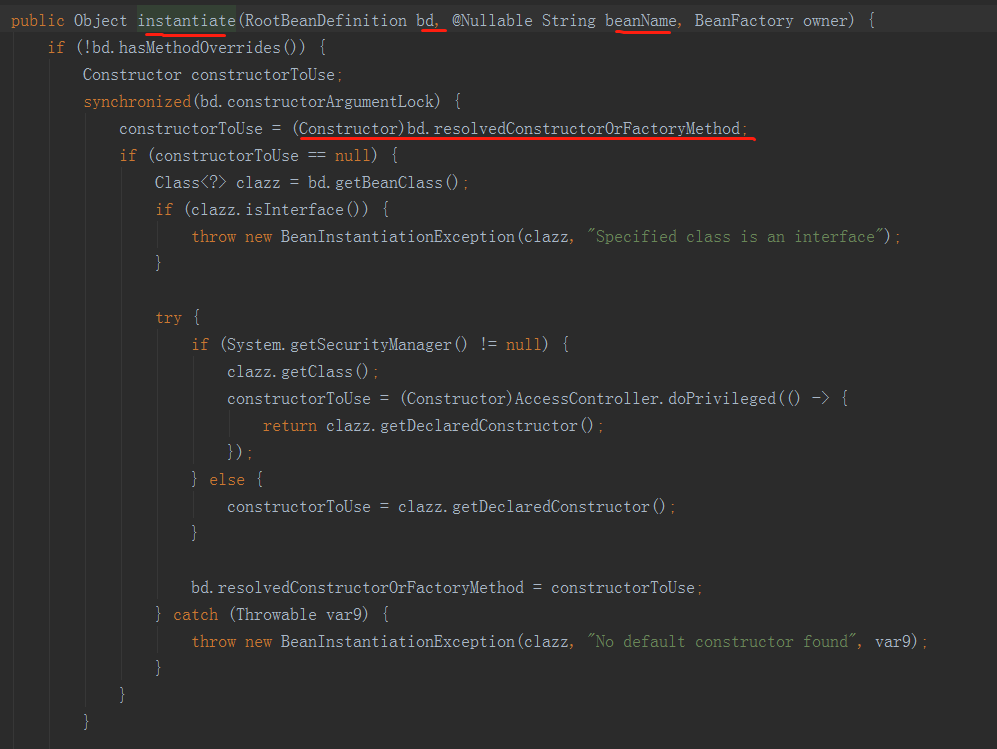
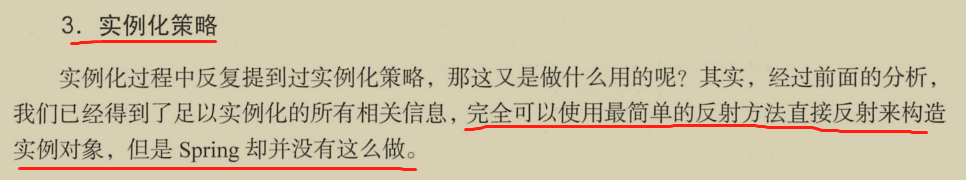
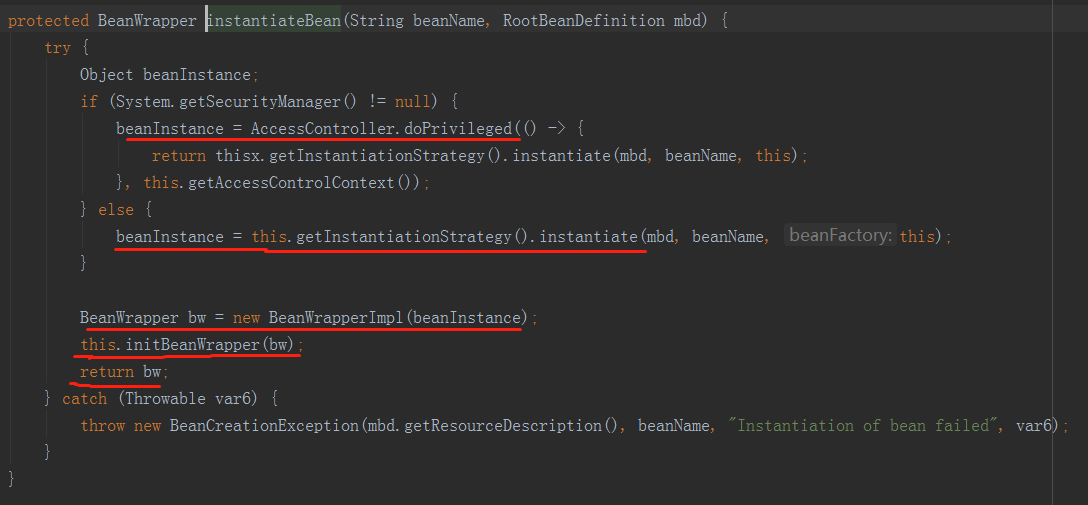


