JAVA动态性之：

反射机制 reflection

加入www.sxt.cn 一起学吧！

讲师：高淇 邮箱：gaoqi110@163.com

JAVA的动态性

- 反射机制
- 动态编译
- 动态执行javascript代码
- 动态字节码操作

动态语言

- 动态语言
  - 程序运行时，可以改变程序结构或变量类型。典型的语言：
    - Python、ruby、javascript等。
    - 如下javascript代码：

```
function test(){
        var s = "var a=3;var b=5;alert(a+b);";
        eval(s);
}
```

- C，C++，JAVA不是动态语言，JAVA可以称之为“准动态语言”。但是JAVA有一定的动态性，我们可以利用反射机制、字节码操作获得类似动态语言的特性。

- JAVA的动态性让编程的时候更加灵活！

反射机制 reflection

- 反射机制
  - 指的是可以于运行时加载、探知、使用编译期间完全未知的类。
  - 程序在运行状态中，可以动态加载一个只有名称的类，对于任意一个已加载的类，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；

```
Class  c = Class.forName("com.bjsxt.test.User");
```

  - 加载完类之后，在堆内存中，就产生了一个 Class 类型的对象（一个类只有一个 Class 对象），这个对象就包含了完整的类的结构信息。我们可以通过这个对象看到类的结构。这个对象就像一面镜子，透过这个镜子看到类的结构，所以，我们形象的称之为：反射。

Class类介绍

- java.lang.Class类十分特殊，用来表示java中类型(class/interface/enum/annotation/primitive type/void)本身。
  - Class类的对象包含了某个被加载类的结构。一个被加载的类对应一个Class对象。
  - 当一个class被加载，或当加载器（class loader）的defineClass()被JVM调用，JVM 便自动产生一个Class 对象。
- Class类是Reflection的根源。
  - 针对任何您想动态加载、运行的类，唯有先获得相应的Class 对象

Class类的对象如何获取?

- 运用getClass()
- 运用Class.forName()（最常被使用）
- 运用.class 语法

反射机制的常见作用

- 动态加载类、动态获取类的信息（属性、方法、构造器）
- 动态构造对象
- 动态调用类和对象的任意方法、构造器
- 动态调用和处理属性
- 获取泛型信息
- 处理注解

反射操作泛型(Generic)

- Java采用**泛型擦除的机制**来引入泛型。Java中的泛型仅仅是给编译器javac使用的，**确保数据的安全性和免去强制类型转换的麻烦**。但是，一旦编译完成，所有的**和泛型有关的类型全部擦除**。

- 为了通过**反射操作这些类型**以迎合实际开发的需要，Java就**新增了ParameterizedType**，**GenericArrayType**，**TypeVariable** 和**WildcardType**几种类型来**代表不能被归一到Class类中的类型但是又和原始类型齐名的类型**。

- ParameterizedType: 表示一种参数化的类型，比如Collection<String>
- GenericArrayType: 表示一种元素类型是参数化类型或者类型变量的数组类型
- TypeVariable: 是各种类型变量的公共父接口
- WildcardType: 代表一种通配符类型表达式，比如?, ? extends Number, ? super Integer【wildcard是一个单词：就是“通配符”】

反射操作注解(annotation)

- 可以通过反射API:getAnnotations, getAnnotation获得相关的注解信息

```
//获得类的所有有效注解
Annotation[] annotations=clazz.getAnnotations();
for (Annotation a : annotations) {
    System.out.println(a);
}
//获得类的指定的注解
SxtTable st = (SxtTable) clazz.getAnnotation(SxtTable.class);
System.out.println(st.value());

//获得类的属性的注解
Field f = clazz.getDeclaredField("studentName");
SxtField sxtField = f.getAnnotation(SxtField.class);
System.out.println(sxtField.columnName()+"--
"+sxtField.type()+"--"+sxtField.length());
```

反射机制性能问题

- setAccessible
  - 启用和禁用访问安全检查的开关,值为 true 则指示反射的对象在使用时应该取消 Java 语言访问检查。值为 false 则指示反射的对象应该实施 Java 语言访问检查。并不是为true就能访问为false就不能访问。
  - 禁止安全检查，可以提高反射的运行速度。

- 可以考虑使用：cglib/javaassist字节码操作