spring AOP解析

通过refresh()方法的registryBeanPostProcessor()方法注册bean后置处理器，aop就是在这个后置处理器加入的

注解:@EnableAspectJAutoProxy

类：AspectJAutoProxyRegistar

方法：registerBeanDefinitions()🡪

AopConfigUtils.registerAspectJAnnotationAutoProxyCreatorIfNecessary()

后置处理器：AnnotationAwareAspectJAutoProxyCreator

用这个注解，让spring再添加上面的后置处理器

spring aop是在运行过程中织入

aspectj在编译过程中织入

refresh()中的initMessageSource()：国际化处理

initApplicationEventMulticaster()：spring的event事件

onRefresh()：空壳方法，没有任何实现

registerListeners()：注册监听器

finishBeanFactoryInitialization()：实例化，从原生对象生成代理对象

**什么时候加载了aop的后置处理器？**

通过@EnableAspectJAutoProxy注解，往spring中动态添加了一个后置处理器AnnotationAwareAspectJAutoProxyCreator，可以对实体类进行动态代理

**实例化过程：**

bean先被new出来，然后在BeanPostProcessor被变为代理对象。在doCreateBean()方法中创建的BeanWrapper,，用其createBeanInstance()方法把bean new出来，然后BeanWrapper.getWrappedInstance()方法获取到原生对象，populateBean()是设置bean的属性，bean是在执行applyBeanPostProcessorAfterInitialization()方法后变为的代理对象

如果@EnableAspectJAutoProxy没有设置proxyTargetClass，默认底层用JdkDynamicAopProxy，如果设置了就用cglib动态代理

interface AopProxy：

CglibAopProxy

JdkDynamicAopProxy

ObjenesisCglibAopProxy

spring源码解析，构造方法如何实例化对象的（自动装配）

IOC容器：微观上理解，就是一个concurrentHashMap，存放了对象的名字和对象的实例。

宏观上理解，指整个spring环境，包括beanFactory、spring内部的处理器、读取器、注册器

preInstantiateSingletones()—>开始了对象的实例化

1、遍历所有的beanDefinitionNames

2、合并父类BeanDefinition

3、判断bd是不是抽象类、单例、懒加载：

（1）如果不是抽象类、是单例、不是懒加载：

判断bd是不是factoryBean

（2）getBean()->doGetBean()

初始化时为什么先调getSingleton(beanName)？

spring在初始化的时候先获取这个对象，判断这个对象是否被实例化好了（普通情况下绝对为空—有一种情况可能不为空，即lazy=true），