**<http://www.cnblogs.com/sunfie/p/5154246.html>**

[**Java学习之javassist**](https://www.cnblogs.com/sunfie/p/5154246.html)

**1、读取和输出字节码**

[复制代码](javascript:void(0);)

1 ClassPool pool = ClassPool.getDefault();

2 //会从classpath中查询该类

3 CtClass cc = pool.get("test.Rectangle");

4 //设置.Rectangle的父类

5 cc.setSuperclass(pool.get("test.Point"));

6 //输出.Rectangle.class文件到该目录中

7 cc.writeFile("c://");

8 //输出成二进制格式

9 //byte[] b=cc.toBytecode();

10 //输出并加载class 类，默认加载到当前线程的ClassLoader中，也可以选择输出的ClassLoader。

11 //Class clazz=cc.toClass();

[复制代码](javascript:void(0);)

这里可以看出，Javassist的加载是依靠ClassPool类，输出方式支持三种。

**2、新增Class**

1 ClassPool pool = ClassPool.getDefault();

2 CtClass cc = pool.makeClass("Point");

3 //新增方法

4 cc.addMethod(m);

5 //新增Field

6 cc.addField(f);

从上面可以看出，对Class的修改主要是依赖于CtClass类。API也比较清楚和简单。

**3、冻结Class**

    当CtClass 调用writeFile()、toClass()、toBytecode() 这些方法的时候，Javassist会冻结CtClass Object，对CtClass object的修改将不允许。这个主要是为了警告开发者该类已经被加载，而JVM是不允许重新加载该类的。如果要突破该限制，方法如下：

1 CtClasss cc = ...;

2 :

3 cc.writeFile();

4 cc.defrost(); //解冻

5 cc.setSuperclass(...); // OK since the class is not frozen.(ok,因为这个类没有被冻结)

    当 ClassPool.doPruning=true的时候，Javassist 在CtClass object被冻结时，会释放存储在ClassPool对应的数据。这样做可以减少javassist的内存消耗。默认情况ClassPool.doPruning=false。例如

1 CtClasss cc = ...;

2 cc.stopPruning(true);

3 :

4 cc.writeFile(); // convert to a class file.

5 // cc没有被释放

提示：当调试时，可以调用debugWriteFile()，该方法不会导致CtClass被释放。

**4、Class 搜索路径**

    从上面可以看出Class 的载入是依靠ClassPool，而ClassPool.getDefault() 方法的搜索Classpath 只是搜索JVM的同路径下的class。当一个程序运行在JBoss或者Tomcat下，ClassPool Object 可能找到用户的classes。Javassist 提供了四种动态加载classpath的方法。如下

[复制代码](javascript:void(0);)

1 //默认加载方式如pool.insertClassPath(new ClassClassPath(this.getClass()));

2 ClassPool pool = ClassPool.getDefault();

3 //从file加载classpath

4 pool.insertClassPath("/usr/local/javalib")

5 //从URL中加载

6 ClassPath cp = new URLClassPath("www.javassist.org", 80, "/java/", "org.javassist.");

7 pool.insertClassPath(cp);

8 //从byte[] 中加载

9 byte[] b = a byte array;

10 String name = class name;

11 cp.insertClassPath(new ByteArrayClassPath(name, b));

12 //可以从输入流中加载class

13 InputStream ins = an input stream for reading a class file;

14 CtClass cc = cp.makeClass(ins);

[复制代码](javascript:void(0);)

**5、ClassPool**

**5.1 减少内存溢出**

     ClassPool是一个CtClass objects的装载容器,当加载了CtClass object后，是不会被ClassPool释放的（默认情况下）,这个是因为CtClass object 有可能在下个阶段会被用到,当加载过多的CtClass object的时候，会造成OutOfMemory的异常。为了避免这个异常，javassist提供几种方法，一种是在上面提到的 ClassPool.doPruning这个参数，还有一种方法是调用CtClass.detach()方法，可以把CtClass object 从ClassPool中移除。例如：

1 CtClass cc = ... ;

2 cc.writeFile();

3 cc.detach();

