# Ioc

1.加载系统/环境变量Environment

2.创建BeanFactory,

加载解析xml propertis yml 解析器接口BeanDefinitionReader

把配置转为BeanDefiniton放入BeanDefinitionMap

如开启注解扫描,创建BeanFactory后置处理器 BeanDefiniton

创建Bean后置处理器 BeanDefiniton

3.执行BeanFactory后置处理器

1) ConfigurationClassPostProcessor

@Configuration @Bean @Component @ComponentScan @ImportResource

@Order @PropertySource @Import(自动装配处理) 生成配置BeanDefinition

4.观察者模式, 初始化多路广播器, 注册Listener到广播器

bean实例化 开辟空间,属性值为默认值

bean初始化 1.填充属性

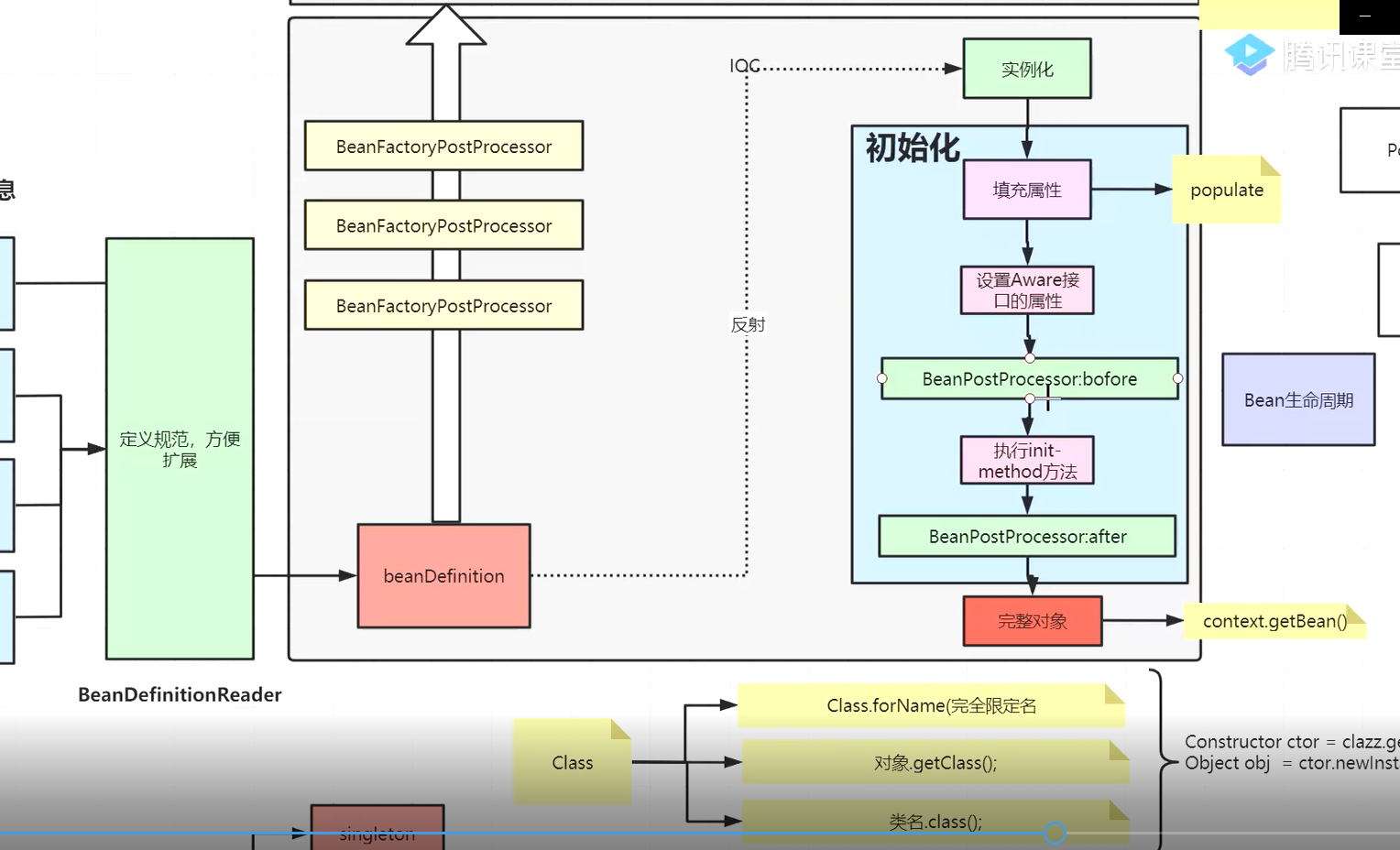
2.设置Aware接口 作用:为bean对象属性设置容器中对象

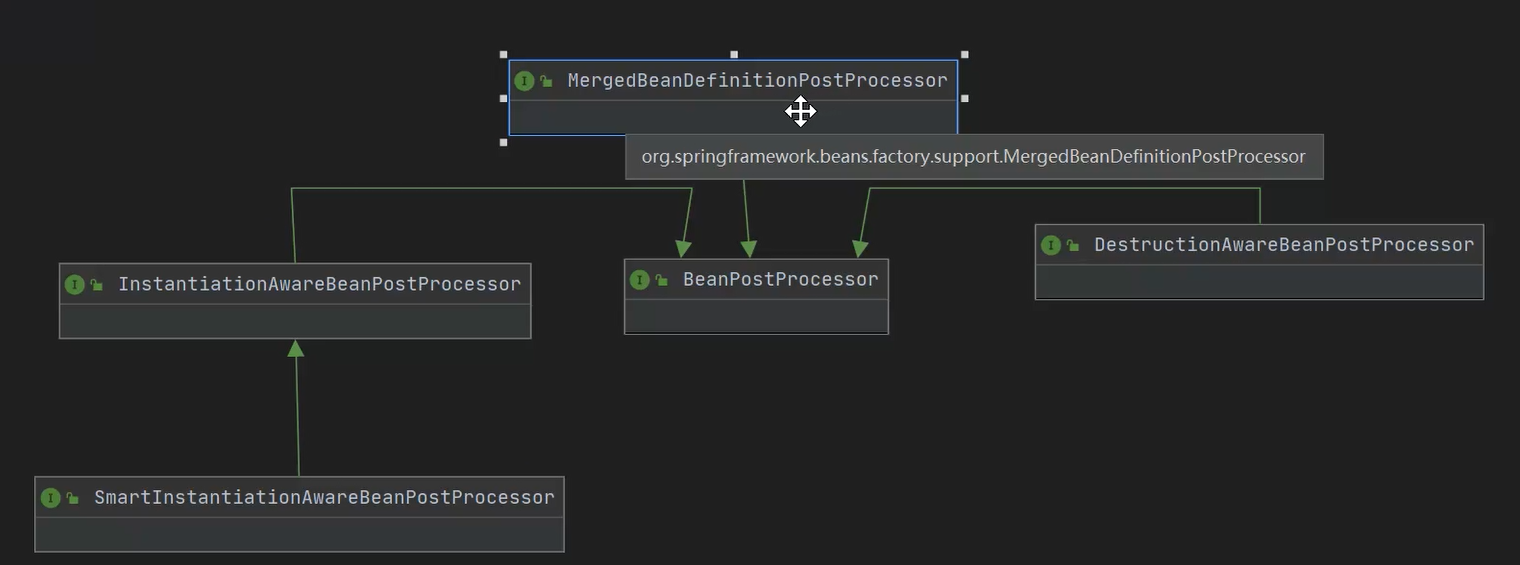
BeanPostProcessor:before 调用aware接口设置属性

执行init-method

BeanPostProcessor:after 切面生成代理对象

Bean完成 -> context.getBean()





## springboot自动装配原理

1. spring启动在执行BeanFactory后置处理器ConfigurationClassPostProcessor,时会处理@Import注解,实例化引入类,并调用其实现方法
2. springboot启动注解中引入@Import(AutoConfigurationImportSelector),其逻辑,会加载META-INF/spring.factories文件,生产BeanDefinition

## 观察者模式

普通观察者模式:

被观察者 implements Observable{ 观察者 implements Observer{

观察者集合 监听方法(事件类型){

**事件1(){}** if(事件1){ 执行1逻辑 }

**事件2(){}** if(事件2){ 执行2逻辑 }

广播(){

遍历观察者结合

调用观察者监听方法(事件类型)

