Atitit .aop 原理理论 架构设计 实现机制 解决方案 理念模式 以及各种语言的实现 java php c#.net js javascript c++ python

[1. 先了解AOP的相关术语:  1](#_Toc6225)

[2. 常见有三种实现方式： 2](#_Toc15556)

[2.1. 1.利用代理模式动态的实现AOP, 2](#_Toc19489)

[2.2. 。采用静态织入的方式，引入特定的语法创建“方面”，@AspectJ注解驱动的切面  2](#_Toc32731)

[2.3. 3.使用自定义加载器的方法动态进行代理。 2](#_Toc30572)

[2.4. Filter 加threadlocal模式 2](#_Toc28652)

[2.5. Plugin设计模式 2](#_Toc25220)

[3. 参考 2](#_Toc18787)

# 先了解AOP的相关术语:

1.通知(Advice):   
通知定义了切面是什么以及何时使用。描述了切面要完成的工作和何时需要执行这个工作。   
2.连接点(Joinpoint):   
程序能够应用通知的一个“时机”，这些“时机”就是连接点，例如方法被调用时、异常被抛出时等等。   
3.切入点(Pointcut)   
通知定义了切面要发生的“故事”和时间，那么切入点就定义了“故事”发生的地点，例如某个类或方法的名称，Spring中允许我们方便的用正则表达式来指定   
4.切面(Aspect)   
通知和切入点共同组成了切面：时间、地点和要发生的“故事”   
5.引入(Introduction)   
引入允许我们向现有的类添加新的方法和属性(Spring提供了一个方法注入的功能）   
6.目标(Target)   
即被通知的对象，如果没有AOP,那么它的逻辑将要交叉别的事务逻辑，有了AOP之后它可以只关注自己要做的事（AOP让他做爱做的事）   
7.代理(proxy)   
应用通知的对象，详细内容参见设计模式里面的代理模式   
8.织入(Weaving)   
把切面应用到目标对象来创建新的代理对象的过程，织入一般发生在如下几个时机:   
(1)编译时：当一个类文件被编译时进行织入，这需要特殊的编译器才可以做的到，例如AspectJ的织入编译器   
(2)类加载时：使用特殊的ClassLoader在目标类被加载到程序之前增强类的字节代码   
(3)运行时：切面在运行的某个时刻被织入,SpringAOP就是以这种方式织入切面的，原理应该是使用了JDK的动态代理技术

# 常见有三种实现方式：

## 1.利用代理模式动态的实现AOP,

从具体的技术细节又可以分为静态代理,动态代理,CGLIB生成子类代理。   
2.使用预编译的方法静态进行代理

## 。采用静态织入的方式，引入特定的语法创建“方面”，@AspectJ注解驱动的切面

从而使得编译器可以在编译期间织入有关“方面”的代码。动态代理实现方式利用.Net的Attribute和.Net Remoting的代理技术，对

## 3.使用自定义加载器的方法动态进行代理。

后两种代理都可以代理更多的内容（比如构造函数，静态方法，静态块，final方法，私有方法等）

## Filter 加threadlocal模式

## Plugin设计模式

# 参考

paip. uapi 过滤器的java php python 实现aop filter - attilax的专栏 - 博客频道 - CSDN.NET.htm

两种AOP实现方式的性能比较 - 小庄 - 博客园.htm

Spring AOP 常用的四种实现方式 - - ITeye技术网站.htm