    另外一种方法是不用默认的ClassPool即不用 ClassPool.getDefault()这个方式来生成,这样当ClassPool没被引用的时候，JVM的垃圾收集会收集该类。例如

1 //ClassPool(true) 会默认加载Jvm的ClassPath

2 ClassPool cp = new ClassPool(true);

3 // if needed, append an extra search path by appendClassPath()

**5.2  级联ClassPools**

     javassist支持级联的ClassPool，即类似于继承。例如：

1 ClassPool parent = ClassPool.getDefault();

2 ClassPool child = new ClassPool(parent);

3 child.insertClassPath("./classes");

**5.3 修改已有Class的name以创建一个新的Class**

    当调用setName方法时，会直接修改已有的Class的类名，如果再次使用旧的类名，则会重新在classpath路径下加载。例如：

1 ClassPool pool = ClassPool.getDefault();

2 CtClass cc = pool.get("Point");

3 cc.setName("Pair");

4 //重新在classpath加载

5 CtClass cc1 = pool.get("Point");

    对于一个被冻结（Frozen)的CtClass object ，是不可以修改class name的，如果需要修改，则可以重新加载，例如：

1 ClassPool pool = ClassPool.getDefault();

2 CtClass cc = pool.get("Point");

3 cc.writeFile(); // has frozened

4 //cc.setName("Pair"); wrong since writeFile() has been called.

5 CtClass cc2 = pool.getAndRename("Point", "Pair");

**6、Class loader**

    上面也提到，javassist同个Class是不能在同个ClassLoader中加载两次的,所以在输出CtClass的时候需要注意下，例如：

[复制代码](javascript:void(0);)

1 // 当Hello未加载的时候，下面是可以运行的。

2 ClassPool cp = ClassPool.getDefault();

3 CtClass cc = cp.get("Hello");

4 Class c = cc.toClass();

5 //下面这种情况，由于Hello2已加载，所以会出错

6 Hello2 h=new Hello2();

7 CtClass cc2 = cp.get("Hello2");

8 Class c2 = cc.toClass();//这里会抛出java.lang.LinkageError 异常

9 //解决加载问题，可以指定一个未加载的ClassLoader

10 Class c3 = cc.toClass(new MyClassLoader());

[复制代码](javascript:void(0);)

**6.1 使用javassist.Loader**

    从上面可以看到，如果在同一个ClassLoader加载两次Class抛出异常，为了方便javassist也提供一个Classloader供使用，例如

1 ClassPool pool = ClassPool.getDefault();

2 Loader cl = new Loader(pool);

3 CtClass ct = pool.get("test.Rectangle");

4 ct.setSuperclass(pool.get("test.Point"));

5 Class c = cl.loadClass("test.Rectangle");

6 Object rect = c.newInstance();        :

   为了方便监听Javassist自带的ClassLoader的生命周期，javassist也提供了一个listener，可以监听ClassLoader的生命周期，例如：

[复制代码](javascript:void(0);)

1 //Translator 为监听器

2 public class MyTranslator implements Translator {

3 void start(ClassPool pool)

4 throws NotFoundException, CannotCompileException {}

5 void onLoad(ClassPool pool, String classname)

6 throws NotFoundException, CannotCompileException

7 {

8 CtClass cc = pool.get(classname);

9 cc.setModifiers(Modifier.PUBLIC);

10 }

11 }

12 //示例

13 public class Main2 {

14 public static void main(String[] args) throws Throwable {

15 Translator t = new MyTranslator();

16 ClassPool pool = ClassPool.getDefault();

17 Loader cl = new Loader();

18 cl.addTranslator(pool, t);

19 cl.run("MyApp", args);

20 }

21 }

22 //输出

23 % java Main2 arg1 arg2...

