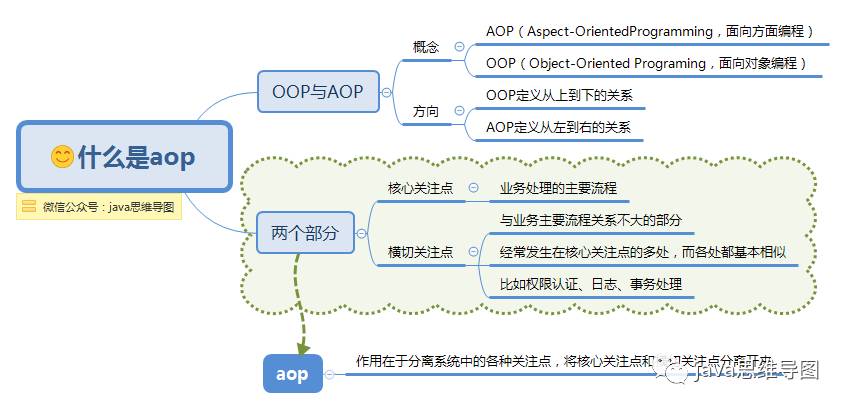
Spring思维导图，让Spring不再难懂（aop篇）

原创 2017-08-07 好玩的 [java思维导图](https://mp.weixin.qq.com/s?__biz=MzI4OTA3NDQ0Nw==&mid=2455544034&idx=1&sn=115144ee04c9184fb7100f339af387e1&chksm=fb9cbc82cceb3594584f938e4121071255cee146fc4828954a054eaa8f9a3d05546c67a1dc1d&scene=0&key=0c5e20dfe5246f3ebf99328499842866871c741cd0ef32aa2cdb7a18e6b8cd4dd4b67ad34632bcb5af113192f179407b78d55f328e304b456a86ea88f6389d0f4e298119e272718de0b3a77c8859cbc3&ascene=14&uin=MjQ1MzQxMjI4NA%3D%3D&devicetype=Windows+7&version=62040549&pass_ticket=PCqZ2NQv5olgssXoxy9dqcygIWu4hP8HJa5N1fMic2%2B63fizseUMJ2oaA%2BADBXBE&winzoom=1##)

什么是aop

AOP（Aspect-OrientedProgramming，面向方面编程），可以说是OOP（Object-Oriented Programing，面向对象编程）的补充和完善。OOP允许你定义从上到下的关系，但并不适合定义从左到右的关系。例如日志功能。日志代码往往水平地散布在所有对象层次中，而与它所散布到的对象的核心功能毫无关系。这种散布在各处的无关的代码被称为横切（cross-cutting）代码，在OOP设计中，它导致了大量代码的重复，而不利于各个模块的重用。

什么是aop.png

而AOP技术则恰恰相反，它利用一种称为“横切”的技术，剖解开封装的对象内部，并将那些影响了多个类的公共行为封装到一个可重用模块，并将其名为“Aspect”，即方面。所谓“方面”，简单地说，就是将那些与业务无关，却为业务模块所共同调用的逻辑或责任封装起来，便于减少系统的重复代码，降低模块间的耦合度，并有利于未来的可操作性和可维护性。

aop使用场景

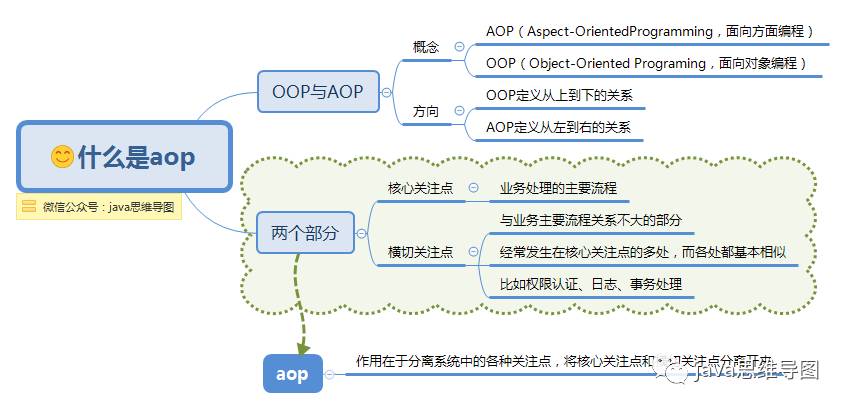
aop框架种类

* AspectJ
* JBoss AOP
* Spring AOP

使用aop可以做的事情有很多。

* 性能监控，在方法调用前后记录调用时间，方法执行太长或超时报警。
* 缓存代理，缓存某方法的返回值，下次执行该方法时，直接从缓存里获取。
* 软件破解，使用AOP修改软件的验证类的判断逻辑。
* 记录日志，在方法执行前后记录系统日志。
* 工作流系统，工作流系统需要将业务代码和流程引擎代码混合在一起执行，那么我们可以使用AOP将其分离，并动态挂接业务。
* 权限验证，方法执行前验证是否有权限执行当前方法，没有则抛出没有权限执行异常，由业务代码捕捉。

*观察一下传统编码方式与使用aop的区别*



3种日志处理.png

核心概念

描述AOP常用的一些术语有通知(Adivce)、切点（Pointcut）、连接点（Join point）、切面（Aspect）、引入（Introduction）、织入（Weaving）、通知（Advice）等。



aop重要概念.png

简单例子

相比xml配置，基于注解的方式更加简洁方便。

@Aspectpublic class TransactionDemo {

@Pointcut(value="execution(\* com.yangxin.core.service.\*.\*.\*(..))") public void point(){

