源自：http://www.iteye.com/topic/1122859?page=2

| **锁定老帖子**[**主题：Spring开闭原则的表现-BeanPostProcessor扩展点-1**](http://www.iteye.com/topic/1122859)  精华帖 (8) :: 良好帖 (2) :: 新手帖 (0) :: 隐藏帖 (0) | | | |
| --- | --- | --- | --- |
| **作者** | | **正文** | |
| * **jinnianshilongnian** * 等级: 五钻会员 * [jinnianshilongnian的博客](http://jinnianshilongnian.iteye.com/) * 性别: http://www.iteye.com/images/icon_minigender_1.gif * 文章: 1112 * 积分: 2280 * http://www.iteye.com/images/status/offline.gif | | 发表时间：2012-04-18   最后修改：2012-04-20  [收藏](http://www.iteye.com/admin/link?user_favorite%5Btitle%5D=Spring%E5%BC%80%E9%97%AD%E5%8E%9F%E5%88%99%E7%9A%84%E8%A1%A8%E7%8E%B0-BeanPostProcessor%E6%89%A9%E5%B1%95%E7%82%B9-1&user_favorite%5Burl%5D=http%3A%2F%2Fwww.iteye.com%2Ftopic%2F1122859)  [<](http://www.iteye.com/topic/1122859) [>](http://www.iteye.com/topic/1122859) 猎头职位: [**上海:**Senior Software Engineer](http://www.iteye.com/jobs/2508)  相关文章:    * [Spring源码浅析 -- 依赖注入](http://www.iteye.com/topic/743998) * [【Spring】IOC核心源码学习（二）：容器初始化过程](http://www.iteye.com/topic/1115391) * [Spring源代码解析(一）：IOC容器](http://www.iteye.com/topic/86339)   推荐群组: [DI](http://weld.group.iteye.com/)  [更多相关推荐](http://www.iteye.com/wiki/topic/1122859)  [Spring](http://www.iteye.com/forums/tag/Spring)    上接[Spring事务处理时自我调用的解决方案及一些实现方式的风险](http://www.iteye.com/topic/1122740)继续分析，在分析上篇的问题之前，我们需要了解下BeanPostProcessor概念和Spring容器创建Bean的流程。    **一、BeanPostProcessor是什么**  接口定义  **Java代码  [收藏代码](javascript:void())**   1. **package** org.springframework.beans.factory.config; 2. **public** **interface** BeanPostProcessor { 3. Object postProcessBeforeInitialization(Object bean, String beanName) **throws** BeansException; 4. Object postProcessAfterInitialization(Object bean, String beanName) **throws** BeansException; 5. }   BeanPostProcessor是Spring容器的一个扩展点，可以进行自定义的实例化、初始化、依赖装配、依赖检查等流程，即可以覆盖默认的实例化，也可以增强初始化、依赖注入、依赖检查等流程，其javadoc有如下描述：      e.g. checking for marker interfaces or wrapping them with proxies.      大体意思是可以检查相应的标识接口完成一些自定义功能实现，如包装目标对象到代理对象。  我们可以看到BeanPostProcessor一共有两个回调方法postProcessBeforeInitialization和postProcessAfterInitialization，那这两个方法会在什么Spring执行流程中的哪个步骤执行呢？还有目前Spring提供哪些相应的实现呢？    Spring还提供了BeanPostProcessor一些其他接口实现，来完成除实例化外的其他功能，后续详细介绍。        **二、通过源代码看看创建一个Bean实例的具体执行流程：**  http://dl.iteye.com/upload/attachment/0066/8083/a7988c9e-c90f-32f6-8d17-ec5e206f48fb.jpg    AbstractApplicationContext内部使用DefaultListableBeanFactory，且DefaultListableBeanFactory继承AbstractAutowireCapableBeanFactory，因此我们此处分析AbstractAutowireCapableBeanFactory即可。    一、AbstractAutowireCapableBeanFactory的createBean方法代码如下：  **Java代码  [收藏代码](javascript:void())**   1. **protected** Object createBean(**final** String beanName, **final** RootBeanDefinition mbd, **final** Object[] args) **throws** BeanCreationException { 2. resolveBeanClass(mbd, beanName); /1解析Bean的**class** 3. mbd.prepareMethodOverrides(); //2 方法注入准备 4. Object bean = resolveBeforeInstantiation(beanName, mbd); //3 第一个BeanPostProcessor扩展点 5. **if** (bean != **null**) { //4 如果3处的扩展点返回的bean不为空，直接返回该bean，后续流程不需要执行 6. **return** bean; 7. } 8. Object beanInstance = doCreateBean(beanName, mbd, args); //5 执行spring的创建bean实例的流程啦 9. **return** beanInstance; 10. }    0.3 第一个BeanPostProcessor扩展点（只有InstantiationAwareBeanPostProcessor接口的实现才会被调用）    二、AbstractAutowireCapableBeanFactory的resolveBeforeInstantiation方法代码如下：  **Java代码  [收藏代码](javascript:void())**   1. **protected** Object resolveBeforeInstantiation(String beanName, RootBeanDefinition mbd) { 2. Object bean = **null**; 3. **if** (!Boolean.FALSE.equals(mbd.beforeInstantiationResolved)) { 4. // Make sure bean class is actually resolved at this point. 5. **if** (mbd.hasBeanClass() && !mbd.isSynthetic() && hasInstantiationAwareBeanPostProcessors()) { 6. //3.1、执行InstantiationAwareBeanPostProcessor的postProcessBeforeInstantiation回调方法 7. bean = applyBeanPostProcessorsBeforeInstantiation(mbd.getBeanClass(), beanName); 8. **if** (bean != **null**) { 9. //3.2、执行InstantiationAwareBeanPostProcessor的postProcessAfterInitialization回调方法 10. bean = applyBeanPostProcessorsAfterInitialization(bean, beanName); 11. } 12. } 13. mbd.beforeInstantiationResolved = (bean != **null**); 14. } 15. **return** bean; 16. }     通过如上代码可以进行实例化的预处理（自定义实例化bean，如创建相应的代理对象）和后处理（如进行自定义实例化的bean的依赖装配）。    