Java8反射获取方法的参数名称 类和接口

java8反射获取方法的参数名称 类和接口

文章目录

1.简要说明

- 演示 `springboot` 版本为 `2.0.3.RELEASE`
- 为啥要分 `java类` 和 `java接口` 两种进行参数获取的演示
  - `java类` 编译成 `class文件` 后， `方法签名` 和 `参数` 会保存在字节码文件中，通过一些字节码的解析框架可以获取方法的参数名称
  - `java接口` 编译成 `class文件` 后，默认不会保存方法的参数签名，jdk这么设计，因为接口的参数名称一般没有意义，但是现在的技术发展导致一些框架需要用到接口的参数名称
    - spring openFeign 中使用接口进行编程
    - mybatis 中的 Mapper 文件接口
    - 其他一些动态代理接口的编程方式
- 在Java8之后，反射包 `package java.lang.reflect;` 新加入了 `Parameter` 类，可以解决这类问题

指导文章
https://docs.oracle.com/javase/tutorial/reflect/member/methodparameterreflection.html
Java 8 + 增加 `javac -parameters` 参数

> 大前提：必须是 Java 8

```
package java.lang.reflect;
...
/**
 * Information about method parameters.
 *
 * A {@code Parameter} provides information about method parameters,
 * including its name and modifiers.  It also provides an alternate
 * means of obtaining attributes for the parameter.
 *
 * @since 1.8
 */
public final class Parameter implements AnnotatedElement {

    private final String name;
    private final int modifiers;
    private final Executable executable;
    private final int index;
    ...
}
```

2.java类获取方法参数名称

测试代码

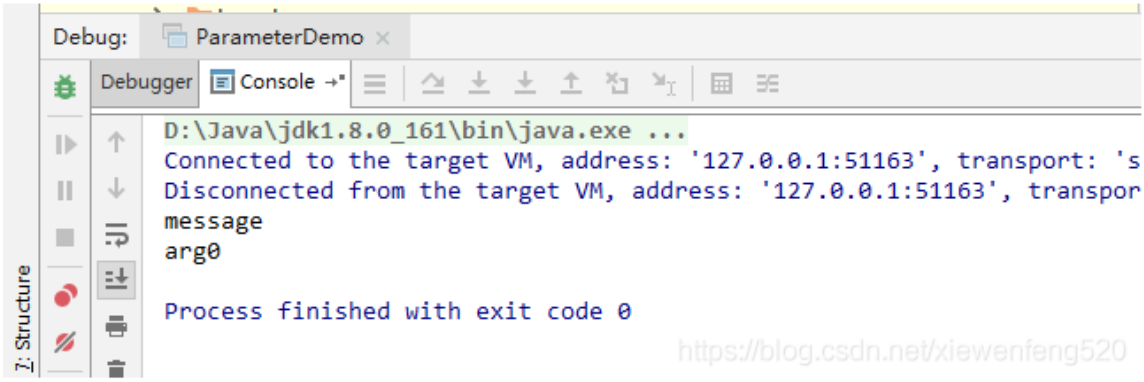
- 使用 `spring` 中的 `ParameterNameDiscoverer` 类获取方法的参数
- 使用传统的 `java反射` 获取参数名称

```
/**
 * 测试反射获取类方法参数
 */
public class ParameterDemo {
    public String say(String message){
        return "hello world:"+message;
    }

    public static void main(String[] args) {
        Method method = ReflectionUtils.findMethod(ParameterDemo.class,"say",String.c
        ParameterNameDiscoverer discoverer = new DefaultParameterNameDiscoverer();
        String[] parameterNames = discoverer.getParameterNames(method);
        Stream.of(discoverer.getParameterNames(method)).forEach(System.out::println);
        methodTest();
    }

    public static void methodTest() {
        Method method = ReflectionUtils.findMethod(ParameterDemo.class,"say",String.c
        Parameter parameter = method.getParameters()[0];
        System.out.println(parameter.getName());
    }
}
```