A）autowireConstructor方法简介

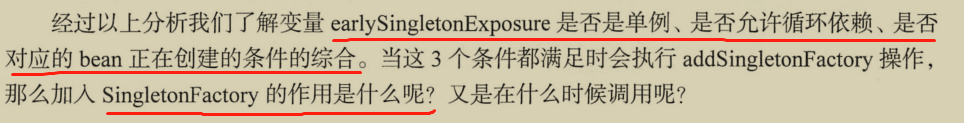
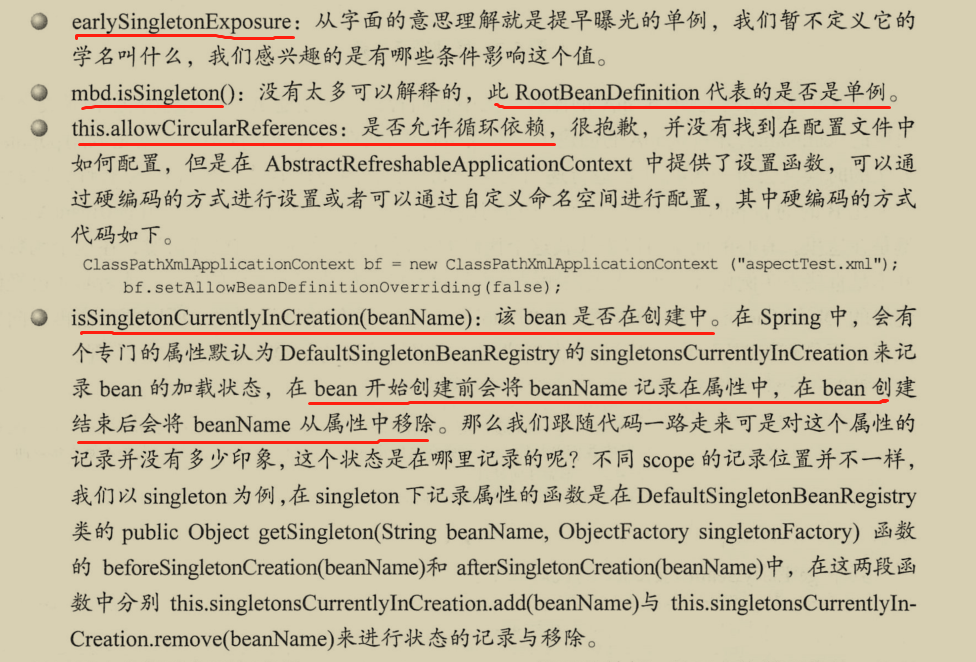
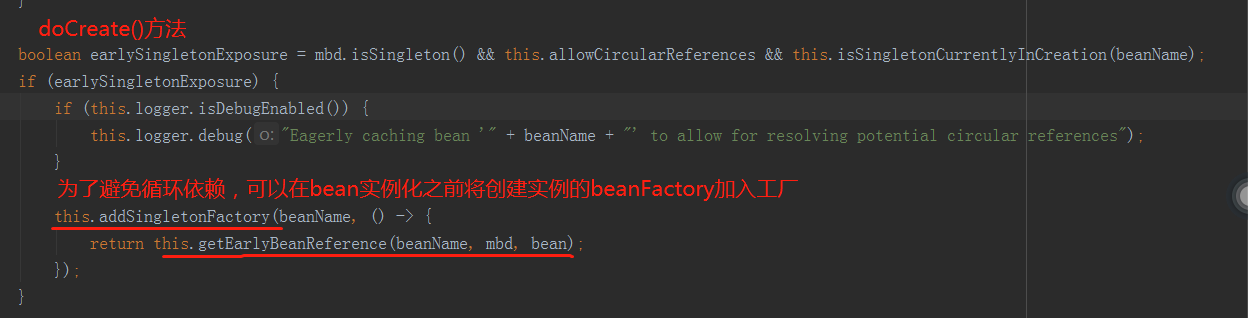