[复制代码](javascript:void(0);)

**6.2 修改系统Class**

    由JVM规范可知，system classloader 是比其他classloader 是优先加载的，而system classloader 主要是加载系统Class，所以要修改系统Class，如果默认参数运行程序是不可能修改的,如果需要修改也有一些办法，即在运行时加入-Xbootclasspath/p: 参数的意义可以参考其他文件。下面修改String的例子如下：

[复制代码](javascript:void(0);)

ClassPool pool = ClassPool.getDefault();

CtClass cc = pool.get("java.lang.String");

CtField f = new CtField(CtClass.intType, "hiddenValue", cc);

f.setModifiers(Modifier.PUBLIC);

cc.addField(f);

cc.writeFile(".");

//运行脚本

% java -Xbootclasspath/p:. MyApp arg1 arg2...

[复制代码](javascript:void(0);)

**6.3 动态重载Class**

    如果JVM运行时开启JPDA（Java Platform Debugger Architecture），则Class是运行被动态重新载入的。具体方式可以参考java.lang.Instrument。javassist也提供了一个运行期重载Class的方法，具体可以看API 中的javassist.tools.HotSwapper。

**7、Introspection和定制**

    javassist封装了很多很方便的方法以供使用，大部分使用只需要用这些API即可，如果不能满足，Javassist也提供了一个低层的API（具体参考javassist.bytecode 包）来修改原始的Class。

**7.1 插入source 文本在方法体前或者后**

     CtMethod 和CtConstructor 提供了 insertBefore()、insertAfter()和 addCatch()方法，它们可以插入一个souce文本到存在的方法的相应的位置。javassist 包含了一个简单的编译器解析这souce文本成二进制插入到相应的方法体里。javassist 还支持插入一个代码段到指定的行数，前提是该行数需要在class 文件里含有。插入的source 可以关联fields 和methods，也可以关联方法的参数。但是关联方法参数的时，需要在程序编译时加上 -g 选项（该选项可以把本地变量的声明保存在class 文件中，默认是不加这个参数的。）。因为默认一般不加这个参数，所以Javassist也提供了一些特殊的变量来代表方法参数：$1,$2,$args...要注意的是，插入的source文本中不能引用方法本地变量的声明，但是可以允许声明一个新的方法本地变量，除非在程序编译时加入-g选项。方法的特殊变量说明：

|  |  |
| --- | --- |
| $0, $1, $2, ... | this and actual parameters |
| $args | An array of parameters. The type of $args is Object[]. |
| $$ | All actual parameters.For example, m($$) is equivalent to m($1,$2,...) |
| $cflow(...) | cflow variable |
| $r | The result type. It is used in a cast expression. |
| $w | The wrapper type. It is used in a cast expression. |
| $\_ | The resulting value |
| $sig | An array of java.lang.Class objects representing the formal parameter types |
| $type | A java.lang.Class object representing the formal result type. |
| $class | A java.lang.Class object representing the class currently edited. |

**7.1.1 $0, $1, $2, ...**

   $0代码的是this，$1代表方法参数的第一个参数、$2代表方法参数的第二个参数,以此类推，$N代表是方法参数的第N个。例如：

1 //实际方法

2 void move(int dx, int dy)

3 //javassist

4 CtMethod m = cc.getDeclaredMethod("move");

5 //打印dx，和i

6 m.insertBefore("{ System.out.println($1); System.out.println($2); }");

注意：如果javassist改变了$1的值，那实际参数值也会改变。

**7.1.2 $args**

    $args 指的是方法所有参数的数组，类似Object[]，如果参数中含有基本类型，则会转成其包装类型。需要注意的时候，$args[0]对应的是$1,而不是$0，$0!=$args[0]，$0=this。

**7.1.3 $$**

    $$是所有方法参数的简写，主要用在方法调用上。例如：

1 //原方法

2 move(String a,String b)

3 move($$) 相当于move($1,$2)

4 如果新增一个方法，方法含有move的所有参数，则可以这些写：

5 exMove($$, context) 相当于 exMove($1, $2, context)

**7.1.4 $cflow**

 $cflow意思为控制流（control flow），是一个只读的变量，值为一个方法调用的深度。例如：

[复制代码](javascript:void(0);)

1 //原方法

2 int fact(int n) {

3 if (n <= 1)

4 return n;

5 else

6 return n \* fact(n - 1);

