

【Spring教程】详解AOP的实现原理（动态代理）

置顶 2019-06-29 22:40:44 码农云帆哥 阅读数 174 更多

版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接和本声明。

本文链接：https://blog.csdn.net/sinat_27933301/article/details/94198281

一、简介

AOP是Aspect-Oriented Programming，即面向切面编程。

它是一种新的模块化机制，用来描述分散在对象/类或函数中的横切关注点。分离关注点使解决特定领域问题的代码从业务逻辑中独立出来，业务逻辑不再含有针对特定领域问题代码的调用，业务逻辑同特定领域问题的关系通过切面来封装、维护，这样原本分散在整个应用程序中的变动逻辑，被很好地封装在切面中。

二、名称解释

1、切面（Aspect）

切面由切点和通知组成，它既包括了横切逻辑的定义，也包括了连接点的定义，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指定的连接点中。

2、连接点（Joinpoint）

连接点是在应用执行过程中能够插入切面（Aspect）的一个点。程序执行的某个特定位置：如类开始初始化前、类初始化后、类某个方法调用前、方法抛出异常后。

3、切点（Pointcut）

切点是指通知（Advice）所要织入（Weaving）的具体位置。每个程序类都拥有多个连接点，如一个拥有两个方法的类，这两个方法都是连接点。连接点不是一对一的关系，一个切点可以匹配多个连接点。AOP通过“切点”定位特定的连接点。

具体举个例子：比如开车经过一条高速公路，这条高速公路上有很多个出口（连接点），但是我们不会每个出口都会出去，只会选择我们需要的某个出口开出去。

简单可以理解为，每个出口都是连接点，但是我们使用的那个出口才是切点。每个应用有多个位置适合织入通知，这些位置都是连接点。但是只有具体的位置才是切点。

4、通知（Advice）

它定义在连接点做什么，为切面增强提供织入接口。例如，日志记录、权限验证、事务控制、性能检测、错误信息检测等。

Spring切面可以应用5种类型的通知：

前置通知（Before）：在目标方法被调用之前调用通知功能；

后置通知（After）：在目标方法完成之后调用通知，此时不会关心方法的输出是什么；

返回通知（After-returning）：在目标方法成功执行之后调用通知；

异常通知（After-throwing）：在目标方法抛出异常后调用通知；

环绕通知（Around）：通知包裹了被通知的方法，在被通知的方法调用之前和调用之后执行自定义的行为。

5、通知器（Advisor）

完成对目标方法的切面增强设计和关注点的设计以后，需要一个对象把它们结合起来，完成这个作用的就是Advisor。

6、代理（Proxy）

它为其他对象提供一种代理以控制对这个对象的访问。在某些情况下，一个对象不适合或者不能直接引用另一个对象，而代理对象可以在客户端和起到中介的作用。代理类的对象本身并不真正实现服务，而是通过调用委托类的对象的相关方法，来提供特定的服务。

三、静态代理和动态代理的区别

- 1、静态代理通常只代理一个类，动态代理是代理一个接口下的多个实现类。
- 2、静态代理事先知道要代理的是什么，而动态代理不知道要代理什么东西，只有在运行时才知道。
- 3、静态代理，在程序运行前，代理类的.class文件就已经存在了；动态代理，在程序运行时，运用反射机制动态创建而成。

四、静态代理实例

1、举一个手机缴话费的例子，TelecomOperator 类是服务类。

```
1 package com.service;
2
3 /**
4  * 定义一个电信运营商接口
5  */
6 public interface TelecomOperator {
7     //查询话费余额
8     public void queryPhoneBal();
9
10    //缴话费
11    public void payPhoneBal();
12 }
```

2、TelecomOperatorImpl是实现类

```
1 package com.controller;
2
3 import com.service.TelecomOperator;
4
5 public class TelecomOperatorImpl implements TelecomOperator {
6     //查询话费余额
7     @Override
8     public void queryPhoneBal(){
9         System.out.println("查话费方法...");
10    }
11
12    //缴话费
13    @Override
14    public void payPhoneBal(){
15        System.out.println("缴话费方法...");
16    }
17 }
```