}

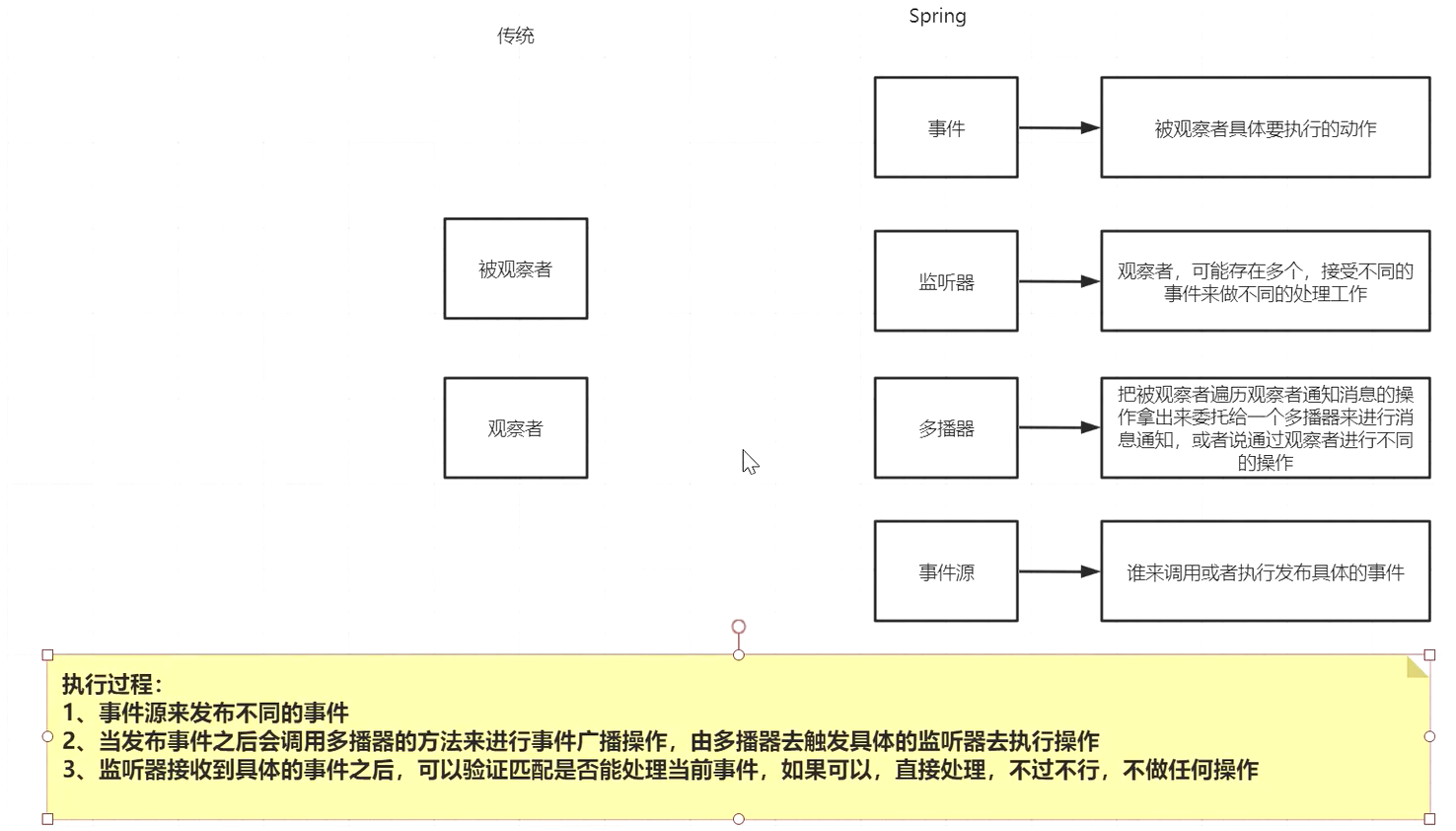
}

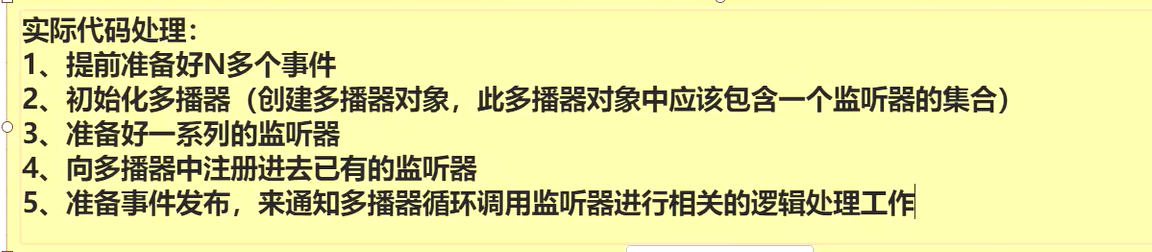
Spring扩展:

**事件接口**

多路广播器

listerner 接口





## BeanFactory 与 FactoryBean区别:

都是用来创建对象

BeanFactory按sping标准流程创建对象的工厂

FactoryBean 实现FactoryBean<自定义类>,由BeanFactory创建对象,此对象为工厂

创建内部对象,单例存放再factoryBeanObjectCache中,多例时时创建

//由spring管理 存入一级缓存

MyFactoryBean myFactoryBean = (MyFactoryBean) ac.getBean( "&myFactoryBean");

//FactoryBean # getObject(){} 中自定义创建的对象 factoryBeanObjectCache中

User user = (User) ac.getBean("myFactoryBean");

MyFactoryBean implements FactoryBean<User> {

@Override

public User getObject() {

return new User("zhangsan");

}

@Override

public Class<?> getObjectType() {

return User.class;

}

@Override

public boolean isSingleton() {

return false;

}