7 }

8 //javassist调用

9 CtMethod cm = ...;

10 //这里代表使用了cflow

11 cm.useCflow("fact");

12 //这里用了cflow，说明当深度为0的时候，就是开始当第一次调用fact的方法的时候，打印方法的第一个参数

13 cm.insertBefore("if ($cflow(fact) == 0)"

14 + " System.out.println(\"fact \" + $1);");

[复制代码](javascript:void(0);)

**7.1.5 $r**

   指的是方法返回值的类型，主要用在类型的转型上。例如：

Object result = ... ;

$\_ = ($r)result;

如果返回值为基本类型的包装类型，则该值会自动转成基本类型，如返回值为Integer，则$r为int。如果返回值为void，则该值为null。

**7.1.6 $w**

$w代表一个包装类型。主要用在转型上。比如：Integer i = ($w)5; 如果该类型不是基本类型，则会忽略。

**7.1.7 $\_**

$\_代表的是方法的返回值。

**7.1.8 $sig**

$sig指的是方法参数的类型（Class）数组，数组的顺序为参数的顺序。

**7.1.9 $class**

$class 指的是this的类型（Class）。也就是$0的类型。

**7.1.10 addCatch()**

   addCatch() 指的是在方法中加入try catch 块，需要注意的是，必须在插入的代码中，加入return 值。$e代表异常值。比如：

1 CtMethod m = ...;

2 CtClass etype = ClassPool.getDefault().get("java.io.IOException");

3 m.addCatch("{ System.out.println($e); throw $e; }", etype);

实际代码如下：

[复制代码](javascript:void(0);)

1 try {

2 the original method body

3 }

4 catch (java.io.IOException e) {

5 System.out.println(e);

6 throw e;

7 }

[复制代码](javascript:void(0);)

**8、修改方法体**

CtMethod 和CtConstructor 提供了 setBody() 的方法，可以替换方法或者构造函数里的所有内容。

支持的变量有：

|  |  |
| --- | --- |
| $0, $1, $2, ... | this and actual parameters |
| $args | An array of parameters. The type of $args is Object[]. |
| $$ | All actual parameters.For example, m($$) is equivalent to m($1,$2,...) |
| $cflow(...) | cflow variable |
| $r | The result type. It is used in a cast expression. |
| $w | The wrapper type. It is used in a cast expression. |
| $sig | An array of java.lang.Class objects representing the formal parameter types |
| $type | A java.lang.Class object representing the formal result type. |
| $class | A java.lang.Class object representing the class currently edited. |

注意 $\_变量不支持。

**8.1 替换方法中存在的source**

javassist 允许修改方法里的其中一个表达式，javassist.expr.ExprEditor 这个class 可以替换该表达式。例如：

[复制代码](javascript:void(0);)

1 CtMethod cm = ... ;

2 cm.instrument(

3 new ExprEditor() {

4 public void edit(MethodCall m)

5 throws CannotCompileException

6 {

7 if (m.getClassName().equals("Point")

8 && m.getMethodName().equals("move"))

9 m.replace("{ $1 = 0; $\_ = $proceed($$); }");

10 }

11 });

[复制代码](javascript:void(0);)

    注意： that the substituted code is not an expression but a statement or a block. It cannot be or contain a try-catch statement.

方法instrument() 可以用来搜索方法体里的内容。比如调用一个方法，field访问，对象创建等。如果你想在某个表达式前后插入方法，则修改的souce如下：

{ before-statements;

  $\_ = $proceed($$);

  after-statements; }

**8.2 javassist.expr.MethodCall**

MethodCall代表的是一个方法的调用。用replace()方法可以对调用的方法进行替换。

|  |  |
| --- | --- |
| $0 | The target object of the method call. This is not equivalent to this, which represents the caller-side this object. $0 is null if the method is static. |
| $1, $2, ... | The parameters of the method call. |
| $\_ | The resulting value of the method call. |
| $r | The result type of the method call. |
| $class | A java.lang.Class object representing the class declaring the method. |
| $sig | An array of java.lang.Class objects representing the formal parameter types |
| $type | A java.lang.Class object representing the formal result type. |
| $proceed | The name of the method originally called in the expression. |

