链滴

Spring BeanPostProcessor 接口使用

作者：umeone

原文链接：https://ld246.com/article/1514960493653

来源网站：链滴

● Spring中提供了很多PostProcessor供开发者进行拓展，例如：BeanPostProcessor、BeanFactoryostProcessor、BeanValidationPostProcessor等一系列后处理器。他们的使用方式大多类似，了解中一个并掌握他的使用方式，其他的可以触类旁通。

● 这里以BeanPostProcessor为例展示其使用方式。

● BeanPostProcessor接口提供了两个供开发者自定义的方法：postProcessBeforeInitialization、pstProcessAfterInitialization。

● postProcessBeforeInitialization：该方法主要针对spring在bean初始化时调用初始化方法前进行定义处理。

● postProcessAfterInitialization：该方法主要针对spring在bean初始化时调用初始化方法后进行自义处理。

● 示例代码：

```
/**
 * 测试bean
 */
public class Cat {

    private String name;

    private int age;

    public void say() {
        System.out.println("name:" + name);
        System.out.println("age:" + age);
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }
}

/**
 * 自定义后处理器
 */
public class CatBeanPostProcessor implements BeanPostProcessor {

    @Nullable
    @Override
    public Object postProcessBeforeInitialization(Object bean, String beanName) throws Beans
```

```
xception {
        if (bean instanceof Cat) {
            //输出原始属性
            Cat cat = (Cat) bean;
            cat.say();
            return bean;
        }
        return bean;
    }

    @Nullable
    @Override
    public Object postProcessAfterInitialization(Object bean, String beanName) throws BeansE
ception {
        if (bean instanceof Cat) {
            //修改属性值，并返回
            Cat cat = (Cat) bean;
            cat.setName("hello maomi");
            cat.setAge(3);
            return cat;
        }
        return bean;
    }

}

/**
 * 运行
 */
public class Run {

    public static void main(String[] args) {
        ApplicationContext applicationContext = new ClassPathXmlApplicationContext("spring-
ean.xml");
        Cat cat = (Cat) applicationContext.getBean("cat");
        cat.say();
    }

}
```

xml配置信息

```
    <!--配置bean并初始化-->
    <bean id="cat" class="com.source.postprocessor.Cat" >
        <property name="name" value="HelloKitty" />
        <property name="age" value="1" />
    </bean>

    <bean id="catBeanPostProcessor" class="com.source.postprocessor.CatBeanPostProcessor
 />
```

输出结果：

```
name:HelloKitty
age:1
```

```
name:hello maomi
age:3
```

- 可以看到通过后处理器处理过后的bean信息已经改变。最后，看看源码中如何调用自定义实现的。
  - 在初始化bean方法中：AbstractAutowireCapableBeanFactory.java

```
  /**
   * 初始化bean
   */
  protected Object initializeBean(final String beanName, final Object bean, @Nullable RootBe
nDefinition mbd) {
      //省略部分无关代码
      Object wrappedBean = bean;
      //初始化前
      if (mbd == null || !mbd.isSynthetic()) {
          wrappedBean = applyBeanPostProcessorsBeforeInitialization(wrappedBean, beanName)

      }

      try {
          //调用初始化方法初始化bean
          invokeInitMethods(beanName, wrappedBean, mbd);
      }
      catch (Throwable ex) {
          throw new BeanCreationException(
                  (mbd != null ? mbd.getResourceDescription() : null),
                  beanName, "Invocation of init method failed", ex);
      }
      //初始化后
      if (mbd == null || !mbd.isSynthetic()) {
          wrappedBean = applyBeanPostProcessorsAfterInitialization(wrappedBean, beanName);
      }
      return wrappedBean;
  }
```

  - postProcessBeforeInitialization方法调用

```
  @Override
  public Object applyBeanPostProcessorsBeforeInitialization(Object existingBean, String bean
ame)
          throws BeansException {

      Object result = existingBean;
      for (BeanPostProcessor beanProcessor : getBeanPostProcessors()) {
          //调用自定义postProcessBeforeInitialization方法
          Object current = beanProcessor.postProcessBeforeInitialization(result, beanName);
          if (current == null) {
              return result;
          }
          result = current;
      }
      return result;
  }
```

- postProcessAfterInitialization方法调用

```
 @Override
 public Object applyBeanPostProcessorsAfterInitialization(Object existingBean, String beanN
me)
        throws BeansException {

    Object result = existingBean;
    for (BeanPostProcessor beanProcessor : getBeanPostProcessors()) {
        //自定义postProcessAfterInitialization方法调用
        Object current = beanProcessor.postProcessAfterInitialization(result, beanName);
        if (current == null) {
            return result;
        }
        result = current;
    }
    return result;
 }
```

- 以上就是spring对自定义方法实现的调用过程。