Javasisst 字节码技术

> Javassist 是一个用于处理 java 字节码的 java语言实现的类库

一、重要类

1、ClassPool -> 类池

> 一个 ClassPool 是存储 CtClass 的容器，一旦 CtClass 对象被创建，它就会被记录在 ClassPool 中，默认系统搜索路径查找类

2、CtClass -> Class

> 一个 CtClass 的实例代表一个类，CtClass 的子类有：CtClassType、CtPrimitiveType 和 CtArray，从 ClassPool 中获取

3、CtField -> Field

> 一个 CtField 的实例代表一个字段，可通过 CtClass#getDeclaredFields() 方法获取类中声明的所有字段，继承的字段不包括在内

4、CtMethod -> Method

> 一个 CtMethod 的实例代表一个方法

5、CtConstructor -> Constructor

> 一个 CtConstructor 的实例代表一个构造函数

6、HotSwapper

> 在运行时重新加载一个类

7、ClassFile

> ClassFile 表示一个 .class 文件，其中由常量池、方法、字段和属性组成

8、FieldInfo

> 表示字节码 field_info 结构信息

9、MethodInfo

> 表示字节码 method_info 结构信息

二、主要类相关方法详细介绍

1、ClassPool

> 1. 基础属性
>
> ```
> // 父 ClassPool
> protected ClassPool parent;
> // 存储 CtClass 的 hash 表，使用了 Hashtable，默认存储了9个类型：8个金部类型 + void 类型
> protected Hashtable classes;
> // hash 表的默认大小
> private static final int INIT_HASH_SIZE = 191;
> ```

2. 常用方法

- getDefault()

返回默认的类池，单例模式

```
ClassPool pool = ClassPool.getDefault();
```

- get()

从源文件中读取类文件并返回 CtClass 引用，原始类型可使用 CtClass 中的静态变量来获取

```
// Object
static final String javaLangObject = "java.lang.Object";
// boolean CtClass
public static CtClass booleanType;
// char CtClass
public static CtClass charType;
// byte CtClass
public static CtClass byteType;
// short CtClass
public static CtClass shortType;
// int CtClass
public static CtClass intType;
// long CtClass
public static CtClass longType;
// float CtClass
public static CtClass floatType;
// double CtClass
public static CtClass doubleType;
// void CtClass
public static CtClass voidType;
```

其他类，需要传类的全限类名

```
CtClass s1 = pool.get(CtClass.shortType.getName());
CtClass s2 = CtClass.shortType;
CtClass o = pool.get(CtClass.javaLangObject);
CtClass i = pool.get("java.lang.Integer");
```

- makeClass()

创建一个新的公共类，如果已经存在同名的类，则新创建类覆盖之前的类，默认没有任何属性和方法

```
CtClass ctClass = pool.makeClass("com.shadow.HelloWorld");
```

- makeInterface()

创建一个新的公共接口，如果已经存在同名的接口，新的接口覆盖以前的接口

```
CtClass anInterface = pool.makeInterface("com.shadow.IHandle");
```

- toClass()

将 CtClass 转为 java 的 Class 对象

> 注意：由于 ClassPool 会缓存，因此大量的 CtClass 对象会消耗内存，可调用 CtClass 的 detach() 方法来将其从 ClassPool 缓存中移除

2、CtClass

1. 基础属性

```
// 类的全限类名，如：java.lang.Integer
protected String qualifiedName;
// Object
static final String javaLangObject = "java.lang.Object";
// boolean 类型的 CtClass
public static CtClass booleanType;
// char 类型的 CtClass
public static CtClass charType;
// byte 类型的 CtClass
public static CtClass byteType;
// short 类型的 CtClass
public static CtClass shortType;
// int 类型的 CtClass
public static CtClass intType;
// long 类型的 CtClass
public static CtClass longType;
// float 类型的 CtClass
public static CtClass floatType;
// double 类型的 CtClass
public static CtClass doubleType;
// void 类型的 CtClass
public static CtClass voidType;
// 存储以上9个 CtClass 类型的数组对象
static CtClass[] primitiveTypes;
```