B）instantiateBean方法实例化bean

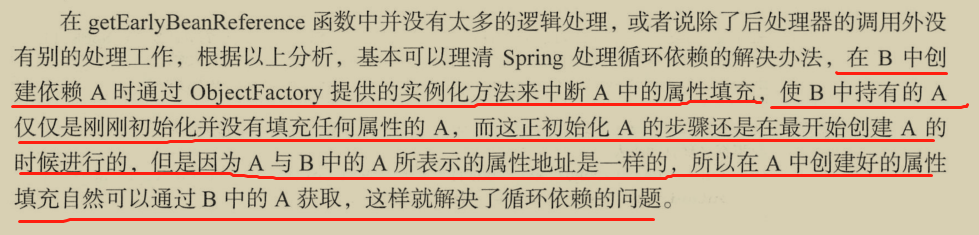
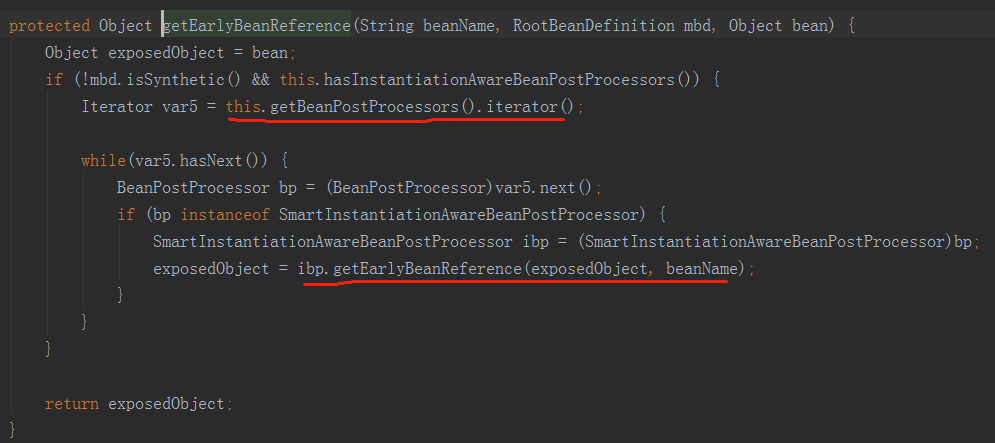
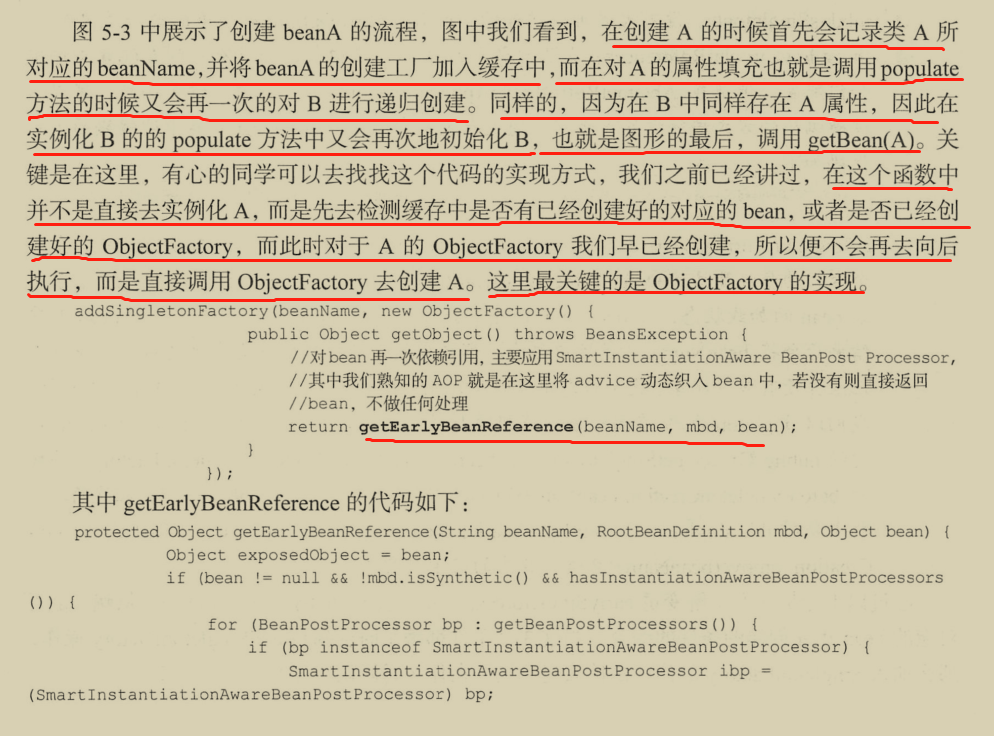
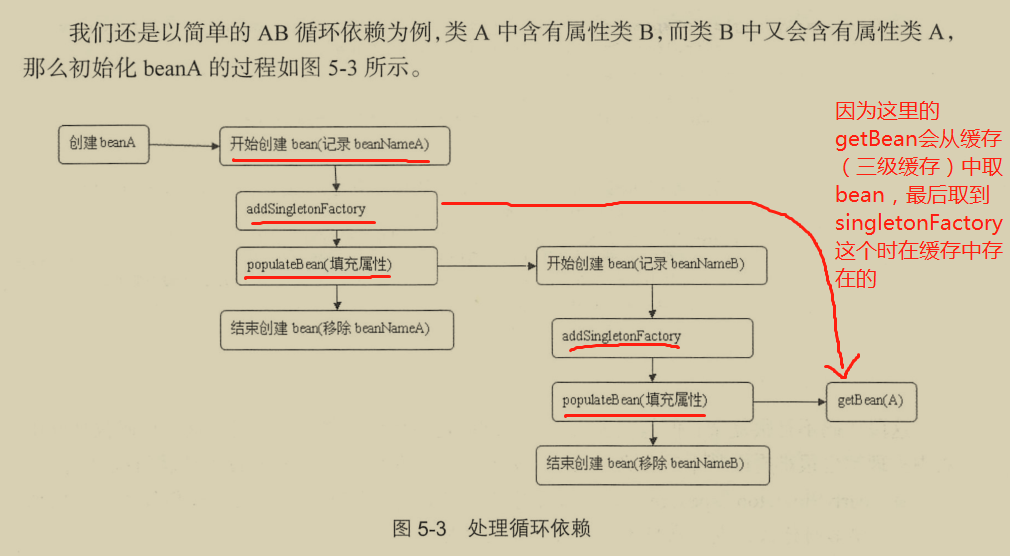
没有参数的实例构造中，直接调用实例化策略实例化就行



