https://blog.csdn.net/lk1728183655/article/details/80565269

**Spring中的InstantiationAwareBeanPostProcessor和BeanPostProcessor**

2018年06月04日 14:19:05 [lk1728183655](https://me.csdn.net/lk1728183655) 阅读数：761

InstantiationAwareBeanPostProcessor接口继承了BeanPostProcessor。

BeanPostProcessor接口中有2个方法

1.Object postProcessBeforeInitialization(Object bean, String beanName) throws BeansException;

2.Object postProcessAfterInitialization(Object bean, String beanName) throws BeansException;

InstantiationAwareBeanPostProcessor中除了上述的两个方法外还有本身接口的方法

1.Object postProcessBeforeInstantiation(Class<?> beanClass, String beanName) throws BeansException;

2.boolean postProcessAfterInstantiation(Object bean, String beanName) throws BeansException;

3.PropertyValues postProcessPropertyValues(  
            PropertyValues pvs, PropertyDescriptor[] pds, Object bean, String beanName)

            throws BeansException;

从表面的意思理解InstantiationAwareBeanPostProcessor实例化前后的处理器，BeanPostProcessor中的方法是初始化类时候的处理器（实例化早于初始化）在spring中初始化指的一般是在调用init-method属性前后。

在AbstractAutowireCapableBeanFactory的createBean方法中

1. @Override
2. protected Object createBean(final String beanName, final RootBeanDefinition mbd, final Object[] args)
3. throws BeanCreationException {
5. if (logger.isDebugEnabled()) {
6. logger.debug("Creating instance of bean '" + beanName + "'");
7. }
8. *// Make sure bean class is actually resolved at this point.*
9. resolveBeanClass(mbd, beanName);
11. *// Prepare method overrides.*
12. try {
13. mbd.prepareMethodOverrides();
14. }
15. catch (BeanDefinitionValidationException ex) {
16. throw new BeanDefinitionStoreException(mbd.getResourceDescription(),
17. beanName, "Validation of method overrides failed", ex);
18. }
20. try {
21. *// Give BeanPostProcessors a chance to return a proxy instead of the target bean instance.*
22. Object bean = resolveBeforeInstantiation(beanName, mbd);
23. if (bean != null) {
24. return bean;
25. }
26. }
27. catch (Throwable ex) {
28. throw new BeanCreationException(mbd.getResourceDescription(), beanName,
29. "BeanPostProcessor before instantiation of bean failed", ex);
30. }
32. Object beanInstance = doCreateBean(beanName, mbd, args);
33. if (logger.isDebugEnabled()) {
34. logger.debug("Finished creating instance of bean '" + beanName + "'");
35. }
36. return beanInstance;
37. }

在resolveBeforeInstantiation方法中会运用实现了InstantiationAwareBeanPostProcessor接口的类实例化前调用postProcessBeforeInstantiation如果返回不为空就会执行所有实现了BeanPostProcessor类（包括实现了InstantiationAwareBeanPostProcessor的类）的postProcessAfterInitialization方法。

在doCreateBean方法中会调用populateBean(String beanName, AbstractBeanDefinition mbd, BeanWrapper bw)该方法主要是用来属性填充的。在开始时会调用实现InstantiationAwareBeanPostProcessor接口类的postProcessAfterInstantiation。返回值还能控制下面是否进行下面的属性填充。没有返回的话后面还会调用postProcessPropertyValues方法

在init-method方法生效前后调用postProcessBeforeInitialization和postProcessAfterInitialization方法

可以看出spring会尽量保证postProcessAfterInitialization能够执行。