链滴

java 代理

作者：terry
原文链接：https://ld246.com/article/1483517610002
来源网站：链滴

java代理分为2种

第一种是基于子类的坊间用的cglib，第二种是基于接口的官方的

一、基于子类代理

子类

```
public class serviceImpl{
    public void save() {}
}
```

代理类

```
import java.lang.reflect.Method;
import net.sf.cglib.proxy.Enhancer;
import net.sf.cglib.proxy.MethodInterceptor;
import net.sf.cglib.proxy.MethodProxy;
public class cglibProxyFactory implements MethodInterceptor {
  //目标对象
  private Object target;
  //构造方法注入目标对象
  public cglibProxyFactory(Object target) {
    super();
    this.target = target;
  }
    //创建代理方法
    public Object createCgligProxyFactory(){
    //建立Enhancer对象
    Enhancer enhancer = new Enhancer();
    //对目标对象建立子类
    enhancer.setSuperclass(target.getClass());
    //传入CallBack,对目标增强
    enhancer.setCallback(this);
    //建立代理对象
    return enhancer.create();
  }
  public Object intercept(Object proxy, Method method, Object[] args, MethodProxy father) th
ows Throwable {
    System.out.println("代理前");
    //Object obj = method.invoke(target, args);
    Object obj = father.invokeSuper(proxy, args);//执行父类的方法
    System.out.println("代理后");
    return obj;
  }
}
```

测试类

```
import org.junit.Test;
public class cglibProxyTest {
  @Test
  public void ProxyTest(){
    serviceImpl s = new serviceImpl();
```

```
    serviceImpl sp = (serviceImpl) new cglibProxyFactory(s).createCgligProxyFactory();
    sp.save();
  }
}
```

二、基于jdk接口代理

接口及接口实现类

```
public interface Service {
  public void save();
}
public class serviceImpl implements Service {
  public void save() {
  }
}
```

代理类

```
import java.lang.reflect.InvocationHandler;
import java.lang.reflect.Method;
import java.lang.reflect.Proxy;
public class jdkproxyFactory implements InvocationHandler{
  //目标的对象
  private Object target;

  //注入目标对象
  public jdkproxyFactory(Object target) {
    super();
    this.target = target;
  }
  //建立代理
  public Object createProxy(){
  //3个参数，类加载器，接口，invocationhandler
    return Proxy.newProxyInstance(target.getClass().getClassLoader(),
      target.getClass().getInterfaces(),this);
  }

  public Object invoke(Object proxy, Method method, Object[] args) throws Throwable {
    System.out.println("代理前");
    Object obj = method.invoke(target, args);
    System.out.println("代理后");
    return obj;
  }
}
```

测试类

```
import org.junit.Test;
public class jdkProxyTest {
  @Test
  public void ProxyTest(){
    Service s = new serviceImpl();
    Service sp = (Service)new jdkproxyFactory(s).createProxy();
```

```
    sp.save();
  }
}
```

以上就是JAVA2种代理的实现方法，本人在工作中接口代理用的居多，子类代理用的比较少。