注意：$w, $args 和 $$也是允许的。$0不是this，是只调用方法的Object。$proceed指的是一个特殊的语法，而不是一个String。

**8.3 javassist.expr.ConstructorCall**

ConstructorCall 指的是一个构造函数，比如：this()、super()的调用。ConstructorCall.replace()是用来用替换一个块当调用构造方法的时候。

|  |  |
| --- | --- |
| $0 | The target object of the constructor call. This is equivalent to this. |
| $1, $2, ... | The parameters of the constructor call. |
| $class | A java.lang.Class object representing the class declaring the constructor. |
| $sig | An array of java.lang.Class objects representing the formal parameter types. |
| $proceed | The name of the constructor originally called in the expression. |

$w, $args 和 $$  也是允许的。

**8.4 javassist.expr.FieldAccess**

FieldAccess代表的是Field的访问类。

|  |  |
| --- | --- |
| $0 | The object containing the field accessed by the expression. This is not equivalent to this. this represents the object that the method including the expression is invoked on. $0 is null if the field is static. |
| $1 | The value that would be stored in the field if the expression is write access. Otherwise, $1 is not available. |
| $\_ | The resulting value of the field access if the expression is read access. Otherwise, the value stored in $\_ is discarded. |
| $r | The type of the field if the expression is read access. Otherwise, $r is void. |
| $class | A java.lang.Class object representing the class declaring the field. |
| $type | A java.lang.Class object representing the field type. |
| $proceed | The name of a virtual method executing the original field access. . |

$w, $args 和 $$  也是允许的。

**8.5 javassist.expr.NewExpr**

NewExpr代表的是一个Object 的操作(但不包括数组的创建)。

|  |  |
| --- | --- |
| $0 | null |
| $1, $2, ... | The parameters to the constructor. |
| $\_ | The resulting value of the object creation. A newly created object must be stored in this variable. |
| $r | The type of the created object. |
| $sig | An array of java.lang.Class objects representing the formal parameter types |
| $type | A java.lang.Class object representing the class of the created object. |
| $proceed | The name of a virtual method executing the original object creation. . |

$w, $args 和 $$  也是允许的。

**8.6 javassist.expr.NewArray**

NewArray 代表的是数组的创建。

|  |  |
| --- | --- |
| $0 | null |
| $1, $2, ... | The size of each dimension. |
| $\_ | The resulting value of the object creation.  A newly created array must be stored in this variable. |
| $r | The type of the created object. |
| $type | A java.lang.Class object representing the class of the created array . |
| $proceed | The name of a virtual method executing the original array creation. . |

$w, $args 和 $$  也是允许的。

例如：

String[][] s = new String[3][4];

 $1 和 $2 的值为 3 和 4, $3 得不到的.

String[][] s = new String[3][];

 $1 的值是 3 ，但 $2 得不到的.

**8.7 javassist.expr.Instanceof**

Instanceof 代表的是Instanceof 表达式。

|  |  |
| --- | --- |
| $0 | null |
| $1 | The value on the left hand side of the original instanceof operator. |
| $\_ | The resulting value of the expression. The type of $\_ is boolean. |
| $r | The type on the right hand side of the instanceof operator. |
| $type | A java.lang.Class object representing the type on the right hand side of the instanceof operator. |
| $proceed | The name of a virtual method executing the original instanceof expression. It takes one parameter (the type is java.lang.Object) and returns true if the parameter value is an instance of the type on the right hand side of the original instanceof operator. Otherwise, it returns false. |

$w, $args 和 $$  也是允许的。

**8.8 javassist.expr.Cast**

Cast 代表的是一个转型表达式。

|  |  |
| --- | --- |
| $0 | null |
| $1 | The value the type of which is explicitly cast. |
| $\_ | The resulting value of the expression. The type of $\_ is the same as the type after the explicit casting, that is, the type surrounded by ( ). |
| $r | the type after the explicit casting, or the type surrounded by ( ). |
| $type | A java.lang.Class object representing the same type as $r. |
| $proceed | The name of a virtual method executing the original type casting. It takes one parameter of the type java.lang.Object and returns it after the explicit type casting specified by the original expression. |

