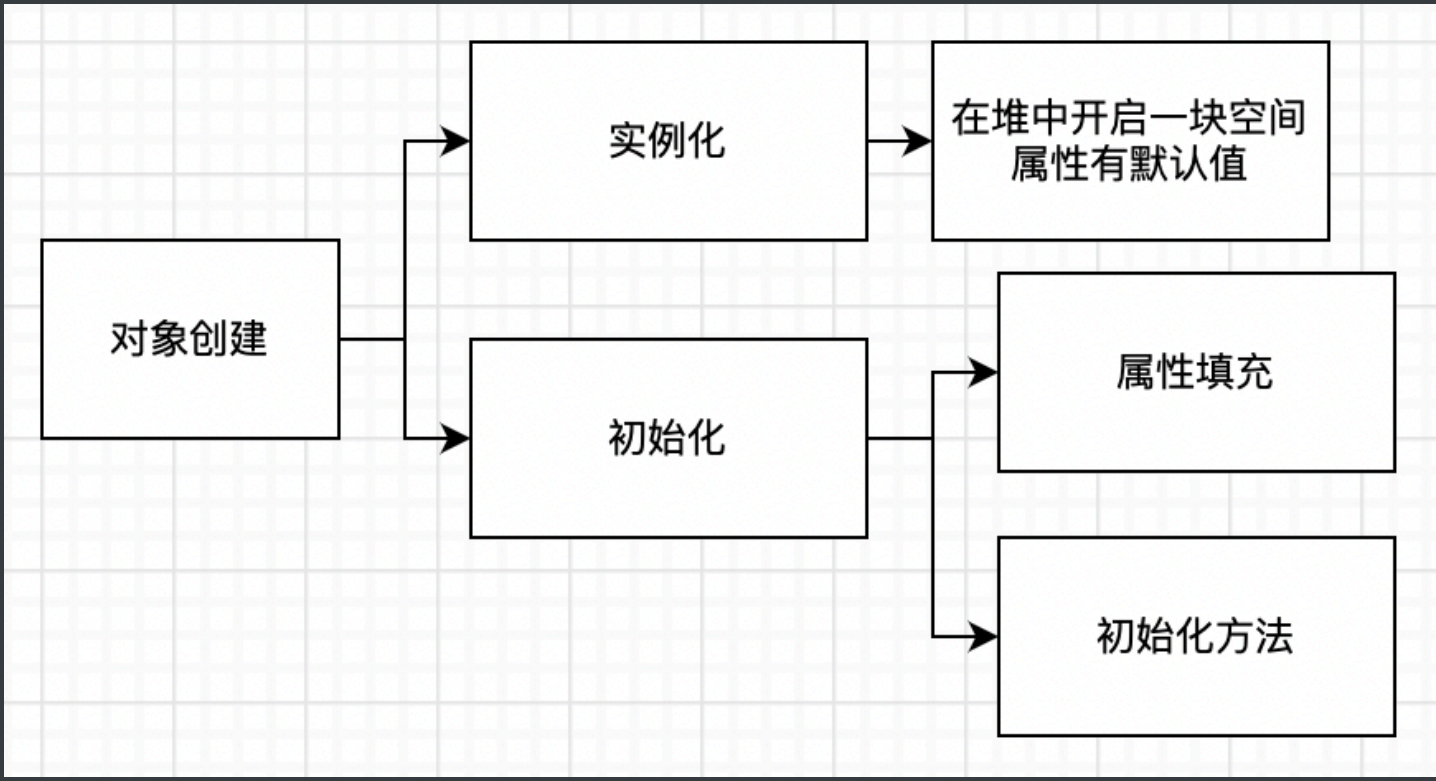
IOC

对象创建



Bean的生命周期

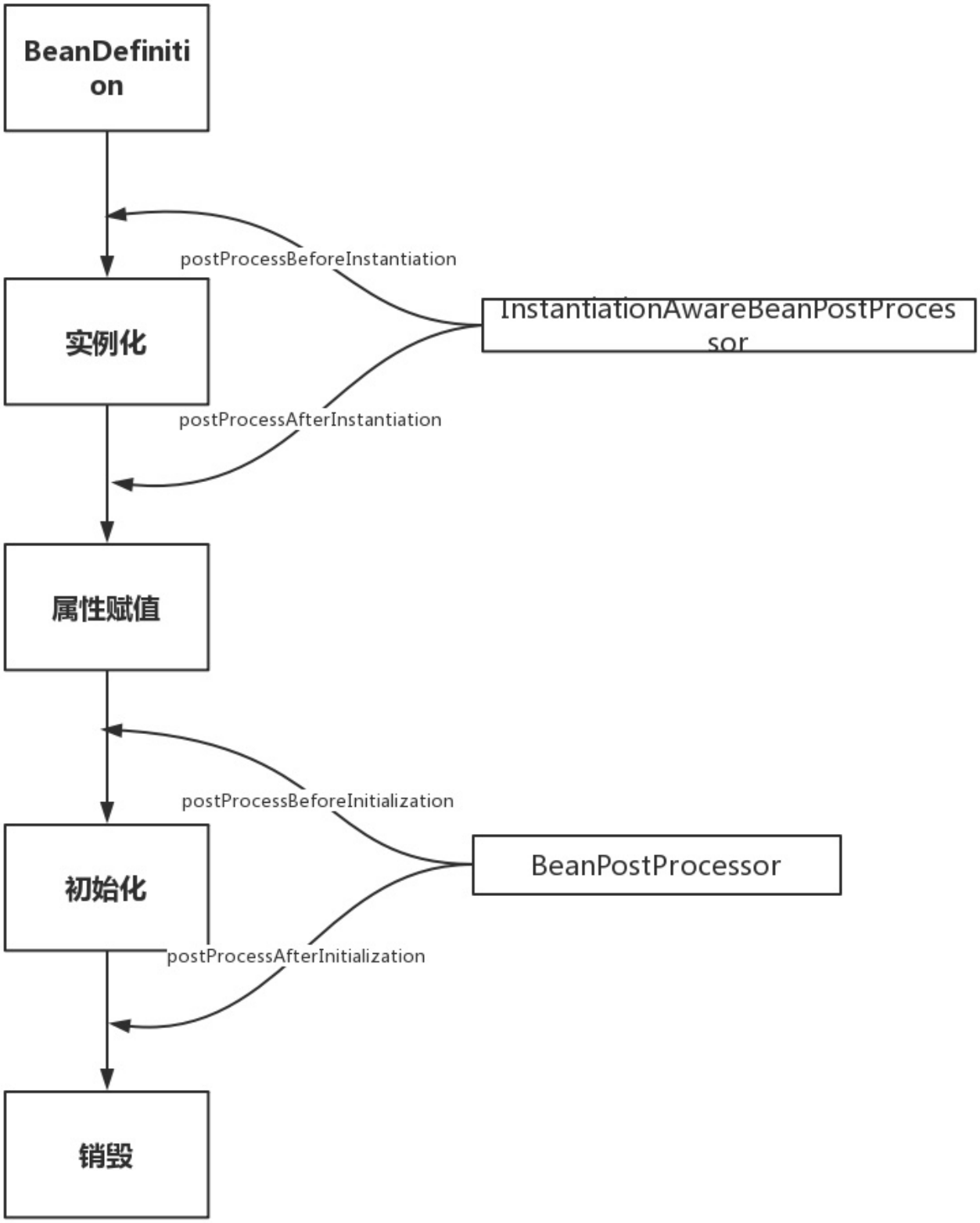
只有四个阶段：

- 实例化
- 属性赋值
- 初始化
- 销毁

常用扩展点

第一大类：影响多个Bean的接口

- BeanPostProcessor 作用于**初始化**阶段前后
- InstantiationAwareBeanPostProcessor 作用于**实例化**阶段前后


BeanDefinition
postProcessBeforeInstantiation
InstantiationAwareBeanPostProcessor
实例化
postProcessAfterInstantiation
属性赋值
postProcessBeforeInitialization
BeanPostProcessor
初始化
postProcessAfterInitialization
销毁