3、TelecomOperatorProxy是服务代理类

```
1 package com.controller;
2
3 import com.service.TelecomOperator;
4
5 /**
6  * 第三方代理商
7  *
8  */
9 public class TelecomOperatorProxy implements TelecomOperator {
10     private TelecomOperatorImpl telecomOperator;
11
12     public TelecomOperatorProxy(TelecomOperatorImpl telecomOperator) {
13         this.telecomOperator = telecomOperator;
14     }
15
16     //查询话费余额
17     @Override
18     public void queryPhoneBal(){
19         System.out.println("切点: 事务控制/日志输出");
20         telecomOperator.queryPhoneBal();
21         System.out.println("切点: 事务控制/日志输出");
22     }
23
24     //缴话费
25     @Override
26     public void payPhoneBal(){
27         System.out.println("切点: 事务控制/日志输出");
28         telecomOperator.payPhoneBal();
29         System.out.println("切点: 事务控制/日志输出");
30     }
31 }
```



2



4、TelecomOperatorTest是测试类

```
1 package com.controller;
2
3 public class TelecomOperatorTest {
4     public static void main(String[] args) {
5         TelecomOperatorImpl telecomOperator = new TelecomOperatorImpl();
6         TelecomOperatorProxy proxy = new TelecomOperatorProxy(telecomOperator);
7         proxy.queryPhoneBal();
8         proxy.payPhoneBal();
9     }
10 }
```

5、控制台输出

```
1 切点: 事务控制/日志输出
2 查话费方法...
3 切点: 事务控制/日志输出
4 切点: 事务控制/日志输出
5 缴话费方法...
6 切点: 事务控制/日志输出
```

五、aop的实现原理（动态代理）

1、JDK动态代理

```
1 package com.controller;
2
3 import java.lang.reflect.InvocationHandler;
4 import java.lang.reflect.Method;
5 import java.lang.reflect.Proxy;
6
7 /**
8  * JDK动态代理类
9  */
10 public class TelecomOperatorJDKProxy implements InvocationHandler {
11     private Object target;
12
13     //返回代理对象
14     public Object newProxy(Object target) {
15         this.target = target;
16         return Proxy.newProxyInstance(target.getClass().getClassLoader(),
17             target.getClass().getInterfaces(), this);
18     }
19
20     /**
21      * @param obj 目标对象代理类的实例
22      * @param method 代理实例上调用父类方法的Method实例
23      * @param args 代入到代理实例上方法参数值的数组
24      */
25     @Override
26     public Object invoke(Object obj, Method method, Object[] args) throws Throwable{
27         Object result = null;
28         System.out.println("切点: 事务控制/日志输出");
29         result=method.invoke(target,args);
30         System.out.println("切点: 事务控制/日志输出");
31         return result;
32     }
33 }
```

```
1 package com.controller;
2
3 import com.service.TelecomOperator;
4
5 public class TelecomOperatorJDKTest {
6     public static void main(String[] args) {
7         TelecomOperatorJDKProxy proxy = new TelecomOperatorJDKProxy();
8         TelecomOperator telecomOperator = (TelecomOperator)proxy.newProxy(new TelecomOperatorImpl());
```