$w, $args 和 $$  也是允许的。

**8.9 javassist.expr.Handler**

Handler 代表的是一个try catch 声明。

|  |  |
| --- | --- |
| $1 | The exception object caught by the catch clause. |
| $r | the type of the exception caught by the catch clause. It is used in a cast expression. |
| $w | The wrapper type. It is used in a cast expression. |
| $type | A java.lang.Class object representing the type of the exception caught by the catch clause. |

**9 新增一个方法或者field**

Javassist 允许开发者创建一个新的方法或者构造方法。新增一个方法，例如：

[复制代码](javascript:void(0);)

1 CtClass point = ClassPool.getDefault().get("Point");

2 CtMethod m = CtNewMethod.make(

3 "public int xmove(int dx) { x += dx; }",

4 point);

5 point.addMethod(m);

6

7 在方法中调用其他方法，例如：

8 CtClass point = ClassPool.getDefault().get("Point");

9 CtMethod m = CtNewMethod.make(

10 "public int ymove(int dy) { $proceed(0, dy); }",

11 point, "this", "move");

12 其效果如下：

13 public int ymove(int dy) { this.move(0, dy); }

[复制代码](javascript:void(0);)

下面是javassist提供另一种新增一个方法（未看明白）：

Javassist provides another way to add a new method. You can first create an abstract method and later give it a method body:

[复制代码](javascript:void(0);)

1 CtClass cc = ... ;

2 CtMethod m = new CtMethod(CtClass.intType, "move",

3 new CtClass[] { CtClass.intType }, cc);

4 cc.addMethod(m);

5 m.setBody("{ x += $1; }");

6 cc.setModifiers(cc.getModifiers() & ~Modifier.ABSTRACT);

7 Since Javassist makes a class abstract if an abstract method is added to the class, you have to explicitly change the class back to a non-abstract one after calling setBody().

[复制代码](javascript:void(0);)

**9.1 递归方法**

[复制代码](javascript:void(0);)

1 CtClass cc = ... ;

2 CtMethod m = CtNewMethod.make("public abstract int m(int i);", cc);

3 CtMethod n = CtNewMethod.make("public abstract int n(int i);", cc);

4 cc.addMethod(m);

5 cc.addMethod(n);

6 m.setBody("{ return ($1 <= 0) ? 1 : (n($1 - 1) \* $1); }");

7 n.setBody("{ return m($1); }");

8 cc.setModifiers(cc.getModifiers() & ~Modifier.ABSTRACT);

[复制代码](javascript:void(0);)

**9.2 新增field**

如下：

[复制代码](javascript:void(0);)

1 CtClass point = ClassPool.getDefault().get("Point");

2 CtField f = new CtField(CtClass.intType, "z", point);

3 point.addField(f);

4 //point.addField(f, "0"); // initial value is 0.

5 或者：

6 CtClass point = ClassPool.getDefault().get("Point");

7 CtField f = CtField.make("public int z = 0;", point);

8 point.addField(f);

[复制代码](javascript:void(0);)

**9.3 移除方法或者field**

1 调用removeField()或者removeMethod()。

**10 注解**

获取注解信息：

[复制代码](javascript:void(0);)

1 //注解

2 public @interface Author {

3 String name();

4 int year();

5 }

6 //javassist代码

7 CtClass cc = ClassPool.getDefault().get("Point");

8 Object[] all = cc.getAnnotations();

9 Author a = (Author)all[0];

10 String name = a.name();

11 int year = a.year();

12 System.out.println("name: " + name + ", year: " + year);

[复制代码](javascript:void(0);)

**11  javassist.runtime**

**12 import**

引用包：

1 ClassPool pool = ClassPool.getDefault();

2 pool.importPackage("java.awt");

3 CtClass cc = pool.makeClass("Test");

4 CtField f = CtField.make("public Point p;", cc);

5 cc.addField(f);

