动态代理

【案例】首先来看看如何获得类加载器：

```
class Test {
    public static void main(String[] args) {
        ClassLoader loader = Test.class.getClassLoader();
        while (loader != null) {
            System.out.println(loader.toString());
            loader = loader.getParent();
        }
    }
}
//~out: sun.misc.Launcher$AppClassLoader@3feba861
        sun.misc.Launcher$ExtClassLoader@2f4d3709
```

在java中有三种类类加载器

1）Bootstrap ClassLoader 此加载器采用c++编写，一般开发中很少见。//不会输出

2）Extension ClassLoader 用来进行扩展类的加载，一般对应的是jre\lib\ext目录中的类。

3）AppClassLoader 加载classpath指定的类，是最常用的加载器。同时也是java中默认的加载器。

如果想要完成动态代理，首先需要定义一个InvocationHandler接口的子类，已完成代理的具体操作。

```
package Reflect;
import java.lang.reflect.*;

//定义项目接口
interface Subject {
    public String say(String name, int age);
}

// 定义真实项目
class RealSubject implements Subject {
    @Override
    public String say(String name, int age) {
        return name + "  " + age;
    }
}

class MyInvocationHandler implements InvocationHandler {
    private Object obj = null;
    public Object bind(Object obj) {
        this.obj = obj;
```

```
        return Proxy.newProxyInstance(obj.getClass().getClassLoader(),
                obj.getClass().getInterfaces(), this);
    }

    @Override
    public Object invoke(Object proxy, Method method, Object[] args)
            throws Throwable {
        Object temp = method.invoke(this.obj, args);
        return temp;
    }
}

class hello {
    public static void main(String[] args) {
        MyInvocationHandler demo = new MyInvocationHandler();
        Subject sub = (Subject) demo.bind(new RealSubject());
        String info = sub.say("Rollen", 20);
        System.out.println(info);
    }
}
```

【运行结果】：

Rollen 20