**Java的反射机制**

java反射其实就是在代码运行时,通过ClassLoader去加载一个存在的类，并可以获取起属性和方法。对于android 来说，只要本地存在的可编译的文件，都可以用个反射来获取。

反射就是把java类中的各种成分映射成一个个的Java对象。

例如：一个类有：成员变量、方法、构造方法、包等等信息，利用反射技术可以对一个类进行解剖，把个个组成部分映射成一个个对象。

**一．反射的原理**

Class加载的时， Class对象的由来是将class文件读入内存，并为之创建一个Class对象。

下图表示了一个class的加载过程：



其中这个Class对象很特殊。我们先了解一下这个Class类

Class类的实例表示正在运行的 Java 应用程序中的类和接口。也就是jvm中有N多的实例每个类每个都有该Class对象。（包括基本数据类型）

Class类没有公共构造方法。Class对象是在加载类时由 Java 虚拟机以及通过调用类加载器中的defineClass方法自动构造的。也就是这不需要我们自己去处理创建，JVM已经帮我们创建好了。

反射的关键就是要获取这个Class对象。

假设有一个People类，获取Class对象有三个方法：

Object —> getClass();每个类都有一个getClass方法。

Class Class = people.getClass();

（包括基本数据类型）都有一个“静态”的class属性。

Class Class2 = People.class;

任何数据类型通过Class类的静态方法：forName（String  className）(常用)

Class Class3 = Class.forName("com.qlj.lakinqiandemo.reflection.People");

第一种对象都有了还要反射干什么。第二种需要导入类的包，依赖太强，不导包就抛编译错误。一般都第三种，一个字符串可以传入也可写在配置文件中等多种方法。

**二．反射的实现**

反射的实现是通过 Class.forName这一静态方法去获取该Class对象，

Class Class3 = Class.forName("com.qlj.lakinqiandemo.reflection.People");

**（1）获取构造方法：**

  1).批量的方法：  
public Constructor[] getConstructors()：所有"公有的"构造方法  
public Constructor[] getDeclaredConstructors()：获取所有的构造方法(包括私有、受保护、默认、公有)  
       
  2).获取单个的方法，并调用：  
public Constructor getConstructor(Class... parameterTypes):获取单个的"公有的"构造方法：  
public Constructor getDeclaredConstructor(Class... parameterTypes):获取"某个构造方法"可以是私有的，或受保护、默认、公有；

调用构造方法：

Constructor-->newInstance(Object... initargs)  
newInstance是 Constructor类的方法（管理构造函数的类）

api的解释为：

newInstance(Object... initargs)  
使用此 Constructor 对象表示的构造方法来创建该构造方法的声明类的新实例，并用指定的初始化参数初始化该实例。

它的返回值是T类型，所以newInstance是创建了一个构造方法的声明类的新实例对象。并为之调用

**（2）获取成员变量**

1.批量获取

1).Field[] getFields():获取所有的"公有字段"

2).Field[] getDeclaredFields():获取所有字段，包括：私有、受保护、默认、公有；

2.获取单个的：

1).public Field getField(String fieldName):获取某个"公有的"字段；

2).public Field getDeclaredField(String fieldName)获取某个字段(可以是私有的) 设置字段的值：

 Field --> public void set(Object obj,Object value)

参数说明： 1.obj:要设置的字段所在的对象； 2.value:要为字段设置的值；

**（3）获取成员方法**

批量的：

public Method[] getMethods():获取所有"公有方法"；（包含了父类的方法也包含Object类）

public Method[] getDeclaredMethods():获取所有的成员方法，包括私有的(不包括继承的)

获取单个的：

public Method getMethod(String name,Class<?>... parameterTypes)

name : 方法名；Class ... : 形参的Class类型对象

public Method getDeclaredMethod(String name,Class<?>... parameterTypes)

调用方法：

Method -> public Object invoke(Object obj,Object... args)

obj : 要调用方法的对象；  args:调用方式时所传递的实参；

**三．反射的使用**

**1. 插件动态加载**

插件更新其实就是通过利用反射实现动态创建对象和编译，利用ClassLoader动态加载插件中的类，做到热更新的。

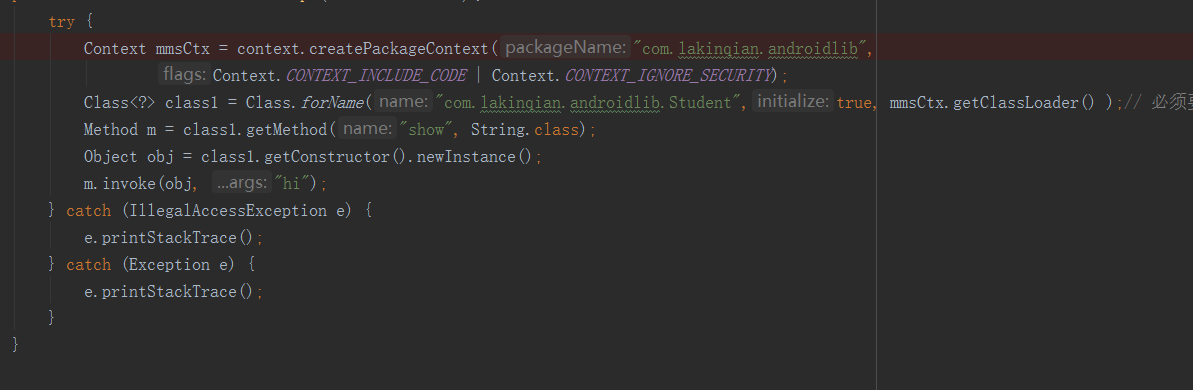
### 2. FrameWork中的反射

### android中FrameWork层的类的某些方法不让第三方应用去调用，比如一些private, protected的方法，还有一些带@hide的注解的方法，通过java反射机制能把这些隐藏方法获取出来并调用，三方应用上我们就很方便的去用这些方法。

### 

### 3. 反射获取其他应用的类

### 通过反射是可以获取任何可编译文件的类和方法的，所以，只要是存在的可执行文件，都可以通过反射获取到。所以，一些系统应用的某些特殊类都是可以通过反射获取到的。



总结：反射是一个很强大的功能，其实反射和之前的ClassLoader是有相关的，一般我们是反射我们自己应用或者FrameWork层的类时，由于该类已经存在于内存中，所以直接去内存中获取，而热修复或者反射第三方apk的类时，这些类有可能是没有被加载的，这时候，通过反射式可以把这些类加载到内存中的。