链滴

设计模式之动态代理模式

作者：sologxl

原文链接：https://ld246.com/article/1543300124701

来源网站：链滴

看到标题，你肯定会问我有没有静态代理模式呢。

是滴，有，但是我就是不说，因为我觉得静态代理没啥可讲的。你们还是自己百度看看静态代理吧，啥东西可讲的。

动态，啥是动态呢，就是会动的状态，哈哈，是不是想打死我呢？没有办法，我就是这么强大。

就喜欢你看我不爽又打不到我的样子。

动态代理：

业务接口：

```
public interface Service { //动态代理必须要接口，不然就GG了
    //目标方法
   public abstract void add();
}
```

业务实现：

```
public class UserServiceImpl implements Service {
    public void add() {
    System.out.println("This is add service");
    }
}
```

代理模式主要代码（不用客气，你可以直接拷贝过去使用）：

```
import  java.lang.reflect.InvocationHandler;
import java.lang.reflect.Method;
import java.lang.reflect.Proxy;

public class MyInvocatioHandler implements InvocationHandler {
private Object target;

public MyInvocatioHandler(Object target) {
    this.target = target;
}

@Override
 public Object invoke(Object proxy, Method method, Object[] args) throws Throwable {
     System.out.println("-----before-----");
       //此处你可以添加任何你需要的业务操作：比如写日志
     Object result = method.invoke(target, args);
       //此处你可以添加任何你需要的业务操作：比如写日志
     System.out.println("-----end-----");
     return result;
}
// 生成代理对象
 public Object getProxy() {
    ClassLoader loader = Thread.currentThread().getContextClassLoader();
    Class[] interfaces = target.getClass().getInterfaces();
    return Proxy.newProxyInstance(loader, interfaces, this);
}
}
```

测试一下：

```
public class ProxyTest {
public static void main(String[] args) {
    Service service = new UserServiceImpl();//创建业务对象实例；
    MyInvocatioHandler handler = new MyInvocatioHandler(service);//将对象实例传入“调用操
类”
    Service serviceProxy = (Service)handler.getProxy();//从“调用操作类”中获取业务代理对象，
记，业务必须要有接口
    serviceProxy.add();//代理调用业务方法，此时会调用MyInvocatioHandler中的invoke，然后inv
ke再调用add，所以一般这种模式可以应用于写日志等附加操作。
}
}
```

ok，动态代理就这样告一段落了，代码拿去用吧，不客气。