JAVA基础－－JAVA中的反射机制详解 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/271/2021_2022_JAVA_E5_9F_BA_E7_A1_80_c104_271838.htm JAVA反射机制 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；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。Java反射机制主要提供了以下功能： 在运行时判断任意一个对象所属的类；在运行时构造任意一个类的对象；在运行时判断任意一个类所具有的成员变量和方法；在运行时调用任意一个对象的方法；生成动态代理。1. 得到某个对象的属性1 public Object getProperty(Object owner, String fieldName) throws Exception {2 Class ownerClass = owner.getClass().3 4 Field field = ownerClass.getField(fieldName).5 6 Object property = field.get(owner).7 8 return property.9 }Class ownerClass = owner.getClass()：得到该对象的Class。Field field = ownerClass.getField(fieldName)：通过Class得到类声明的属性。Object property = field.get(owner)：通过对象得到该属性的实例，如果这个属性是非公有的，这里会报IllegalAccessException。2. 得到某个类的静态属性 1 public Object getStaticProperty(String className, String fieldName) 2 throws Exception { 3 Class ownerClass = Class.forName(className). 4 5 Field field = ownerClass.getField(fieldName). 6 7 Object property = field.get(ownerClass). 8 9 return property.10 }Class ownerClass = Class.forName(className) ：首先得到这个类的Class。Field

field = ownerClass.getField(fieldName)：和上面一样，通过Class得到类声明的属性。Object property = field.get(ownerClass)：这里和上面有些不同，因为该属性是静态的，所以直接从类的Class里取。3. 执行某对象的方法 1 public Object invokeMethod(Object owner, String methodName, Object[] args) throws Exception { 2 3 Class ownerClass = owner.getClass(). 4 5 Class[] argsClass = new Class[args.length]. 6 7 for (int i = 0, j = args.length. i 8 argsClass[i] = args[i].getClass(). 9 }10 11 Method method = ownerClass.getMethod(methodName, argsClass).12 13 return method.invoke(owner, args).14 }Class owner_class = owner.getClass()：首先还是必须得到这个对象的Class。5～9行：配置参数的Class数组，作为寻找Method的条件。Method method = ownerClass.getMethod(methodName, argsClass)：通过Method名和参数的Class数组得到要执行的Method。method.invoke(owner, args)：执行该Method，invoke方法的参数是执行这个方法的对象，和参数数组。返回值是Object，也既是该方法的返回值。4. 执行某个类的静态方法 1 public Object invokeStaticMethod(String className, String methodName, 2 Object[] args) throws Exception { 3 Class ownerClass = Class.forName(className). 4 5 Class[] argsClass = new Class[args.length]. 6 7 for (int i = 0, j = args.length. i 8 argsClass[i] = args[i].getClass(). 9 }10 11 Method method = ownerClass.getMethod(methodName, argsClass).12 13 return method.invoke(null, args).14 }基本的原理和实例3相同，不同点是最后一行，invoke的一个参数是null，因为这是静态方法，不需要借助实例运行。5. 新建实例 1 2 public Object

newInstance(String className, Object[] args) throws Exception { 3 Class newoneClass = Class.forName(className). 4 5 Class[] argsClass = new Class[args.length]. 6 7 for (int i = 0, j = args.length. i 8 argsClass[i] = args[i].getClass(). 9 }10 11 Constructor cons = newoneClass.getConstructor(argsClass).12 13 return cons.newInstance(args).14 15 }这里说的方法是执行带参数的构造函数来新建实例的方法。如果不需要参数，可以直接使用newoneClass.newInstance()来实现。Class newoneClass = Class.forName(className)：第一步，得到要构造的实例的Class。第5～第9行：得到参数的Class数组。Constructor cons = newoneClass.getConstructor(argsClass)：得到构造子。cons.newInstance(args)：新建实例。6. 判断是否为某个类的实例1 public boolean isInstance(Object obj, Class cls) {2 return cls.isInstance(obj).3 }7. 得到数组中的某个元素1 public Object getByArray(Object array, int index) {2 return Array.get(array,index).3 } 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com