执行结果



- `spring` 通过字节码解析技术获取到了方法的参数名称
- 传统的反射方法无法获取参数名称

解决方法

增加 `javac -parameters` 参数

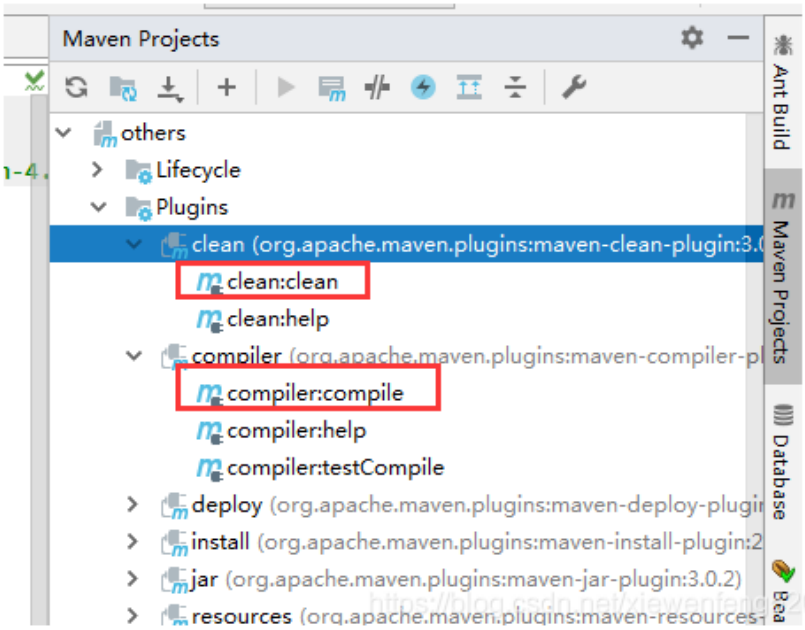
通过 `maven` 的方式

```
<build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.5.1</version>
                <configuration>
                    <source>1.8</source>
                    <target>1.8</target>
                    <debug>true</debug>
                    <debuglevel>lines,vars,source</debuglevel>
                    <compilerArgs>
                        <arg>-parameters</arg>
                    </compilerArgs>
                </configuration>
            </plugin>
        </plugins>
    </build>
```

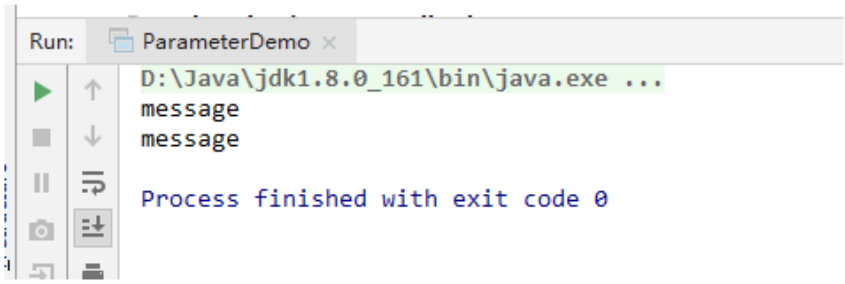
关键代码

```
<compilerArgs>
     <arg>-parameters</arg>
</compilerArgs>
```

先 clean 再 compiler



执行结果



3.接口获取方法参数名称

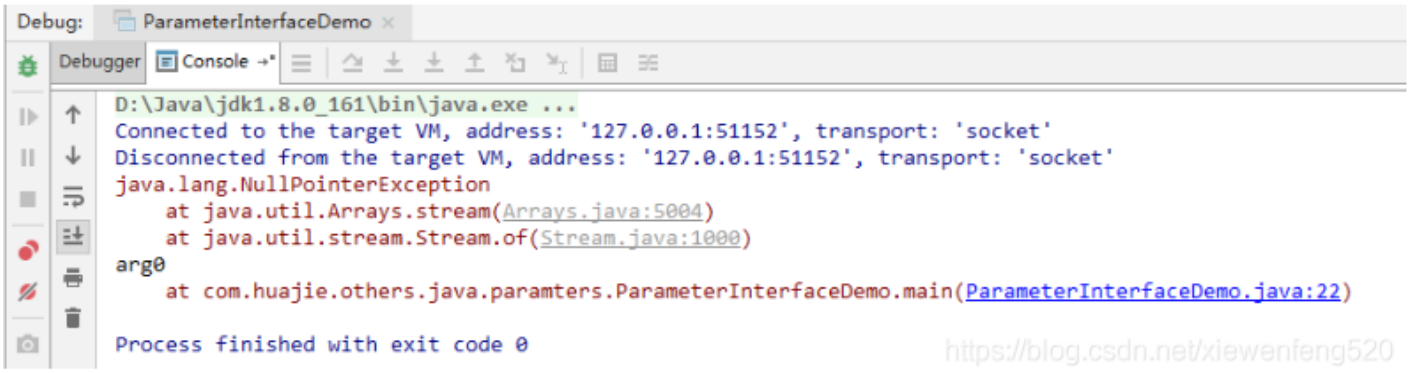
测试代码

```
1  /**
2   * 测试反射获取接口方法参数
3   */
4  public interface ParameterInterfaceDemo {
5      String say(String message);
6
7      public static void main(String[] args) {
8          Method method = ReflectionUtils.findMethod(ParameterInterfaceDemo.class,"say"
9          ParameterNameDiscoverer discoverer = new DefaultParameterNameDiscoverer();
10         String[] parameterNames = discoverer.getParameterNames(method);
11         try {
12             Stream.of(discoverer.getParameterNames(method)).forEach(System.out::print
13         }catch (Exception e){
14             e.printStackTrace();
15         }
16         methodTest();
```

```
    }

    public static void methodTest() {
        Method method = ReflectionUtils.findMethod(ParameterInterfaceDemo.class,"say"
        Parameter parameter = method.getParameters()[0];
        System.out.println(parameter.getName());
    }
}
```