2



```
9         telecomOperator.queryPhoneBal();
10     }
11 }
```

控制台输出：

```
1 切点：事务控制/日志输出
2 查话费方法...
3 切点：事务控制/日志输出
```

2、Cglib动态代理

```
1 package com.controller;
2
3 import org.springframework.cglib.proxy.Enhancer;
4 import org.springframework.cglib.proxy.MethodInterceptor;
5 import org.springframework.cglib.proxy.MethodProxy;
6
7 import java.lang.reflect.Method;
8
9 /**
10  * Cglib动态代理类
11  */
12 public class TelecomOperatorCglibProxy implements MethodInterceptor {
13     private Object target;//代理的目标对象
14
15     //创建目标对象的代理对象
16     public Object newProxy(Object target) {
17         this.target = target;
18         Enhancer enhancer = new Enhancer();//该类用于生成代理对象
19         enhancer.setSuperclass(this.target.getClass());//设置父类
20         enhancer.setCallback(this);//回调方法，设置回调对象为本身
21         return enhancer.create();//创建代理对象
22     }
23
24     /**
25      * @param obj 目标对象代理类的实例（增强过）
26      * @param method 代理实例上调用父类方法的Method实例
27      * @param args 代入到代理实例上方法参数值的数组
28      * @param proxy 使用它调用父类的方法
29      * @throws Throwable
30      */
31     @Override
32     public Object intercept(Object obj, Method method, Object[] args, MethodProxy proxy) throws Throwable {
33         System.out.println("切点：事务控制/日志输出");
34         Object object = proxy.invokeSuper(obj, args);
35         //Object object = proxy.invoke(target,args);
36         System.out.println("切点：事务控制/日志输出");
37         return object;
38     }
39 }

```

```
1 package com.controller;
2
3 public class TelecomOperatorCglibTest {
4     public static void main(String[] args) {
5         TelecomOperatorCglibProxy proxy = new TelecomOperatorCglibProxy();
6         TelecomOperatorImpl telecomOperatorimpl = (TelecomOperatorImpl)proxy.newProxy(new TelecomOperatorImpl());
7         telecomOperatorimpl.queryPhoneBal();
8     }
9 }
```

控制台输出：

```
1 切点：事务控制/日志输出
2 查话费方法...
3 切点：事务控制/日志输出
```



2



invoke方法调用的对象（target）没有增强过，invokeSuper方法调用的对象（obj）已经是增强了的，所以会再走一遍 MyMethod方法，如果是个拦截器链条，就会重新在走一次拦截器链。如果使用invoke(obj,args)就会循环调用，造成死循环，并抛异常java.lang.Ste

2

reporterflow

六、两种动态代理方式区别

- 1、java动态代理是利用反射机制生成一个实现代理接口的匿名类，在调用具体方法前调用InvokeHandler来处理。而Cglib动态代理对对象类的class文件加载进来，通过修改其字节码生成子类来处理。
- 2、JDK动态代理只能对实现了接口的类生成代理，而不能针对类；Cglib是针对类实现代理，主要是对指定的类生成一个子类，覆盖该方法，所以该类或方法最好不要声明成final。
- 3、Cglib一个目标类方法会生成两个代理方法，一个重写目标方法，并实现代理逻辑，还有一个直接调用目标类方法。

有 0 个人打赏

文

布于: 201

Spring AOP底层实现原理（动态代理）

阅读数 28

2019独角兽企业重金招聘Python工程师标准>>>博文 来自： weixin_33717...

想对作者说点什么

Spring AOP的实现原理

阅读数 12

原文出处：ListenAOP（AspectOrientProgramming），我们一般称为面向方面（切面）编程，作为...博文 来自： weixin_30265...

Spring 之 AOP 动态代理实现原理

阅读数 1163

pom.xml4.0.0cn.etSpringAopPrinciple0.0.1-SNAPSHOT org.springframework spring-beans 4.3....博文 来自： phone131448...

细说Spring——AOP详解（动态代理实现AOP）

阅读数 2327

前言嗯，我应该是有一段实现没有写过博客了，在写完了细说Spring——AOP详解（AOP概览）之后...博文 来自： 啦啦啦的博客

Spring AOP 的实现原理

阅读数 742

SpringAOP的实现原理 原理概述:织入的时机1.编译期(AspectJ)2.类加载时(AspectJ5+)3.运行时(Spring...博文 来自： 小炫风技术旅行

探析Spring AOP（三）：Spring AOP的底层实现原理

阅读数 2万+

一、前言 前面第一篇我们讲到了AOP的概念和使用，第二篇也讲到了AOP的实现机制，在第一篇，...博文 来自： 刘剑峰的博客

Spring AOP的实现原理（二）

阅读数 2035

二、AOP的设计与实现1、JVM的动态代理特性在SpringAOP实现中，使用的核心技术时动态代理...博文 来自： Felix_阳的博客

Spring框架IOC和AOP的实现原理

阅读数 2076

Spring框架IOC和AOP的实现原理https://www.cnblogs.com/cyhzuz/p/6644981.htmlSpring面试，I...博文 来自： nanxuan_hen...

