

Spring:IOC AOP

IOC 控制反转：用来解决层层之间 耦合

```
A B
@Autowired
IB b;

Spring ioc(A ,B)
```

BeanFactory Spring顶层核心接口,使用了简单工厂模式，负责生产Bean

BeanDefinition Spring顶层核心接口,封装了生产Bean的一切原料

BeanFactoryPostProcessor:对bean定义做修改或者注册

BeanPostProcessor:对bean生产过程做扩展

1.配置类 xml @注解 javaconfig
2.加载spring上下文
xml: new ClassPathXmlApplicationContext("xml");
@: new AnnotationConfigApplicationContext(config.class)
3.getBean()

自定义类，实现BeanFactoryPostProcessor，重写postProcessorBeanFactory方法，该方法有BeanFactory参数，可以获得bean定义，然后自己可以对相应bean定义进行相应修改

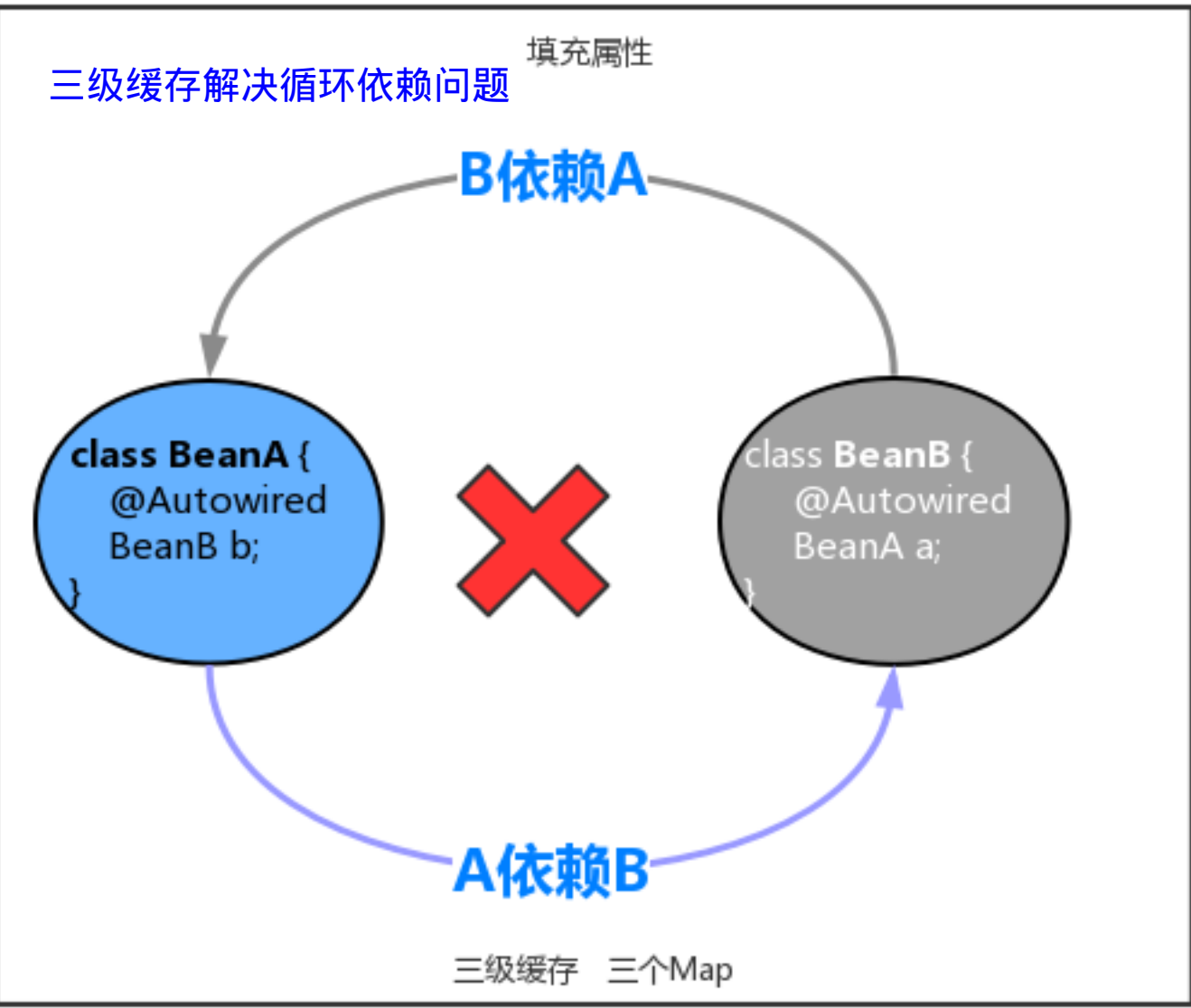
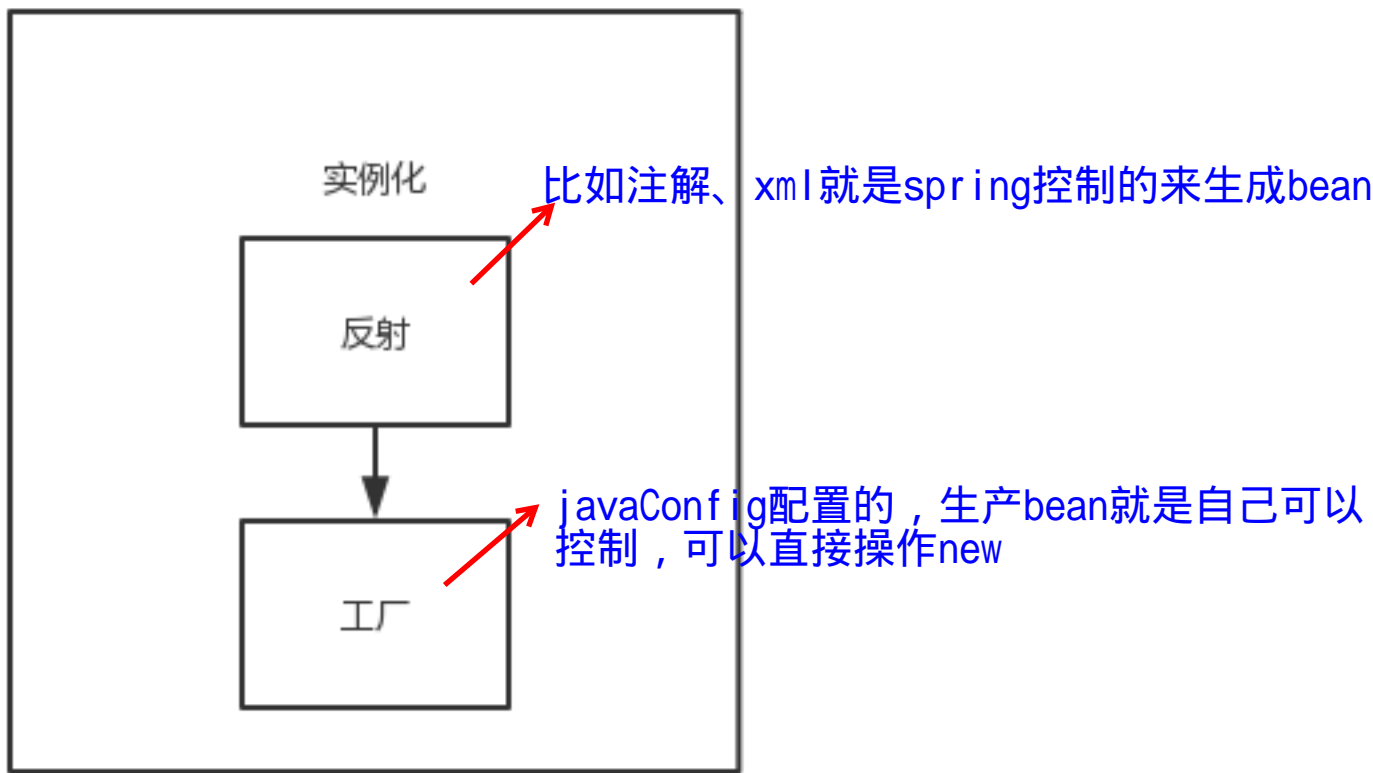
注：
BeanDefinitionRegistryPostProcessor是BeanFactoryPostProcessor的子接口

自定义类，实现BeanDefinitionRegistryPostProcessor，重写postProcessorBeanDefinitionRegistry方法，该方法拿到BeanDefinitionRegistry参数，自己就可以在这里操作比如，多注册一个bean等

两个重要扩展节点，在applicationContext中完成

扩展接口，Bean工厂的后置处理器，可以修改Bean定义当spring集成其他框架时会用到下面两个扩展接口

在这之前属于材料准备，接下去就是真正生产bean的地方



BeanFactory和ApplicationContext的异同点：

- 相同：
- 都有生产bean能力（ApplicationContext实现了BeanFactory）
- 异同：个人理解
- BeanFactory生产Bean比较傻逼，功能比较简单，一步一步的需要手动去组装
 - ApplicationContext比较智能，可以理解为智能化封装好
- 从代码角度：
- BeanFactory只能根据BeanDefinition通过getBean生产Bean
 - Application则智能化，能根据配置文件读取配置，再封装成BeanDefinition，注册BeanDefinition,然后再根据getBean生产出Bean

搞装修 柜子