三、AbstractAutowireCapableBeanFactory的doCreateBean方法代码如下：    **Java代码  [收藏代码](javascript:void())**   1. // 6、通过BeanWrapper实例化Bean 2. BeanWrapper instanceWrapper = **null**; 3. **if** (mbd.isSingleton()) { 4. instanceWrapper = **this**.factoryBeanInstanceCache.remove(beanName); 5. } 6. **if** (instanceWrapper == **null**) { 7. instanceWrapper = createBeanInstance(beanName, mbd, args); 8. } 9. **final** Object bean = (instanceWrapper != **null** ? instanceWrapper.getWrappedInstance() : **null**); 10. Class beanType = (instanceWrapper != **null** ? instanceWrapper.getWrappedClass() : **null**); 12. //7、执行MergedBeanDefinitionPostProcessor的postProcessMergedBeanDefinition流程 13. **synchronized** (mbd.postProcessingLock) { 14. **if** (!mbd.postProcessed) { 15. applyMergedBeanDefinitionPostProcessors(mbd, beanType, beanName); 16. mbd.postProcessed = **true**; 17. } 18. } 19. // 8、及早暴露单例Bean引用，从而允许setter注入方式的循环引用 20. **boolean** earlySingletonExposure = (mbd.isSingleton() && **this**.allowCircularReferences && 21. isSingletonCurrentlyInCreation(beanName)); 22. **if** (earlySingletonExposure) { 23. //省略log 24. addSingletonFactory(beanName, **new** ObjectFactory() { 25. **public** Object getObject() **throws** BeansException { 26. //8.1、调用SmartInstantiationAwareBeanPostProcessor的getEarlyBeanReference返回一个需要暴露的Bean（例如包装目标对象到代理对象） 27. **return** getEarlyBeanReference(beanName, mbd, bean); 28. } 29. }); 30. } 32. Object exposedObject = bean; 33. **try** { 34. populateBean(beanName, mbd, instanceWrapper); //9、组装-Bean依赖 35. **if** (exposedObject != **null**) { 36. exposedObject = initializeBean(beanName, exposedObject, mbd); //10、初始化Bean 37. } 38. } 39. **catch** (Throwable ex) { 40. //省略异常 41. }  44. //11如果是及早暴露单例bean，通过getSingleton触发3.1处的getEarlyBeanReference调用获取要及早暴露的单例Bean 45. **if** (earlySingletonExposure) { 46. Object earlySingletonReference = getSingleton(beanName, **false**); 47. **if** (earlySingletonReference != **null**) { 48. **if** (exposedObject == bean) { 49. exposedObject = earlySingletonReference; 50. } 51. **else** **if** (!**this**.allowRawInjectionDespiteWrapping && hasDependentBean(beanName)) { 52. String[] dependentBeans = getDependentBeans(beanName); 53. Set<String> actualDependentBeans = **new** LinkedHashSet<String>(dependentBeans.length); 54. **for** (String dependentBean : dependentBeans) { 55. **if** (!removeSingletonIfCreatedForTypeCheckOnly(dependentBean)) { 56. actualDependentBeans.add(dependentBean); 57. } 58. } 59. **if** (!actualDependentBeans.isEmpty()) { 60. **throw** **new** BeanCurrentlyInCreationException(beanName, 61. "Bean with name '" + beanName + "' has been injected into other beans [" + 62. StringUtils.collectionToCommaDelimitedString(actualDependentBeans) + 63. "] in its raw version as part of a circular reference, but has eventually been " + 64. "wrapped. This means that said other beans do not use the final version of the " + 65. "bean. This is often the result of over-eager type matching - consider using " + 66. "'getBeanNamesOfType' with the 'allowEagerInit' flag turned off, for example."); 67. } 68. } 69. } 70. } 71. //12、注册Bean的销毁回调 72. **try** { 73. registerDisposableBeanIfNecessary(beanName, bean, mbd); 74. } 75. **catch** (BeanDefinitionValidationException ex) { 76. **throw** **new** BeanCreationException(mbd.getResourceDescription(), beanName, "Invalid destruction signature", ex); 77. } 79. **return** exposedObject; 80. }       四、AbstractAutowireCapableBeanFactory的populateBean方法代码如下：  **Java代码  [收藏代码](javascript:void())**   1. //9、组装-Bean 2. **protected** **void** populateBean(String beanName, AbstractBeanDefinition mbd, BeanWrapper bw) { 3. PropertyValues pvs = mbd.getPropertyValues(); 4. //省略部分代码 5. //9.1、通过InstantiationAwareBeanPostProcessor扩展点允许自定义装配流程（如@Autowired支持等） 6. //执行InstantiationAwareBeanPostProcessor的postProcessAfterInstantiation 7. **boolean** continueWithPropertyPopulation = **true**; 8. **if** (!mbd.isSynthetic() && hasInstantiationAwareBeanPostProcessors()) { 9. **for** (BeanPostProcessor bp : getBeanPostProcessors()) { 10. **if** (bp **instanceof** InstantiationAwareBeanPostProcessor) { 11. InstantiationAwareBeanPostProcessor ibp = (InstantiationAwareBeanPostProcessor) bp; 12. **if** (!ibp.postProcessAfterInstantiation(bw.getWrappedInstance(), beanName)) { 13. continueWithPropertyPopulation = **false**; 14. **break**; 15. } 16. } 17. } 18. } 19. **if** (!continueWithPropertyPopulation) { 20. **return**; 21. } 22. **if** (mbd.getResolvedAutowireMode() == RootBeanDefinition.AUTOWIRE\_BY\_NAME || 23. mbd.getResolvedAutowireMode() == RootBeanDefinition.AUTOWIRE\_BY\_TYPE) { 24. MutablePropertyValues newPvs = **new** MutablePropertyValues(pvs); 25. // 9. 