Spring：AOP

作者：Emile

原文链接：https://ld246.com/article/1536600089057

来源网站：链滴

摘要

本文内容为我在网上搜集Spring AOP资料的汇总、摘抄。

AOP是一种编程思想，其对不同对象进行了横向的抽象，将不同对象的、和主流程无关的公共逻辑抽出来以方便维护。AOP的实现基础为AOP动态代理，动态代理又可以由JDK动态代理和CGLIB实现。Spring中AOP的编程模型是定义组件、定义切入点、定义增强功能。

关键词：**Spring**，**AOP**

一、AOP简介

1. 什么是AOP

AOP（Aspect Oriented Programming，面向切面编程），横向抽象的一种编程思想，是对OOP（Object Oriented Programming，面向对象编程）的补充和完善。

OOP中的基本三个基本特征：**封装**、**继承**、**多态**，其对象关系的描述是**纵向**的，对于分散在不同对象同一层面的代码却无能为力。比如需要在代码开头和结尾处都打印相同的日志，就没有办法描述这种务了。

AOP则可以解决这个问题，它可以将不同代码同一层次的业务无关的公共逻辑抽象出来，再把代码逻辑**入**到对应的切面点，这样就可以减少重复代码，以及方便维护了。

AOP的应用可以有权限认证、日志、事务等。

2. AOP的一些概念

概念	**解释**
核心关注点	业务处理的主要流程为核心关注点
横切关注点	与业务处理主要流程不相关的部分为切关注点（特征：**分散**）
切面(aspect)	切面为对横切关注点的抽象
连接点(joinpoint)	被拦截的点，在Spring中为拦到的方法
切入点(pointcut)	对连接点进行拦截的定义
通知(adivce)	指拦截到连接点后，要执行的代码可以分为：**前置**、**后置**、**异常**、**最终**、**环绕**
目标对象	代理的目标对象
织入(weave)	将切面应用到目标对象，产生代理象的过程。（即将行为织入到业务流程中）
引入(introduction)	不修改代码，在运行期(runtime)将类动态添加一些方法或字段。

二、Spring中AOP的实现原理

Spring AOP实现的基石在于代理，也就是应用AOP后实际上是生成**AOP代理对象**，Spring中AOP代及其依赖关系由IoC容器负责生成、管理。

代理对象=目标对象+切面处理（通知）

1. 代理

1.1 静态代理

静态代理产生于代码的编译阶段，一旦代码运行就不可改变；

```
// 接口
public interface IPerson {
  public void doSth();
}

// 接口实现
public class Person implements IPerson {
  public void doSth(){
    System.out.println("do sth");
  }
}

// 代理实现
public class PersonProxy {
  private IPerson person;

  public PersonProxy(IPerson person) {
    this.person = person;
  }

  public void doSth() {
    System.out.println("before do sth");
    person.doSth();
    System.out.println("after do sth");
  }
}

public static void main(String[] args) {
  PersonProxy personProxy = new PersonProxy(new Person());
  personProxy.doSth();
}
/*
result console:
before do sth
do sth
after do sth
*/
```

1.2 动态代理

这里特指JDK动态代理，区别与静态代理，动态代理产生于代码的运行时阶段。JDK动态代理利用了Ja

a的反射机制，生成一个实现代理接口的类，调用具体方法由InvocationHandler来处理。这种动态代需要依赖接口来实现。

```
public class PersonProxy implements InvocationHandler {
  private Object delegate;

  public Object bind(Object delegate) {
    this.delegate = delegate;
    // 指明类加载器、指明生成哪个对象的代理对象（接口指定）、指明Handler
    return Proxy.newProxyInstance(delegate.getClass().getClassLoader(), delegate.getClass().get
nterfaces(), this);
  }

  public Object invoke(Object proxy, Method method, Object[] args) throws Throwable {
    Object result = null;
    try {
      System.out.println("before do sth");
      result = method.invoke(delegate, args);
      System.out.println("after do sth");
    } catch(Exception e) {
      throw e;
    }
    return result;
  }
}

public static void main(String[] args) {
  PersonProxy personProxy = new PersonProxy();
  IPerson iPerson = (IPerson) personProxy.bind(new Person());
  iPerson.doSth();
}
/*
result console:
before do sth
do sth
after do sth
*/
```

1.3 CGLIB

如果没有接口，动态代理可以借助CGLIB来实现，它可以用生成代理目标子类（因此**不能代理final修的类**）的方式来实现代理，而不是接口，规避来接口的局限性。

实现MethodInterceptor接口来实现。

```
public class PersonProxy implements MethodInterceptor {
  private Object delegate;
  public Object intercept(Object proxy, Method method, Object[] args, MethodProxy methodP
oxy) throws Throwable {
    System.out.println("before do sth");
    Object result = methodProxy.invokeSuper(method, args);
    System.out.println("after do sth");
    return result;
  }
```

```
  public static Person getProxyInstance() {
    Enhancer enhancer = new Enhancer();
    enhancer.setSuperclass(Person.class);
    enhancer.setCallback(new PersonProxy());
    return (Person) enhancer.create();
  }
}

public static void main(String[] args) {
  Person person = PersonProxy.getProxyInstance();
  person.doSth();
}
/*
result console:
before do sth
do sth
after do sth
*/
```

2. Spring AOP代理创建规则

Spring提供了两种方式来生成代理对象：**JDKProxy**和**CGLIB**

使用方式由AopProxyFactory根据AdvisedSupport对象配置来决定。

代理默认创建规则：

1. 如果目标类是接口，则使用Java动态代理创建AOP代理；
2. 如果目标类不是接口，则使用Cglib生成代理；

三、在Spring中使用AOP

AOP编程步骤：

1. 定义普通业务组件；
2. 定义切入点（切入点和业务组件的对应关系可以是一对多）；
3. 定义增强处理，即为普通业务组件织入的动作；

1. 使用XML文件配置AOP

参考：Spring 中基于 AOP 的 XML架构

2. 使用注解配置AOP

参考：Spring 中基于 AOP 的 @AspectJ

四、总结

1. AOP（Aspect Oriented Programming，面向切面编程），横向抽象的一种编程思想，是对OOP Object Oriented Programming，面向对象编程）的补充和完善。

2. Spring AOP实现的基石在于代理，也就是应用AOP后实际上是生成 **AOP代理对象**，Spring中AP代理及其依赖关系由IoC容器负责生成、管理。

3. Spring中AOP的编程模型是定义组件、定义切入点、定义增强功能。

参考资料

1. Spring3：AOP
2. Spring AOP原理分析一次看懂
3. Spring-aop 全面解析（从应用到原理）
4. Spring的两种代理JDK和CGLIB的区别浅谈
5. Spring 教程