37_精选面试题：Spring AOP 的设计模式有哪些？

1、开篇

上节通过 Spring AOP 实现了登陆信息拦截的功能，该功能包括两个部分：一、编写拦截器类，其中包括拦截规则以及拦截以后的处理；二、为 spring-web.xml 中配置拦截器的信息，拦截路径、例外拦截路径以及拦截器的实现类。本节课来讲讲 Spring AOP 的设计模式有哪些。今天的内容：

- 策略模式在 AOP 中的使用
- 模板模式在 AOP 中的使用
- 适配器模式在 AOP 中的使用

2、策略模式在 AOP 中的使用

策略模式（Strategy Pattern），将各种算法封装到具体的类中，作为一个抽象策略类的子类，使得它们可以互换。客户端可以自行决定使用哪种算法。

如图 1 所示，策略模式角色划分为如下三个部分：

- Strategy 策略接口或者（抽象策略类），定义策略执行接口
- ConcreteStrategy 具体策略类
- Context 上下文类，持有具体策略类的实例，并负责调用相关的算法

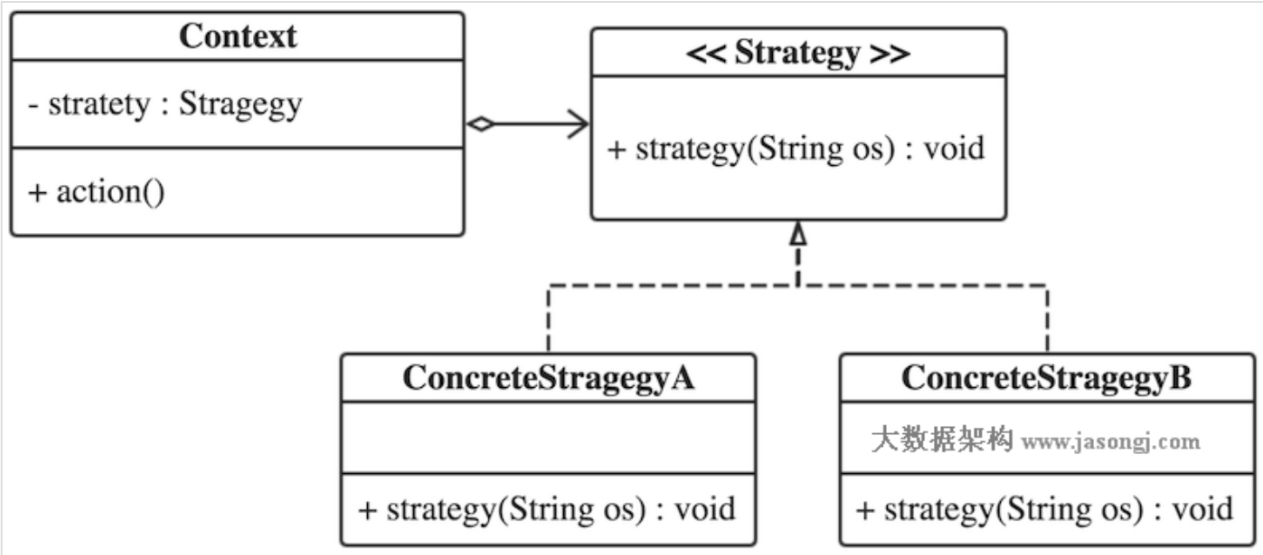

图 1 策略模式

策略模式提供了对“开闭原则”的完美支持，用户可以在不修改原有系统的基础上选择算法（策略），并且可以灵活地增加新的算法（策略）。

策略模式通过 Context 类提供了管理具体策略类（算法族）的办法。

结合简单工厂模式和 Annotation，策略模式可以方便的在不修改客户端代码的前提下切换算法（策略）。

在 Spring AOP 中策略模式也起到了重要的作用，还接的 Spring AOP 的两种动态代理机制吗？JDK 动态代理和 CGLIB 动态代理，这里两种代理就是两种 AOP 代理的策略。如图 2 琐事，AOP 中定义了 DefaultAopProxyFactory 的代理类，这个类分别关联了两种代理：JdkDynamicAopProxy（JDK 动态代理）和 CglibProxyFactory（CGLIB 动态代理）。