**13 限制**

（1）不支持java5.0的新增语法。不支持注解修改，但可以通过底层的javassist类来解决，具体参考：javassist.bytecode.annotation

（2）不支持数组的初始化，如String[]{"1","2"}，除非只有数组的容量为1

（3）不支持内部类和匿名类

（4）不支持continue和btreak 表达式。

（5）对于继承关系，有些不支持。例如

class A {}

class B extends A {}

class C extends B {}

class X {

    void foo(A a) { .. }

    void foo(B b) { .. }

}

如果调用  x.foo(new C())，可能会调用foo(A) 。

（6）推荐开发者用#分隔一个class name和static method或者 static field。例如：

javassist.CtClass.intType.getName()推荐用javassist.CtClass#intType.getName()

**14.完整实例**

**14.1 创建类实例**

[复制代码](javascript:void(0);)

1 package com.swust.javassist;

2

3 import javassist.ClassPool;

4 import javassist.CtClass;

5 import javassist.CtConstructor;

6 import javassist.CtField;

7 import javassist.CtMethod;

8

9 public class Example1 {

10 public static void main(String[] args) throws Exception {

11 ClassPool pool = ClassPool.getDefault();

12 CtClass cc = pool.makeClass("bean.User");

13

14 //创建属性

15 CtField field01 = CtField.make("private int id;",cc);

16 CtField field02 = CtField.make("private String name;", cc);

17 cc.addField(field01);

18 cc.addField(field02);

19

20 //创建方法

21 CtMethod method01 = CtMethod.make("public String getName(){return name;}", cc);

22 CtMethod method02 = CtMethod.make("public void setName(String name){this.name = name;}", cc);

23 cc.addMethod(method01);

24 cc.addMethod(method02);

25

26 //添加有参构造器

27 CtConstructor constructor = new CtConstructor(new CtClass[]{CtClass.intType,pool.get("java.lang.String")},cc);

28 constructor.setBody("{this.id=id;this.name=name;}");

29 cc.addConstructor(constructor);

30 //无参构造器

31 CtConstructor cons = new CtConstructor(null,cc);

32 cons.setBody("{}");

33 cc.addConstructor(cons);

34

35 cc.writeFile("E:/workspace/TestCompiler/src");

36 }

37 }

[复制代码](javascript:void(0);)

**14.2 访问类实例变量**

[复制代码](javascript:void(0);)

1 package com.swust.javassist;

2

3 import java.lang.reflect.Field;

4 import java.lang.reflect.Method;

5 import java.util.Arrays;

6

7 import javassist.ClassPool;

8 import javassist.CtClass;

9 import javassist.CtConstructor;

10 import javassist.CtField;

11 import javassist.CtMethod;

12 import javassist.CtNewMethod;

13 import javassist.Modifier;

14

15 public class Example2 {

16 //获取类的简单信息

17 public static void test01() throws Exception{

18 ClassPool pool = ClassPool.getDefault();

19 CtClass cc = pool.get("com.swust.beans.Person");

20 //得到字节码

21 byte[] bytes = cc.toBytecode();

22 System.out.println(Arrays.toString(bytes));

23 System.out.println(cc.getName());//获取类名

24 System.out.println(cc.getSimpleName());//获取简要类名

25 System.out.println(cc.getSuperclass());//获取父类

26 System.out.println(cc.getInterfaces());//获取接口

27 System.out.println(cc.getMethods());//获取

28 }

29 //新生成一个方法

30 public static void test02() throws Exception{

31 ClassPool pool = ClassPool.getDefault();

32 CtClass cc = pool.get("com.swust.beans.Person");

33 //第一种

34 //CtMethod cm = CtMethod.make("public String getName(){return name;}", cc);

35 //第二种

36 //参数：返回值类型，方法名，参数，对象

37 CtMethod cm = new CtMethod(CtClass.intType,"add",new CtClass[]{CtClass.intType,CtClass.intType},cc);

38 cm.setModifiers(Modifier.PUBLIC);//访问范围

39 cm.setBody("{return $1+$2;}");

