1. Class基础用法

String s = "3";  
//获取某个对象的Class对象  
Class clazz = s.getClass();  
//获取某个类的Class对象  
Class clazz0 = String.class;  
Class clazz1 = String[].class;  
Class clazz2 = int.class;  
Class clazz3 = int[].class;  
//获取某个类名的Class对象  
//可以使用此技巧在程序启动后再加载类，加快启动速度  
Class clazz4 = Class.*forName*("java.lang.String");  
  
//获取类的名字，包含包名  
String className = clazz.getName();  
//数组获取的类名前面会多一个'[L'字符串  
String className0 = clazz1.getName();  
//基本类型的名字是直接是类型名，如int类型返回的是int  
String className1 = clazz2.getName();  
//基本数据类型的数组用getName会返回奇奇怪怪的名字  
//如int返回[I, long 返回[J, boolean返回[Z  
String className2 = clazz3.getName();  
  
//因为虚拟机每种类型只会管理一个Class对象  
//所以可以用==来比较类型是否相同  
//下面4个String的Class对象都指向一个引用  
boolean a = clazz == clazz0;  
boolean b = clazz0 == String.class;  
  
Class userClass = Class.*forName*("com.pojo.User");  
//可以动态的创建类的一个实例  
//newInstance方法调用默认的构造器（没有参数的构造器）初始化新创建的对象  
//如果没有默认构造器，就会抛出一个异常  
User s0 = (User) userClass.newInstance();

1. Reflect包又三个类Field、Method和Constructor分别用于描述类的域、方法和构造器。

3． 1

ClassDemo classDemo = new ClassDemo();  
Class userClass = User.class;  
  
User user = new User(1, "3321", "312", "312",3);  
String[] strings = {"333","31231","312","aaa"};  
int[] ints = {1,2,3,4};  
Class stringsClass = strings.getClass();  
Product product = new Product();  
product.setName("666");  
product.setDate(new Date());  
  
  
//获取封闭该类的类,如这个类是一个内部类(接口)或者匿名类  
Class enclosingClass = ClassDemo.ClassA.class.getEnclosingClass();  
Class enclosingClass0 = ClassDemo.IA.class.getEnclosingClass();  
//通过匿名类获取封闭该匿名类的类，一般为实现这个匿名类的所在的类  
Class enclosingClass1 = new ClassDemo.IA() { public void ia(int a) {} }.getClass().getEnclosingClass();  
//判断这个类对象是否描述一个数组  
boolean isArray = stringsClass.isArray();  
//获取数组元素的类型  
Class arrayType = stringsClass.getComponentType();  
  
  
  
//获取类中的公共域，包含超类的公有成员  
Field[] fields = userClass.getFields();  
//获取类中的全部域、方法和构造器（包括private和protected）  
//不包括超类的成员  
Field[] fields1 = userClass.getDeclaredFields();  
Field field = userClass.getField("sex");  
//获取域的类型  
Class fieldClass = field.getType();  
//获取域名的字符串  
String fieldName = field.getName();  
//判断是否为原始类型（如int，long。。。）  
boolean isPrimitive = fieldClass.isPrimitive();  
  
//直接获取私有域的对象的方法  
Field productField = userClass.getDeclaredField("product");  
//使能够获取私有域对象  
productField.setAccessible(true);  
User user1 = new User(1, "31","333","wjx",1);  
//获取私有域对象  
Product value = (Product) productField.get(user1);  
//设置私有域的对象新值  
productField.set(user1, product);  
  
  
  
  
//同域  
Method[] methods = userClass.getMethods();  
Method[] methods0 = userClass.getDeclaredMethods();  
//通过方法名获取方法  
Method method = userClass.getMethod("setId", int.class);  
Method method0 = userClass.getDeclaredMethod("declaredMethod");  
//调用对应方法  
method.invoke(user, 10);  
//内部类或者匿名类的包围方法,带参数  
Method method1 = classDemo.getClassE().getClass().getEnclosingMethod();  
Method method2 = classDemo.getIAImpl().getClass().getEnclosingMethod();  
//获取该方法的异常类数组  
Class[] exceptionClasses = userClass.getMethod("exceptionMethod").getExceptionTypes();  
//获取方法名的字符串  
String methodName = method.getName();  
//获取描述参数类型的Class对象数组  
Class[] methodParamClasses = method.getParameterTypes();  
//获取返回类型的Claass对象  
Class returnType = method.getReturnType();  
  
  
  
  
//获取本类的公共构造器，通过参数类型  
Constructor constructor = userClass.getConstructor(  
 int.class, String.class, String.class, String.class, int.class);  
//获取本类的公共或私有构造器，通过参数类型  
Constructor constructor0 = userClass.getDeclaredConstructor(String.class, int.class);  
//获取封闭该内部类,匿名类的构造器，包括参数构造器的参数  
Constructor constructor1 = classDemo.classC.getClass().getEnclosingConstructor();  
Constructor constructor2 = classDemo.classD.getClass().getEnclosingConstructor();  
//获取所有本类的公有构造器  
Constructor[] constructors = userClass.getConstructors();  
//获取本类的全部构造器，包括的私有的  
//不获取父类的构造器  
Constructor[] constructors1 = userClass.getDeclaredConstructors();  
//获取该构造器所在的类  
Class constructorClass = constructor.getDeclaringClass();  
//获取该构造器的异常类数组  
Class[] exceptionClasses0 = constructor.getExceptionTypes();  
//获取构造器名的字符串  
String constructorName = constructor.getName();  
//获取描述参数类型的Class对象数组  
Class[] constructorParamClasses = constructor.getParameterTypes();  
  
  
  
//用于获取构造器，方法，域的修饰符的整型数值  
int modifiers = constructor0.getModifiers();  
String type = Modifier.*toString*(modifiers);  
Modifier.*isAbstract*(modifiers);  
Modifier.*isFinal*(modifiers);  
Modifier.*isInterface*(modifiers);  
Modifier.*isNative*(modifiers);  
Modifier.*isPrivate*(modifiers);  
Modifier.*isProtected*(modifiers);  
Modifier.*isPublic*(modifiers);  
Modifier.*isStatic*(modifiers);  
Modifier.*isStrict*(modifiers);  
Modifier.*isSynchronized*(modifiers);  
Modifier.*isVolatile*(modifiers);  
Modifier.*isTransient*(modifiers);  
  
  
  
//设置所以field为可以访问私有变量  
AccessibleObject.*setAccessible*(fields, true);  
//获取一个数组的长度，这个方法经常用于反射的数组  
int len = Array.*getLength*(strings);  
//动态创建一个新的arrayType类型，长度为100的数组  
Object newArray = Array.*newInstance*(arrayType, 100);  
//获取某个数组的某个下表的元素  
String element = (String) Array.*get*(strings, 1);  
//获取数组的对应数据类型和下标的元素，如int，long，byte等  
int element0 = Array.*getInt*(ints, 1);  
//设置数组对应下标的元素  
Array.*set*(strings, 3, "5");  
//设置数组对应下标和类型的元素，如int，long，byte等  
Array.*setInt*(ints, 2, 3);