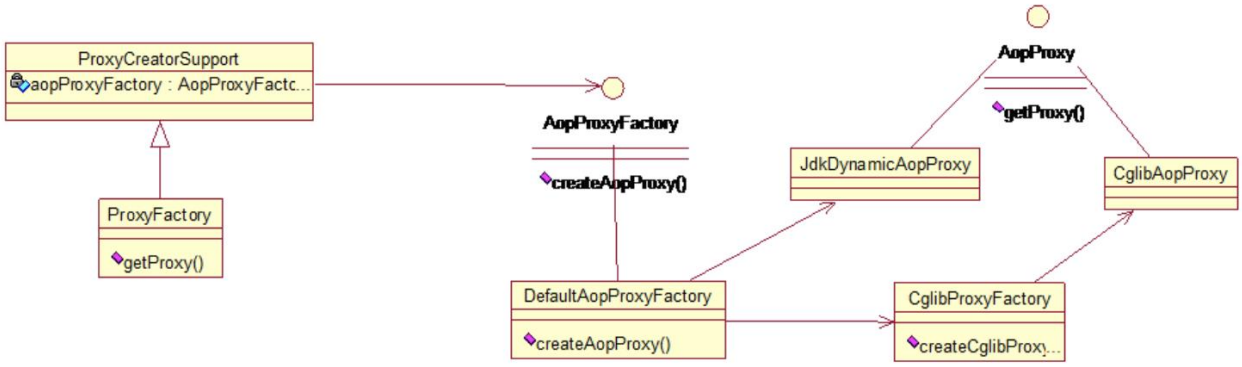

图 2 代理策略

3、模板模式在 AOP 中的使用

模板模式（Template Pattern）中，一个抽象类公开定义了执行它的方法的方式/模板。它的子类可以按需要重写方法实现，但调用将以抽象类中定义的方式进行。这种类型的设计模式属于行为型模式。

如图 3 所示，模版模式包括两部分内容：

- 抽象类（AbstractClass）：实现了模板方法，定义了算法的骨架。
- 具体类（ConcreteClass)：实现抽象类中的抽象方法，已完成完整的算法。

