# 一、反射的概述

JAVA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；对于任意一个对象，都能够调用它的任意一个方法和属性；这种动态获取的信息以及动态调用对象的方法的功能称为

java语言的反射机制。

要想解剖一个类,必须先要获取到该类的字节码文件对象。而解剖使用的就是Class类中的方法.所以先要获取到每一个字节码文件对应的

Class类型的对象.

## 以上的总结就是什么是反射

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。

  （其实：一个类中这些成员方法、构造方法、在加入类中都有一个类来描述）

如图是类的正常加载过程：反射的原理在与class对象。

熟悉一下加载的时候：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。

Animal animal =new Dog（）；//在编译时期确定 静态

比如：有一个宠物的商城

Animal animal =new ??? // 在运行时期确定 动态、

**所有类都是Object的子类**

**所有类都是Class类的实例**

**反射机制： 动态的获取类的信息和动态的调用对象方法或者属性的功能！**

* 1. 在程序运行期间，对于任意一个类，都能获取这个类中的所有属性和方法；
  2. 对于任意一个对象，都能获取这个对象的所有属性和方法；

### 获取对象的三个方法

System.***out***.println(**"类的完整限定名来加载类"** + Class.*forName*(**"Reflect.Student"**));  
System.***out***.println(**"任何类名。class获取类"** + Student.**class**.getName());  
System.***out***.println(**"任何对象。getclass（）获取了class类"** + **new** String().getClass().getName());

### 获取对象的属性

Class c = Class.*forName*(**"Reflect.Student"**);  
System.***out***.println(**"Student类所在包"** + c.getPackage().getName());  
System.***out***.println(**"Student全类名"** + c.getName());  
System.***out***.println(**"Student简写类名"** + c.getSimpleName());

### getFields()和getDeclaredFields()。我们先来看看这两者的区别吧：

getFields()：获得某个类的所有的公共（public）的字段，包括父类中的字段。   
getDeclaredFields()：获得某个类的所有声明的字段，即包括public、private和

protected，但是不包括父类的申明字段。

Field[] fields = c.getDeclaredFields();  
**for** (**int** i = 0; i < fields.**length**; i++) {  
 System.***out***.println(fields[i]);  
 *// 获取所有字段的访问修饰符* System.***out***.println(Modifier.*toString*(fields[i].getModifiers()));  
 System.***out***.println(fields[i].getModifiers());  
}

获取类中所有的方法相关信息 不包含构造方法 **1.c.genNethod（方法名称，参数类型） 只能是public修饰的  
 2. c.getMethods（） 只能说public修饰的  
 3.c.getDeclareMethods（）所有**  
 Class c = Class.*forName*(**"Reflect.Student"**); *//加载* Method[] methods = c.getDeclaredMethods(); *//获取所有 包含private* **for** (**int** i = 0; i < methods.**length**; i++) {  
 System.***out***.println(**"方法的名字"** + methods[i].getName());  
 System.***out***.println(**"方法的修饰符值是"** + methods[i].getModifiers());  
 System.***out***.println(**"方法的修饰是"** + Modifier.*toString*(methods[i].getModifiers()));  
 System.***out***.println(**"方法的返回值类型是"** + methods[i].getReturnType());  
 }