Spring AOP实现原理

阅读数 5468

1、SpringAOPspring的面向切面编程，是面向对象编程的一种补充，用于处理系统中分布的各个模块...博文 来自： yanweihipu的...

Spring AOP底层实现原理

阅读数 428

1、spring的AOP底层是由JDK提供的动态代理技术和CGLIB(动态字节码增强技术)实现。2、JDK动态代...博文 来自： 学亮编程手记

【深入理解JVM】GC垃圾回收面试热点 - 码农云帆哥的博客 - CSDN博客

spring AOP的实现原理(动态代理) - jayzym的博客 - CSDN博客

Spring AOP 实现原理

阅读数 2360

SpringAOP属于第二代AOP，采用动态代理机制和字节码生成技术实现。与最初的AspectJ采用...博文 来自： OrPis的专栏

weixin_33717117

4672篇文章

排名:千里之外

关注

weixin_30265171

4474篇文章

排名:千里之外

关注

phone13144830339

309篇文章

排名:千里之外

关注

阿古拉

160篇

排名:千

关注

Spring AOP原理之动态代理 - Tyshawnn的博客 - CSDN博客

https://blog.csdn.net/sinat_27933301/article/details/94198281

5/9

Spring AOP之动态代理原理解析 - 大头哥的哥 - CSDN博客	<div><div></div><div>2</div></div>
Spring AOP的实现原理解析及代码实例 <div>SpringAOP的实现原理解析及实例spring实现AOP是依赖JDK动态代理和CGLIB代理实现的。以下是JD... 博文 来自： 小朱</div>	<div>阅读数 451</div>
Spring中AOP实现的两种方式之JDK和cglib的动态代理 <div>AOP的实现原理：都是基于代理模式，都是生成一个大代理对象静态AOP：AspectJ实现的AOP，将切... 博文 来自： 逍遥飞鹤的专栏</div>	<div>阅读数 5117</div>
Spring AOP概念和原理是两种动态代理的实现方式 - 小驴 - CSDN博客	
SpringAOP动态代理底层源码详解 - LiuY521的博客 - CSDN博客	
Spring Aop详解 <div>Springaop详解springaop的概述aop面向切面（方面）编程，扩展功能不修改源代码的实现。aop采取... 博文 来自： 程序小黑马的...</div>	<div>阅读数 178</div>
重新学习Spring2——IOC和AOP原理彻底搞懂 <div>一、AOP1SpringAOP的实现原理是对OOP编程方式的一种补充。翻译过来为“面向切面编程”。1As... 博文 来自： devlgdg1924...</div>	<div>阅读数 6</div>
Spring AOP的实现原理及应用场景(通过动态代理) - 阿顾..._CSDN博客	
细说Spring——AOP详解(动态代理实现AOP) - l啦啦啦的..._CSDN博客	
Spring 容器AOP的实现原理——动态代理 <div>SpringAOP实现原理解析 博文 来自： qq756161569...</div>	<div>阅读数 303</div>
spring04 spring中aop思想动态代理实现演示 <div>一，spring中的aop思想详解1.思想：横向重复，纵向抽取spring能够为容器中管理对象生成动态代理... 博文 来自： qq_41826183...</div>	<div>阅读数 25</div>
Spring AOP底层实现原理(动态代理) - 寒武没有纪 - CSDN博客	
spring源码剖析（六）AOP实现原理剖析 <div>Spring的AOP实现原理，酝酿了一些日子，写博客之前信心不是很足，所以重新阅读了一边AOP的实现... 博文 来自： Fighter168的...</div>	<div>阅读数 3万+</div>
AOP的实现原理与应用场景 <div>SpringAOP的实现原理与应用场景作用：在不修改源代码的情况下，可以实现功能的增强。传统的纵向... 博文 来自： 阅后即焚</div>	<div>阅读数 418</div>
Spring AOP原理解析以及实现AOP的实现方式 <div>在介绍AOP之前，首先了解一下OOP（面向对象的编程），所谓“对象”就是再面向对象的语言中，一... 博文 来自： “花花”公子...</div>	<div>阅读数 206</div>
程序员实用工具网站 <div>目录1、搜索引擎2、PPT3、图片操作4、文件共享5、应届生招聘6、程序员面试题库7、办公、开发软... 博文 来自： 不脱发的程序猿</div>	<div>阅读数 8万+</div>
我花了一夜用数据结构给女朋友写个H5走迷宫游戏 <div>起因又到深夜了，我按照以往在csdn和公众号写着数据结构！这占用了我大量的时间！我的超越妹妹严... 博文 来自： bigsai</div>	<div>阅读数 3万+</div>
别再翻了，面试二叉树看这 11 个就够了~ <div>写在前边数据结构与算法：不知道你有没有这种困惑，虽然刷了很多算法题，当我去面试的时候，面试... 博文 来自： 一个不甘平凡...</div>	<div>阅读数 1万+</div>
Java 13 来袭，最新最全新特性解读 <div>2017年8月，JCP执行委员会提出将Java的发布频率改为每六个月一次，新的发布周期严格遵循时间点... 博文 来自： HollisChuang'</div>	<div>阅读数 4万+</div>
代码整洁 vs 代码肮脏 <div>写出整洁的代码，是每个程序员的追求。《cleancode》指出，要想写出好的代码，首先得知道什么是... 博文 来自： www.bysocket...</div>	<div>阅读数 8万+</div>
我在快手认识了 4 位工程师，看到了快速发展的公司和员工如何彼此成就！ <div>作者 胡巍巍出品 CSDN（ID：CSDNnews）从西二旗地铁站B口出来，步行700多米可以看到一个工业... 博文 来自： CSDN资讯</div>	<div>阅读数 2万+</div>