2、自动装配（根据name/type） 26. **if** (mbd.getResolvedAutowireMode() == RootBeanDefinition.AUTOWIRE\_BY\_NAME) { 27. autowireByName(beanName, mbd, bw, newPvs); 28. } 29. **if** (mbd.getResolvedAutowireMode() == RootBeanDefinition.AUTOWIRE\_BY\_TYPE) { 30. autowireByType(beanName, mbd, bw, newPvs); 31. } 32. pvs = newPvs; 33. } 34. **boolean** hasInstAwareBpps = hasInstantiationAwareBeanPostProcessors(); 35. **boolean** needsDepCheck = (mbd.getDependencyCheck() != RootBeanDefinition.DEPENDENCY\_CHECK\_NONE);  38. //9. 3、执行InstantiationAwareBeanPostProcessor的postProcessPropertyValues 39. **if** (hasInstAwareBpps || needsDepCheck) { 40. PropertyDescriptor[] filteredPds = filterPropertyDescriptorsForDependencyCheck(bw); 41. **if** (hasInstAwareBpps) { 42. **for** (BeanPostProcessor bp : getBeanPostProcessors()) { 43. **if** (bp **instanceof** InstantiationAwareBeanPostProcessor) { 44. InstantiationAwareBeanPostProcessor ibp = (InstantiationAwareBeanPostProcessor) bp; 45. pvs = ibp.postProcessPropertyValues(pvs, filteredPds, bw.getWrappedInstance(), beanName); 46. **if** (pvs == **null**) { 47. **return**; 48. } 49. } 50. } 51. } 52. //9. 4、执行依赖检查 53. **if** (needsDepCheck) { 54. checkDependencies(beanName, mbd, filteredPds, pvs); 55. } 56. } 57. //9. 5、应用依赖注入 58. applyPropertyValues(beanName, mbd, bw, pvs); 59. }     五、AbstractAutowireCapableBeanFactory的initializeBean方法代码如下：  **Java代码  [收藏代码](javascript:void())**   1. //10、实例化Bean 2. **protected** Object initializeBean(**final** String beanName, **final** Object bean, RootBeanDefinition mbd) { 3. //10.1、调用Aware接口注入（BeanNameAware、BeanClassLoaderAware、BeanFactoryAware） 4. invokeAwareMethods(beanName, bean);//此处省略部分代码 5. //10.2、执行BeanPostProcessor扩展点的postProcessBeforeInitialization进行修改实例化Bean 6. Object wrappedBean = bean; 7. **if** (mbd == **null** || !mbd.isSynthetic()) { 8. wrappedBean = applyBeanPostProcessorsBeforeInitialization(wrappedBean, beanName); 9. } 10. //10.3、执行初始化回调（1、调用InitializingBean的afterPropertiesSet  2、调用自定义的init-method） 11. **try** { 12. invokeInitMethods(beanName, wrappedBean, mbd); 13. } 14. **catch** (Throwable ex) { 15. //异常省略 16. } 17. //10.4、执行BeanPostProcessor扩展点的postProcessAfterInitialization进行修改实例化Bean 18. **if** (mbd == **null** || !mbd.isSynthetic()) { 19. wrappedBean = applyBeanPostProcessorsAfterInitialization(wrappedBean, beanName); 20. } 21. **return** wrappedBean;   **三、创建一个Bean实例的执行流程简化：**  protected Object createBean(final String beanName, final RootBeanDefinition mbd, final Object[] args); 创建Bean  **（1、resolveBeanClass(mbd, beanName);**解析Bean class，若class配置错误将抛出CannotLoadBeanClassException；    **（2、mbd.prepareMethodOverrides();**准备和验证配置的方法注入，若验证失败抛出BeanDefinitionValidationException  有关方法注入知识请参考[【第三章】 DI 之 3.3 更多DI的知识 ——跟我学spring3](http://jinnianshilongnian.iteye.com/blog/1415461) 3.3.5 方法注入；    **（3、Object bean = resolveBeforeInstantiation(beanName, mbd);**第一个BeanPostProcessor扩展点，此处只执行InstantiationAwareBeanPostProcessor类型的BeanPostProcessor Bean；  （3.1、bean = applyBeanPostProcessorsBeforeInstantiation(mbd.getBeanClass(), beanName);执行InstantiationAwareBeanPostProcessor的实例化的预处理回调方法postProcessBeforeInstantiation（自定义的实例化，如创建代理）；  （3.2、bean = applyBeanPostProcessorsAfterInitialization(bean, beanName);执行InstantiationAwareBeanPostProcessor的实例化的后处理回调方法postProcessAfterInitialization（如依赖注入），如果3.1处返回的Bean不为null才执行；    **（4、如果3处的扩展点返回的bean不为空，直接返回该bean，后续流程不需要执行；**    **（5、Object beanInstance = doCreateBean(beanName, mbd, args);**执行spring的创建bean实例的流程；      **（6、createBeanInstance(beanName, mbd, args);**实例化Bean  （6.1、instantiateUsingFactoryMethod 工厂方法实例化；请参考【<http://jinnianshilongnian.iteye.com/blog/1413857>】  （6.2、构造器实例化，请参考【<http://jinnianshilongnian.iteye.com/blog/1413857>】；  （6.2.1、如果之前已经解析过构造器  （6.2.1.1 autowireConstructor：有参调用autowireConstructor实例化  （6.2.1.2、instantiateBean：无参调用instantiateBean实例化；  （6.2.2、如果之前没有解析过构造器：    （6.2.2.1、通过SmartInstantiationAwareBeanPostProcessor的determineCandidateConstructors回调方法解析构造器，第二个BeanPostProcessor扩展点，返回第一个解析成功（返回值不为null）的构造器组，如AutowiredAnnotationBeanPostProcessor实现将自动扫描通过@Autowired/@Value注解的构造器从而可以完成构造器注入，请参考[【第十二章】零配置 之 12.2 注解实现Bean依赖注入 ——跟我学spring3](http://jinnianshilongnian.iteye.com/blog/1457224) ；  （6.2.2.2、autowireConstructor：如果（6.2.2.1返回的不为null，且是有参构造器，调用autowireConstructor实例化；  （6.2.2.3、instantiateBean： 否则调用无参构造器实例化；    **（7、applyMergedBeanDefinitionPostProcessors(mbd, beanType, beanName);**第三个BeanPostProcessor扩展点，执行Bean定义的合并；  （7.1、执行MergedBeanDefinitionPostProcessor的postProcessMergedBeanDefinition回调方法，进行bean定义的合并；    **（8、addSingletonFactory(beanName, new ObjectFactory() {**  **public Object getObject() throws BeansException {**  **return getEarlyBeanReference(beanName, mbd, bean);**  **}**  **});** 及早暴露单例Bean引用，从而允许setter注入方式的循环引用  （8.1、SmartInstantiationAwareBeanPostProcessor的getEarlyBeanReference；第四个BeanPostProcessor扩展点，当存在循环依赖时，通过该回调方法获取及早暴露的Bean实例；    **（9、populateBean(beanName, mbd, instanceWrapper);**装配Bean依赖  （9.1、InstantiationAwareBeanPostProcessor的postProcessAfterInstantiation；第五个BeanPostProcessor扩展点，在实例化Bean之后，所有其他装配逻辑之前执行，如果false将阻止其他的InstantiationAwareBeanPostProcessor的postProcessAfterInstantiation的执行和从（9.2到（9.5的执行，通常返回true；  （9.2、autowireByName、autowireByType：根据名字和类型进行自动装配，自动装配的知识请参考[【第三章】 DI 之 3.3 更多DI的知识 ——跟我学spring3](http://jinnianshilongnian.iteye.com/blog/1415461)  3.3.3  自动装配；  （9.3、InstantiationAwareBeanPostProcessor的postProcessPropertyValues：第六个BeanPostProcessor扩展点，完成其他定制的一些依赖注入，如AutowiredAnnotationBeanPostProcessor执行@Autowired注解注入，CommonAnnotationBeanPostProcessor执行@Resource等注解的注入，PersistenceAnnotationBeanPostProcessor执行@ PersistenceContext等JPA注解的注入，RequiredAnnotationBeanPostProcessor执行@ Required注解的检查等等，请参考[【第十二章】零配置 之 12.2 注解实现Bean依赖注入 ——跟我学spring3](http://jinnianshilongnian.iteye.com/blog/1457224)；  （9.4、checkDependencies：依赖检查，请参考[【第三章】 DI 之 3.3 更多DI的知识 ——跟我学spring3](http://jinnianshilongnian.iteye.com/blog/1415461)  3.3.4  依赖检查；  （9.5、applyPropertyValues：应用明确的setter属性注入，请参考[【第三章】 DI 之 3.1 DI的配置使用 ——跟我学spring3](http://jinnianshilongnian.iteye.com/blog/1415277)；    **（10、exposedObject = initializeBean(beanName, exposedObject, mbd);**执行初始化Bean流程；  （10.1、invokeAwareMethods（BeanNameAware、BeanClassLoaderAware、BeanFactoryAware）：调用一些Aware标识接口注入如BeanName、BeanFactory；  （10.2、BeanPostProcessor的postProcessBeforeInitialization：第七个扩展点，在调用初始化之前完成一些定制的初始化任务，如BeanValidationPostProcessor完成JSR-303 @Valid注解Bean验证，InitDestroyAnnotationBeanPostProcessor完成@PostConstruct注解的初始化方法调用，ApplicationContextAwareProcessor完成一些Aware接口的注入（如EnvironmentAware、ResourceLoaderAware、ApplicationContextAware），其返回值将替代原始的Bean对象；  （10.3、invokeInitMethods ： 调用初始化方法；  （10.3.1、InitializingBean的afterPropertiesSet ：调用InitializingBean的afterPropertiesSet回调方法；  （10.3.2、通过xml指定的自定义init-method ：调用通过xml配置的自定义init-method  （10.3.3、BeanPostProcessor的postProcessAfterInitialization ：第八个扩展点，AspectJAwareAdvisorAutoProxyCreator（完成xml风格的AOP配置(<aop:config>)的目标对象包装到AOP代理对象）、AnnotationAwareAspectJAutoProxyCreator（完成@Aspectj注解风格（<aop:aspectj-autoproxy> @Aspect）的AOP配置的目标对象包装到AOP代理对象），其返回值将替代原始的Bean对象；    （11、if (earlySingletonExposure) {                       Object earlySingletonReference = getSingleton(beanName, false);              ……        } ：如果是earlySingleExposure，调用getSingle方法获取Bean实例；  earlySingleExposure =(mbd.isSingleton() && this.allowCircularReferences && isSingletonCurrentlyInCreation(beanName))  只要单例Bean且允许循环引用（默认true）且当前单例Bean正在创建中  （11.1、如果是earlySingletonExposure调用getSingleton将触发【8】处ObjectFactory.getObject()的调用，通过【8.1】处的getEarlyBeanReference获取相关Bean（如包装目标对象的代理Bean）；（在循环引用Bean时可能引起[Spring事务处理时自我调用的解决方案及一些实现方式的风险](http://www.iteye.com/topic/1122740)）；    **（12、registerDisposableBeanIfNecessary(beanName, bean, mbd) ：** 注册Bean的销毁方法（只有非原型Bean可注册）；  （12.1、单例Bean的销毁流程  （12.1.1、DestructionAwareBeanPostProcessor的postProcessBeforeDestruction ： 第九个扩展点，如InitDestroyAnnotationBeanPostProcessor完成@PreDestroy注解的销毁方法注册和调用；  （12.1.2、DisposableBean的destroy：注册/调用DisposableBean的destroy销毁方法；  （12.1.3、通过xml指定的自定义destroy-method ： 注册/调用通过XML指定的destroy-method销毁方法；  （12.1.2、Scope的registerDestructionCallback：注册自定义的Scope的销毁回调方法，如RequestScope、SessionScope等；其流程和【12.1 单例Bean的销毁流程一样】，关于自定义Scope请参考[【第三章】 DI 之 3.4 Bean的作用域 ——跟我学spring3](http://jinnianshilongnian.iteye.com/blog/1415463)    **（13、到此Bean实例化、依赖注入、初始化完毕可以返回创建好的bean了。**    从上面的流程我们可以看到BeanPostProcessor一个使用了九个扩展点，其实还一个扩展点（SmartInstantiationAwareBeanPostProcessor的predictBeanType在下一篇介绍），接下来我们看看BeanPostProcessor这些扩展点都主要完成什么功能及常见的BeanPostProcessor。    我将在下一帖子中使用例子来解析这八个扩展点的主要功能，及一些Spring默认提供的BeanPostProcessor主要作用。    欢迎大家一起探讨学习。    下一篇地址：[**Spring开闭原则的表现-BeanPostProcessor扩展点-2**](http://www.iteye.com/topic/1122937)   * [查看图片附件](http://www.iteye.com/topic/1122859)   声明：ITeye文章版权属于作者，受法律保护。没有作者书面许可不得转载。  推荐链接 | |
