Java 动态代理模式InvocationHandler

在Java中动态代理依赖反射机制，两种重要的类为InvocationHandler和Proxy。每个动态代理类必须实现InvocationHandler接口，而且代理类的实例都关联到一个Handler，通过代理对象调用一个方法的时候，这个方法的调用会被转发为由InvocationHandler这个接口的invoke方法调用。

# 1.示例

1. 定义接口和实现类

接口Subject，方法rent和hello

*public interface Subject {*

*public void rent();*

*public void hello(String str);*

*}*

定义实现类：

*public class RealSubject implements Subject {*

*@Override*

*public void rent() {*

*System.out.println("I want to rent a house");*

*}*

*@Override*

*public void hello(String str) {*

*System.out.println("Hello: " + str);*

*}*

*}*

1. 实现动态代理类，DynamicProxy

*public class DynamicProxy implements InvocationHandler {*

*private Object subject;*

*public DynamicProxy(Object subject ) {*

*this.subject = subject;*

*}*

*@Override*

*public Object invoke(Object proxy, Method method, Object[] args) throws Throwable {*

*System.out.println("before rent house");*

*System.out.println("Method:" + method);*

*method.invoke(subject, args);*

*System.out.println("after rent house");*

*return null;*

*}*

*}*

3) 测试程序

*public class Client {*

*public static void main(String[] args) {*

*Subject realSubject = new RealSubject();*

*InvocationHandler handler = new DynamicProxy(realSubject);*

*Subject subject = (Subject) Proxy.newProxyInstance(*

*handler.getClass().getClassLoader(),*

*realSubject.getClass().getInterfaces(),*

*handler);*

*System.out.println(subject.getClass().getName());*

*subject.rent();*

*subject.hello("world");*

*}*

*}*

输出如下：

*com.sun.proxy.$Proxy0*

*before rent house*

*Method:public abstract void com.fys.proxy.Subject.rent()*

*I want to rent a house*

*after rent house*

*before rent house*

*Method:public abstract void com.fys.proxy.Subject.hello(java.lang.String)*

*Hello: world*

*after rent house*

# 2.详解

代理是通过Proxy 对象来调用实体的方法，Java的反射lib中提供了两个类Proxy和InvocationHandler，通过这两个类可以实现一个动态代理类。Proxy类可以创建给定参数的动态类，InvocationHandler调用动态代理类的方法。

1. Proxy

java.lang.reflect.Proxy提供创建动态代理类的静态方法，方法newProxyInstance，定义如下：

*/\*\**

*\* Returns an instance of a proxy class for the specified interfaces*

*\* that dispatches method invocations to the specified invocation*

*\* handler.\*/*

*public static Object newProxyInstance(ClassLoader loader,*

*Class<?>[] interfaces,*

*InvocationHandler h)*

*throws IllegalArgumentException*

* ClassLoader，用于定义dynamic proxy class
* Interfaces，动态代理类的实现的接口
* InvocationHandler，将执行类传递给InvocationHandler

对于动态代理类：

* 类属性为public和final，不能是抽象类
* 每个类继承java.lang.reflect.Proxy
* 如果有non-public接口，那么这个接口必须在同一个包中
* 每个代理类有一个参数的public 构造器，通过这个构造器设置InvocationHandler
* 代理类与InvocationHandler相关

1. InvocationHandler

用户类实现InvocationHandler，调用动态代理类中的方法，语法如下：

*Object invoke(Object proxy, Method m, Object[] args)*

参数如下：

* Object，被调用的代理类
* Method，被调用代理类中的接口方法
* Object[]，传递的参数

https://www.concretepage.com/java/dynamic-proxy-with-proxy-and-invocationhandler-in-java