让程序员崩溃的瞬间（非程序员勿入）

今天给大家带来点快乐，程序员才能看懂。来源：https://zhuanlan.zhihu.com/p/470665211.公司实... 博文 来自：strongerHuang

MySQL经典面试题

1、MySQL的复制原理以及流程(1)、复制基本原理流程1.主：binlog线程——记录下所有改变了数据库... 博文 来自：Java知音

七个开源的 Spring Boot 前后端分离项目，一定要收藏！

前后端分离已经在慢慢走进各公司的技术栈，根据松哥了解到的消息，不少公司都已经切换到这个技术... 博文 来自：江南一点雨的...

接私活必备的 10 个开源项目！

点击蓝色“GitHubDaily”关注我加个“星标”，每天下午18:35，带你逛GitHub！作者|SevDot来源|h... 博文 来自：GitHubDaily

阿里资深工程师教你如何优化 Java 代码！

作者|王超责编|伍杏玲明代王阳明先生在《传习录》谈为学之道时说：私欲日生，如地上尘，一日不扫... 博文 来自：CSDN资讯

全中！七大初学者易踩的坑！

【CSDN编者按】作为初学者，你是否有犯过不知道有现成的API，而自己重复造轮子呢？本文作者详解... 博文 来自：CSDN资讯

周杰伦新歌《说好不哭》上线，程序员哭了.....

欢迎添加华为云小助手微信（微信号：HWCloud002或HWCloud003），输入关键字“加群”，加入... 博文 来自：华为云官方博客

GitHub开源的10个超棒后台管理面板

目录1、AdminLTE2、vue-Element-Admin3、tabler4、Gentelella5、ng2-admin6、ant-design-pr... 博文 来自：不脱发的程序猿

手绘知识点——指针入门

距离上篇博客整整一个月了，秋招的黄金期，感觉自己的节奏和去年此时的师兄师姐完全不在一个频道... 博文 来自：想不出一个好...