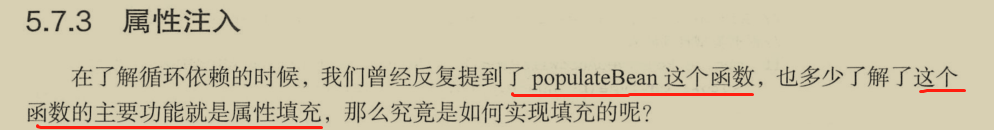
循环依赖解决：记录bean的ObjectFactory

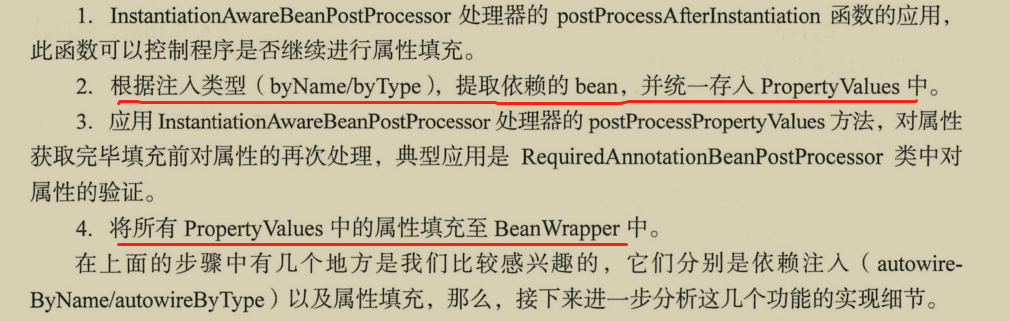
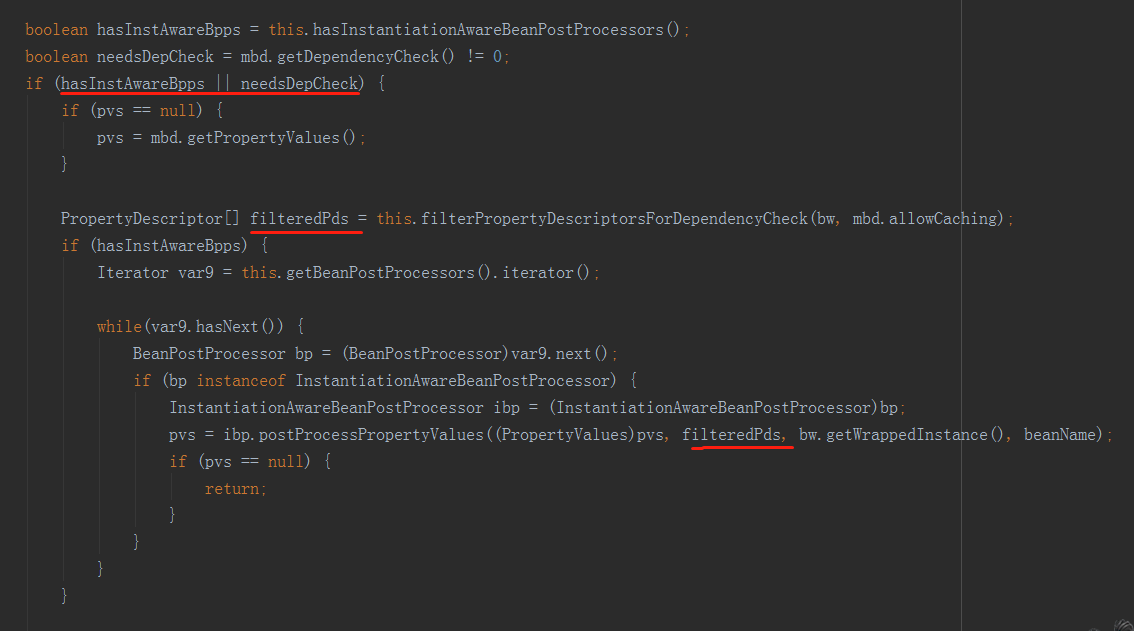
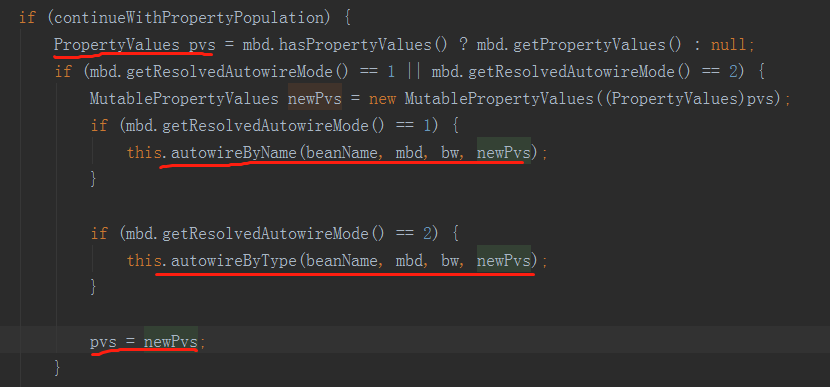