执行结果



在接口中，不管是 `spring` 通过字节码技术还是传统的反射方法都无法获取

解决方法

增加 `javac -parameters` 参数

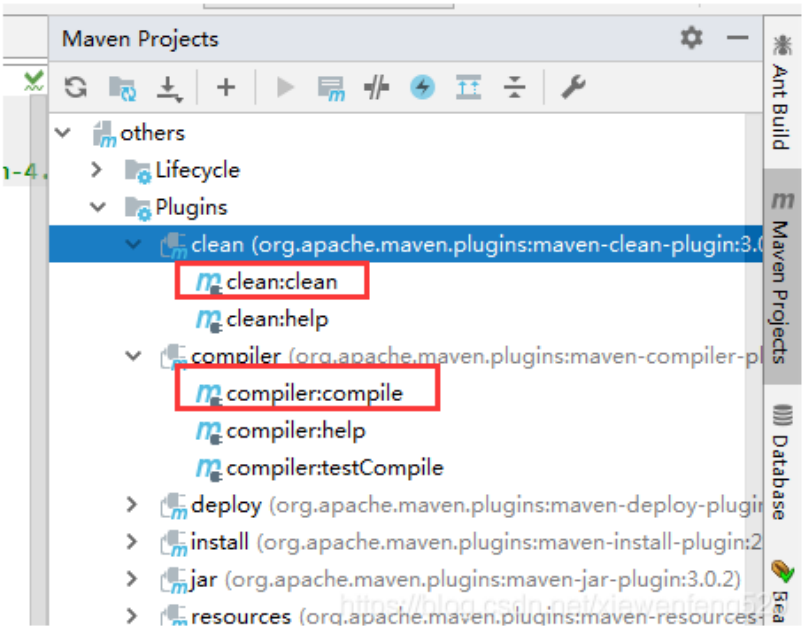
通过 `maven` 的方式

```
<build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.5.1</version>
                <configuration>
                    <source>1.8</source>
                    <target>1.8</target>
                    <debug>true</debug>
                    <debuglevel>lines,vars,source</debuglevel>
                    <compilerArgs>
                        <arg>-parameters</arg>
                    </compilerArgs>
                </configuration>
            </plugin>
        </plugins>
    </build>
```

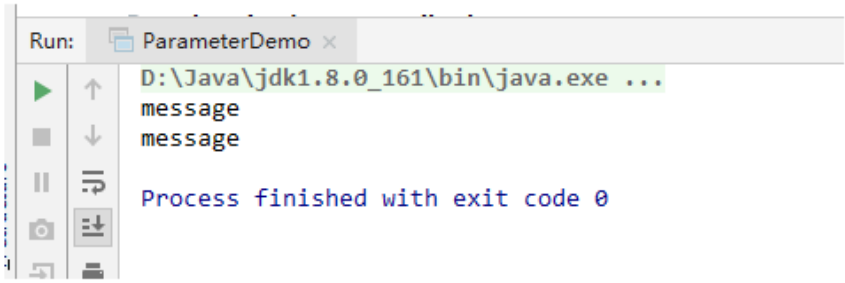
关键代码

```
<compilerArgs>
     <arg>-parameters</arg>
</compilerArgs>
```

先 clean 再 compiler



执行结果



感谢

mercyblitz - 小马哥的讲解 博客地址 (https://mercyblitz.github.io)