| [返回顶楼](http://www.iteye.com/topic/1122859) | |  | |
|  | | | |
| * **xiangdefei** * 等级: 初级会员 * [xiangdefei的博客](http://xiangdefei.iteye.com/) * 性别: http://www.iteye.com/images/icon_minigender_1.gif * 文章: 74 * 积分: 0 * 来自: 南京 * http://www.iteye.com/images/status/offline.gif | | 发表时间：2012-04-19  [收藏](http://www.iteye.com/admin/link?user_favorite%5Btitle%5D=xiangdefei%E5%9B%9E%E5%B8%96%3ASpring%E5%BC%80%E9%97%AD%E5%8E%9F%E5%88%99%E7%9A%84%E8%A1%A8%E7%8E%B0-BeanPostProcessor%E6%89%A9%E5%B1%95%E7%82%B9-1&user_favorite%5Burl%5D=http%3A%2F%2Fwww.iteye.com%2Ftopic%2F1122859%232342503)  beanPostProcesso为对象实例化前后提供了可扩展的空间，spring里面最典型的beanPostProcessor实现就是属性文件占位（PropertyPlaceholderConfigurer）。 | |
| [返回顶楼](http://www.iteye.com/topic/1122859) | | [回帖地址](http://www.iteye.com/topic/1122859#2342503)  [0](http://www.iteye.com/topic/1122859) [0](http://www.iteye.com/topic/1122859) 已投票 | |
|  | | | |
| * **jinnianshilongnian** * 等级: 五钻会员 * [jinnianshilongnian的博客](http://jinnianshilongnian.iteye.com/) * 性别: http://www.iteye.com/images/icon_minigender_1.gif * 文章: 1112 * 积分: 2280 * http://www.iteye.com/images/status/offline.gif | | 发表时间：2012-04-19  [收藏](http://www.iteye.com/admin/link?user_favorite%5Btitle%5D=jinnianshilongnian%E5%9B%9E%E5%B8%96%3ASpring%E5%BC%80%E9%97%AD%E5%8E%9F%E5%88%99%E7%9A%84%E8%A1%A8%E7%8E%B0-BeanPostProcessor%E6%89%A9%E5%B1%95%E7%82%B9-1&user_favorite%5Burl%5D=http%3A%2F%2Fwww.iteye.com%2Ftopic%2F1122859%232342505)  **xiangdefei 写道**  beanPostProcesso为对象实例化前后提供了可扩展的空间，spring里面最典型的beanPostProcessor实现就是属性文件占位（PropertyPlaceholderConfigurer）。  下一篇帖子详细介绍Spring内置的BeanPostProcessor 及其功能  如  AnnotationAwareAspectJAutoProxyCreator   AspectJAwareAdvisorAutoProxyCreator   InitDestroyAnnotationBeanPostProcessor  ApplicationContextAwareProcessor  AutowiredAnnotationBeanPostProcessor  CommonAnnotationBeanPostProcessor  RequiredAnnotationBeanPostProcessor  PersistenceAnnotationBeanPostProcessor  等等………… | |
| [返回顶楼](http://www.iteye.com/topic/1122859) | | [回帖地址](http://www.iteye.com/topic/1122859#2342505)  [0](http://www.iteye.com/topic/1122859) [0](http://www.iteye.com/topic/1122859) 已投票 | |
|  | | | |
| * **xiangdefei** * 等级: 初级会员 * [xiangdefei的博客](http://xiangdefei.iteye.com/) * 性别: http://www.iteye.com/images/icon_minigender_1.gif * 文章: 74 * 积分: 0 * 来自: 南京 * http://www.iteye.com/images/status/offline.gif | | 发表时间：2012-04-19  [收藏](http://www.iteye.com/admin/link?user_favorite%5Btitle%5D=xiangdefei%E5%9B%9E%E5%B8%96%3ASpring%E5%BC%80%E9%97%AD%E5%8E%9F%E5%88%99%E7%9A%84%E8%A1%A8%E7%8E%B0-BeanPostProcessor%E6%89%A9%E5%B1%95%E7%82%B9-1&user_favorite%5Burl%5D=http%3A%2F%2Fwww.iteye.com%2Ftopic%2F1122859%232342506)  LZ的好学的精神真值得学习，我工作了三年后都静不下来心去研究这些东西了。 | |
| [返回顶楼](http://www.iteye.com/topic/1122859) | | [回帖地址](http://www.iteye.com/topic/1122859#2342506)  [0](http://www.iteye.com/topic/1122859) [0](http://www.iteye.com/topic/1122859) 已投票 | |
|  | | | |
| * **jinnianshilongnian** * 等级: 五钻会员 * [jinnianshilongnian的博客](http://jinnianshilongnian.iteye.com/) * 性别: http://www.iteye.com/images/icon_minigender_1.gif * 文章: 1112 * 积分: 2280 * http://www.iteye.com/images/status/offline.gif | | 发表时间：2012-04-19  [收藏](http://www.iteye.com/admin/link?user_favorite%5Btitle%5D=jinnianshilongnian%E5%9B%9E%E5%B8%96%3ASpring%E5%BC%80%E9%97%AD%E5%8E%9F%E5%88%99%E7%9A%84%E8%A1%A8%E7%8E%B0-BeanPostProcessor%E6%89%A9%E5%B1%95%E7%82%B9-1&user_favorite%5Burl%5D=http%3A%2F%2Fwww.iteye.com%2Ftopic%2F1122859%232342509)  **xiangdefei 写道**  LZ的好学的精神真值得学习，我工作了三年后都静不下来心去研究这些东西了。  呵呵，营造个好的学习氛围，一起学习探讨 才有动力 | |
| [返回顶楼](http://www.iteye.com/topic/1122859) | | [回帖地址](http://www.iteye.com/topic/1122859#2342509)  [0](http://www.iteye.com/topic/1122859) [0](http://www.iteye.com/topic/1122859) 已投票 | |
|  | | | |
| * **kjj** * 等级: 二星会员 * [kjj的博客](http://docs.iteye.com/) * 性别: http://www.iteye.com/images/icon_minigender_1.gif * 文章: 1017 * 积分: 266 * 来自: 陕西 * http://www.iteye.com/images/status/offline.gif | | 发表时间：2012-04-19  [收藏](http://www.iteye.com/admin/link?user_favorite%5Btitle%5D=kjj%E5%9B%9E%E5%B8%96%3ASpring%E5%BC%80%E9%97%AD%E5%8E%9F%E5%88%99%E7%9A%84%E8%A1%A8%E7%8E%B0-BeanPostProcessor%E6%89%A9%E5%B1%95%E7%82%B9-1&user_favorite%5Burl%5D=http%3A%2F%2Fwww.iteye.com%2Ftopic%2F1122859%232342664)  aop思想的延伸,参与bean创建事件,给用户一个修改bean的机会!! | |
