

● Java Reflection

Reflection（反射）是被视为动态语言的关键，反射机制允许程序在执行期借助于Reflection API取得任何类的内部信息，并能直接操作任意对象的内部属性及方法

● Java反射机制提供的功能

- 在运行时判断任意一个对象所属的类
- 在运行时构造任意一个类的对象
- 在运行时判断任意一个类所具有的成员变量和方法
- 在运行时调用任意一个对象的成员变量和方法
- 生成动态代理

anana.atguigu.com

Java反射机制研究及应用

● 反射相关的主要API:

- java.lang.Class:代表一个类
- java.lang.reflect.Method:代表类的方法
- java.lang.reflect.Field:代表类的成员变量
- java.lang.reflect.Constructor:代表类的构造方法
- ...

在有反射以前，如果创建一个类的对象，并调用其中的方法和属性

实例:

```
package thread;

import java.lang.reflect.Field;
import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;

import org.junit.Test;

public class testfanshe {
```

```

@Test
    public void test2() throws NoSuchFieldException, SecurityException,
InstantiationException, IllegalAccessException, NoSuchMethodException,
IllegalArgumentException, InvocationTargetException{
    //有了反射，可以通过反射创建一个类的对象，并调用其中的结构
    //1, 创建class1对应的运行时类fanshe类的对象
    Class class1 = fanshe.class;
    System.out.println("反射调用属性：");
        fanshe fanshe = (fanshe)class1.newInstance();
        System.out.println(fanshe);
        Field field = class1.getDeclaredField("nameString");
        field.set(fanshe, "linmingjun");
        System.out.println(fanshe);
        Field field2 = class1.getDeclaredField("age");
        field2.setAccessible(true);
        field2.set(fanshe, 88);
        System.out.println(fanshe);
        System.out.println("反射调用方法：");
        Method m1 = class1.getMethod("show", String.class);
        m1.invoke(fanshe, "美国");

    }

@Test
public void test1(){
    //在有反射之前，如何创建一个对象，并调用其中的方法和属性
    fanshe fanshe = new fanshe("anqili", 21, "信科16-2");
    fanshe.setAge(32);
    System.out.println(fanshe.getAge());
    System.out.println(fanshe);
}
}

```