29 反射_动态代理

JAVAEE高级

一：动态代理描述

> 1、动态代理是指客户通过代理类来调用其它对象的方法，并且是在程序运行时根据需要动态创建目标类的代理对象。
>
> 2、代理设计模式的原理:
> ①.使用一个代理将对象包装起来, 然后用该代理对象取代原始对象
> ②.任何对原始对象的调用都要通过代理
> ③.代理对象决定是否以及何时将方法调用转到原始对象上

1、Proxy ：

> 1、专门完成代理的操作类，是所有动态代理类的父类
>
> 2、通过此类为一个或多个接口动态地生成实现类

2、常用方法

> 1、提供用于创建动态代理类和动态代理对象的静态方法

```
1.
2.  static Class<?>   getProxyClass(ClassLoader loader, Class<?
    >... interfaces)
3.  //直接创建一个动态代理对象
4.  static Object   newProxyInstance(ClassLoader loader, Class<?>
    [] interfaces, InvocationHandler h)  直接创建一个动态代理对象
```

二：动态代理实现

> 1、创建一个接口

```
public interface Student {
    public void login();
    public void submit();
}
```

> 2、创建一个接口的实现类

```
public class StudentImp implements Student {
    @Override
    public void login() {
        System.out.println("登录");
    }

    @Override
    public void submit() {
        System.out.println("提交");
    }
}
```

> 3、创建有个实现接口InvocationHandler的类
> 必须实现invoke方法，以完成代理的具体操作

```
public class MyInvocationHandler implements InvocationHandler {
    private Object target;
    public MyInvocationHandler(Object target) {
        this.target = target;
    }
    @Override
    public Object invoke(Object proxy, Method method, Object[] args) th
rows Throwable {
        System.out.println("权限校验");
        method.invoke(target, args); // 执行被代理target对象的方法
        System.out.println("日志记录");
        return null;
```

```
    }
}
```

4、测试

```
    public static void main(String[] args) {
        StudentImp si = new StudentImp();
        si.login();
        si.submit();
        System.out.println("--------------------------------");
        MyInvocationHandler m = new MyInvocationHandler(si);
        Student s = (Student)Proxy.newProxyInstance(si.getClass().getCl
assLoader(), si.getClass().getInterfaces(), m);
        s.login();
        s.submit();
    }
```