| [返回顶楼](http://www.iteye.com/topic/1122859) | | [回帖地址](http://www.iteye.com/topic/1122859#2342664)  [0](http://www.iteye.com/topic/1122859) [0](http://www.iteye.com/topic/1122859) 已投票 | |
|  | | | |
| * **jinnianshilongnian** * 等级: 五钻会员 * [jinnianshilongnian的博客](http://jinnianshilongnian.iteye.com/) * 性别: http://www.iteye.com/images/icon_minigender_1.gif * 文章: 1112 * 积分: 2280 * http://www.iteye.com/images/status/offline.gif | | 发表时间：2012-04-19  [收藏](http://www.iteye.com/admin/link?user_favorite%5Btitle%5D=jinnianshilongnian%E5%9B%9E%E5%B8%96%3ASpring%E5%BC%80%E9%97%AD%E5%8E%9F%E5%88%99%E7%9A%84%E8%A1%A8%E7%8E%B0-BeanPostProcessor%E6%89%A9%E5%B1%95%E7%82%B9-1&user_favorite%5Burl%5D=http%3A%2F%2Fwww.iteye.com%2Ftopic%2F1122859%232342675)  **kjj 写道**  aop思想的延伸,参与bean创建事件,给用户一个修改bean的机会!!  应该是Spring一直遵循的开闭原则，对扩展开放，对修改关闭。就如SpringMVC一样。提供很多扩展点 | |
| [返回顶楼](http://www.iteye.com/topic/1122859) | | [回帖地址](http://www.iteye.com/topic/1122859#2342675)  [3](http://www.iteye.com/topic/1122859) [0](http://www.iteye.com/topic/1122859) 已投票 | |
|  | | | |
| * **jinnianshilongnian** * 等级: 五钻会员 * [jinnianshilongnian的博客](http://jinnianshilongnian.iteye.com/) * 性别: http://www.iteye.com/images/icon_minigender_1.gif * 文章: 1112 * 积分: 2280 * http://www.iteye.com/images/status/offline.gif | | 发表时间：2012-04-19  [收藏](http://www.iteye.com/admin/link?user_favorite%5Btitle%5D=jinnianshilongnian%E5%9B%9E%E5%B8%96%3ASpring%E5%BC%80%E9%97%AD%E5%8E%9F%E5%88%99%E7%9A%84%E8%A1%A8%E7%8E%B0-BeanPostProcessor%E6%89%A9%E5%B1%95%E7%82%B9-1&user_favorite%5Burl%5D=http%3A%2F%2Fwww.iteye.com%2Ftopic%2F1122859%232342863)  **xiangdefei 写道**  beanPostProcesso为对象实例化前后提供了可扩展的空间，spring里面最典型的beanPostProcessor实现就是属性文件占位（PropertyPlaceholderConfigurer）。  PropertyPlaceholderConfigurer是 BeanFactoryPostProcessor 实现，是Bean工厂的后处理 | |
| [返回顶楼](http://www.iteye.com/topic/1122859) | | [回帖地址](http://www.iteye.com/topic/1122859#2342863)  [0](http://www.iteye.com/topic/1122859) [0](http://www.iteye.com/topic/1122859) 已投票 | |
|  | | | |
| * **kjj** * 等级: 二星会员 * [kjj的博客](http://docs.iteye.com/) * 性别: http://www.iteye.com/images/icon_minigender_1.gif * 文章: 1017 * 积分: 266 * 来自: 陕西 * http://www.iteye.com/images/status/offline.gif | | 发表时间：2012-04-20  [收藏](http://www.iteye.com/admin/link?user_favorite%5Btitle%5D=kjj%E5%9B%9E%E5%B8%96%3ASpring%E5%BC%80%E9%97%AD%E5%8E%9F%E5%88%99%E7%9A%84%E8%A1%A8%E7%8E%B0-BeanPostProcessor%E6%89%A9%E5%B1%95%E7%82%B9-1&user_favorite%5Burl%5D=http%3A%2F%2Fwww.iteye.com%2Ftopic%2F1122859%232343109)  **jinnianshilongnian 写道**  **kjj 写道**  aop思想的延伸,参与bean创建事件,给用户一个修改bean的机会!!  应该是Spring一直遵循的开闭原则，对扩展开放，对修改关闭。就如SpringMVC一样。提供很多扩展点  你可以得到bean,对修改关闭,怎么理解,怎么个关闭法!!?? | |
| [返回顶楼](http://www.iteye.com/topic/1122859) | | [回帖地址](http://www.iteye.com/topic/1122859#2343109)  [0](http://www.iteye.com/topic/1122859) [0](http://www.iteye.com/topic/1122859) 已投票 | |
|  | | | |
| * **jinnianshilongnian** * 等级: 五钻会员 * [jinnianshilongnian的博客](http://jinnianshilongnian.iteye.com/) * 性别: http://www.iteye.com/images/icon_minigender_1.gif * 文章: 1112 * 积分: 2280 * http://www.iteye.com/images/status/offline.gif | | 发表时间：2012-04-20  [收藏](http://www.iteye.com/admin/link?user_favorite%5Btitle%5D=jinnianshilongnian%E5%9B%9E%E5%B8%96%3ASpring%E5%BC%80%E9%97%AD%E5%8E%9F%E5%88%99%E7%9A%84%E8%A1%A8%E7%8E%B0-BeanPostProcessor%E6%89%A9%E5%B1%95%E7%82%B9-1&user_favorite%5Burl%5D=http%3A%2F%2Fwww.iteye.com%2Ftopic%2F1122859%232343114)  **kjj 写道**  **jinnianshilongnian 写道**  **kjj 写道**  aop思想的延伸,参与bean创建事件,给用户一个修改bean的机会!!  应该是Spring一直遵循的开闭原则，对扩展开放，对修改关闭。就如SpringMVC一样。提供很多扩展点  你可以得到bean,对修改关闭,怎么理解,怎么个关闭法!!??  整体实例化、初始化、依赖装配是 流程的关闭  通过BeanPostProcessor进行扩展（如包装目标对象到AOP对象） | |
| [返回顶楼](http://www.iteye.com/topic/1122859) | | [回帖地址](http://www.iteye.com/topic/1122859#2343114)  [0](http://www.iteye.com/topic/1122859) [0](http://www.iteye.com/topic/1122859) 已投票 | |
| **锁定老帖子**[**主题：Spring开闭原则的表现-BeanPostProcessor扩展点-1**](http://www.iteye.com/topic/1122859)  精华帖 (8) :: 良好帖 (2) :: 新手帖 (0) :: 隐藏帖 (0) | | | |
| **作者** | | **正文** | |