**在这里解决了循环依赖：**

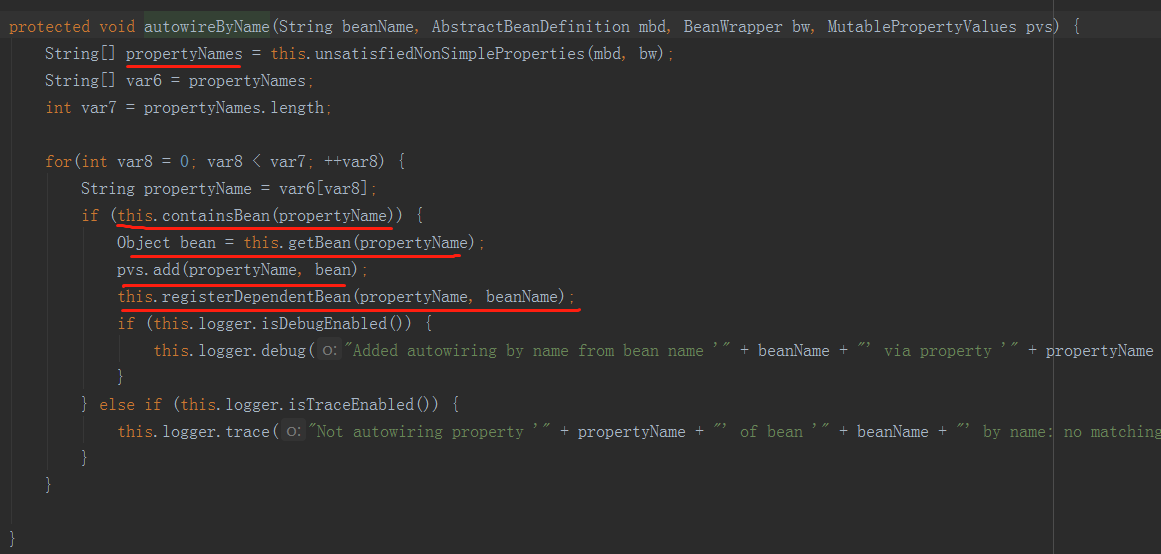


###### 2、populateBean()：依赖注入or属性注入

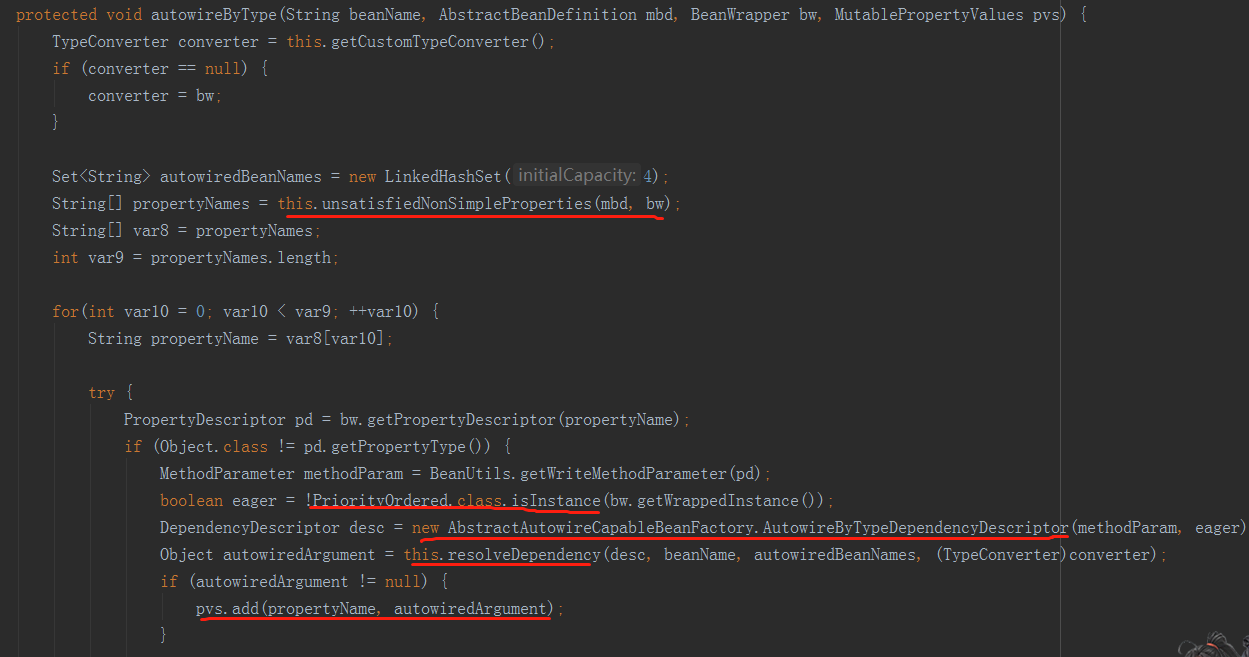
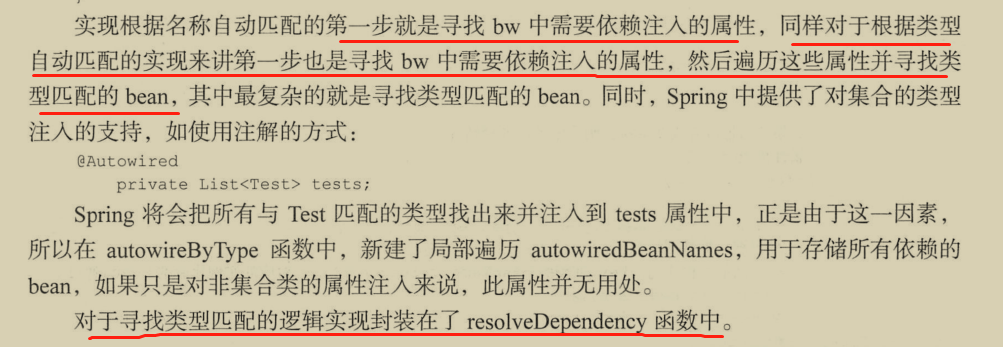
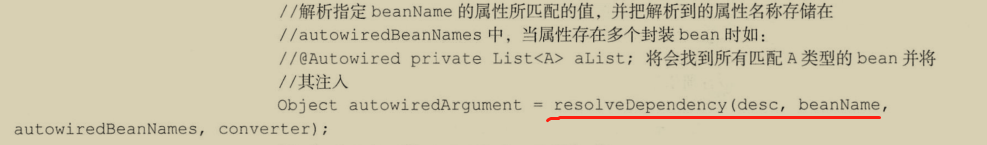




A）autowireByName



B）autowireByType

**认识几个类：**

ParameterNameDiscoverer：用于提取方法参数名称。

DependencyDescriptor：封装了依赖注入点的详细信息，可以是字段 Field，也可以是方法参数 MethodParameter。

AutowireCandidateResolver：判断 DependencyDescriptor 是否是可注入对象。Spring IoC 容器默认实现为 ContextAnnotationAutowireCandidateResolver。

