1. 反射调用成员

·获取成员

类结构之中最后一个核心组成就是成员（Field），大部分情况下都会将其称为成员属性，对于成员信息的获取也是通过Class类完成，在这个类之中提供有两组操作方法:

|  |  |
| --- | --- |
| 获取本类全部成员 | public Field getDeclaredField​(String name)  throws NoSuchFieldException,SecurityException |
| 获取本类指定成员 | public Field[] getDeclaredFields() throws SecurityException |
| 获取父类指定成员 | public Field getField​(String name)  throws NoSuchFieldException,SecurityException |
| 获取父类全部成员 | public Field[] getFields()  throws SecurityException |

范例:

|  |
| --- |
|  |

但是在Field类中最重要的不是获取成员，而是如下三个方法:

·设计属性内容：public void set​(Object obj,Object value)

throws IllegalArgumentException,IllegalAccessException

·获取属性内容： public Object get​(Object obj)

throws IllegalArgumentException,IllegalAccessException

所有成员是在实例化对象之后分配空间，所以此时一定要先有实例化对象之后才可以进行成员的操作。

**·解除封装: public void setAccessible​(boolean flag)**

在我们的类中大部分属性是已经封装了（使用private进行了封装）如果直接使用上面两个操作则会报IllegalAccessException错误。（同样构造方法，和普通方法也可以解除封装）

范例:直接调用Person类之中的name私有成员

|  |
| --- |
| **package com.company.反射机制;  import java.lang.reflect.Field;  */\*\*  \* 通过反射获取实例化对象的成员  \* @author john  \*/* public class JavaDemo6 {  public static void main(String[] args) throws Exception {  Class<?> aClass = Class.*forName*("com.company.反射机制.entity.Person");  //实例化对象  Object instance = aClass.getDeclaredConstructor().newInstance();  //获取成员变量  Field field = aClass.getDeclaredField("name") ;  //解除成员封装  field.setAccessible(true);  //设置成员变量  field.set(instance,"我是超人");  //获取成员  System.*out*.println(field.get(instance));  } }** |

通过一系列分析我们可以发现，类之中的构造、方法、成员属性都可以通过反射进行调用，但是对于成员的反射调用我们很少直接这样处理，大部分操作应该通过setter，getter完成，所以上面的代码只能说明反射的特点不能作为实际开发。而实际中Field只有一个方法最为常用也就是获取成员类型:

·获取成员类型：public Class<?> getType()；

范例:获取Person类中的name的类型

|  |
| --- |
|  |