2. 常用方法

- addField()

给类添加属性字段，关注 CtField.make()

```
// new CtField 的方式创建字段
// 1、添加常量 name
CtField ctField = new CtField(pool.get(String.class.getName()), "name",
ctClass);
// public static final
ctField.setModifiers(Modifier.PUBLIC + Modifier.STATIC +
Modifier.FINAL);
// 添加到类中并设置值
ctClass.addField(ctField, CtField.Initializer.constant("javassist"));
// 2、添加静态变量 args
ctField = new CtField(pool.get(Object[].class.getName()), "args",
ctClass);
// public static
ctField.setModifiers(Modifier.PUBLIC + Modifier.STATIC);
ctClass.addField(ctField);
// 3、添加成员变量 count
ctField = new CtField(pool.get(int.class.getName()), "count", ctClass);
// private
```

```
ctField.setModifiers(Modifier.PRIVATE);
ctClass.addField(ctField);
// CtField.make() 的方式创建字段
ctField = CtField.make("private String password;", ctClass);
ctClass.addField(ctField);
// 4、添加带注解的属性
// ctField = new CtField(pool.get(ApplicationContext.class.getName()),
"count", ctClass);
ctField = CtField.make("private
org.springframework.context.ApplicationContext ioc;", ctClass);
ctClass.addField(ctField);
// Lorg/springframework/context/ApplicationContext;
System.out.println(ctField.getSignature());
ctField = ctClass.getDeclaredField("ioc");
List<AttributeInfo> attributes =
ctField.getFieldInfo().getAttributes();
AnnotationsAttribute annotationsAttribute = !attributes.isEmpty() ?
(AnnotationsAttribute) attributes.get(0) :
new AnnotationsAttribute(ctField.getFieldInfo().getConstPool(),
AnnotationsAttribute.visibleTag);
Annotation annotation = new
Annotation("org.springframework.beans.factory.annotation.Autowired",
ctField.getFieldInfo().getConstPool());
annotationsAttribute.addAnnotation(annotation);
ctField.getFieldInfo().addAttribute(annotationsAttribute);
```

- addMethod()

给类添加方法，关注 CtMethod.make()

```
// new CtMethod 的方式创建方法
// 1、添加静态方法
// $1、$2...表示获取方法参数的第一个、第二个...参数值，$e 表示获取异常参数值
// void main(String[],String)
CtMethod mainMethod = new CtMethod(CtClass.voidType, "main", new
CtClass[]{pool.get(String[].class.getName()),
pool.get(String.class.getName())}, ctClass);
// public static
mainMethod.setModifiers(Modifier.PUBLIC + Modifier.STATIC);
mainMethod.setBody("{System.out.println(\"javassist hello world!\" + $2
+ name);}");
// 定义局部变量 int age
mainMethod.addLocalVariable("age", CtClass.intType);
// 插入到方法最前面
mainMethod.insertBefore("{age = 100; System.out.println(\" age = \" +
age);}");
// 添加异常捕获
mainMethod.addCatch("{throw $e;}", pool.get("java.lang.Exception"));
ctClass.addMethod(mainMethod);
// 2、添加实例方法 boolean check(int)
CtMethod checkMethod = new CtMethod(CtClass.booleanType, "check", new
CtClass[]{pool.get(int.class.getName())}, ctClass);
// public
checkMethod.setModifiers(Modifier.PUBLIC);
checkMethod.setBody("{System.out.println(\"javassist hello world!\" +
$1); return $1 > 0;}");
ctClass.addMethod(checkMethod);
```

```
// CtMethod.make() 的方式创建字段
CtMethod testMethod = CtMethod.make("private String test(String
username) { return \"this is a test method\" + $1;}", ctClass);
ctClass.addMethod(testMethod);
```

- addConstructor()

给类添加构造方法

```
// 1、添加无参构造方法
CtConstructor ctConstructor = new CtConstructor(new CtClass[]{},
ctClass);
ctConstructor.setBody("{}");
ctClass.addConstructor(ctConstructor);
// 2、添加有参构造方法
CtConstructor constructor = new CtConstructor(new CtClass[]
{CtClass.intType}, ctClass);
constructor.setBody("{count = $1;}");
ctClass.addConstructor(constructor);
```