**resolveDependency本质：**

根据类型查找依赖，调用 beanFactory#beanNamesForType 方法根据类型查找依赖名称。

resolveDependency依赖查找解决了以下场景：

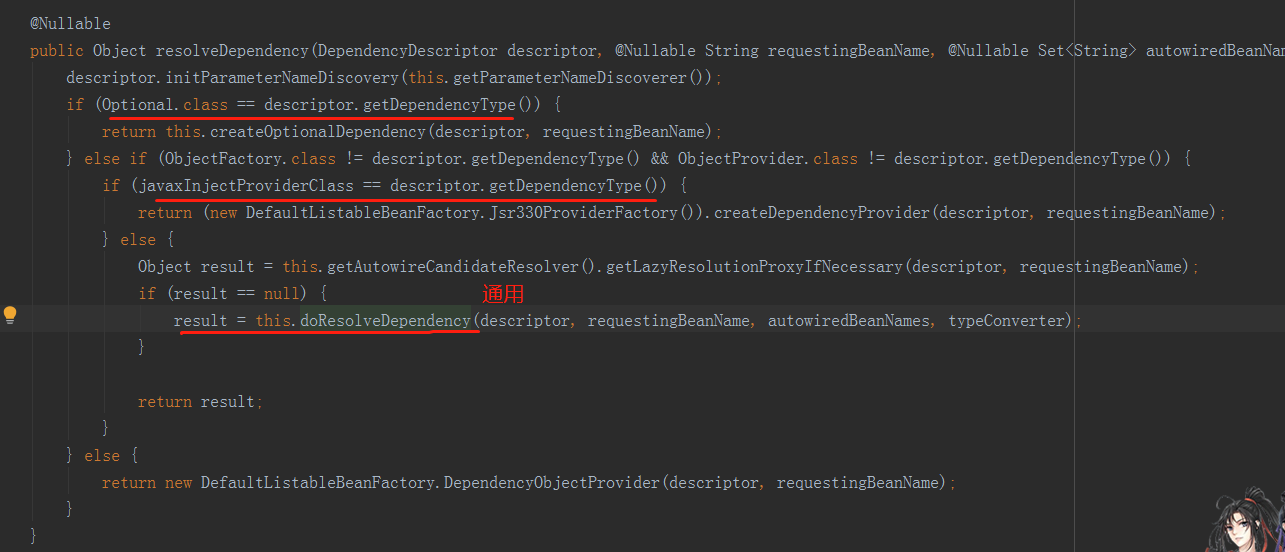
1）Optional：JDK8 提供了 API。主要是将依赖设置非强制依赖，即 descriptor.required=false。

2）延迟依赖注入支持：ObjectFactory、ObjectProvider、javax.inject.Provider 没有本质的区别。

3）另一种延迟注入的支持 - @Lazy 属性。

4）**根据类型查找依赖 -** **doResolveDependency**。

**A）resolveDependency**



**B）doResolveDependency**

通用逻辑处理：根据类型查找依赖

doResolveDependency 封装了依赖查找的各种情况：

**1）快速查找：**

@Autowired 注解处理场景。AutowiredAnnotationBeanPostProcessor 处理 @Autowired 注解时，如果注入的对象只有一个，会将该 bean 对应的名称缓存起来，下次直接通过名称查找会快很多。

1. **注入指定值：**

@Value 注解处理场景。QualifierAnnotationAutowireCandidateResolver 处理 @Value 注解时，会读取 @Value 对应的值进行注入。

如果是 String 要经过三个过程：

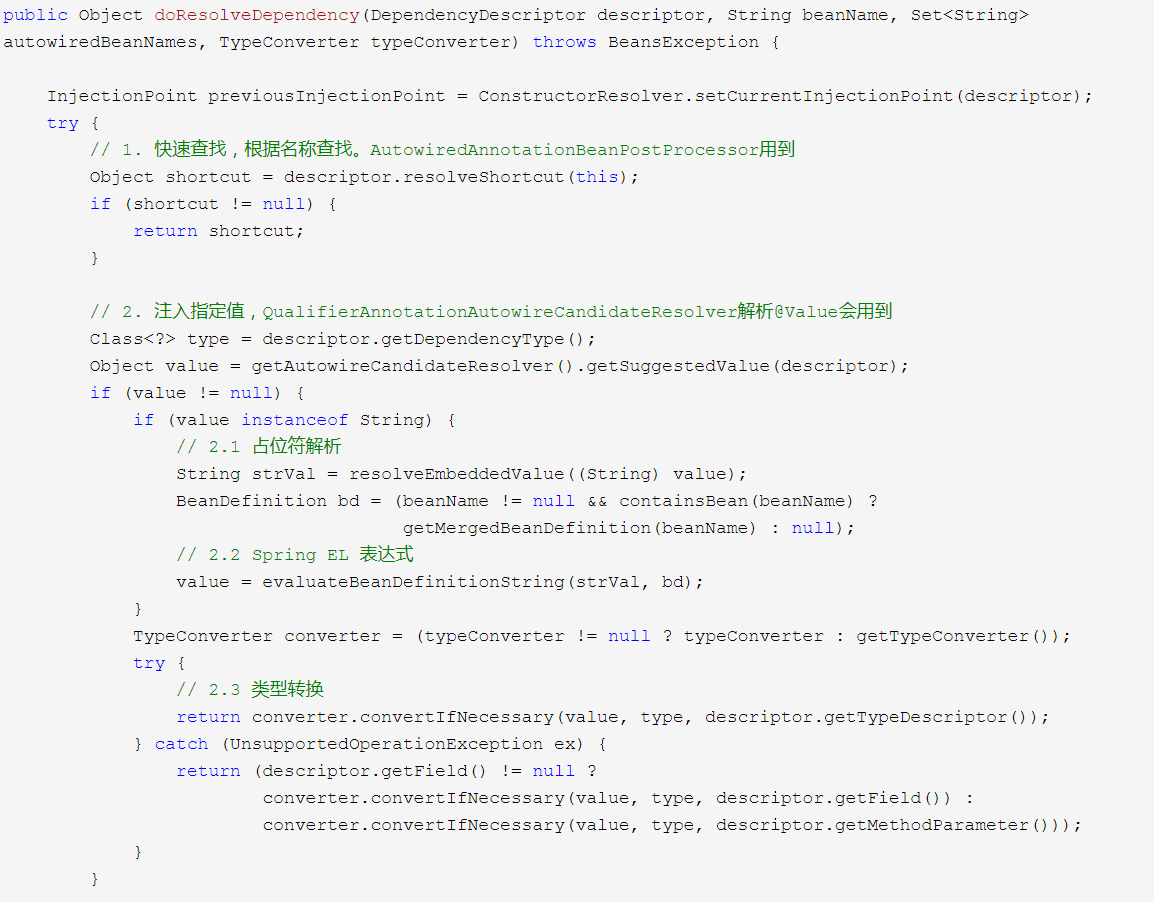
①占位符处理 -> ②EL 表达式解析 -> ③类型转换，这也是一般的处理过程，BeanDefinitionValueResolver 处理 String 对象也是这个过程。