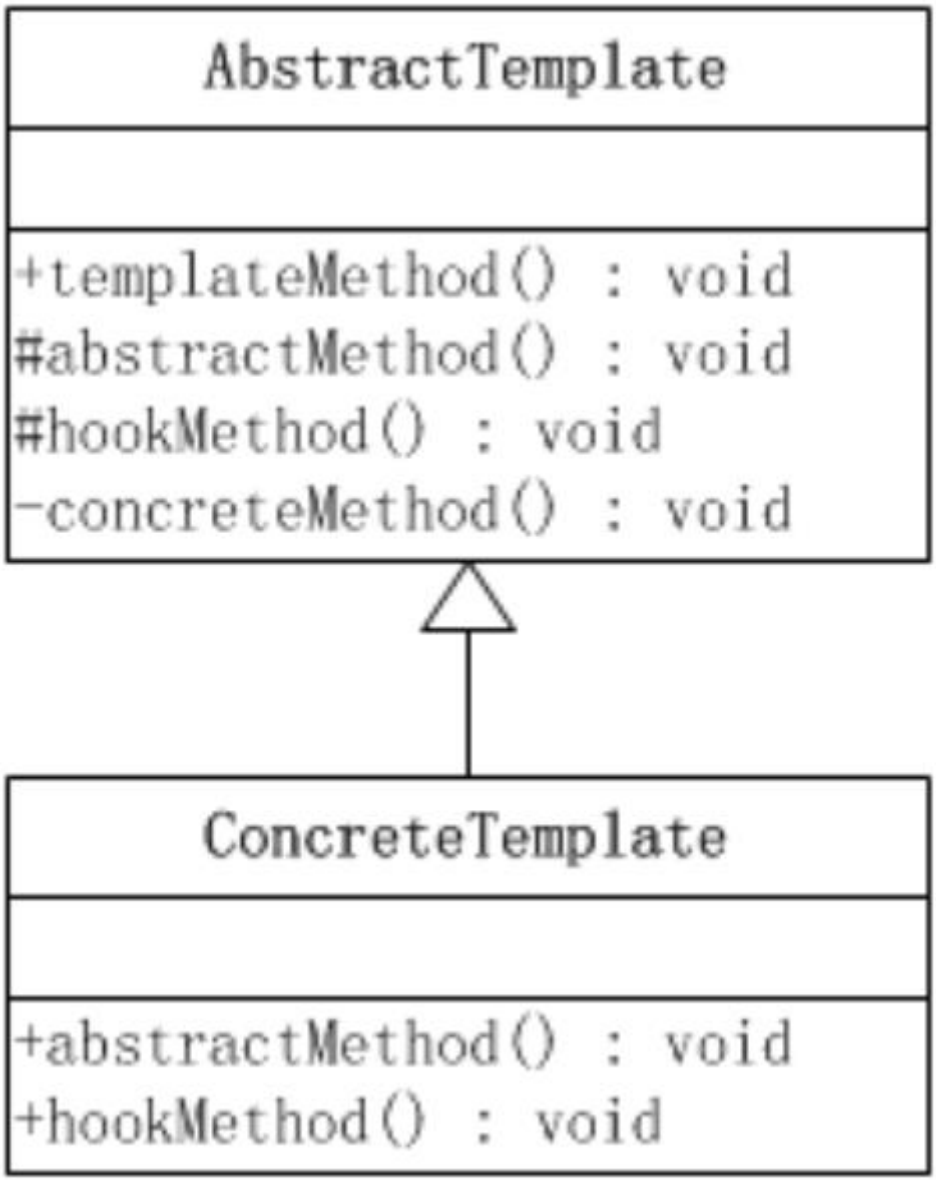

图 3 模版模式

模板方法模式通过把不变的行为搬移到超类，去除了子类中的重复代码。子类实现算法的某些细节，有助于算法的扩展。通过一个父类调用子类实现的操作，通过子类扩展增加新的行为，符合“开放-封闭原则”。

回到 Spring AOP 中在代理创建的时候用到了 AbstractAutoProxyCreator，如图 4 所示，它位于图的上部定义了一些通用的行为，例如：postProcessBeforeInstantiation、getAdvicesAndAdvisorsForBean、CreateProxy 等，这些行为是所有代理创建者都需要使用的，因此在父类中进行统一定义。

AbstractAdvisorAutoProxyCreator，是根据增强器自动代理的创建者，它加入针对增强器特有的方法。再往下有三个类都继承了AbstractAdvisorAutoProxyCreator，分别是 DefaultAdvisorAutoProxyCreator、AspectJAwareAdvisorAutoProxyCreator 和 InfrastructureAdvisorAutoProxyCreator，它们又根据自身处理场景扩展了父类的模版方法。