- addInterface()

给类添加接口实现

```
// 添加接口
CtClass cloneInterface = pool.get("java.lang.Cloneable");
ctClass.addInterface(cloneInterface);
```

- toClass()

将 CtClass 对象转为 Class 对象

注意：此方法调用后会冻结 CtClass 对象，不能再去修改信息，如果想再次修改，需要调用 defrost() 方法

```
Class<?> helloClass = ctClass.toClass();
// 解冻
ctClass.defrost();
```

- writeFile()

将字节码写入文件，方法内部调用 toBytecode()，这两个方法调用后也会冻结 CtClass 对象，如果想再次修改 CtClass ，需要调用 defrost() 方法

3、CtField

1. 基础属性

```
static final String javaLangString = "java.lang.String";
// 字段信息，对应字节码 field_info
protected FieldInfo fieldInfo;
```

2. 常用方法

- make()

编译给定的源代码并创建一个字段，可以将生成的字段加入到 CtClass 中（通过 CtClass 的 addField() ）

```
CtField ctField = CtField.make("private String password;", ctClass);
```

- setModifiers()

javassist.Modifier 设置字段的修饰符：public 、private、protected、static...

```
CtField ctField = new CtField(pool.get(Object[].class.getName()),
"args", ctClass);
// public static
ctField.setModifiers(Modifier.PUBLIC + Modifier.STATIC);
```

注：CtField 有多种重载的构造方法，根据需要使用

4、CtMethod & CtConstructor

1. 基础属性

```
// 父类中的属性，对应字节码 method_info
protected MethodInfo methodInfo;
```

2. 常用方法

- make()

编译给定的源代码并创建一个方法，可以将生成的方法加入到 CtClass 中（通过 CtClass 的 addMethod() 或 addConstructor()）

```
CtMethod logMethod = new CtMethod(CtClass.voidType, "log", new
CtClass[]{pool.get(String.class.getName()), CtClass.intType}, ctClass);
```

- setModifiers()

javassist.Modifier 设置方法的修饰符：public 、private、protected、static...

```
CtMethod logMethod = new CtMethod(CtClass.voidType, "log", new
CtClass[]{pool.get(String.class.getName()), CtClass.intType}, ctClass);
logMethod.setModifiers(Modifier.PUBLIC | Modifier.STATIC);
```

- setBody()

设置方法体

```
CtMethod logMethod = new CtMethod(CtClass.voidType, "log", new
CtClass[]{pool.get(String.class.getName()), CtClass.intType}, ctClass);
logMethod.setModifiers(Modifier.PUBLIC | Modifier.STATIC);
logMethod.setBody("{System.out.println(\"this is a log params = \" + $1
+ \" & \" + $2);}");
```

- insertBefore()

在方法开始处插入字节码

```
CtMethod logMethod = new CtMethod(CtClass.voidType, "log", new
CtClass[]{pool.get(String.class.getName()), CtClass.intType}, ctClass);
logMethod.setModifiers(Modifier.PUBLIC | Modifier.STATIC);
logMethod.setBody("{System.out.println(\"this is a log params = \" + $1
+ \" & \" + $2);}");
logMethod.addLocalVariable("start", CtClass.longType);
// 插入方法最开始处
logMethod.insertBefore("start = System.currentTimeMillis();");
```

- insertAfter()

在方法结尾处插入字节码

```
CtMethod logMethod = new CtMethod(CtClass.voidType, "log", new
CtClass[]{pool.get(String.class.getName()), CtClass.intType}, ctClass);
logMethod.setModifiers(Modifier.PUBLIC | Modifier.STATIC);
logMethod.setBody("{System.out.println(\"this is a log params = \" + $1
+ \" & \" + $2);}");
logMethod.addLocalVariable("start", CtClass.longType);
logMethod.insertBefore("start = System.currentTimeMillis();");
// 插入到方法结尾处
logMethod.insertAfter("System.out.println(\"cost time : \" +
(System.currentTimeMillis() - start));");
```

