## 反射 Reflection

| | | | |
|---|---|---|---|
| **笔记本：** | JavaSE | | |
| **创建时间：** | 2019.05.28 10:08 | **更新时间：** | 2020.08.25 07:20 |
| **作者：** | 195330205@qq.com | | |
| **URL：** | about:blank | | |

---

**反射：程序运行期，动态性（通用性）**
**应用：动态代理**

---

### 什么是反射

Java反射就是在运行状态中，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方法和属性；并且能改变它的属性。而这也是Java被视为动态（或准动态，因为一般而言的动态语言定义是程序运行时，允许改变程序结构或变量类型，这种语言称为动态语言。从这个观点看，Perl，Python，Ruby是动态语言，C++，Java，C#不是动态语言）语言的一个关键性质。

---

### 反射的用途

我们知道反射机制允许程序在运行时取得任何一个已知名称的class的内部信息，包括其modifiers(修饰符)，fields(属性)，methods(方法)等，并可于运行时改变fields内容或调用methods。那么我们便可以更灵活的编写代码，代码可以在运行时装配，无需在组件之间进行源代码链接，降低代码的耦合度；还有动态代理的实现等等；但是需要注意的是反射使用不当会造成很高的资源消耗！

---

#### 使用反射注意事项

尽管反射非常强大，但也不能滥用。如果一个功能可以不用反射完成，那么最好就不用。

- 性能第一：反射包括了一些动态类型，所以JVM无法对这些代码进行优化。因此，反射操作的效率要比那些非反射操作低得多。我们应该避免在经常被执行的代码或对性能要求很高的程序中使用反射。
- 安全限制：使用反射要求程序必须在一个没有安全限制的环境中运行。
- 内部暴露：由于反射允许代码执行一些在正常情况下不被允许的操作（比如访问私有的属性和方法），所以使用反射可能会导致意料之外的副作用----代码有功能上的错误，降低可移植性。反射代码破坏了抽象性，因此当平台发生改变的时候，代码的行为就有可能也随着变化。

灵活使用反射能让我们代码更加灵活，比如JDBC原生代码注册驱动，Hibernate 的实体类，Spring 的 AOP等等都有反射的实现。但是凡事都有两面性，反射也会消耗系统的性能，增加复杂性等。

---

java编译器javac.exe 将java源代码文件*.java 编译成java字节码文件*.class

java.exe对字节码文件*.class解释运行，相当于将字节码文件加载到内存中（类的加载），加载到内存中的类称为运行时类（Class类的实例）

---

获取运行时类的方式：加载到内存中的运行时类会缓存一定的时间，在此时间内可以通过以下4中方式来获取
1.运行时类的属性：Person.class
2.运行时类的对象的方法：person.getClass()
3.Class类的静态方法：Class.forName("full class name")　更好地体现了动态性
4.类加载器：getClassLoader().loadClass("full class name")

**一个类在 JVM 中只会有一个 Class 的实例**

---

属性：
getFields()
getDeclaredFields()
　getModifiers()
　　Modifier.toString(getModifiers())
　getType()
　getName()

---

方法：
getMethods()
getDeclaredMethods()
　getModifiers()
　　Modifier.toString(getModifiers())
　getAnnotations()
　getReturnType().getName()
　getName()
　getParameterTypes()
　getParameters()
　getExceptionTypes()

---

构造方法：
　getConstructors()
　getDeclaredConstructors()

---

父类：
getSuperClass()
getGenericSuperClass()
泛型：
((ParameterizedType) getGenericSuperClass()).getActualTypeArguments()

---

接口：
getInterfaces()

---

包：
getPackage()

注解：
getAnnotations()

setAccessible()