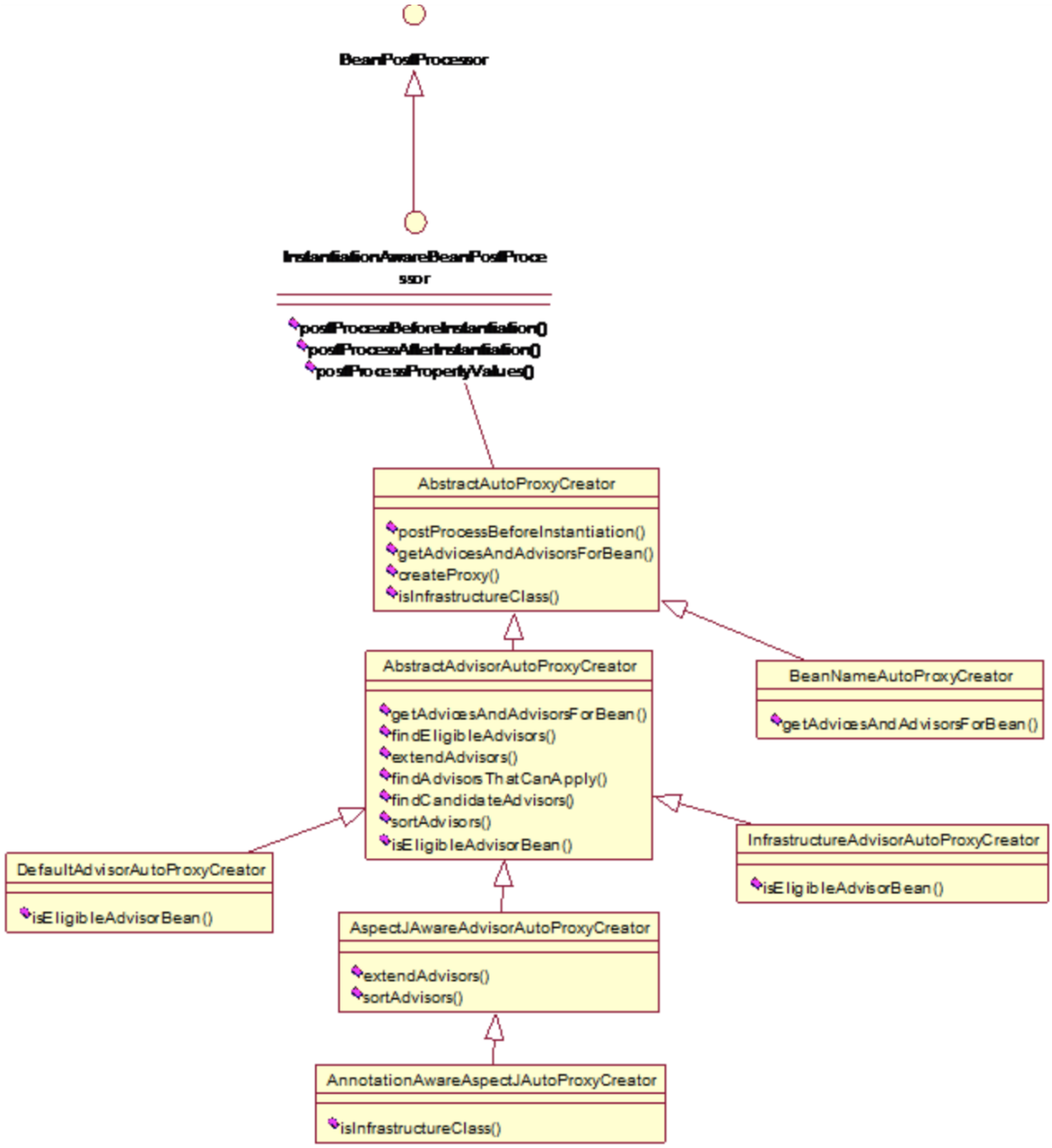

图 4 模版模式

4、适配器模式在 AOP 中的使用

适配器模式（Adapter Pattern），将一个类的接口转换成客户希望的另外一个接口。适配器模式使得原本由于接口不兼容而不能一起工作的那些类可以一起工作。如图 5 所示，适配器模式包括如下部分：

