Java反射

概念

1.反射机制是在运行状态中，对于任意一个类，都能够知道这个类的所有属性和方法。

2.反射可以在一个类运行的时候获取类的信息的机制，可以获取在编译期不可能获得的类的信息。

3.对于任意一个对象，都能调用它的任意一个方法和属性。

4.因为类的信息是保存在Class对象中的，而这个Class对象是在程序运行时被类加载器（ClassLoader）动态加载的。

5.当类加载器装载运行了类后，动态获取Class对象的信息以及动态操作Class对象的属性和方法的功能称为Java语音的反射机制。

作用

1.反编译：.class —> .java。

2.通过反射机制访问Java对象中的属性、方法、构造方法等。

涉及到的类

1.Class —— 类的创建；

- 1.获取Class对象的方法
  - 1.Class c1 = Class.forName("com.mxm.Reflect");
    会让ClassLoader装载类，并进行
  - 2.Class c2 = Reflect.class;
    返回类对象运行时真正所指的对象、所属类型
  - 3.Class c3 = new Reflect().getClass();
    ClassLoader装载入内存，不对类
- 2.无参数创建对象
  - Class c4 = Class。forName("com.mxm.Reflect");
    Object o = c4.newInstance();
    newInstance式使用类的加载机制
- 3.有参数创建对象
  - Constructor<?> csr = c4.getConstructor(String.class,,int.class);
    Object o = csr.newInstance("王",28);
    getConstructor方法返回了一 个Constructor对象，它反映了此 Class对象所表示的类的指定的公共构造

2.Constructor —— 反射类中构造方法；

3.Field —— 反射方法；

- 1.获取属性
  - Field field = class.getDeclaredField("name");
    使用setAccseeible取消封装，特别是可以取消私有字段的访问权限。
    field.setAccessible(true);
    field.set(object,"老王");
- 2.Field类描述
  - 1.Field类描述的是属性对象，其中可以获取到很多属性信息，包括名字、属性类型、属性的注解。
- 3.安全管理
  - 1.在安全管理器中会使用checkPermission方法来检查权限 ，而setAccessible(true)并不是将方法的权限改为public，而是取消Java的权限控制检查，所以即使是public方法，其accessible属性默认也是false。
- 4.修改属性中的修饰符
  - Field field = class.getDeclaredField("name");
    String prive = Modeifier.toString(field.getModofoers());
    getModofoers()返回的是一个 代表类、成员变量、方法的修饰符

4.Method —— 反射方法；

```
Method m = class.getDeclaredMethod("setName",String.class);
m.setAccessible(true); //同样需要忽略访问权限的限制
m.invoke(class,"老王");
```

5.Modifier —— 访问修饰符的信息。

反射进阶

- 1.获取不到Class
  - 当Class.foeName()中路径获取不到对应的Class时，会抛出异常。
- 2.获取不到Field
  - 1.确实不存在这个Field，抛出异常。
  - 2.修饰符导致的权限问题，抛出相同异常。
- 3.获取父类修饰符
  - 1.getField只能获取对象和父类的public修饰的属性。
  - 2.getDeclaredField获取对象中的各种修饰符属性，但是不能获取父类的任何属性。
  - 3.先使用getSupperclass方法可以获取父类的suppereClass对象，再使用getDeclaredField方法获取父类的全部属性
- 4.获取不到父类的非public的方法
- 5.获取不到父类的构造方法
- 6.newInstance方法创建类对象的两种方法
  - 1.Class.newInstance() —— 使用受到限制，对应的Class中必须存在一个无参数的构造方法，并且必须要有访问权限。
  - 2.Contructor.newInstance() —— 适应任何类型的构造方法，无论是否有参数都可以调用，只需要setAccessible()方法控制访问权限。
- 7.反射静态方法

```
public class TestMethod{
    static void test ( ) {}
}
Class cla = Class.foeName("TestMethod");
Method m = cla.getDeclaredMethod("test");
m.invoke(null);
```

关键是Method.invoke的第一个 static方法因为属于类本身 所以不需要填写对象，填写null就可以

- 8.反射泛型参数方法

```
1.Java的泛型擦除概念，泛型T在编译时会自动向上转型为Object

public class Test<T>{
    public void test(T t){}
}
Class cla = Test.class;
Method m = cla.getDeclaredMethod("test",Object.class);
m.invoke(new Test<Integer>(),1);
```

- 9.反射框架：jOOR