“不给钱就删库”的勒索病毒，程序员该如何防护？

作者|阿木责编|郭芮出品|CSDN（ID：CSDNnews）近期一家名为ProPublica的外媒批露了两家号称专... 博文 来自：CSDN资讯

100 个网络基础知识普及，看完成半个网络高手

欢迎添加华为云小助手微信（微信号：HWCloud002或HWCloud003），输入关键字“加群”，加入... 博文 来自：华为云官方博客

动画：面试如何轻松手写链表？

写在前边暑假参加的第一个公司的就让我手写一个双向链表，并完成插入数据和删除数据的操作。当时... 博文 来自：一个不甘平凡...

栈和队列：面试题（Java）

两个队列实现一个栈使用两个队列完成栈的功能.思路：如上图，入队顺序为：12345，如果要模拟栈的... 博文 来自：博客

Google离开我们快十年了

2010年1月13日，Google离开中国。掐指算来，Google已经离开我们快十年了。2010年是个特殊的年... 博文 来自：阿朱=行业趋势...

c#读蓝牙数据 c# exe反编译成源码 c#流程控制语句 c#调用静态类里的变量 c# csv库 c# int 小端 c# 文件在线管理 c# 精确小数点以为 c#只保留字符串的汉字 .net c# 程序部署

没有更多推荐了，返回首页

©2019 CSDN 皮肤主题: 酷酷鲨 设计师: CSDN官方博客

码农云帆哥

4 粉丝

私信

已关注

博客专家

TA的个人主页 >






原创176

粉丝2451

获赞407

评论526

等级 博客 访问123万+ 排名1440

勋章     

https://blog.csdn.net/sinat_27933301/article/details/94198281

7/9

最新文章

【Spring Boot架构】AOP的两种动态代理（JDK和Cglib）

【Spring Boot架构】JdbcTemplate的使用

【Spring Boot架构】全局异常处理 @ExceptionHandler+@ControllerAdvice的使用

【Spring Boot架构】Controller的使用及获取请求参数的示例

【Spring Boot架构】自定义事件及监听

联系方式

yanyunfan22@qq.com

分类专栏

Java42篇

并发编程艺术6篇

深入理解JVM5篇

Spring Boot架构13篇

Spring教程7篇

展开

热门文章

Tomcat服务器设置用户名和密码
阅读数 43567

Java-驼峰命名与下划线命名互转
阅读数 26008

创建线程的三种方式优缺点
阅读数 25345

【Oracle笔记】一个完美的JDBC连接
Oracle数据库的示例代码
阅读数 20833

线程的生命周期
阅读数 19571

归档

2019年10月4篇

2019年9月4篇

2019年8月1篇

2019年7月9篇

2019年6月3篇

2019年4月4篇

2019年3月8篇

2019年1月4篇

展开

最新评论

新人一看就懂：Dubbo+Zook...

2





















sinat_27933301 : [reply]qq_37217713[/reply]
已发送到你的邮箱。同意，欢迎转载~

新人一看就懂：Dubbo+Zook...
qq_37217713 : 1027700603@qq.com 作者您好，今天刷到这篇文章。受益匪浅哈，想转载- ...

新人一看就懂：Dubbo+Zook...
qq_36803704 : [reply]qq_36803704[/reply] 谢谢楼主

【Spring Boot架构】自定...
sinat_27933301 : [reply]ljfphp[/reply] 互相学习，一起进步，哈哈

【Spring Boot架构】自定...
ljfphp : 哈哈，向云帆哥学习



CSDN学院



CSDN企业招聘

QQ客服

客服论坛

kefu@csdn.net

400-660-0108

工作时间 8:30-22:00

关于我们 招聘 广告服务 网站地图

百度提供站内搜索 京ICP备19004658号

©1999-2019 北京创新乐知网络技术有限公司

网络110报警服务 经营性网站备案信息

北京互联网违法和不良信息举报中心

中国互联网举报中心 家长监护 版权申诉

👍
2

🔗

💬

📄

🔖

📱

<

>

👑

🎧

🛡