39 图说AOP---妈妈再也不担心我概念混淆了

更新时间：2020-08-20 09:27:06

> 耐心是一切聪明才智的基础。——柏拉图

背景

假设我们想将多个木板组装一个家具，我们该怎么做呢？

简单一点，用锤子将钉钉起来。

这种属于比较简单暴力的方式，容易将木板弄坏。

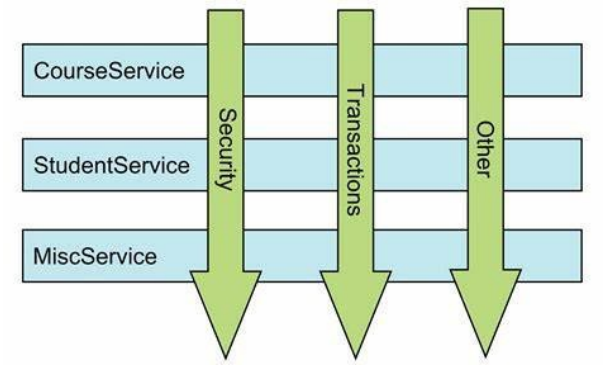
另外一种方式则比较常用，通过在木板特定的位置打孔，

然后通过螺栓固定在一起，组装成一个有用的家具。

AOP 的方式类似与家具的组装。

图说**AOP**概念

AOP（Aspect-Oriented Programming），即面向切面编程，它与 OOP（Object-Oriented Programming，面向对象编程）相辅相成，提供了与 OOP 不同的抽象软件结构的视角。在 OOP 中，我们以类（class）作为我们的基本单元，而 AOP 中的基本单元是 Aspect（切面）。SpringAOP的在实际应用中场景有哪些？



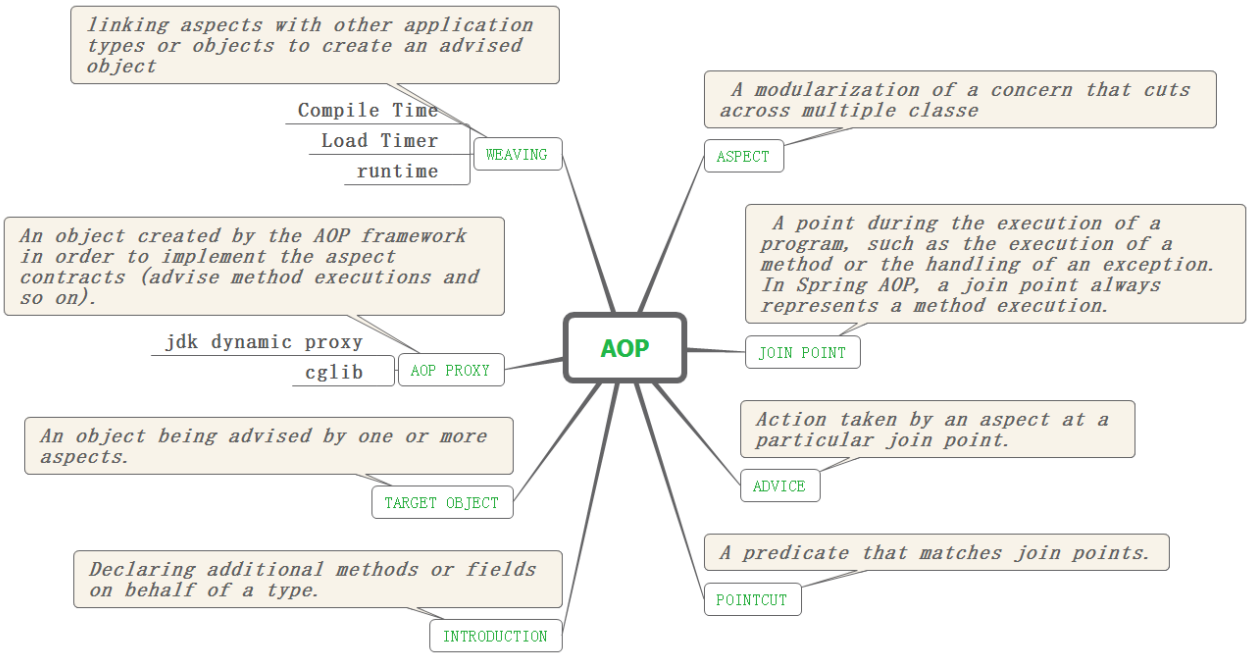
1. Authentication 认证：检测用户是否登录，未登录则返回登录页面。
2. Caching 缓存：
3. Context passing 上下文传递
4. Error handling 错误处理如SpringMVC 中*@ControllerAdvice* 注解。
5. Lazy loading延迟加载
6. Debugging 调试
7. logging，tracing，profiling and monitoring 记录跟踪 优化 校准
8. Performance optimization 性能优化
9. Persistence 持久化
10. Resource pooling 资源池
11. Synchronization 同步

