BeanFactory与ApplicationContext的关系

1. BeanFactory是Spring的早期接口，称为Spring的Bean工厂，ApplicationContext是后期更高级接口，称之为Spring容器；
2. ApplicationContext在BeanFactory基础上对功能进行扩展，例如：监听功能、国际化功能等。BeanFactory的API更加偏向底层，ApplicationContext的API大多数是对这些底层API的封装；
3. Bean创建的主要逻辑和功能都被封装在BeanFactory中，ApplicationContext不仅继承了BeanFactory，而且ApplicationContext内部还维护这BeanFactory的引用，所以ApplicationContext与BeanFactory既有继承关系，又有融合关系。
4. Bean的初始化时机不同，原始BeanFactory是在首次调用getBean时才进行Bean的创建，而ApplicationContext则是配置文件加载，容器一创建就讲Bean都实例化并初始化好。



Bean的范围配置

默认情况下，单纯的Spring环境Bean的作用范围有两个：Singleton和Prototype

Singleton：单例，默认值，Spring容器创建的时候，就会进行Bean的实例化，并存储到容器内部的单例池中，每次getBean时都是从单例池中获取相同的Bean实例；

Prototype：原型，Spring容器初始化时不会创建Bean实例，当调用getBean时候才会实例化Bean，每次getBean都会创建一个新的Bean实例。

Bean的延迟加载

当lazy-init设置为ture的时候进行懒加载，也就是当Spring容器创建的时候，不会立刻创建Bean实例，而是等待用到时在创建Bean实例并存储到单例池中去，后续在使用该Bean直接从单例池获取即可，本质上该Bean还是单例的。

Bean实例化的基本流程

Spring容器在进行初始化时，会将xml配置的<bean>的信息封装成一个BeanDefinition对象，所有的BeanDefinition存储到一个名为beanDefinitionMap的Map集合中去，

Spring框架在对该Map进行遍历，使用反射创建Bean实例对象，

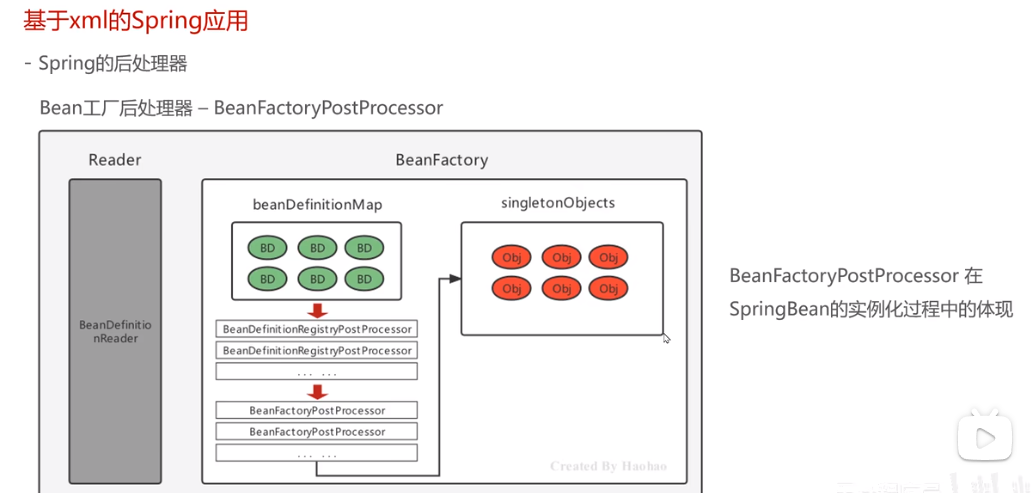
创建好的Bean对象存储在一个名为singletonObjects的Map集合中，当调用getBean方法时则最终从该Map集合中取出Bean实例对象返回。