1. **集合依赖查询：**

直接全部委托给 resolveMultipleBeans 方法。

1. **单个依赖查询：**

先调用 findAutowireCandidates 查找所有可用的依赖，如果有多个依赖，则根据规则匹配： @Primary -> @Priority -> ③方法名称或字段名称。



**C）findAutowireCandidates**



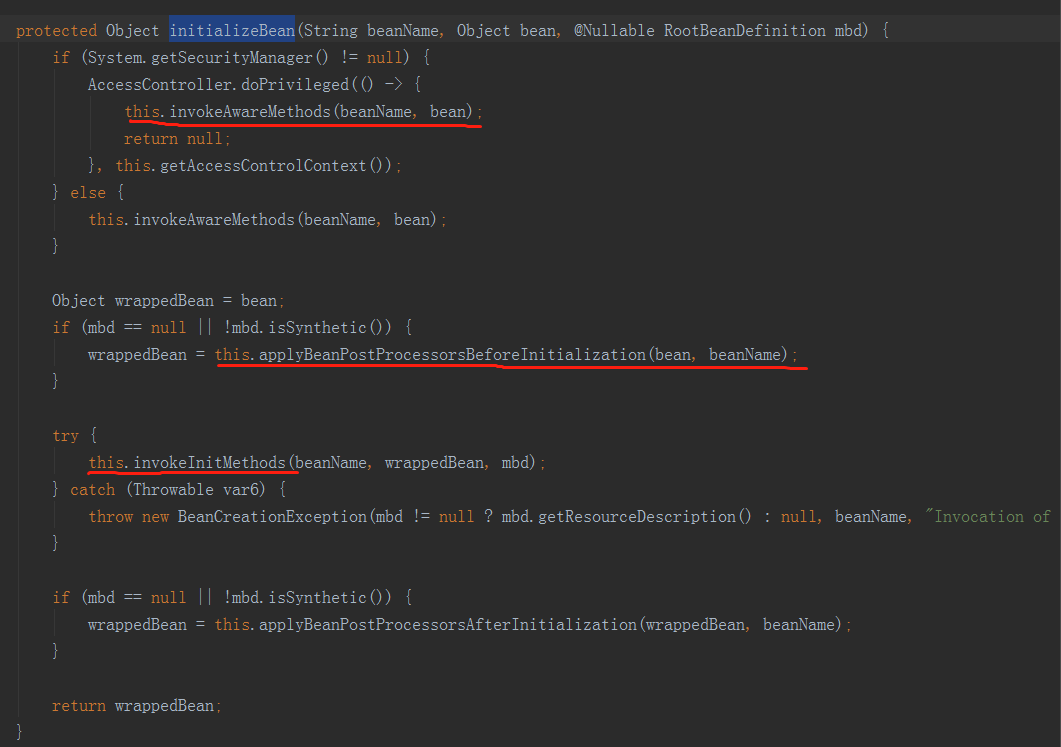
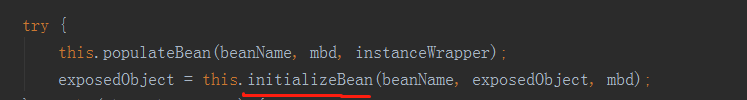
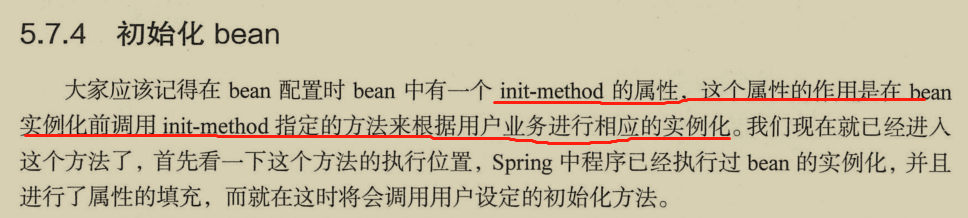
**D）addCandidateEntry**