12. Transactions 事务
13. Logging 日志

AOP 给人难以理解的一个关键点是它的概念比较多，而且坑爹的是，这些概念经过了中文翻译后，变得面目全非，相同的一个术语，在不同的翻译下，含义总有着各种莫名其妙的差别. 鉴于此，我们直接拿官方的说明来看。

概述

下图是从Spring AOP 源码上摘取的 aop 的组件：



切面Aspect

> A modularization of a concern that cuts across multiple classe

切面是一个横切关注点的模块化，一个切面能够包含同一个类型的不同增强方法，比如说事务处理和日志处理可以理解为两个切面。
Aspect 由 Advice 和 PointCut 组成，它既包含了横切逻辑的定义，也包括了切入点的定义。
Spring AOP就是负责实施切面的框架，它将切面所定义的横切逻辑织入到切面所指定的连接点中。

通俗的说，Aspect相当于一个螺栓，它可以穿过多块木板。

Join point(连接点)

> 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.

程序执行过程中明确的点，如方法的调用或特定的异常被抛出。连接点由两个信息确定：

方法(表示程序执行点，即在哪个目标方法)

相对点(表示方位，即目标方法的什么位置，比如调用前，后等)

通俗的说，joint point就是木板中的孔。

PointCut(切点)

> A predicate that matches join points.

切入点是对连接点进行拦截的条件定义。它和连接点匹配。示例：

```
//表示匹配所有方法
1）execution(* *(..))
//表示匹配com.savage.server.UserService中所有的公有方法
2）execution(public * com. savage.service.UserService.*(..))
//表示匹配com.savage.server包及其子包下的所有方法
3）execution(* com.savage.server..*.*(..))
```

通俗的说，就是下图的螺栓部件。

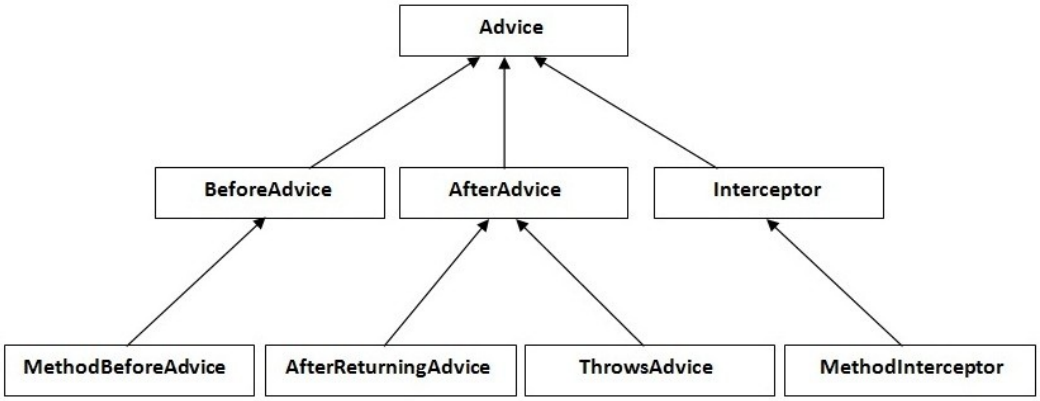
Advice(增强)

> Action taken by an aspect at a particular join point.

