java反射详解

反射的概念

反射就是把Java类中的各个成分映射成一个个的Java对象，即在运行状态中，对于任意一个类，都能够知道这个类的所以属性和方法；对于任意一个对象，都能调用它的任意一个方法和属性。这种**动态获取信息**及**动态调用对象方法的功能**叫Java的反射机制。

反射的作用

1. 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。
2. 在运行状态中，获取类的信息，可以获取在编译期不可能获得的类的信息。
3. 对于任意一个对象，都能调用它的任意一个方法和属性。
4. 动态获取Class对象的信息以及动态操作Class对象的属性和方法的功能称为Java的反射机制。

实现反射机制的类

Java中主要由以下的类来实现Java反射机制（这些类都位于java.lang.reflect包中）：

- Class类：代表一个类。 Field类：代表类的成员变量（成员变量也称为类的属性）。
- Method类：代表类的方法。
- Constructor类：代表类的构造方法。
- Array类：提供了动态创建数组，以及访问数组的元素的静态方法。

反射的使用

要使用反射，得先了解Class类

Class类

java语言是面向对象编程，我们操作的所有类都是Class类的实例化对象。这种实例化对象有三种表示方式：

```
public class Demo4 {
    public static void main(String[] args) throws ClassNotFoundException {
        F f = new F();
        //第一种表达方式
        Class c1 = F.class;//这种表达方式同时也告诉了我们任何一个类都有一个隐含的静态成员变量class
        //第二种表达方式
        Class c2 = f.getClass();//这种表达方式在已知了该类的对象的情况下通过getClass方法获取
        //第三种表达方式
        //类的全称
        Class c3 = Class.forName("com.jzc.problem.jvm.F");
    }
}
class F {}
```

以上三种表达方式，c1,c2,c3都表示了F类的类类型，也就是官方解释的Class Type。

Class类主要方法

- getName()：获得类的完整名字。
- getFields()：获得类的public类型的属性。
- getDeclaredFields()：获得类的所有属性。包括private 声明的和继承类
- getMethods()：获得类的public类型的方法。
- getDeclaredMethods()：获得类的所有方法。包括private 声明的和继承类
- getMethod(String name, Class[] parameterTypes)：获得类的特定方法，name参数指定方法的名字，parameterTypes 参数指定方法的参数类型。
- getConstructors()：获得类的public类型的构造方法。
- getConstructor(Class[] parameterTypes)：获得类的特定构造方法，parameterTypes 参数指定构造方法的参数类型。
- newInstance()：通过类的构造方法创建这个类的一个对象。

Class能实现的功能

1.判断对象属于哪个类

```
Person person = new Person();
Class class2= person.getClass();
System.out.println("class2: "+class2);
输出：class2：class reflect.Person
```

2.获取类信息

```
Class class1 = Person.class;
// 获取public方法
Method[] methods = class1.getMethods();
// 获取所有方法
Method[] declaredMethods = class1.getDeclaredMethods();
// 获取所有属性
Field[] declaredFields = class1.getDeclaredFields();
```

3.构建对象

```
Person person = new Person();
Class class2= person.getClass();
Object o = class2.newInstance();
//强转前先用instanceof判断
if (o instanceof Person) {
    Person o1 = ((Person) o);
}
```

4.动态执行方法

```
Class class1 = Class.forName("reflect.Person");
Method work = class1.getDeclaredMethod("work");
Person person = new Person();
work.invoke(person);
```

5.动态操作属性

```
Class class1 = Class.forName("reflect.Person");
Person person = new Person();
Field field = class1.getDeclaredField("username");
field.set(person,"pine");
```

6.动态代理

在java的动态代理机制中，有两个重要的类或接口，一个是 InvocationHandler(Interface)、另一个则是 Proxy(Class)，这一个类和接口是实现我们动态代理所必须用到的。首先我们先来看看java的API帮助文档是怎么样对这两个类进行描述的：

InvocationHandler:

```
InvocationHandler is the interface implemented by the invocation handler of a proxy instance.

Each proxy instance has an associated invocation handler. When a method is invoked on a proxy instance, the method invocation is encoded and dispatched to the invoke method of its invocation handler.
```

每一个动态代理类都必须要实现InvocationHandler这个接口，并且每个代理类的实例都关联到了一个handler，当我们通过代理对象调用一个方法的时候，这个方法的调用就会被转发为由InvocationHandler这个接口的 invoke 方法来进行调用。我们来看看InvocationHandler这个接口的唯一一个方法 invoke 方法：

```
Object invoke(Object proxy, Method method, Object[] args) throws Throwable
```

我们看到这个方法一共接受三个参数，那么这三个参数分别代表什么呢?

```
Object invoke(Object proxy, Method method, Object[] args) throws Throwable

proxy:    指代我们所代理的那个真实对象
method:    指代的是我们所要调用真实对象的某个方法的Method对象
args:    指代的是调用真实对象某个方法时接受的参数
```

如果不是很明白，等下通过一个实例会对这几个参数进行更深的讲解。

接下来我们来看看Proxy这个类：

```
Proxy provides static methods for creating dynamic proxy classes and instances, and it
is also the superclass of all dynamic proxy classes created by those methods.
```

Proxy这个类的作用就是用来动态创建一个代理对象的类，它提供了许多的方法，但是我们用的最多的就是newProxyInstance 这个方法：

```
public static Object newProxyInstance(ClassLoader loader, Class<?>[] interfaces,
 InvocationHandler h)  throws IllegalArgumentException
Returns an instance of a proxy class for the specified interfaces that dispatches
method invocations to the specified invocation handler.
```

这个方法的作用就是得到一个动态的代理对象，其接收三个参数，我们来看看这三个参数所代表的含义：

```
public static Object newProxyInstance(ClassLoader loader, Class<?>[] interfaces,
InvocationHandler h) throws IllegalArgumentException

loader:    一个ClassLoader对象，定义了由哪个ClassLoader对象来对生成的代理对象进行加载

interfaces:    一个Interface对象的数组，表示的是我将要给我需要代理的对象提供一组什么接口，如果我提供了一组接口给它，那么这个代理对象就宣称实现了该接口(多态)，这样我就能调用这组接口中的方法了

h:    一个InvocationHandler对象，表示的是当我这个动态代理对象在调用方法的时候，会关联到哪一个InvocationHandler对象上
```

Java反射面试题

1. java反射的作用是什么?
2. Java反射创建对象效率高还是通过new创建对象的效率高?
3. 除了使用new创建对象之外，还可以用什么方法创建对象?
4. 反射的实现方式都有什么?
5. 实现java反射的类有什么?
6. 反射机制的优缺点?