| * **jinnianshilongnian** * 等级: 五钻会员 * [jinnianshilongnian的博客](http://jinnianshilongnian.iteye.com/) * 性别: http://www.iteye.com/images/icon_minigender_1.gif * 文章: 1112 * 积分: 2280 * http://www.iteye.com/images/status/offline.gif | | 发表时间：2012-04-20  [收藏](http://www.iteye.com/admin/link?user_favorite%5Btitle%5D=jinnianshilongnian%E5%9B%9E%E5%B8%96%3ASpring%E5%BC%80%E9%97%AD%E5%8E%9F%E5%88%99%E7%9A%84%E8%A1%A8%E7%8E%B0-BeanPostProcessor%E6%89%A9%E5%B1%95%E7%82%B9-1&user_favorite%5Burl%5D=http%3A%2F%2Fwww.iteye.com%2Ftopic%2F1122859%3Fpage%3D2%232343115)  **kjj 写道**  **jinnianshilongnian 写道**  **kjj 写道**  aop思想的延伸,参与bean创建事件,给用户一个修改bean的机会!!  应该是Spring一直遵循的开闭原则，对扩展开放，对修改关闭。就如SpringMVC一样。提供很多扩展点  你可以得到bean,对修改关闭,怎么理解,怎么个关闭法!!??  7、BeanValidationPostProcessor  默认不自动注册，Spring3.0开始支持。   提供对JSR-303验证规范支持。   根据afterInitialization是false/true决定调用postProcessBeforeInitialization或postProcessAfterInitialization来通过JSR-303规范验证Bean，默认false。  8、MethodValidationPostProcessor  Spring3.1开始支持，且只支持Hibernate Validator 4.2及更高版本，从Spring 3.2起可能将采取自动检测Bean Validation 1.1兼容的提供商且自动注册（Bean Validation 1.1 (JSR-349)正处于草案阶段，它将提供方法级别的验证，提供对方法级别的验证），目前默认不自动注册。   Bean Validation 1.1草案请参考http://jcp.org/en/jsr/detail?id=349    http://beanvalidation.org/。   提供对方法参数/方法返回值的进行验证（即前置条件/后置条件的支持），通过JSR-303注解验证，使用方式如：  public @NotNull Object myValidMethod(@NotNull String arg1, @Max(10) int arg2)   默认只对@org.springframework.validation.annotation.Validated注解的Bean进行验证，我们可以修改validatedAnnotationType为其他注解类型来支持其他注解验证。而且目前只支持Hibernate Validator实现，在未来版本可能支持其他实现。   有了这东西之后我们就不需要在进行如Assert.assertNotNull（）这种前置条件/后置条件的判断了。  9、ScheduledAnnotationBeanPostProcessor  当配置文件中有<task:annotation-driven>自动注册或@EnableScheduling自动注册。   提供对注解@Scheduled任务调度的支持。   postProcessAfterInitialization：通过查找Bean对象类上的@Scheduled注解来注册任务调度方法（仅返回值为void且方法是无形式参数的才可以）。   可参考Spring官方文档的任务调度章节学习@Scheduled注解任务调度。    如在写的其他BeanPostProccessor。 可以看出可以扩展N多功能 欢迎讨论 | |
| [返回顶楼](http://www.iteye.com/topic/1122859?page=2) | | [回帖地址](http://www.iteye.com/topic/1122859?page=2#2343115)  [0](http://www.iteye.com/topic/1122859?page=2) [0](http://www.iteye.com/topic/1122859?page=2) 已投票 | |
|  | | | |
| * **grzrt** * 等级: 初级会员 * [grzrt的博客](http://grzrt.iteye.com/) * 性别: http://www.iteye.com/images/icon_minigender_1.gif * 文章: 13 * 积分: 30 * 来自: 北京 * http://www.iteye.com/images/status/offline.gif | | 发表时间：2012-04-20  [收藏](http://www.iteye.com/admin/link?user_favorite%5Btitle%5D=grzrt%E5%9B%9E%E5%B8%96%3ASpring%E5%BC%80%E9%97%AD%E5%8E%9F%E5%88%99%E7%9A%84%E8%A1%A8%E7%8E%B0-BeanPostProcessor%E6%89%A9%E5%B1%95%E7%82%B9-1&user_favorite%5Burl%5D=http%3A%2F%2Fwww.iteye.com%2Ftopic%2F1122859%3Fpage%3D2%232343356)  LZ的好学的精神真值得学习，顶你 | |
| [返回顶楼](http://www.iteye.com/topic/1122859?page=2) | | [回帖地址](http://www.iteye.com/topic/1122859?page=2#2343356)  [0](http://www.iteye.com/topic/1122859?page=2) [0](http://www.iteye.com/topic/1122859?page=2) 已投票 | |
|  | | | |
| * **jinnianshilongnian** * 等级: 五钻会员 * [jinnianshilongnian的博客](http://jinnianshilongnian.iteye.com/) * 性别: http://www.iteye.com/images/icon_minigender_1.gif * 文章: 1112 * 积分: 2280 * http://www.iteye.com/images/status/offline.gif | | 发表时间：2012-04-21  [收藏](http://www.iteye.com/admin/link?user_favorite%5Btitle%5D=jinnianshilongnian%E5%9B%9E%E5%B8%96%3ASpring%E5%BC%80%E9%97%AD%E5%8E%9F%E5%88%99%E7%9A%84%E8%A1%A8%E7%8E%B0-BeanPostProcessor%E6%89%A9%E5%B1%95%E7%82%B9-1&user_favorite%5Burl%5D=http%3A%2F%2Fwww.iteye.com%2Ftopic%2F1122859%3Fpage%3D2%232343407)  **grzrt 写道**  LZ的好学的精神真值得学习，顶你  别光顶，来点实质的干货 http://www.iteye.com/images/smiles/icon_biggrin.gif ，一起探讨下 | |
| [返回顶楼](http://www.iteye.com/topic/1122859?page=2) | | [回帖地址](http://www.iteye.com/topic/1122859?page=2#2343407)  [0](http://www.iteye.com/topic/1122859?page=2) [0](http://www.iteye.com/topic/1122859?page=2) 已投票 | |
|  | | | |
| * **yxb1990** * 等级: 初级会员 * [yxb1990的博客](http://yxb1990.iteye.com/) * 性别: http://www.iteye.com/images/icon_minigender_1.gif * 文章: 48 * 积分: 80 * 来自: 深圳 * http://www.iteye.com/images/status/offline.gif | | 发表时间：2012-04-23  [收藏](http://www.iteye.com/admin/link?user_favorite%5Btitle%5D=yxb1990%E5%9B%9E%E5%B8%96%3ASpring%E5%BC%80%E9%97%AD%E5%8E%9F%E5%88%99%E7%9A%84%E8%A1%A8%E7%8E%B0-BeanPostProcessor%E6%89%A9%E5%B1%95%E7%82%B9-1&user_favorite%5Burl%5D=http%3A%2F%2Fwww.iteye.com%2Ftopic%2F1122859%3Fpage%3D2%232344304)  Spring 的方法，类名字又多又长，看的头大，看过就忘。 | |
| [返回顶楼](http://www.iteye.com/topic/1122859?page=2) | | [回帖地址](http://www.iteye.com/topic/1122859?page=2#2344304)  [0](http://www.iteye.com/topic/1122859?page=2) [0](http://www.iteye.com/topic/1122859?page=2) 已投票 | |
|  | | | |
