

# 12 Spring\_AOP\_铺垫

Spring

## 一：什么是AOP

- 1、AOP为Aspect Oriented Programming的缩写，意为：面向切面编程，通过预编译方式和运行期动态代理实现程序功能的统一维护的一种技术
- 2、利用AOP可以对业务逻辑的各个部分进行隔离，从而使得业务逻辑各部分之间的<耦合度降低>，提高程序的可重用性，同时提高了开发的效率



- 3、横向重复,纵向抽取

## 二：为什么要学AOP

- 1、对程序进行增强:不修改源码的情况下.
  - \* AOP 可以进行权限校验,日志记录,性能监控,事务控制.
- 2、Spring帮忙我们生成动态代理对象

## 三：动态代理(优先)

- 1、被代理对象必须要实现接口,才能产生代理对象.如果没有接口将不能使用动态代理技

## 1、创建接口

```
1.  public interface IUserService {
2.      void save() throws Exception;
3.      void del() throws Exception;
4.      void upde() throws Exception;
5.      void add() throws Exception;
6.  }
```

## 2、实现类

```
1.  public class UserServiceImp implements IUserService {
2.      @Override
3.      public void save() throws Exception {
4.          System.out.println("save:查询用户");
5.      }
6.      //其他方法同样
7.  }
```

## 3、创建动态代理工厂

```
1.  import java.lang.reflect.InvocationHandler;
2.  import java.lang.reflect.Method;
3.  import java.lang.reflect.Proxy;
4.
5.  import com.spring.service.IUserService;
6.  import com.spring.service.impl.UserServiceImp;
7.
8.  public class UsersServiceProxyFactory implements InvocationHandler {
9.
10.     private IUserService ius;
11.
12.     public UsersServiceProxyFactory(IUserService ius) {
13.         super();
14.         ius = ius;
15.     }
16. }
```

```

17.     public IUserService getUsersServiceProxyFactory() {
18.         // 生成动态代理
19.         IUserService userService = (IUserService) Proxy.newProxyInstance(
20.             UsersServiceProxyFactory.class.getClassLoader(), UserSe
21.             rviceImp.class.getInterfaces(), this);
22.         // 返回对象
23.         return userService;
24.     }
25.     @Override
26.     public Object invoke(Object proxy, Method method, Object[] args) th
27.     rows Throwable {
28.         // 业务方法之前
29.         System.out.println("====开启事务");
30.         Object invoke = method.invoke(Ius, args);
31.         // 业务方法之后
32.         System.out.println("关闭事务====");
33.         return invoke;
34.     }

```

## 4、测试

```

1.     @Test
2.     public void getVoid() throws Exception{
3.         //获取接口对象
4.         IUserService userService=new UserServiceImp();
5.         //获取动态代理工厂
6.         UsersServiceProxyFactory proxyFactory = new
7.         UsersServiceProxyFactory(userService);
8.         //获取代理对象
9.         IUserService serviceProxyFactory = proxyFactory.getUsersService
10.        ProxyFactory();
11.        serviceProxyFactory.add();
12.        //代理对象与被代理对象实现相同的接口,
13.        //代理对象与被代理对象没有继承关系
14.        System.out.println(serviceProxyFactory instanceof UserService
15.        Imp);
16.    }

```

<terminated> TestProxy.getVoid [JUnit] D:\JDK\jdk-8u101-windows-x64\jdk\bin\javaw.exe (2018年8月15日 上午4:32:5

=====开启事务

**add:**添加用户

关闭事务=====

**false**

## 四：cglib代理(没有接口)

1、第三方代理技术,cglib代理.可以对任何类生成代理.代理的原理是对目标对象进行继承代理. >2、如果目标对象被final修饰.那么该类无法被cglib代理.

### 1、实现类同上

### 2、创建cglib代理工厂

```
1.  import java.lang.reflect.Method;
2.  import org.springframework.cglib.proxy.Enhancer;
3.  import org.springframework.cglib.proxy.MethodInterceptor;
4.  import org.springframework.cglib.proxy.MethodProxy;
5.  import com.spring.serivce.IUserSerivce;
6.  import com.spring.serivce.impl.UserSerivceImp;
7.
8.  public class CglibProxyFactory implements MethodInterceptor {
9.      public IUserSerivce getCglibProxyFactory() {
10.         // 帮我生成代理对象
11.         Enhancer enhancer = new Enhancer();
12.         // 设置对谁进行代理
13.         enhancer.setSuperclass(UserSerivceImp.class);
14.         // 代理要做什么
15.         enhancer.setCallback(this);
16.         IUserSerivce create = (IUserSerivce) enhancer.create();
17.         return create;
18.     }
```

```

19.
20.     @Override
21.     public Object intercept(Object obj, Method method, Object[] arg, Me
thodProxy methodProxy) throws Throwable {
22.         System.out.println("-----开启事务");
23.         // 调用原有方法
24.         Object invokeSuper = methodProxy.invokeSuper(obj, arg);
25.         System.out.println("提交事务-----");
26.
27.         return invokeSuper;
28.     }
29. }

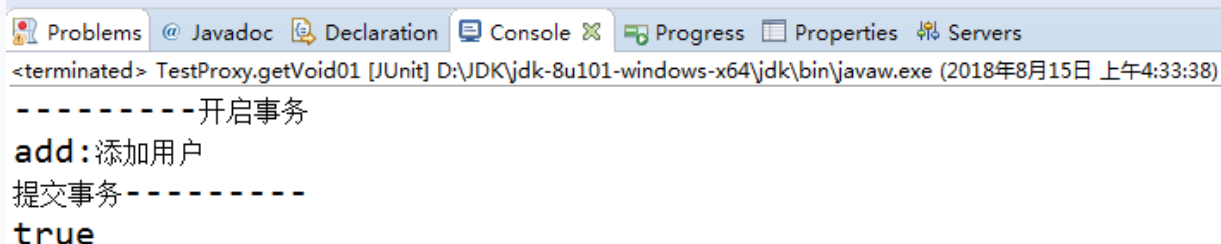
```

### 3、测试

```

1.     @Test
2.     public void getVoid01() throws Exception{
3.         //获取接口对象
4.         IUserService userService=new UserServiceImp();
5.         //获取动态代理工厂
6.         CglibProxyFactory proxyFactory = new CglibProxyFactory();
7.         //获取代理对象
8.         IUserService serviceProxyFactory = proxyFactory.getCglibProxyFa
ctory();
9.         serviceProxyFactory.add();
10.        //判断代理对象是否属于被代理对象类型
11.        //代理对象继承了被代理对象 true
12.        System.out.println(serviceProxyFactory instanceof UserService
Imp);
13.    }

```



```

<terminated> TestProxy.getVoid01 [JUnit] D:\JDK\jdk-8u101-windows-x64\jdk\bin\javaw.exe (2018年8月15日 上午4:33:38)
-----开启事务
add:添加用户
提交事务-----
true

```