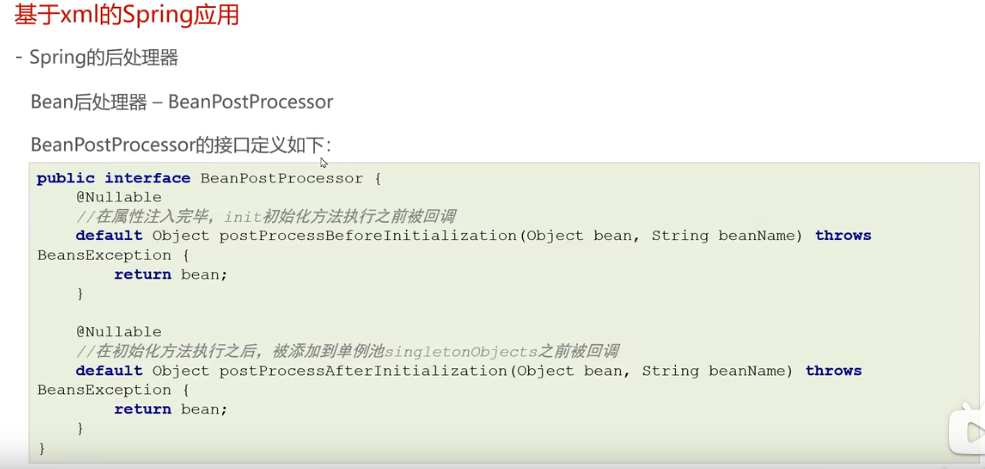
Spring的后处理器

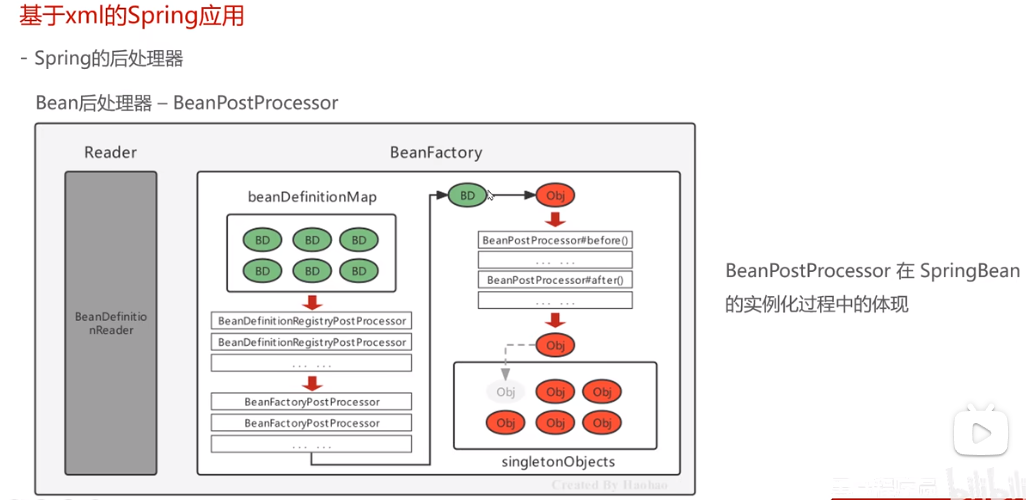
Spring的后处理器是Spring对外开发的重要扩展点，允许我们介入到Bean的整个实例化流程中来，以达到动态注册BeanDefinition，动态修改BeanDefinition，以及动态修改Bean的作用。Spring主要有两种后处理器：

BeanFactoryPostProcessor：Bean工厂后处理器，在BeanDefinitionMap填充完毕，Bean实例化之前执行；（用于对BeanDefiniton注册和修改的功能）

BeanPostProcessor：Bean后处理器，一般在Bean实例化之后，填充到单例池singletonObjects之前执行。









Spring Bean的生命周期

由于Bean的初始化阶段的步骤比较复杂，所以着重研究Bean的初始化阶段

Spring Bean的初始化过程涉及如下几个过程：

Bean实例的属性填充

Aware接口属性注入

BeanPostProcessor的before()方法回调

InitializingBean接口的初始化方法回调

自定义初始化方法init回调

BeanPostProcessor的after()方法回调



Bean实例属性填充

Spring在进行属性注入时，会分为如下几种情况：

注入普通属性，String、int或存储基本数据类型的集合时，直接通过set方法的反射设置进去；

注入单向对象引用属性时，从容器中getBean获取后通过set方法反射设置进去，如果容器中没有，则先创建被注入对象Bean实例（完成整个生命周期）后，在进行注入操作；

注入双向对象引用属性时，就比较复杂了，涉及了循环引用（循环依赖）问题。

三级缓存解决循环依赖

UserService和UserDap循环依赖的过程结合上述三级缓存描述一下

UserService实例化对象，但尚未初始化，将UserService存储到三级缓存；

UserService属性注入，需要UserDao，从缓存中获取，没有UserDao；

UserDao实例化对象，但尚未初始化，讲UserDao存储到三级缓存；

UserDao属性注入，需要UserService，从三级缓存中获取UserService，UserService从三级缓存移入二级缓存；

UserDao执行其他生命过程，最终成为一个完整Bean存储到一级缓存，删除二三级缓存

UserService注入UserDao；

UserService执行其他生命过程，最终成为一个完整Bean存储到一级缓存，删除二三级缓存



