#### Spring Framework源码阅读

##### 之Bean加载：加载流程

问题描述

Spring Boot 2.2.5

Spring Framework 5.2.6

JDK 1.8

Spring Bean的加载流程是深入理解Spring IoC容器的一个基础问题。

Spring Bean的加载流程主要包含以下步骤：

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。

解决

【调用栈】postProcessBeforeInitialization 系统类一

initializeBean:1776, AbstractAutowireCapableBeanFactory (org.springframework.beans.factory.support)

doCreateBean:595, AbstractAutowireCapableBeanFactory (org.springframework.beans.factory.support)

createBean:517, AbstractAutowireCapableBeanFactory (org.springframework.beans.factory.support)

lambda$doGetBean$0:323, AbstractBeanFactory (org.springframework.beans.factory.support)

getObject:-1, 23197359 (org.springframework.beans.factory.support.AbstractBeanFactory$$Lambda$32)

getSingleton:226, DefaultSingletonBeanRegistry (org.springframework.beans.factory.support)

doGetBean:321, AbstractBeanFactory (org.springframework.beans.factory.support)

getBean:207, AbstractBeanFactory (org.springframework.beans.factory.support)

invokeBeanFactoryPostProcessors:90, PostProcessorRegistrationDelegate (org.springframework.context.support)

invokeBeanFactoryPostProcessors:706, AbstractApplicationContext (org.springframework.context.support)

refresh:532, AbstractApplicationContext (org.springframework.context.support)

<init>:89, AnnotationConfigApplicationContext (org.springframework.context.annotation)

main:9, App (org.bunny.spring.source)

;;

org.springframework.context.annotation.internalConfigurationAnnotationProcessor

org.springframework.context.event.internalEventListenerProcessor

org.springframework.context.event.internalEventListenerFactory

[]

【调用栈】postProcessBeforeInitialization 系统类二

initializeBean:1776, AbstractAutowireCapableBeanFactory (org.springframework.beans.factory.support)

doCreateBean:595, AbstractAutowireCapableBeanFactory (org.springframework.beans.factory.support)

createBean:517, AbstractAutowireCapableBeanFactory (org.springframework.beans.factory.support)

lambda$doGetBean$0:323, AbstractBeanFactory (org.springframework.beans.factory.support)

getObject:-1, 23197359 (org.springframework.beans.factory.support.AbstractBeanFactory$$Lambda$32)

getSingleton:226, DefaultSingletonBeanRegistry (org.springframework.beans.factory.support)

doGetBean:321, AbstractBeanFactory (org.springframework.beans.factory.support)

getBean:207, AbstractBeanFactory (org.springframework.beans.factory.support)

registerBeanPostProcessors:208, PostProcessorRegistrationDelegate (org.springframework.context.support)

registerBeanPostProcessors:722, AbstractApplicationContext (org.springframework.context.support)

refresh:535, AbstractApplicationContext (org.springframework.context.support)

<init>:89, AnnotationConfigApplicationContext (org.springframework.context.annotation)

main:9, App (org.bunny.spring.source)

;;

org.springframework.context.annotation.internalAutowiredAnnotationProcessor

org.springframework.context.annotation.internalCommonAnnotationProcessor

org.springframework.context.annotation.internalPersistenceAnnotationProcessor

[]

【调用栈】postProcessBeforeInitialization 用户类

initializeBean:1776, AbstractAutowireCapableBeanFactory (org.springframework.beans.factory.support)

doCreateBean:595, AbstractAutowireCapableBeanFactory (org.springframework.beans.factory.support)

createBean:517, AbstractAutowireCapableBeanFactory (org.springframework.beans.factory.support)

lambda$doGetBean$0:323, AbstractBeanFactory (org.springframework.beans.factory.support)

getObject:-1, 23197359 (org.springframework.beans.factory.support.AbstractBeanFactory$$Lambda$32)

getSingleton:226, DefaultSingletonBeanRegistry (org.springframework.beans.factory.support)

doGetBean:321, AbstractBeanFactory (org.springframework.beans.factory.support)

getBean:202, AbstractBeanFactory (org.springframework.beans.factory.support)

preInstantiateSingletons:895, DefaultListableBeanFactory (org.springframework.beans.factory.support)

finishBeanFactoryInitialization:878, AbstractApplicationContext (org.springframework.context.support)

refresh:550, AbstractApplicationContext (org.springframework.context.support)

