Java基础加强-动态代理

# 代理模式介绍

即Proxy Pattern，23种常用的面向对象软件的设计模式之一

代理模式的定义：为其他对象提供一种[代理](http://baike.baidu.com/view/24001.htm)以控制对这个对象的访问。在某些情况下，一个对象不适合或者不能直接引用另一个对象，而代理对象可以在客户端和目标对象之间起到中介的作用。

## 组成

抽象角色：通过接口或抽象类声明真实角色实现的业务方法。

代理角色：实现抽象角色，是真实角色的代理，通过真实角色的业务逻辑方法来实现抽象方法，并可以附加自己的操作。

真实角色：实现抽象角色，定义真实角色所要实现的业务逻辑，供代理角色调用。

## 分类

### 静态代理

是由程序员创建或工具生成代理类的源码，再编译代理类。所谓静态也就是在程序运行前就已经存在代理类的字节码文件，代理类和委托类的关系在运行前就确定了。

### 动态代理

是在实现阶段不用关心代理类，而在运行阶段才指定哪一个对象。

## 优点

(1).职责清晰

真实的角色就是实现实际的[业务逻辑](http://baike.baidu.com/view/1642754.htm)，不用关心其他非本职责的事务，通过后期的代理完成一件完成事务，附带的结果就是编程简洁清晰。

(2).代理对象可以在客户端和目标对象之间起到中介的作用，这样起到了中介的作用和保护了目标对象的作用。

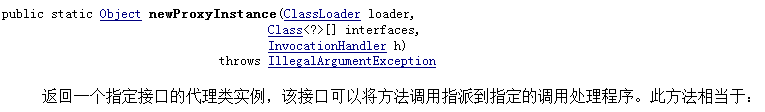
(3).高扩展性

# Java中动态代理

在jdk的api中提供了java.lang.reflect.Proxy它可以帮助我们完成动态代理创建

注意：在java中使用Proxy来完成动态代理对象的创建，只能为目标实现了接口的类创建代理对象。

动态代理是在内存中直接生成代理对象。



通过这个方法可以直接创建一个代理对象。

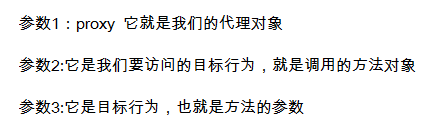


InvocationHandler详解

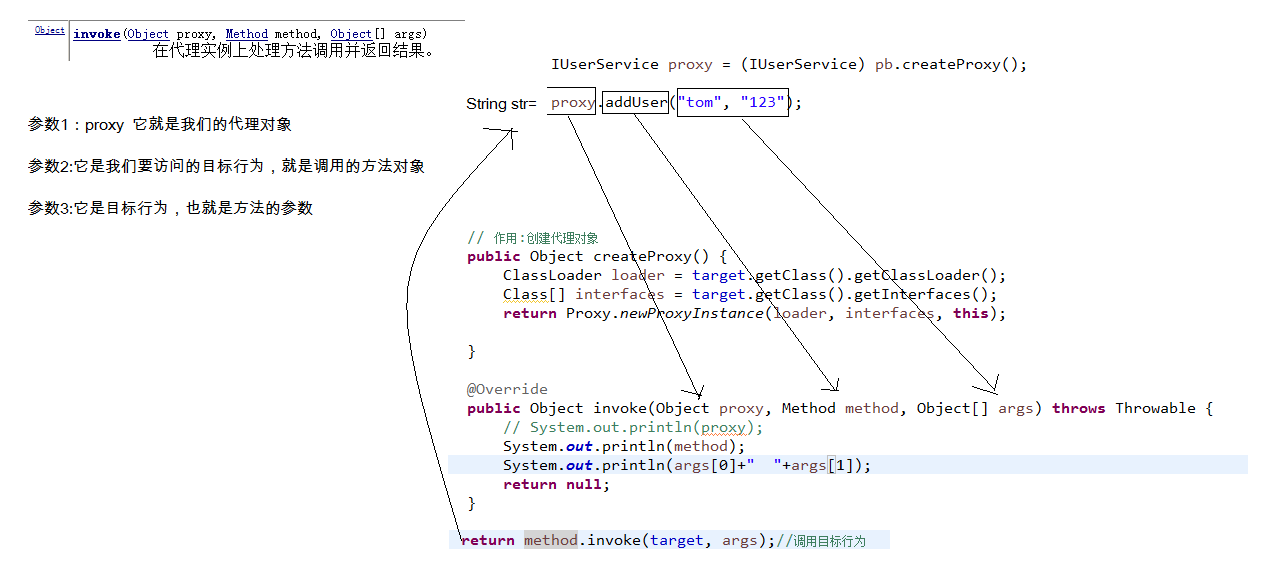
它是一个接口，接口中声明了一个方法



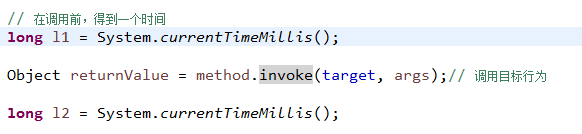
Invoke方法，它是在代理对象调用行为时，会执行的方法，而invoke方法上有三个参数



这个方法的主要作用是，当我们通过代理对象调用行为时，来控制目标行为是否可以被调用。



案例:测试addUser方法的运行时间



在开发中，我们使用动态代理可以完成性能监控，权限控制，日志记录等操作s