1. 不使用SingletonObjects作为一级缓存解决循环依赖的原因：若使用SingletonObjects解决循环依赖，则需要在实例化完成后立即放入缓存中。则未填充属性且未初始化的Bean可能被提前获取到。
2. 三级缓存：SingletonObjects、EarlySingletonObjects、SingletonFactories为3级缓存。分别存放成品、只完成实例化的Bean、工厂对象。
3. Bean创建过程
   1. 首先尝试对Bean进行实例化前预解析。预解析使用InstantiationAwarePostProcessors进行实例化前置处理，若处理过程中产生了Bean，则使用BeanPsotProcessors进行初始化后置处理，最后直接返回。
   2. 实例化时会根据参数类型选择合适的构造器，并使用该构造器结合实例化策略进行实例化。实例化策略有JDK实例化和Cglib实例化两种。
   3. 实例化完成的单例Bean会被包装成ObjectFactory并加入3级缓存SingletonFactories中。
   4. 加入3级缓存后，会使用InstantiationAwarePostProcessors进行实例化后置处理。处理过程中如果处理器表示不需要填充属性，则会直接返回Bean。
   5. 填充属性前会使用InstantiationAwarePostProcessors的PostProcessPropertyValues来处理填充属性。@Autowire和@Value的属性填充就是发生在这一步骤中。
   6. 属性填充，基本值直接填充，引用属性BeanReference则根据BeanName去获取Bean。
   7. Bean初始化的步骤是：为感知类注入可感知的属性->BeanPostProcesser对Bean进行初始化前置处理，ApplicationContextAwareProcessor会在这个步骤为ApplicationContextAware类注入ApplicationContext->Initializing的Bean执行afterPropertiesSet方法->初始化方法(BeanDefinition.getInitMethodName)调用