- 目标接口， Target
- 具体目标实现，如 ConcreteTarget
- 适配器，Adapter
- 待适配类，Adaptee

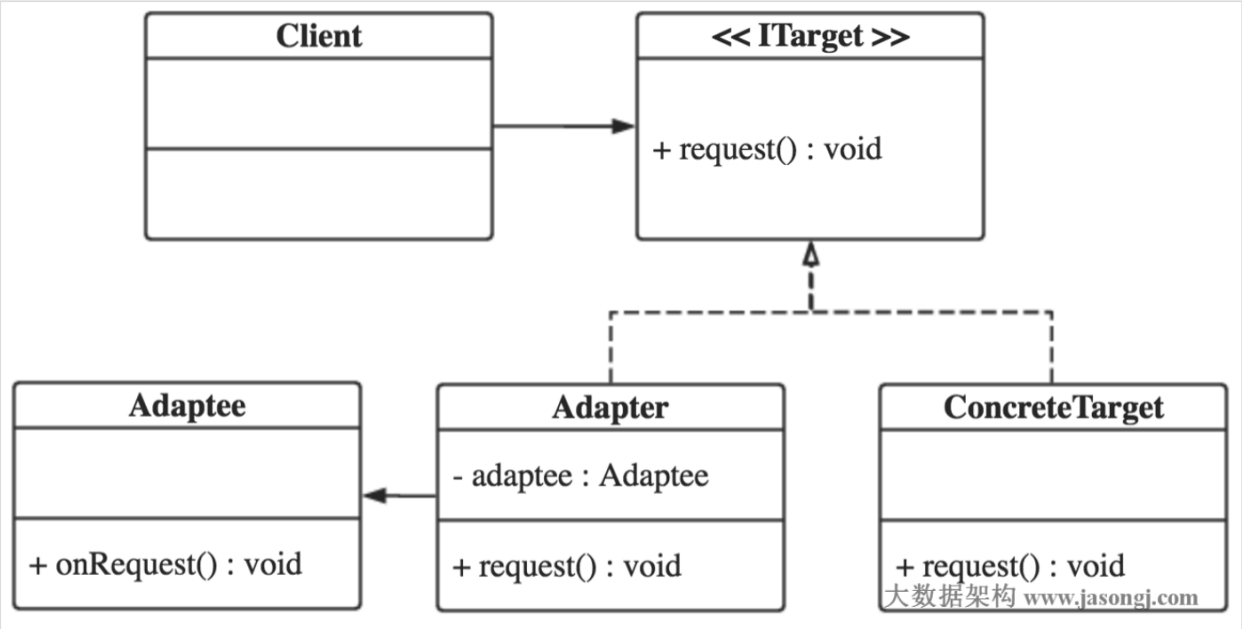

图 5 适配器模式

回到 Spring AOP 如图 6 琐事，其中 Advisor 使用了 AdvisorAdapter 适配器，该适配器关联另外两个适配器：ThrowsAdviceAdapter 和 AfterReturningAdviceAdapter，这两个适配会分别依赖 ThrowsAdviceInterceptor 和 AfterRuturningAdviceInterceptor。也就是说 Advisor 增强器通过适配器模式使用了不同接口的 Interceptor 使用的拦截器中的功能，通过适配器模式做到了代码解耦和复用。

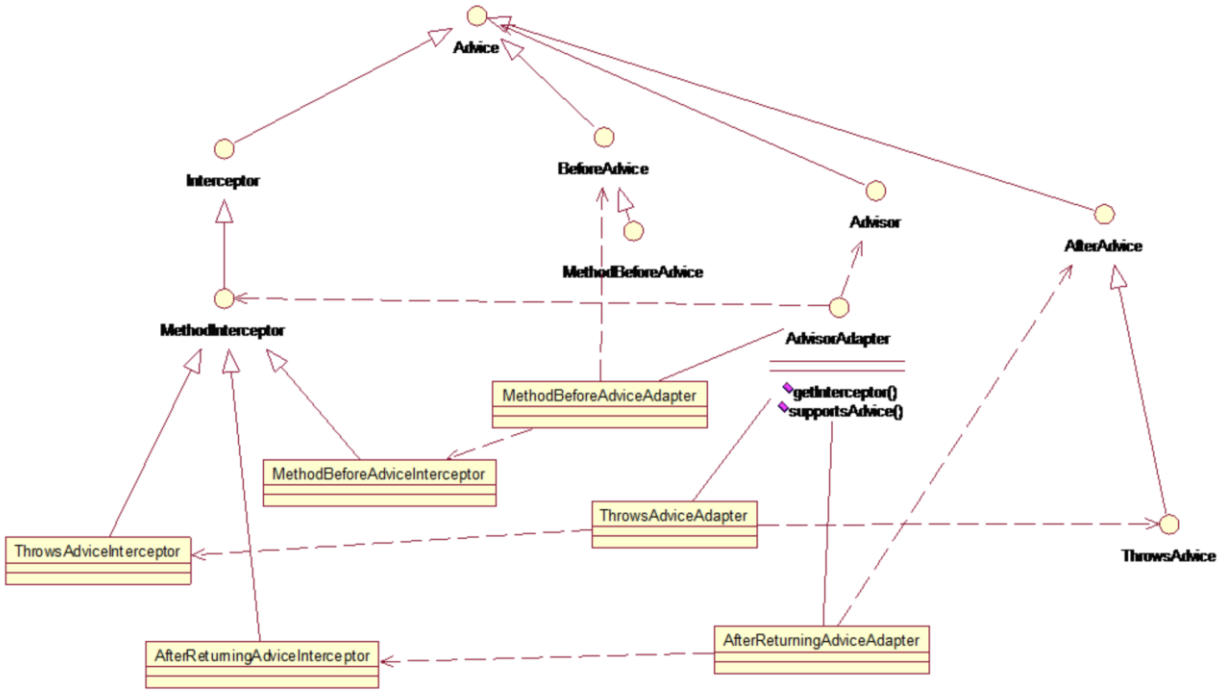

图 6 适配器模式

5、总结

本节课介绍了策略模式、模版模式、适配器模式在 Spring AOP 中的应用，在介绍三种模式的同时，也通过 UML 图的方式列出了 Spring AOP 中使用模式的精髓。

鉴于本节课是这周最后一节课，在这里将本周课程做一个总结。

本周以 Spring AOP 在项目中的应用为主，通过 logback 框架为切入点引出 Spring AOP 拦截并打印日志的功能，同时提出处理异常的问题，又使用 Spring AOP 处理系统的全局异常，最后通过拦截器的方式判断登陆时信息是否合乎要求。在主线中还穿插了@Aspect 运作原理和 Spring AOP 设计模式的讲解。

下周课将专注于另外一个系统级别的功能介绍：数据库访问。下期见，拜拜。