对象生命周期

2021年9月19日 15:18

Spring容器提供了一系列接口，让我们可以在整个对象的生命周期的不同阶段来做一些处理

Bean级生命周期接口

只会影响特定的一个Bean的生命周期，即只会在特定的一个Bean的生命周期中才会起作用

五个Bean级生命周期接口

- BeanNameAware：用于获得Bean对象名称

- BeanFactoryAware：用来得到BeanFactory对象

- ApplicationContextAware：用来得到ApplicationContext容器对象

- InitializingBean：Bean创建时的接口

- DispoableBean：Bean废弃时的接口

这五个接口只针对特定的Bean对象（实现这些接口的）起作用，如果某个Bean对象需要使用这些接口，Bean对象就可以去实现implements这些接口，这些接口中的方法就会在生命周期的不同阶段被调用

容器级生命周期接口

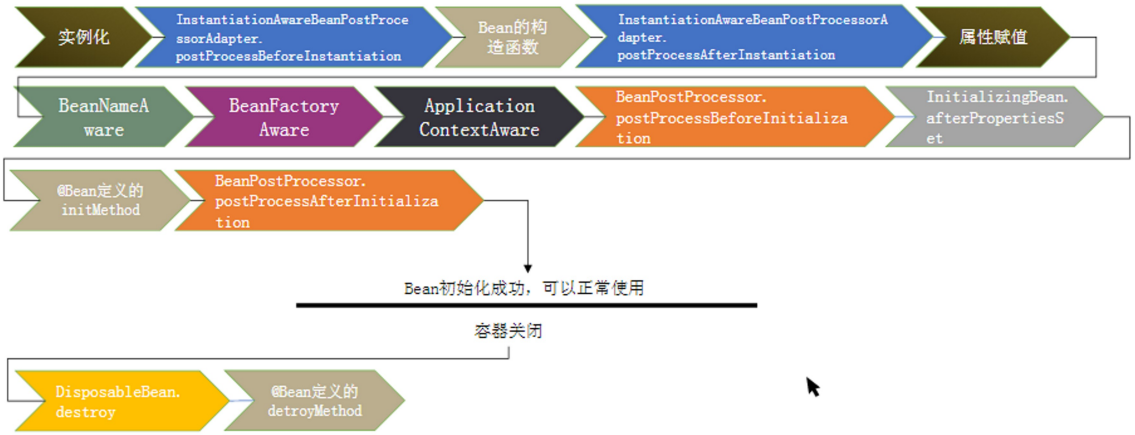
会对容器中所有的Bean起作用

主要的两个容器级生命周期接口

- InstantiationAwareBeanPostProcessorAdapter

- BeanPostProcessor

两类接口的调用顺序



当我们实例化一个对象时，首先要用InstantiationAwareBeanPostProcessorAdapter中的*postProcessBeforeInstantiation*

~~然后调用**Bean对象自身的构造函数**，然后调用InstantiationAwareBeanPostProcessorAdapter中的*postProcessPropertyValues*（该接口已被废弃）~~

然后调用InstantiationAwareBeanPostProcessorAdapter的*postProcessAfterInstantiation*（注意此处与课程视频不同）

（*postProcessAfterInstantiation*的调用在Bean对象的构造函数之后，Bean对象属性赋值之前）

然后**对Bean对象的属性赋值**，这些属性往往来自配置中告诉它的某些关联，或者是某些配置文件的信息是需要放到Bean的属性中的

赋值完成后，调用BeanNameAware的*setBeanName*，获得Bean对象的名称

然后调用BeanFactoryAware的*setBeanFactory*，拿到BeanFactory对象

然后调动ApplicationAware的*setApplicationContext*，拿到ApplicationContext对象

然后调用BeanPostProcessor的*postProcessBeforeInitialization*

然后调用InitializingBean的*afterPropertiesSet*（如果我们想让某些方法或属性在容器运行起来之后就加载，可以放在afterProperties里）

然后调用**@Bean**注解中指定的**initMethod**方法

然后调用BeanPostProcessor的*postProcessAfterInitialization*

至此，Bean初始化完成，可以正常使用

当容器关闭时，首先调用DisposableBean的*destroy*方法

然后调用**@Bean**注解中指定的**destroyMethod**方法