【第36话：什么是AOP你的项目哪里使用了AOP】

Hello 小伙伴们，这节课给大家讲解一下Spring AOP。这是一个经典又常见的面试题。

我们先来看看Spring官网中的**官方说明:**

官方链接： https://docs.spring.io/spring-framework/docs/5.3.24/reference/html/core.html#aop

5. Aspect Oriented Programming with Spring

Aspect-oriented Programming (AOP) complements Object-oriented Programming (OOP) by providing another way of thinking about program structure. The key unit of modularity in OOP is the class, whereas in AOP the unit of modularity is the aspect. Aspects enable the modularization of concerns (such as transaction management) that cut across multiple types and objects. (Such concerns are often termed "crosscutting" concerns in AOP literature.)

One of the key components of Spring is the AOP framework. While the Spring IoC container does not depend on AOP (meaning you do not need to use AOP if you don't want to), AOP complements Spring IoC to provide a very capable middleware solution.

翻译含义：

面向切面编程 (AOP) 通过提供另一种思考程序结构的方式来补充面向对象编程 (OOP)。OOP 中模块化的关键单位是类，而 AOP 中模块化的单位是切面。切面能够实现跨越多种类型和对象的关注点（例如事务管理）的模块化.(这种关注点在 AOP 文献中通常被称为“横切”关注点。）

Spring 的关键组件之一是 AOP 框架。虽然 Spring IoC 容器不依赖于 AOP（意味着如果您不想使用 AOP，则不需要使用 AOP），但 AOP 补充了 Spring IoC 以提供一个非常强大的中间件解决方案。

下面对上面官方解释的总结：官方在强调AOP时强调了下面几点：

1. AOP 叫做面向切面编程
2. AOP 是对OOP的补充
3. AOP的核心是切面
4. AOP是对IoC的补充

想要把AOP解释清楚，就必须把AOP中的专业术语搞清楚。

- Aspect: A modularization of a concern that cuts across multiple classes. Transaction management is a good example of a crosscutting concern in enterprise Java applications. In Spring AOP, aspects are implemented by using regular classes (the schema-based approach) or regular classes annotated with the `@Aspect` annotation (the @AspectJ style).
- Join point: A point during the execution of a program, such as the execution of a method or the handling of an exception. In Spring AOP, a join point always represents a method execution.
- Advice: Action taken by an aspect at a particular join point. Different types of advice include "around", "before" and "after" advice. (Advice types are discussed later.) Many AOP frameworks, including Spring, model an advice as an interceptor and maintain a chain of interceptors around the join point.
- Pointcut: A predicate that matches join points. Advice is associated with a pointcut expression and runs at any join point matched by the pointcut (for example, the execution of a method with a certain name). The concept of join points as matched by pointcut expressions is central to AOP, and Spring uses the AspectJ pointcut expression language by default.
- Introduction: Declaring additional methods or fields on behalf of a type. Spring AOP lets you introduce new interfaces (and a corresponding implementation) to any advised object. For example, you could use an introduction to make a bean implement an `IsModified` interface, to simplify caching. (An introduction is known as an inter-type declaration in the AspectJ community.)
- Target object: An object being advised by one or more aspects. Also referred to as the "advised object". Since Spring AOP is implemented by using runtime proxies, this object is always a proxied object.
- AOP proxy: An object created by the AOP framework in order to implement the aspect contracts (advise method executions and so on). In the Spring Framework, an AOP proxy is a JDK dynamic proxy or a CGLIB proxy.
- Weaving: linking aspects with other application types or objects to create an advised object. This can be done at compile time (using the AspectJ compiler, for example), load time, or at runtime. Spring AOP, like other pure Java AOP frameworks, performs weaving at runtime.

Aspect：切面。即join point + Advice

join point: 切入点。就是我们平时说的目标方法，或说对哪个方法做扩展，做增强。

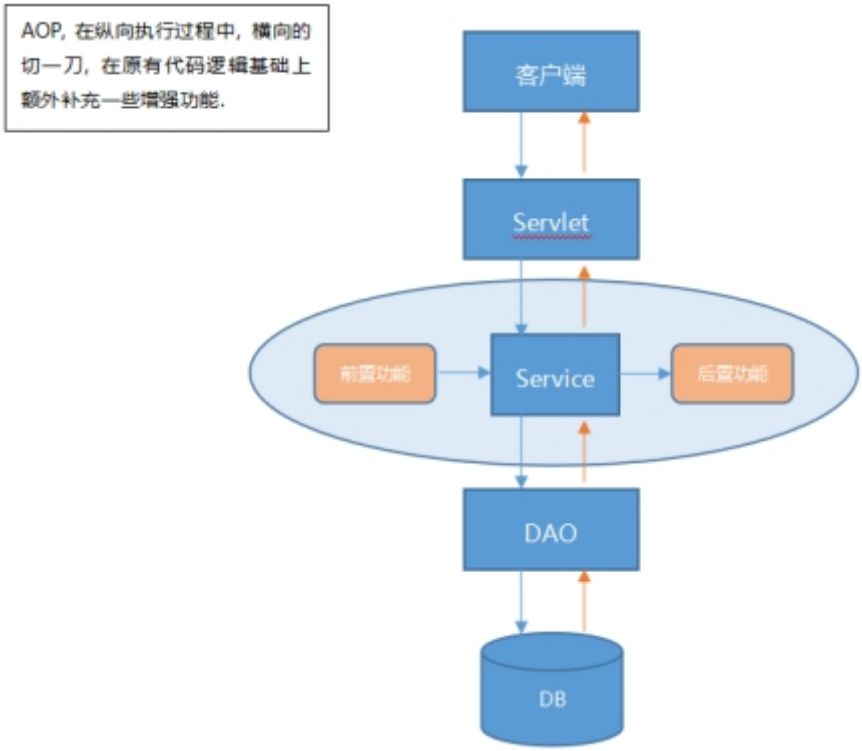
Advice：通知，增强内容。

Pointcut：切点。就是表达式，通过表达式说明哪些方法是join point

AOP Proxy：代理。Spring支持JDK动态代理和cglib动态代理两种方式，可以通过proxy-target-class=true把默认的JDK动态代理修改为Cglib动态代理。

Weaving：织入。织入就是把Advice添加到join point的过程。

在实际开发中AOP主要应用在Service层。



如果面试官问：什么是AOP?

AOP叫做面向切面编程，属于对OOP的扩展。其实现是基于动态代理设计模式，在IoC基础上实现的。

AOP就是对某个切入点做了通知进行增强扩展，形成横切面。可以实现在不修改原有代码的情况下，做额外扩展。

实现AOP的两种方式

在Spring中提供了两种方式实现AOP：

- Schema-based:所有的通知都需要实现特定类型的接口。
- AspectJ：可以使用普通Java类结合特定的配置标签实现通知。

上面是对AOP的解释，项目中哪里可以使用AOP呢?

- 声明式事务
- AOP的日志记录
- 统一权限管理
- 异常捕获和处理