# Java反射技术栈

类似与现实生活中的X射线，看透所有。如果Java没有反射机制，则星空则会暗淡无光，也是IDE提示功能的重要支持，还有servlet里面配置文件中class标签，用来动态生成一个类的对象，用到的也是反射的机制。

## 反射的概念

### 反射是什么

反射就是在程序运行过程中，动态创建对象。只要知道该类的类名称，就可以使用他的字节码对象创建该类的一个对象。对于该类中的任何一个对象或者属性，我们都可以访问和使用它，

### 字节码文件

.class

.java 🡪.class 🡪 jvm Class对象

Class对象加载到内存的时机，而且只会加载一次：

1. new一个对象时，也是通过字节码文件创建的，然后加载到内存；
2. new一个父类的子类对象时，把父类的字节码文件加载到内存；
3. 使用一个类的静态成员时
4. Java命令执行一个字节码对象的实现；
5. 用反射的时候

### 获得字节码对象的三种方式

1. 通过Object类的“实例.getClass”方法，获得类的全路径；
2. 通过“类型.class”，获得类的全路径；
3. 通过Class类的静态方法forName(“类的全路径”)；

### 字节码对象

类里面有什么东西，字节码就有什么东西。

1. 成员变量

Field对象

1. 成员方法

Method对象

1. 构造方法

Constructor对象

## 字节码对象的详解

### Constuctor对象

创建一个对象的方法

1. new调用的是public的构造方法，可以调用无参或者是有参的构造方法；
2. 利用.newInstance()方法创建一个实例，底层是调用有参的构造方法；
3. 通过构造方法对象创建一个类的实例Constructor

应用实例

1. 获得单个公共方法

Constructor constructor=类的class对象.getConstructor(参数的Class对象)；

类名 clazz =（类名）constructor. newInstance(参数)；

1. 获得单个方法

上述方法只能获得一个公共构造方法，若要获得私有的构造方法，需要使用增强方法getDeclareConstructor()方法，只要声明了就能找到：

Constructor constructor=类的class对象. getDeclareConstructor (参数的Class对象)；

constructor.setAccessible(true); //暴力访问；

类名 clazz =（类名）constructor. newInstance(参数)；

1. 获得所有公共方法

Constructor[] constructors=类的class对象.getConstructors()；

for(Constructor constructor: constructors){

System.out.println(constructor);

}

1. 获得所有方法

Constructor[] constructors=类的class对象.getDeclareConstructors()；

for(Constructor constructor: constructors){

constructor.setAccessible(true);

System.out.println(constructor);

}

### Field对象

1. 获得所有公共成员变量：

Class clazz = Class.forName(“com.lvhongbin.bean.User”);

Field[] fields=clazz.getFileds();

For(Filed field: fields){

System.out.println(field);

}

1. 获得单个成员变量（不管是私有的或者是共有的）

Class clazz = Class.forName(“com.lvhongbin.bean.User”);

类名 obj =（类名）clazz. newInstance(参数)；

Field field=clazz.getDeclareFiled(“成员变量名称”);

field.setAccessible(true);

System.out.println(field);

//给field赋值

field.set(obj, 值)；

### Method对象

1. 获得单个成员变量（不管是私有的或者是共有的）

Class clazz = Class.forName(“com.lvhongbin.bean.User”);

类名 obj =（类名）clazz. newInstance(参数)；

Method method =clazz.getDeclareMethod (“成员方法名称”，参数的类对象);

method.setAccessible(true);

System.out.println(method);

// 执行method方法

field.invoke (obj, 参数值)；

1. 获得所有公共成员变量：

Class clazz = Class.forName(“com.lvhongbin.bean.User”);

Field[] fields=clazz.getFileds();

For(Filed field: fields){

System.out.println(field);

}