第二大类：只调用一次的接口

- 各种Aware接口

  当前属性对象需要获取当前容器的，某些属性值的时候，需要实现Aware接口。Aware之前的名字就是可以拿到什么资源，例如BeanNameAware可以拿到BeanName。**所有Aware方法都是在初始化阶段之前调用的**

  Group1：

  - BeanNameAware
  - BeanClassLoaderAware
  - BeanFactoryAware

  Group2：

  - EnvironmentAware
  - EmbeddedValueResolverAware
  - ApplicationContextAware

  执行顺序：第一组在第二组之前执行

- 生命周期接口
  - InitializingBean 对应生命周期的初始化阶段
  - DisposableBean 对应生命周期的销毁阶段，实现是通过循环获取所有实现了DisposableBean接口的Bean然后调用其destroy方法。

BeanPostProcessor注册时机与执行顺序：

有两个排序相关的接口：PriorityOrdered、Ordered。

首先PriorityOrdered，接着Ordered，最后是都没有实现

生命周期总结

Spring Bean的生命周期分为**四个阶段**和**多个扩展点**。扩展点又可以分为**影响多个Bean**和**影响单个Bean**。

四个阶段：

- 实例化 Initialization
- 属性赋值 Populate
- 初始化 Initialization
- 销毁 Detruction

多个扩展点：

- 影响多个Bean

  BeanPostProcesser

  InstantiationAwareBeanPostProcessor
- 影响单个Bean

  Aware:
  - Group1:

    BeanNameAware

    BeanClassLoaderAware

    BeanFactoryAware
  - Group2:

    EnvironmentAware

    EmbeddedValueResolverAware

    ApplicationContextAware
- 生命周期

  IntializingBean

  DisposableBean

面试回答：

1. **Spring启动，查找并加载需要被Spring管理的bean，进行Bean的实例化**
2. **Bean实例化后对将Bean的引入和值注入到Bean的属性中**
3. **如果Bean实现了BeanNameAware接口的话，Spring将Bean的Id传递给setBeanName()**

方法

4. 如果Bean实现了BeanFactoryAware接口的话，Spring将调用setBeanFactory()方法，将BeanFactory容器实例传入
5. 如果Bean实现了ApplicationContextAware接口的话，Spring将调用Bean的setApplicationContext()方法，将bean所在应用上下文引用传入进来
6. 如果Bean实现了BeanPostProcessor接口，Spring就将调用他们的postProcessBeforeInitialization()方法。
7. 如果Bean 实现了InitializingBean接口，Spring将调用他们的afterPropertiesSet()方法。类似的，如果bean使用init-method声明了初始化方法，该方法也会被调用
8. 如果Bean 实现了BeanPostProcessor接口，Spring就将调用他们的postProcessAfterInitialization()方法。
9. 此时，Bean已经准备就绪，可以被应用程序使用了。他们将一直驻留在应用上下文中，直到应用上下文被销毁。
10. 如果bean实现了DisposableBean接口，Spring将调用它的destory()接口方法，同样，如果bean使用了destory-method 声明销毁方法，该方法也会被调用。

Refresh过程

1. PrepareRefresh 准备工作
2. obtainFreshBeanFactory
    - 创建容器
    - 加载Bean定义信息
3. prepareBeanFactory 设置BeanFactory属性值
4. postPrecessBeanFactory 空的，留给子类扩展
5. invokeBeanFactoryPostProcessor 实例化及执行所有注册的BeanFactoryPostProcessor Beans
6. registerBeanPostProcessor 实例化并注册所有BeanPostProcessor Beans
7. initMessageSource 国际化
8. initApplicationEventMulticaster 初始化事件监听器（观察者模式）

9. onRefresh 空的，留给子类实现
10. registerListeners 注册事件监听器
11. finishBeanFactoryInitialization 实例化所有剩余非懒加载的单例对象

    beanFactory.preInstantiateSingletons( ):
12. finishRefresh