40 //cc.removeMethod(m) 删除一个方法

41 cc.addMethod(cm);

42 //通过反射调用方法

43 Class clazz = cc.toClass();

44 Object obj = clazz.newInstance();//通过调用无参构造器，生成新的对象

45 Method m = clazz.getDeclaredMethod("add", int.class,int.class);

46 Object result = m.invoke(obj, 2,3);

47 System.out.println(result);

48 }

49

50 //修改已有的方法

51 public static void test03() throws Exception{

52 ClassPool pool = ClassPool.getDefault();

53 CtClass cc = pool.get("bean.User");

54

55 CtMethod cm = cc.getDeclaredMethod("hello",new CtClass[]{pool.get("java.lang.String")});

56 cm.insertBefore("System.out.println(\"调用前\");");//调用前

57 cm.insertAt(29, "System.out.println(\"29\");");//行号

58 cm.insertAfter("System.out.println(\"调用后\");");//调用后

59

60 //通过反射调用方法

61 Class clazz = cc.toClass();

62 Object obj = clazz.newInstance();

63 Method m = clazz.getDeclaredMethod("hello", String.class);

64 Object result = m.invoke(obj, "张三");

65 System.out.println(result);

66 }

67

68 //修改已有属性

69 public static void test04() throws Exception{

70 ClassPool pool = ClassPool.getDefault();

71 CtClass cc = pool.get("bean.User");

72

73 //属性

74 CtField cf = new CtField(CtClass.intType,"age",cc);

75 cf.setModifiers(Modifier.PRIVATE);

76 cc.addField(cf);

77 //增加响应的get set方法

78 cc.addMethod(CtNewMethod.getter("getAge",cf));

79 cc.addMethod(CtNewMethod.setter("setAge",cf));

80

81 //访问属性

82 Class clazz = cc.toClass();

83 Object obj = clazz.newInstance();

84 Field field = clazz.getDeclaredField("age");

85 System.out.println(field);

86 Method m = clazz.getDeclaredMethod("setAge", int.class);

87 m.invoke(obj, 16);

88 Method m2 = clazz.getDeclaredMethod("getAge", null);

89 Object resutl = m2.invoke(obj,null);

90 System.out.println(resutl);

91 }

92

93 //操作构造方法

94 public static void test05() throws Exception{

95 ClassPool pool = ClassPool.getDefault();

96 CtClass cc = pool.get("com.swust.beans.Person");

97

98 CtConstructor[] cons = cc.getConstructors();

99 for(CtConstructor con:cons){

100 System.out.println(con);

101 }

102 }

103 public static void main(String[] args) throws Exception {

104 test01();

105 //test02();

106 //test03();

107 //test04();

108 test05();

109 }

110 }

[复制代码](javascript:void(0);)

调用方法1获取类的基本信息，结果如下：

[复制代码](javascript:void(0);)

1 完整类名为：com.swust.beans.Person

2 类名为：Person

3 父类名称为：java.lang.Object

4 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

5 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

6 属性方法为：wait

7 属性方法为：wait

8 属性方法为：setName

9 属性方法为：notifyAll

10 属性方法为：wait

11 属性方法为：toString

12 属性方法为：getName

13 属性方法为：setAge

14 属性方法为：equals

15 属性方法为：main

16 属性方法为：getAge

17 属性方法为：getClass

18 属性方法为：clone

19 属性方法为：finalize

20 属性方法为：hashCode

21 属性方法为：notify

[复制代码](javascript:void(0);)

调用方法2添加新方法：

1 方法执行结果为：5

1 这是在原有方法体执行之前增加的内容

2 张三

3 这是在原有方法体执行之后增加的内容

4 null

调用方法4修改已有属性：

1 增添的属性为：private int com.swust.beans.Person.age

2 getAge方法执行后的结果为：16

3 增添的属性为：private int com.swust.beans.Person.height

4 getHeight方法执行后的结果为：176

调用方法5操作构造函数：

1 javassist.CtConstructor@180cb01[public Person ()V]

分类: [Java基础](http://www.cnblogs.com/sunfie/category/704485.html)