24_Aspect、Join Points、Pointcuts 和 Advice 语法

1、开篇

上节课我们介绍 AOP 技术，以及其应用原理，从而引出 Spring AOP 的动态代理机制，包括：JDK 代理和 GCLIB 代理，并且通过例子对它们进行描述。本节课依旧围绕 Spring AOP 应用中的几个概念展开，内容包括：

- Spring AOP 核心概念综述
- Spring AOP 举例

2、Spring AOP 核心概念综述

课程标题中提到了 Aspect、Join Points、Pointcuts 以及 Advice，这些都属于 Spring AOP 的核心概念，Spring AOP 是通过这些概念的组合完成代码的织入工作的。

如图 1 所示，这张图将 Spring AOP 的概念进行了总体描述。先从向右的箭头说起，这个箭头包含了一些小方块，我们称之为 Join Points，它是可以用来进行增强的方法点，例如上节课中 Service 类中的 help 方法，说白了就是对什么方法进行增强。

看完了 Join points 之后在往上看，有一个向下的箭头，上面标注着 Pointcut，我们称之为切入点。切入点是告诉我们在方法的什么位置进行增强，比如在方法执行之前增强，还是执行之后增强，或者两者皆有。Advice 是最上面的方块，它表示需要增强的功能，上节课的例子中在“买书”的前面和后面都加入了“买书之前”和“买书之后”的打印语句。

这里的“买书之前”和“买书之后”就是增强功能，再例如我们如果需要编写日志也可以放在这个增强方法里面完成。最后，Advice 和 Pointcut 的组合就是切面 Aspect。

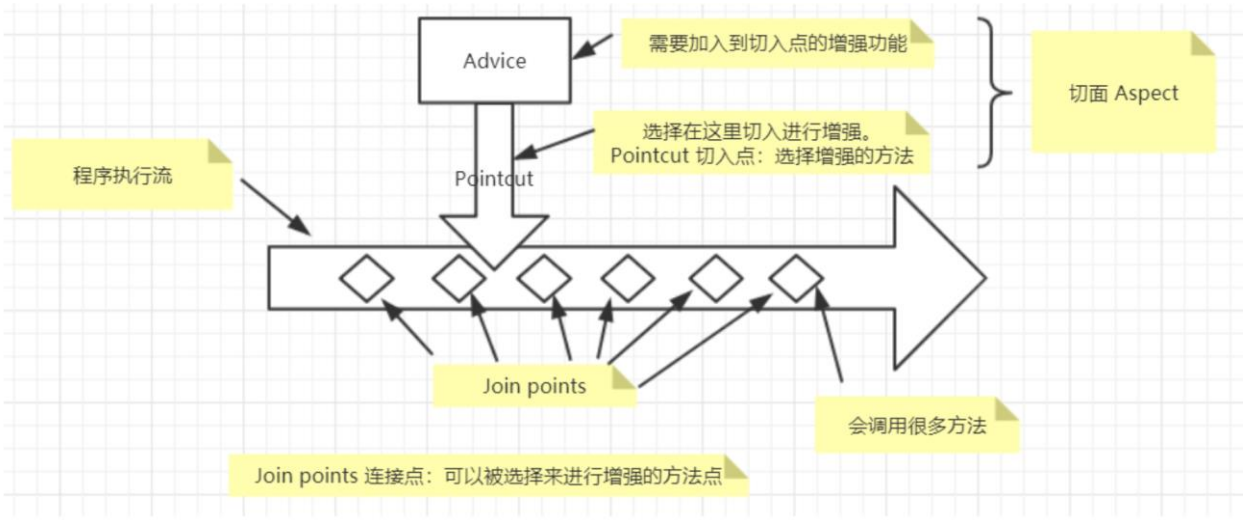

图 1 Spring AOP 核心概念

3、Spring AOP 举例

上面对 Spring AOP 的核心概念进行了描述，Aspect、Join Points、Pointcuts 以及 Advice 这几个概念，看上去还是比较抽象，这里会通过一个例子讲这几个概念进行说明。

如图 2 所示，依旧是 Biz 类，它作为需要增强的目标类，其中包括 help 方法打印出“买书”，service 方法打印出“提供买书服务”。假设对这两个方法进行增强，那么这两个方法就是 Join Points 了。

```
public class Biz {
    public void help() {
        System.out.println("买书");
    }

    public void service() {
        System.out.println("提供买书服务");
    }
}
```

图 2 目标类

上面描述了目标类和 Point cuts 也就是需要增强的方法，这里说明一下具体的增强内容，也就是 Advice。如图 3 所示，这里定义了两个类 BeforeAdvice、ArroundAdvice 分别实现了 MethodBeforeAdvice 和 MethodInterceptor。