- addLocalVariable()

添加局部变量

```
CtMethod logMethod = new CtMethod(CtClass.voidType, "log", new
CtClass[]{pool.get(String.class.getName()), CtClass.intType}, ctClass);
logMethod.setModifiers(Modifier.PUBLIC | Modifier.STATIC);
logMethod.setBody("{System.out.println(\"this is a log params = \" + $1
+ \" & \" + $2);}");
// 添加局部变量 start
logMethod.addLocalVariable("start", CtClass.longType);
logMethod.insertBefore("start = System.currentTimeMillis();");
logMethod.insertAfter("System.out.println(\"cost time : \" +
(System.currentTimeMillis() - start));");
```

- addCatch()

添加 try...catch 语句块

```
CtMethod logMethod = new CtMethod(CtClass.voidType, "log", new
CtClass[]{pool.get(String.class.getName()), CtClass.intType}, ctClass);
logMethod.setModifiers(Modifier.PUBLIC | Modifier.STATIC);
logMethod.setBody("{System.out.println(\"this is a log params = \" + $1
+ \" & \" + $2);}");
logMethod.addLocalVariable("start", CtClass.longType);
logMethod.insertBefore("start = System.currentTimeMillis();");
logMethod.insertAfter("System.out.println(\"cost time : \" +
(System.currentTimeMillis() - start));");
// try...catch
logMethod.addCatch("{System.out.println(\"error : \" +
$e.getMessage()); $e.printStackTrace(); throw $e;}",
pool.get("java.lang.Exception"));
```

- addParameter()

给方法添加一个参数，末尾的位置上追加，可以多次添加

```
CtMethod logMethod = new CtMethod(CtClass.voidType, "log", new
CtClass[]{pool.get(String.class.getName()), CtClass.intType}, ctClass);
logMethod.setModifiers(Modifier.PUBLIC | Modifier.STATIC);
logMethod.setBody("{System.out.println(\"this is a log params = \" + $1
+ \" & \" + $2);}");
logMethod.addLocalVariable("start", CtClass.longType);
logMethod.insertBefore("start = System.currentTimeMillis();");
logMethod.insertAfter("System.out.println(\"cost time : \" +
(System.currentTimeMillis() - start));");
logMethod.addCatch("{System.out.println(\"error : \" +
$e.getMessage()); $e.printStackTrace(); throw $e;}",
pool.get("java.lang.Exception"));
// 方法添加参数
logMethod.addParameter(CtClass.booleanType);
logMethod.addParameter(pool.get("java.lang.String"));
```

5、HotSwapper

动态重新加载类的实用程序类 java 平台调试器架构（JPDA），或 HotSwap，它只适用于JDK1.4及更高版本

1. 使用方法

   java 服务启动需要开启 JDWP 服务

```
-agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=8000
```

2. 测试方法类

```
public class ApiTest {
    public String query(String account, int x) {
        return account + " 账号余额:  " + (new Random().nextInt(10000) +
1 + x) + " 元";
    }
}
```

3. 测试类，加入启动 vm 参数：-agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=8000

```
public class TestHotSwap {
    public static void main(String[] args) throws Exception {
        ClassPool pool = ClassPool.getDefault();
        ApiTest apiTest = new ApiTest();
        // 启动一个线程模拟查询
        new Thread(() -> {
            for (; ; ) {
                System.out.println(apiTest.query("oracle", 10));
                try {
                    Thread.sleep(2000);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
```

```
        }).start();

        // 开启热交换服务
        HotSwapper hotSwapper = new HotSwapper(8000);
        CtClass ctClass = pool.get(ApiTest.class.getName());

        CtMethod ctMethod = ctClass.getDeclaredMethod("query", new
CtClass[]{pool.get("java.lang.String"), CtClass.intType});
        // 重写方法体
        ctMethod.setBody("{return $1 + \"hacker 余额：\" + $2 + \" 美元
\";}");
        hotSwapper.reload(ApiTest.class.getName(),
ctClass.toBytecode());
    }
}
```