Advice是指拦截到连接点之后要执行的代码，包括了“around”、“before”和“after”等不同类型的通知。Spring AOP框架以拦截器来实现通知模型，并维护一个以连接点为中心的拦截器链。

通俗的说，Advice就是螺栓的头，决定了pointcut的位置。around 相当于螺栓的两个头(不一定贴切)。

Spring AOP 包括以下类型的通知：



1. Before advice：在连接点之前运行的通知，但是不能阻止执行流继续到连接点(除非抛出异常)。
2. After returning advice：在连接点正常完成后运行的通知(例如，如果一个方法没有抛出异常返回)。
3. After throwing advice：如果方法通过抛出异常退出，则执行通知。
4. After (finally) advice：不管连接点以何种方式退出(正常或异常返回)，都要执行的通知
5. Around advice：围绕连接点(如方法调用)的通知。这是最有力的建议。Around通知可以在方法调用前后执行自定义行为。它还负责选择是继续到连接点，还是通过返回它自己的返回值或抛出异常来简化建议的方法执行

Target object(目标对象)

> An object being advised by one or more aspects.

目标对象指将要被增强的对象，即包含主业务逻辑的类对象。或者说是被一个或者多个切面所通知的对象。

通俗的讲就是我们下图的木板。

Weaving(织入)

> linking aspects with other application types or objects to create an advised object。

织入是将切面和业务逻辑对象连接起来， 并创建通知代理的过程。织入可以在编译时，类加载时和运行时完成。在编译时进行织入就是静态代理，而在运行时进行织入则是动态代理。

Introduction

> Declaring additional methods or fields on behalf of a type.

静态代理，其它为动态代理。

AOP proxy

> An object created by the AOP framework in order to implement the aspect contracts (advise method executions and so on).

Spring AOP 默认使用标准的 JDK 动态代理（Dynamic Proxy）技术来实现 AOP 代理，通过它，我们可以为任意的接口实现代理。如果需要为一个类实现代理，那么可以使用 CGLIB 代理。当一个业务逻辑对象没有实现接口时，那么Spring AOP 就默认使用 CGLIB 来作为 AOP 代理了。即如果我们需要为一个方法织入 advice，但是这个方法不是一个接口所提供的方法，则此时 Spring AOP 会使用 CGLIB 来实现动态代理。鉴于此，Spring AOP 建议基于接口编程，对接口进行 AOP 而不是类。

总结

Spring AOP 是在纯 Java 中实现的。不需要特殊的编译过程。Spring AOP 不需要控制类装入器层次结构，因此适合在 Servlet 容器或应用服务器中使用。

Spring AOP 目前只支持方法执行连接点（建议在Spring Bean上执行方法）。虽然可以在不破坏核心 Spring AOP API 的情况下添加对字段拦截的支持，但是没有实现字段拦截。如果需要通知字段访问和更新连接点，请考虑 AspectJ 之类的语言。

Spring AOP 的 AOP 方法与大多数其他 AOP 框架不同。

其目的不是提供最完整的 AOP 实现（尽管Spring AOP非常强大）。相反，其目标是提供 AOP 实现和 Spring IoC 之间的紧密集成，以帮助解决企业应用程序中的常见问题。

因此，例如，Spring 框架的 AOP 功能通常与 Spring IoC 容器一起使用。Aspects 是通过使用普通的 bean 定义语法来配置的（尽管这允许强大的“自动代理”功能）。这是与其他 AOP 实现的一个重要区别。使用 Spring AOP 不能轻松或有效地完成某些事情，比如通知非常细粒度的对象(通常是域对象)。在这种情况下，AspectJ 是最佳选择。然而，我们的经验是，Spring AOP 为企业 Java 应用程序中大多数适合 AOP 的问题提供了一个优秀的解决方案。

Spring AOP 从未试图与 AspectJ 竞争来提供全面的 AOP 解决方案。我们认为基于代理的框架（如Spring AOP）和成熟的框架（如AspectJ）都是有价值的，它们是互补的，而不是竞争的。Spring 将 Spring AOP 和 IoC 与 AspectJ 无缝集成，从而在一致的基于 Spring 的应用程序体系结构中支持 AOP 的所有使用。这种集成并不影响 Spring AOP AP I或 AOP Alliance API。Spring AOP 保持向后兼容。

}

38 SpEL总结及常见面试题解析

40 Spring AOP策略模式使用及示例实战