}

@Before(value="point()") public void before(){

System.out.println("transaction begin");

}

@AfterReturning(value = "point()") public void after(){

System.out.println("transaction commit");

}

@Around("point()") public void around(ProceedingJoinPoint joinPoint) throws Throwable{

System.out.println("transaction begin");

joinPoint.proceed();

System.out.println("transaction commit");

}

}

在applicationContext.xml中配置。

<aop:aspectj-autoproxy />

<bean id = "transactionDemo" class = "com.yangxin.core.transaction.TransactionDemo" />

spring aop原理

通过前面介绍可以知道：AOP 代理其实是由 AOP 框架动态生成的一个对象，该对象可作为目标对象使用。AOP 代理包含了目标对象的全部方法，但 AOP 代理中的方法与目标对象的方法存在差异：AOP 方法在特定切入点添加了增强处理，并回调了目标对象的方法。

代理的方法与目标对象的方法.png

Spring 的 AOP 代理由 Spring 的 IoC 容器负责生成、管理，其依赖关系也由 IoC 容器负责管理。因此，AOP 代理可以直接使用容器中的其他 Bean 实例作为目标，这种关系可由 IoC 容器的依赖注入提供。

aop开发时，其中需要程序员参与的只有 3 个部分：

* 定义普通业务组件。
* 定义切入点，一个切入点可能横切多个业务组件。
* 定义增强处理，增强处理就是在 AOP 框架为普通业务组件织入的处理动作。

*为了理清关系，先来个类关系图。*

Spring中主要的AOP组件.png

两种动态代理方式

Spring默认采取的动态代理机制实现AOP，当动态代理不可用时（代理类无接口）会使用CGlib机制。

Spring提供了两种方式来生成代理对象: JDKProxy和Cglib，具体使用哪种方式生成由AopProxyFactory根据AdvisedSupport对象的配置来决定。默认的策略是如果目标类是接口，则使用JDK动态代理技术，否则使用Cglib来生成代理。

*JDK动态代理*

* JDK动态代理主要涉及到java.lang.reflect包中的两个类：Proxy和InvocationHandler。InvocationHandler是一个接口，通过实现该接口定义横切逻辑，并通过反射机制调用目标类的代码，动态将横切逻辑和业务逻辑编制在一起。
* Proxy利用InvocationHandler动态创建一个符合某一接口的实例，生成目标类的代理对象。

*CGLib动态代理*

* CGLib全称为Code Generation Library，是一个强大的高性能，高质量的代码生成类库，可以在运行期扩展Java类与实现Java接口，CGLib封装了asm，可以再运行期动态生成新的class。和JDK动态代理相比较：JDK创建代理有一个限制，就是只能为接口创建代理实例，而对于没有通过接口定义业务方法的类，则可以通过CGLib创建动态代理。

知识拓展

通过上面的分析，大家是否有种熟悉的感觉，似乎和拦截器、过滤器的功能相似。那么问题来了，aop与拦截器、过滤器是什么关系。

先来回顾一下拦截器与过滤器。如下图一网友的测试，在web.xml中注册了TestFilter1和TestFilter2。然后在spring的配置文件中配置了BaseInterceptor和TestInterceptor。得到的结果如下图所示。从图中可以看出，拦截器和过滤器都横切了业务方法，看似符合aop的思想。

拦截器和过滤器.png

Filter过滤器：拦截web访问url地址。  
Interceptor拦截器：拦截以 .action结尾的url，拦截Action的访问。  
Spring AOP拦截器：只能拦截Spring管理Bean的访问（业务层Service）

拦截器与过滤器.png

写在最后

下篇文章将会写Spring cache的内容，同样以思维导图的方式编写。可视化学习，让java不再难懂。

关注java思维导图微信公众号，下载导图源文件，与我共同学习吧。

上篇文章阅读

* [Spring思维导图，让Spring不再难懂（mvc篇）](http://mp.weixin.qq.com/s?__biz=MzI4OTA3NDQ0Nw==&mid=2455544029&idx=1&sn=5d63c342cc82ef8dc87043c8f15b8ea4&chksm=fb9cbcbdcceb35ab4250db25a310c34e8b6bad2794682bb20371c61cad27835994eb4b27e5a8&scene=21#wechat_redirect)
* [Spring思维导图，让Spring不再难懂（ioc篇）](http://mp.weixin.qq.com/s?__biz=MzI4OTA3NDQ0Nw==&mid=2455544007&idx=1&sn=01b6ecc0ab7ba10491bf22000f39dc35&chksm=fb9cbca7cceb35b16e97b0d7b610be68ee515f9a06004e0922e9de4fec0ca773d4f826a73b01&scene=21#wechat_redirect)
* [Spring思维导图，让spring不再难懂（一）](http://mp.weixin.qq.com/s?__biz=MzI4OTA3NDQ0Nw==&mid=2455543996&idx=1&sn=8cd68bda48dc234852fa4199cab06adf&chksm=fb9cbcdccceb35caba863b9cac46bc06fdaabfbbdc980f12d5c7e3ce31a496d13c1d4dcdaba1&scene=21#wechat_redirect)