| * **jinnianshilongnian** * 等级: 五钻会员 * [jinnianshilongnian的博客](http://jinnianshilongnian.iteye.com/) * 性别: http://www.iteye.com/images/icon_minigender_1.gif * 文章: 1112 * 积分: 2280 * http://www.iteye.com/images/status/offline.gif | | 发表时间：2012-04-24  [收藏](http://www.iteye.com/admin/link?user_favorite%5Btitle%5D=jinnianshilongnian%E5%9B%9E%E5%B8%96%3ASpring%E5%BC%80%E9%97%AD%E5%8E%9F%E5%88%99%E7%9A%84%E8%A1%A8%E7%8E%B0-BeanPostProcessor%E6%89%A9%E5%B1%95%E7%82%B9-1&user_favorite%5Burl%5D=http%3A%2F%2Fwww.iteye.com%2Ftopic%2F1122859%3Fpage%3D2%232344330)  **yxb1990 写道**  Spring 的方法，类名字又多又长，看的头大，看过就忘。  还好吧 spring分模块 分包做的到位，包前缀直接忽略 http://www.iteye.com/images/smiles/icon_biggrin.gif | |
| [返回顶楼](http://www.iteye.com/topic/1122859?page=2) | | [回帖地址](http://www.iteye.com/topic/1122859?page=2#2344330)  [0](http://www.iteye.com/topic/1122859?page=2) [0](http://www.iteye.com/topic/1122859?page=2) 已投票 | |
|  | | | |
| * **liuwuhen** * 等级: 初级会员 * [liuwuhen的博客](http://liuwuhen.iteye.com/) * 性别: http://www.iteye.com/images/icon_minigender_1.gif * 文章: 12 * 积分: 60 * 来自: 深圳 * http://www.iteye.com/images/status/offline.gif | | 发表时间：2012-08-06  [收藏](http://www.iteye.com/admin/link?user_favorite%5Btitle%5D=liuwuhen%E5%9B%9E%E5%B8%96%3ASpring%E5%BC%80%E9%97%AD%E5%8E%9F%E5%88%99%E7%9A%84%E8%A1%A8%E7%8E%B0-BeanPostProcessor%E6%89%A9%E5%B1%95%E7%82%B9-1&user_favorite%5Burl%5D=http%3A%2F%2Fwww.iteye.com%2Ftopic%2F1122859%3Fpage%3D2%232366544)  lz我想问下  假如我们使用了多个的BeanPostProcessor的实现类，那么如何确定处理顺序呢？ | |
| [返回顶楼](http://www.iteye.com/topic/1122859?page=2) | | [回帖地址](http://www.iteye.com/topic/1122859?page=2#2366544)  [0](http://www.iteye.com/topic/1122859?page=2) [0](http://www.iteye.com/topic/1122859?page=2) 已投票 | |
|  | | | |
| * **liuwuhen** * 等级: 初级会员 * [liuwuhen的博客](http://liuwuhen.iteye.com/) * 性别: http://www.iteye.com/images/icon_minigender_1.gif * 文章: 12 * 积分: 60 * 来自: 深圳 * http://www.iteye.com/images/status/offline.gif | | 发表时间：2012-08-06  [收藏](http://www.iteye.com/admin/link?user_favorite%5Btitle%5D=liuwuhen%E5%9B%9E%E5%B8%96%3ASpring%E5%BC%80%E9%97%AD%E5%8E%9F%E5%88%99%E7%9A%84%E8%A1%A8%E7%8E%B0-BeanPostProcessor%E6%89%A9%E5%B1%95%E7%82%B9-1&user_favorite%5Burl%5D=http%3A%2F%2Fwww.iteye.com%2Ftopic%2F1122859%3Fpage%3D2%232366546)  BeanFactory中和ApplicationContext中对BeanPostProcessor的处理的不同处在于那些？ | |
| [返回顶楼](http://www.iteye.com/topic/1122859?page=2) | | [回帖地址](http://www.iteye.com/topic/1122859?page=2#2366546)  [0](http://www.iteye.com/topic/1122859?page=2) [0](http://www.iteye.com/topic/1122859?page=2) 已投票 | |
|  | | | |
| * **jinnianshilongnian** * 等级: 五钻会员 * [jinnianshilongnian的博客](http://jinnianshilongnian.iteye.com/) * 性别: http://www.iteye.com/images/icon_minigender_1.gif * 文章: 1112 * 积分: 2280 * http://www.iteye.com/images/status/offline.gif | | 发表时间：2012-08-06  [收藏](http://www.iteye.com/admin/link?user_favorite%5Btitle%5D=jinnianshilongnian%E5%9B%9E%E5%B8%96%3ASpring%E5%BC%80%E9%97%AD%E5%8E%9F%E5%88%99%E7%9A%84%E8%A1%A8%E7%8E%B0-BeanPostProcessor%E6%89%A9%E5%B1%95%E7%82%B9-1&user_favorite%5Burl%5D=http%3A%2F%2Fwww.iteye.com%2Ftopic%2F1122859%3Fpage%3D2%232366549)  **liuwuhen 写道**  lz我想问下  假如我们使用了多个的BeanPostProcessor的实现类，那么如何确定处理顺序呢？  实现Order接口 | |
| [返回顶楼](http://www.iteye.com/topic/1122859?page=2) | | [回帖地址](http://www.iteye.com/topic/1122859?page=2#2366549)  [0](http://www.iteye.com/topic/1122859?page=2) [0](http://www.iteye.com/topic/1122859?page=2) 已投票 | |
|  | | | |
| * **jinnianshilongnian** * 等级: 五钻会员 * [jinnianshilongnian的博客](http://jinnianshilongnian.iteye.com/) * 性别: http://www.iteye.com/images/icon_minigender_1.gif * 文章: 1112 * 积分: 2280 * http://www.iteye.com/images/status/offline.gif | | 发表时间：2012-08-06   最后修改：2012-08-06  [收藏](http://www.iteye.com/admin/link?user_favorite%5Btitle%5D=jinnianshilongnian%E5%9B%9E%E5%B8%96%3ASpring%E5%BC%80%E9%97%AD%E5%8E%9F%E5%88%99%E7%9A%84%E8%A1%A8%E7%8E%B0-BeanPostProcessor%E6%89%A9%E5%B1%95%E7%82%B9-1&user_favorite%5Burl%5D=http%3A%2F%2Fwww.iteye.com%2Ftopic%2F1122859%3Fpage%3D2%232366550)  **liuwuhen 写道**  BeanFactory中和ApplicationContext中对BeanPostProcessor的处理的不同处在于那些？  BeanFactory仅仅是个Bean的工厂（因此核心在提供Bean），ApplicationContext接口继承了BeanFactory。   BeanPostProcessor仅仅是ApplicationContext在创建Bean时 插入的扩展点。 | |
| [返回顶楼](http://www.iteye.com/topic/1122859?page=2) | | [回帖地址](http://www.iteye.com/topic/1122859?page=2#2366550)  [0](http://www.iteye.com/topic/1122859?page=2) [0](http://www.iteye.com/topic/1122859?page=2) 已投票 | |