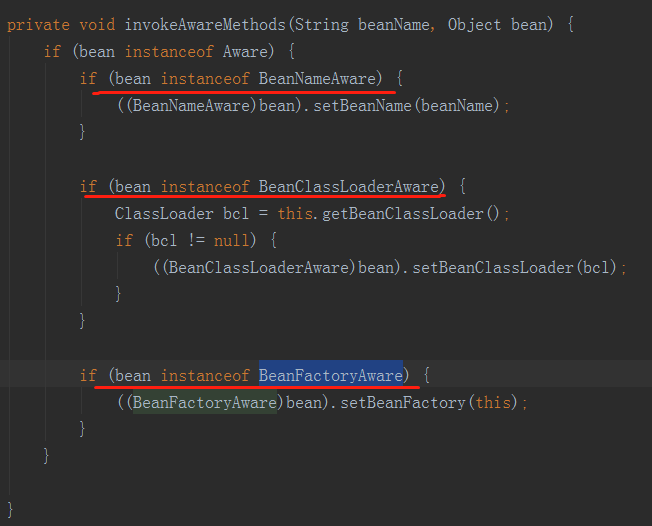
**E）isAutowireCandidate**



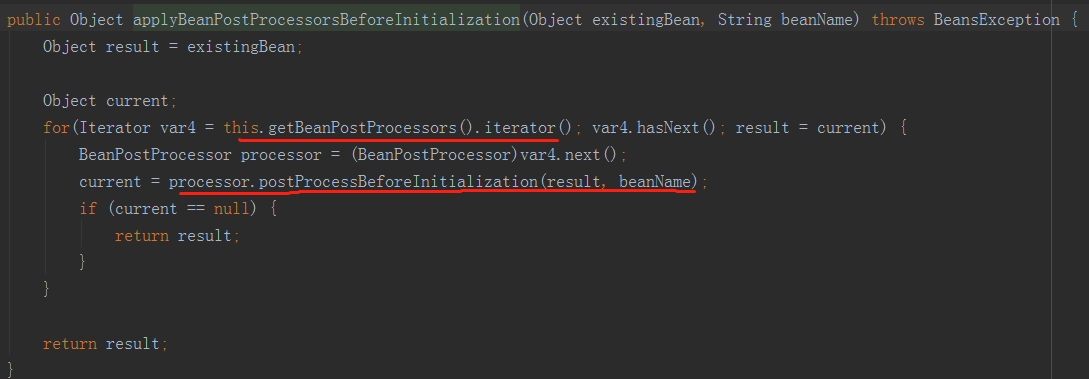
###### 3、initializeBean()：初始化bean



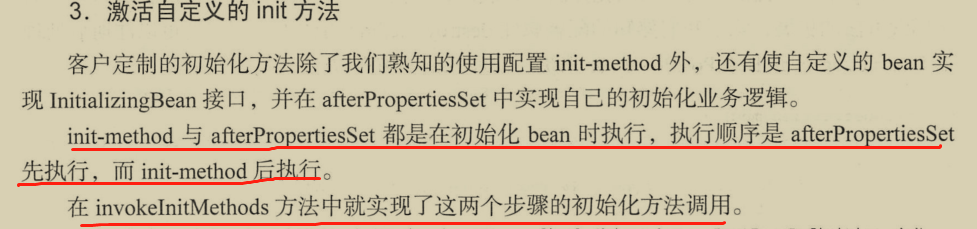
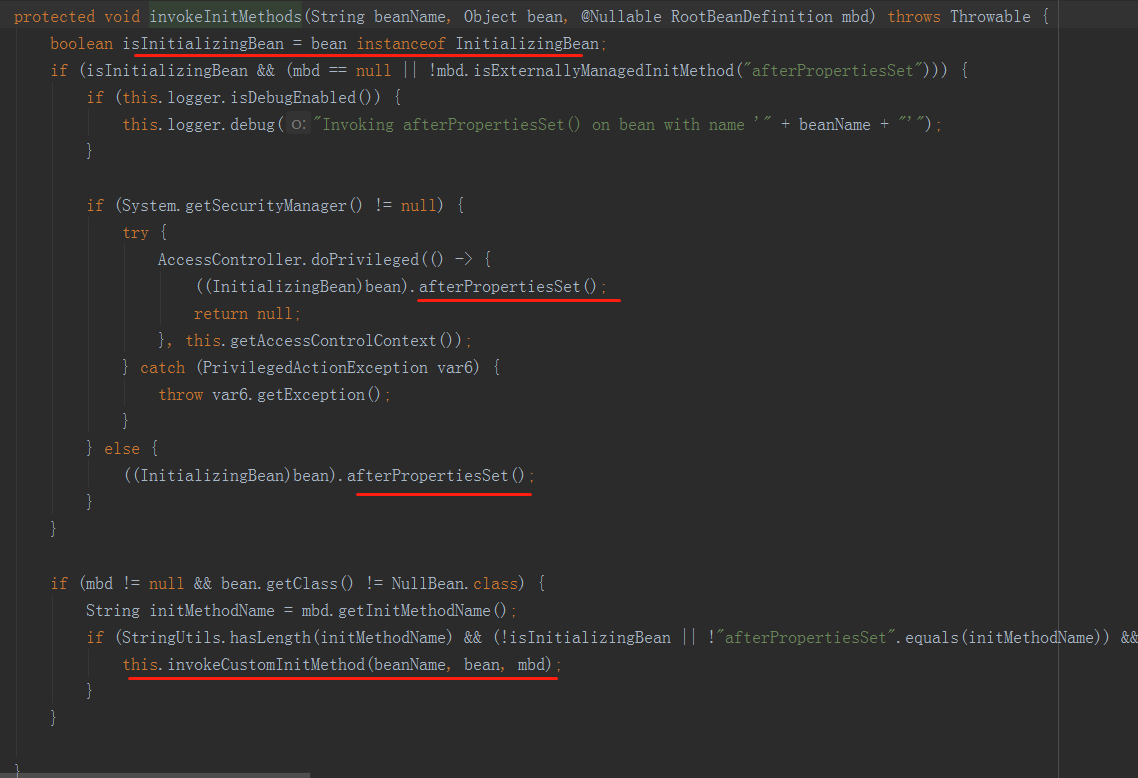
//调用所有BeanNameAware、BeanClassLoaderAware、BeanFactoryAware接口方法



//执行初始化之前的 前置操作



//调用初始化方法

###### 4、registerDisposableBeanIfNecessary()：注册disposableBean

