反射机制介绍

java反射机制是在**运行状态中**，对于任意一个类，都能够知道**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；这种**动态获取的信息以及动态调用对象的方法的功能称之为java语言的反射机制。**

获取Class对象的两种方式

知道具体类的情况下可以使用： Class alunbarClass = TargetObject.class;

通过Class.forname()传入类的路径获取 Class alunbarClass1 = Class.forName("cn.javaguide.TargetObject");

代码实例

1. 创建一个我们要使用的反射操作的类TargetObject：

   public class TargetObject{ private String value;

   ```
   public TargetObject(){
       this.value = "zyq";
   }

   public void publicMethod(){
       System.out.println("value is " + value);
   }
   public void publicMethod(String s){
       System.out.println("value is " + s);
   }
   ```

   }

2. 使用反射操作这个类的方法以及参数

   ```
   public class Main {
       public static void main(String[] args) throws ClassNotFoundException,
   NoSuchMethodException, IllegalAccessException, InstantiationException,
   InvocationTargetException, NoSuchFieldException {
           /**
           * 获取TargetObject类的Class对象并且创建TargetObject类实例
           */
           Class<?> tagetClass = Class.forName("cn.javaguide.TargetObject");
           TargetObject targetObject = (TargetObject)
   tagetClass.newInstance();
           /**
           * 获取所有类中所有定义的方法
           */
           Method[] methods = tagetClass.getDeclaredMethods();
   ```

```
            for (Method method : methods) {
                System.out.println(method.getName());
            }
            /**
            * 获取指定方法并调用
            */
            Method publicMethod = tagetClass.getDeclaredMethod("publicMethod",
                    String.class);

            publicMethod.invoke(targetObject, "JavaGuide");
            /**
            * 获取指定参数并对参数进行修改
            */
            Field field = tagetClass.getDeclaredField("value");
            //为了对类中的参数进行修改我们取消安全检查
            field.setAccessible(true);
            field.set(targetObject, "JavaGuide");
            /**
            * 调用 private 方法
            */
            Method privateMethod =
    tagetClass.getDeclaredMethod("privateMethod");
            //为了调用private方法我们取消安全检查
            privateMethod.setAccessible(true);
            privateMethod.invoke(targetObject);
        }
    }

    publicMethod
    privateMethod
    I love JavaGuide
    value is JavaGuide
```

反射机制优缺点

优点：运行期类型的判断，动态加载类，提高代码灵活的 **确定**：1. 性能瓶颈：反射相当于一系列解释操作，通知JVM要做的事情，性能比直接的java代码要慢汗多 2. 安全问题，让我们可以动态操作改变类的属性同时增加类的安全隐患。

反射是框架设计的灵魂