**3.2  建立AopProxy代理对象**

**3.2.1  配置ProxyFactoryBean**

从这部分开始，我们进入到Spring AOP的实现部分，在分析Spring AOP的实现原理中，我们主要以ProxyFactoryBean的实现作为例子和实现的基本线索进行分析。很大一个原因是因为ProxyFactoryBean是在Spring IoC环境中，创建AOP应用的最底层方法，也是最灵活的方法，Spring通过它完成了对AOP使用的封装，以它的实现为入口逐层深入，是很好的一条帮助我们理解Spring AOP实现的学习路径。

在了解ProxyFactoryBean的实现之前，我们先简要地了解一下ProxyFactoryBean的配置和使用，在基于XML配置Spring的Bean的时候，往往需要一系列的配置步骤来使用ProxyFactoryBean和AOP，比如以下这些步骤：

1）定义使用的通知器Advisor，这个通知器应该作为一个Bean来定义。很重要的一点，这个通知器的实现定义了需要对目标对象进行增强的切面行为，也就是Advice通知。

2）定义proxyFactoryBean，把它作为另一个Bean来定义，它是封装AOP功能的主要类。在配置ProxyFactoryBean时，需要设定与AOP实现相关的重要属性，比如ProxyInterface、interceptorNames和target等。从属性名称可以看出，interceptorNames属性的值往往设置为需要定义的通知器，因为这些通知器在ProxyFactoryBean的AOP配置下，是通过使用代理对象的拦截器机制起作用的。所以，这里依然沿用了拦截器这个名字，也算是旧瓶装新酒吧。

3）定义target属性，作为target属性注入的Bean，是需要用AOP通知器中的切面应用来增强的对象，也就是前面我们提到的base对象。

有了这些配置，就可以使用ProxyFactoryBean完成AOP的基本功能了。关于配置的例子，如代码清单3-10所示。与前面提到的配置步骤相对应，除了定义了ProxyFactoryBean的AOP封装，还定义了一个Advisor，取名为testAdvisor。作为ProxyFactory配置的一部分，还需要配置拦截的方法调用接口和目标对象。这些基本的配置，是使用ProxyFactoryBean实现AOP功能的重要组成，对于这些配置的实现和作用机制，也是我们后面重点分析的内容。

代码清单3-10  配置ProxyFactoryBean

1. <bean id="testAdvisor" **class**="com.abc.TestAdvisor"/>
2. <bean id="testAOP" **class**="org.springframework.aop.ProxyFactoryBean>
3. <property name="proxyInterfaces"><value>com.test.AbcInterface"
4. </value></property>
5. <property name="target">
6. <bean **class**="com.abc.TestTarget"/>
7. </property>
8. <property name="interceptorNames">
9. <list><value> testAdvisor</value></list>
10. </property>
11. </bean>

掌握这些配置后，就可以具体看一看这些AOP是如何实现的。也就是说，切面应用是怎样通过ProxyFactoryBean对target对象起作用的。对于这个ProxyFactoryBean，我们先了解一下ProxyFactoryBean的类层次关系，如图3-8所示。在这个类层次关系中，可以看到用来完成AOP应用的类，比如AspectJProxyFactory、ProxyFactory和ProxyFactoryBean，它们都在同一个类的继承体系下，都是ProxyConfig、AdvisedSupport和ProxyCreatorSupport的子类。作为大家的共同基类，ProxyConfig可以看成是一个数据类，这个数据基类为像ProxyFactoryBean这样的子类提供了配置属性。在另一个基类AdvisedSupport的实现中，它封装了AOP中对通知和通知器的相关操作，这些操作对于不同的AOP的代理对象的生成都是一样的，但对于具体的AOP代理对象的创建，AdvisedSupport把它交给它的子类们去完成。ProxyCreatorSupport看成是其子类创建AOP代理对象的一个辅助类，通过继承以上提到的基类的功能实现，具体的AOP代理对象的生成，根据不同的需要分别由ProxyFactoryBean、AspectJProxyFactory和ProxyFactory来完成。对于需要使用AspectJ的AOP应用，AspectJProxyFactory起到集成Spring和AspectJ的作用；对于使用Spring AOP的应用，ProxyFactoryBean和ProxyFactoy都提供了AOP功能的封装，只是如果使用ProxyFactoryBean，可以在IoC容器中完成声明式配置，而使用ProxyFactory，则需要编程式的使用Spring AOP的功能；对于它们是如何封装实现AOP功能的，在本章后面的小节中给出详细的分析，在这里，通过这些类层次关系的了解，希望能先给读者留下一个大致的印象。

|  |
| --- |
| <http://images.51cto.com/files/uploadimg/20091222/143558743.jpg> |
|  |