其中 BeforeAdvice 类中 Override 了 before 方法，顾名思义是在方法执行之前运行 before 方法中的内容。再看 ArroundAdvice 类实现了 MethodInterceptor 类并且 Override 了 invoke 方法，通过 invocation.proceed() 调用代理的方法，并且在该方法执行的前后加上了“方法执行之前增强”和“方法执行之后增强”的打印语句。

```
public class BeforeAdvice implements MethodBeforeAdvice {

    @Override
    public void before(Method method, Object[] args, Object target) throws Throwable {
        System.out.println("方法执行之前增强");
    }
}

public class ArroundAdvice implements MethodInterceptor {

    @Override
    public Object invoke(MethodInvocation invocation) throws Throwable {
        System.out.println("方法执行之前增强");
        Object ret = invocation.proceed();
        System.out.println("方法执行之后增强");
        return ret;
    }
}
```

图 3 具体增强内容

有了 Join points 和 advice 的定义，再来看在什么地方进行增强，前面说了可以在方法执行之前、之后或者是两者都有。我们在 advice 的定义中包含了方法执行之前和前后皆有的 advice，这里通过配置文件看看它们是如何与目标类以及 Advice 进行关联的。

如图 4 所示，在 bean 的定义中，定义目标类 Biz，同时也定义了对应 advice 类包括：beforeAdvice 和 arroundAdvice。在 aop：config 节点中定义了第一个 pointcut，id 为“doMethods”，在 expression 中定义了这个 pointcut 的执行范围，显示是可以作用于对应 namespace 中的 help 方法的。

在 advisor 节点中定义了 beforeAdvice，在 pointcut-ref 中引用了 doMethods 的 pointcut，也就是说对于所有 namespace 在 com.aop.* 的类中使用 help 方法的时候，会使用 beforeAdvice 的增强方法，也就是在 help 方法之前加入增强代码。同理，定义另外一个 advisor 节点，使用了 arroundAdvice 增强方法，针对所有的 namespace 为“com.aop.*”的类中方法为 service 进行增强，增强的方法遵循 arroundAdvice 中描述的：在方法执行之前和之后都插入打印语句。

```
<beans>
    <bean id="Biz" class="com.aop.Biz" />
    <!--配置advice  -->
    <bean id="beforeAdvice" class="com.aop.BeforeAdvice" />
    <bean id="arroundAdvice" class="com.aop.ArroundAdvice" />
    <!--配置pointcut  -->
    <aop:config >
        <aop:pointcut id="doMethods" expression="execution(* com.aop.*.help(..))" />
        <aop:advisor advice-ref="beforeAdvice" pointcut-ref="doMethods" />
        <aop:advisor advice-ref="arroundAdvice" pointcut="execution(* com.aop.*.service(..))"/>
    </aop:config>
</beans>
```

图 4 point cut 配置文件

完成配置以后，再来看看测试代码，如图 5 所示，首先引入配置的 xml 文件。然后获取 Biz 类的实例，并且调用 help 和 service 方法。

```
public static void main(String[] args) {
    ApplicationContext context = new GenericXmlApplicationContext(
            "classpath:com/aop/application.xml");
    Biz biz = context.getBean(Biz.class);
    biz.help();
    biz.service();
}
```

图 5 测试增强方法

执行测试代码之后显示如下内容，由于“买书”使用了 BeforeAdvice 的增强方法，因此在执行 help 方法打印出“买书”之前会先打印出“方法执行之前增强”。同样，“提供买书服务”使用了 ArroundAdvice 的环绕增强方法，因此在执行 service 方法的时候，会在“提供买书服务”前面和后面分别加入“方法执行之前增强”和“方法执行之后增强”的打印语句。

方法执行之前增强

买书

方法执行之前增强

提供买书服务

方法执行之后增强

4、总结

本节课延续上节课的 Spring AOP 的主题，介绍了 Spring AOP 中的几个核心概念：Aspect、Join Points、Pointcuts 以及 Advice。并且通过代码例子告诉大家它们的基本用法。由于本节课是本周最后一节课，这里对本周的课程做一个总结。

本周主要对 Spring IOC 和 Spring AOP 进行了概要性描述，由于两者是 Spring 核心开发的基础，因此从依赖查找、依赖注入以及依赖来源的方法让大家了解了 Spring IOC 的概念和工作原理。同样通过 JDK 动态代理和 CGLIB 动态代理以及 Spring AOP 的核心概念让大家对，Spring AOP 有深入的了解。

希望通过本周课程让大家对 Spring IOC 和 AOP 这两大 Spring 开发基石有全面的了解和掌握。下周我们会进入正式代码实战环节。下期见，拜拜。