<init>:89, AnnotationConfigApplicationContext (org.springframework.context.annotation)

main:9, App (org.bunny.spring.source)

;;

containerConfig

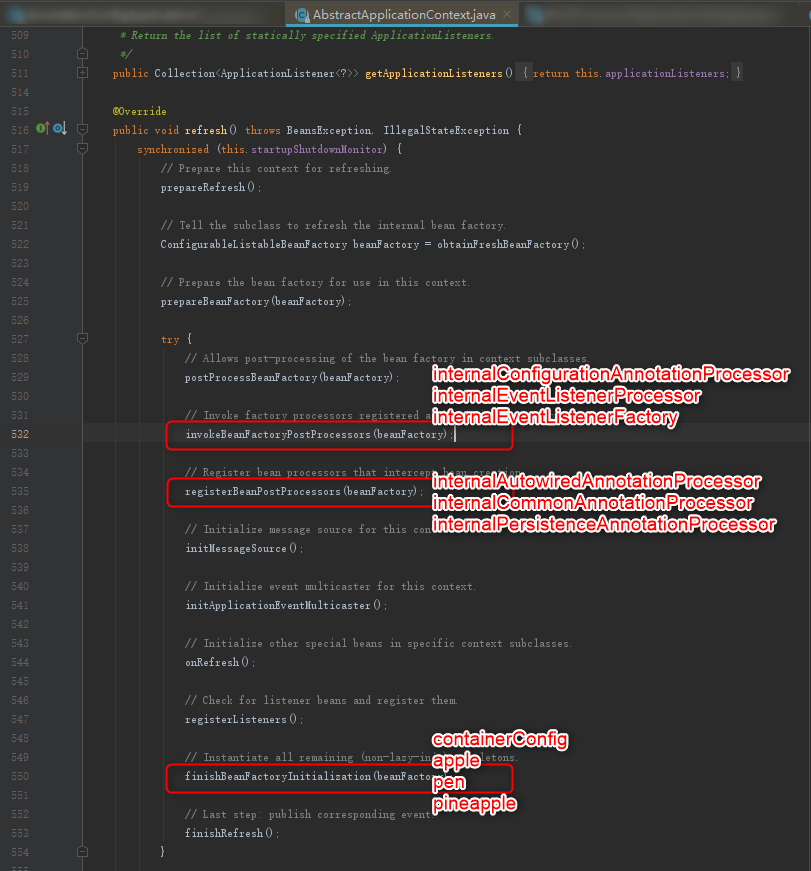
apple

pen

pineapple

[]

调用栈图解



Spring Bean加载的关键类：

org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory

关键方法：

protected Object initializeBean(final String beanName, final Object bean, @Nullable RootBeanDefinition mbd) {  
 if (System.*getSecurityManager*() != null) {  
 AccessController.*doPrivileged*((PrivilegedAction<Object>) () -> {  
 invokeAwareMethods(beanName, bean);  
 return null;  
 }, getAccessControlContext());  
 }  
 else {  
 invokeAwareMethods(beanName, bean);  
 }  
  
 Object wrappedBean = bean;  
 if (mbd == null || !mbd.isSynthetic()) {  
 wrappedBean = applyBeanPostProcessorsBeforeInitialization(wrappedBean, beanName);  
 }  
  
 try {  
 invokeInitMethods(beanName, wrappedBean, mbd);  
 }  
 catch (Throwable ex) {  
 throw new BeanCreationException(  
 (mbd != null ? mbd.getResourceDescription() : null),  
 beanName, "Invocation of init method failed", ex);  
 }  
 if (mbd == null || !mbd.isSynthetic()) {  
 wrappedBean = applyBeanPostProcessorsAfterInitialization(wrappedBean, beanName);  
 }  
  
 return wrappedBean;  
}

【问题描述】中Spring Bean的加载流程即是根据这个函数的执行流程总结出来的。

□