Bean 后置处理器允许在调用初始化方法前后对 Bean 进行额外的处理。

BeanPostProcessor 接口定义回调方法，可以实现该方法来提供自己的实例化逻辑，依赖解析逻辑等。

也可以在 Spring 容器通过插入一个或多个 BeanPostProcessor 的实现来完成实例化，配置和初始化一个bean之后实现一些自定义逻辑回调方法。通过设置 BeanPostProcessor 实现的 Ordered 接口提供的 order 属性来控制这些 BeanPostProcessor 接口的执行顺序。

Spring IoC 容器实例化一个 bean 实例，然后 BeanPostProcessor 接口进行它们的工作。

ApplicationContext 会自动检测由 BeanPostProcessor 接口的实现定义的 bean，注册这些 bean 为后置处理器，然后通过在容器中创建 bean，在适当的时候调用它。

Spring提供了两种常用的后处理器：

- Bean后处理器: 这种后处理器会对容器中Bean进行后处理，对Bean进行额外加强。
- 容器后处理器: 这种后处理器会对IoC容器进行后处理，用于增强容器功能。

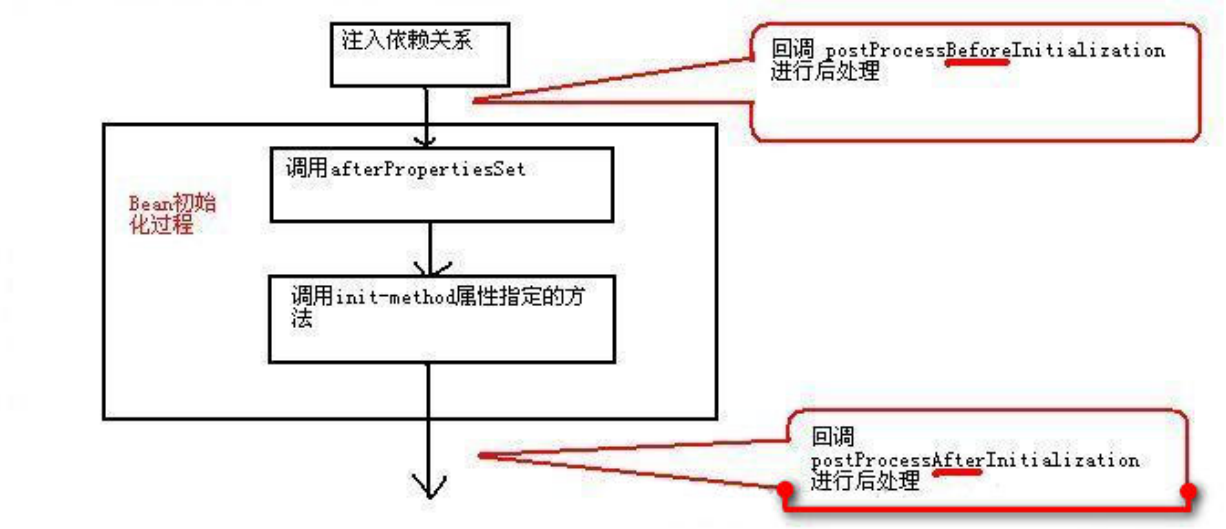
Bean后处理器

Bean后处理器是一种特殊的Bean，这种特殊的Bean并不对外提供服务，它甚至可以无须id属性，它主要负责对容器中的其他Bean执行后处理，例如为容器中的目标Bean生成代理等，这种Bean称为Bean后处理器。Bean后处理器会在Bean实例创建成功之后，对Bean实例进行进一步的增强处理。Bean后处理器必须实现BeanPostProcessor接口，同时必须实现该接口的两个方法。

1. Object postProcessBeforeInitialization(Object bean, String name) throws BeansException: 该方法的第一个参数是系统即将进行后处理的Bean实例，第二个参数是该Bean的配置id

2. Object postProcessAfterinitialization(Object bean, String name) throws BeansException: 该方法的第一个参数是系统即将进行后处理的Bean实例，第二个参数是该Bean的配置id

容器中一旦注册了Bean后处理器，Bean后处理器就会自动启动，在容器中每个Bean创建时自动工作，Bean后处理器两个方法的回调时机如下图：



注意一点，如果使用BeanFactory作为Spring容器，则必须手动注册Bean后处理器，程序必须获取Bean后处理器实例，然后手动注册。

```
BeanPostProcessor bp = (BeanPostProcessor)beanFactory.getBean("bp");
beanFactory.addBeanPostProcessor(bp);
Person p = (Person)beanFactory.getBean("person");
```

容器后处理器

Bean后处理器负责处理容器中的所有Bean实例，而容器后处理器则负责处理容器本身。容器后处理器必须实现BeanFactoryPostProcessor接口，并实现该接口的一个方法postProcessBeanFactory(ConfigurableListableBeanFactory beanFactory)实现该方法的方法体就是对Spring容器进行的处理，这种处理可以对Spring容器进行自定义扩展，当然也可以对Spring容器不进行任何处理。

类似于BeanPostProcessor，ApplicationContext可自动检测到容器中的容器后处理器，并且自动注册容器后处理器。但若使用BeanFactory作为Spring容器，则必须手动调用该容器后处理器来处理BeanFactory容器。