三、使用案列

1、方法执行耗时统计

1. 测试方法类

```
public class TestController {

    public void testTimer(String name, int age) throws Exception {
        // service logic
        Thread.sleep(2000);
        System.out.println("name:" + name + " age:" + age);
    }
}
```

2. 测试类

```
public class TimerTest {
    public static void main(String[] args) throws Exception {
        ClassPool pool = ClassPool.getDefault();
        // 1、得到 CtClass
        CtClass ctClass = pool.get(TestController.class.getName());
        // 2、得到 CtMethod
        CtMethod ctMethod = ctClass.getDeclaredMethod("testTimer", new
CtClass[]{pool.get(String.class.getName()), CtClass.intType});
        // 3、添加局部变量
        ctMethod.addLocalVariable("start", CtClass.longType);
        // 4、插入到方法最开始
        ctMethod.insertBefore("{start = System.currentTimeMillis();}");
        // 5、出入到方法末尾
        ctMethod.insertAfter("{System.out.println(\"执行耗时：\" +
(System.currentTimeMillis() - start));}");
        // 6、写入文件
        ctClass.writeFile();
        // 7、模拟方法调用
        TestController testController = new TestController();
        new Thread(() -> {
            for (; ; ) {
                try {
                    testController.testTimer("测试", 20);
                    Thread.sleep(2000);
```

```
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        }).start();
        // 8、热交换
        // vm 参数：-
agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=8000
        HotSwapper hotSwapper = new HotSwapper(8000);
        Thread.sleep(2000);
        hotSwapper.reload(TestController.class.getName(),
ctClass.toBytecode());
    }
}
```

2、获取方法参数

1. 测试方法类

```
public class TestController {

    public boolean test(String name, int age, long salary) throws Exception
{
        // service logic
        Thread.sleep(2000);
        System.out.println("name:" + name + " age:" + age + " salary:" +
salary);
        return salary > 100;
    }
}
```

2. 测试类

```
public class TestParam {
    public static void main(String[] args) throws Exception{
        Class<TestController> testControllerClass = TestController.class;
        String className = testControllerClass.getName();
        String methodName = "test";
        Object[] params = {"shadow", 22, 6666};
        Map<String, Object> map = getParamMap(testControllerClass,
className, methodName, params);
        map.forEach((k, v) -> System.out.println(k + " = " + v));
    }

    private static Map<String, Object> getParamMap(Class<?> aClass, String
className, String methodName, Object[] params) throws Exception {
        Map<String, Object> map = new HashMap<>();
        // 1、获取类
        ClassPool pool = ClassPool.getDefault();
        ClassPath classPath = new ClassClassPath(aClass);
        pool.insertClassPath(classPath);
        // 1.1 通过类名获取类信息
        CtClass ctClass = pool.get(className);
        // 1.2 获取方法
        CtMethod cm = ctClass.getDeclaredMethod(methodName);
```

```
        // 1.3 方法信息
        MethodInfo methodInfo = cm.getMethodInfo();
        // 1.4 方法：入参信息，名称和类型
        CodeAttribute codeAttribute = methodInfo.getCodeAttribute();
        // LocalVariableAttribute 获取方法的入参的名称
        LocalVariableAttribute attribute = (LocalVariableAttribute)
codeAttribute.getAttribute(LocalVariableAttribute.tag);
        if (attribute == null) {
            return map;
        }
        // 判断方法是静态方法还是实例方法
        // 1-非静态方法 0-静态方法
        int pos = Modifier.isStatic(cm.getModifiers()) ? 0 : 1;
        // 非静态方法下标0的参数是 this
        if (pos == 1 && !attribute.variableName(0).equals("this")) {
            while (true) {
                if (attribute.variableName(pos++).equals("this"))
                    break;
            }
        }
        // parameterTypes 获取方法入参的类型
        for (int i = 0; i < cm.getParameterTypes().length; i++) {
            map.put(attribute.variableName(i + pos), params[i]);
        }
        return map;
    }
}
```