

使用 javassist 生成新类

分类: [javassist](#) 2013-11-26 18:06 922 人阅读 评论(0) 收藏 举报

JAVAJavassist

1、简介

javassist 是一个开源的分析、编辑和创建 java 字节码的类库。不需要了解虚拟机指令，就能动态生成类或者改变类的结构。


2、下载

- (1) 下载链接 <http://www.csg.ci.i.u-tokyo.ac.jp/~chiba/javassist/>
- (2) 使用的版本是 javassist-3.18.0-GA。

3、实验

此实验的目的是通过 javassist 生成一个新类 Emp.java

- (1) 生成的目标类 Emp.java

```
1. [java] view plaincopyprint?
2. package com.study.javassist;
3.
4. public class Emp {
5.
6.     private String ename;
7.     private int eno;
8.
9.     public Emp(){
10.         ename="yy";
11.         eno=001;
12.     }
13.
14.     public String getEname() {
15.         return ename;
16.     }
17.
18.     public void setEname(String ename) {
19.         this.ename = ename;
20.     }
21.
22.     public int getEno() {
23.         return eno;
24.     }
25.
26.     public void setEno(int eno) {
27.         this.eno = eno;
28.     }
29.
30.
31.     //添加一个自定义方法
32.     public void printInfo(){
33.         System.out.println("begin!");
34.         System.out.println(ename);
35.         System.out.println(eno);
36.         System.out.println("over!");
37.     }
38. }
```

(2) 主类 GenerateNewClassByJavassist.java

```
39. [java] view plaincopyprint?
40. package com.study.javassist;
41.
42. import java.io.File;
43. import java.io.FileOutputStream;
44. import java.lang.reflect.Modifier;
45.
46. import javassist.ClassPool;
47. import javassist.CtClass;
48. import javassist.CtConstructor;
49. import javassist.CtField;
50. import javassist.CtMethod;
51. import javassist.CtNewMethod;
52.
53. /**
54.  *使用 javassist 动态生成一个 java 类
55.  * @author yy
56.  * @version 1.0
57.  *
58.  */
59. public class GenerateNewClassByJavassist {
60.
61.     public static void main(String[] args) throws Exception{
62.
63.         //ClassPool: CtClass 对象的容器
64.         ClassPool pool = ClassPool.getDefault();
65.
66.         //通过 ClassPool 生成一个 public 新类 Emp.java
67.         CtClass ctClass = pool.makeClass("com.study.javassist.Emp");
68.
69.         //添加字段
70.         //首先添加字段 private String ename
71.         CtField enameField = new CtField(pool.getCtClass("java.lang.String"), "ename", ctClass);
72.         enameField.setModifiers(Modifier.PRIVATE);
73.         ctClass.addField(enameField);
74.
75.         //其次添加字段 private int eno
76.         CtField enoField = new CtField(pool.getCtClass("int"), "eno", ctClass);
77.         enoField.setModifiers(Modifier.PRIVATE);
78.         ctClass.addField(enoField);
79.
80.         //为字段 ename 和 eno 添加 getXXX 和 setXXX 方法
81.         ctClass.addMethod(CtNewMethod.getter("getEname", enameField));
82.         ctClass.addMethod(CtNewMethod.setter("setEname", enameField));
83.         ctClass.addMethod(CtNewMethod.getter("getEno", enoField));
84.         ctClass.addMethod(CtNewMethod.setter("setEno", enoField));
85.
86.         //添加构造函数
87.         CtConstructor ctConstructor = new CtConstructor(new CtClass[] {}, ctClass);
88.         //为构造函数设置函数体
89.         StringBuffer buffer = new StringBuffer();
90.         buffer.append("{\n")
91.             .append("    ename=\"yy\";\n")
92.             .append("    eno=001;\n}");
93.         ctConstructor.setBody(buffer.toString());
94.         //把构造函数添加到新的类中
95.         ctClass.addConstructor(ctConstructor);
```

```

96.
97.         //添加自定义方法
98.         CtMethod ctMethod = new CtMethod(CtClass.voidType,"printInfo",new
CtClass[] {},ctClass);
99.         //为自定义方法设置修饰符
100.        ctMethod.setModifiers(Modifier.PUBLIC);
101.        //为自定义方法设置函数体
102.        StringBuffer buffer2 = new StringBuffer();
103.        buffer2.append("\nSystem.out.println(\"begin!\");\n");
104.        .append("System.out.println(ename);\n");
105.        .append("System.out.println(eno);\n");
106.        .append("System.out.println(\"over!\");\n");
107.        .append("{}");
108.        ctMethod.setBody(buffer2.toString());
109.        ctClass.addMethod(ctMethod);
110.
111.        //为了验证效果，下面使用反射执行方法 printInfo
112.        Class<?> clazz = ctClass.toClass();
113.        Object obj = clazz.newInstance();
114.        obj.getClass().getMethod("printInfo", new Class[] {} ).invoke(obj
, new Object[] {});
115.
116.        //把生成的 class 文件写入文件
117.        byte[] byteArr = ctClass.toBytecode();
118.        FileOutputStream fos = new FileOutputStream(new File("D://Emp.c
lass"));
119.        fos.write(byteArr);
120.        fos.close();
121.    }
122. }

```

(3) 实验结果

首先打印结果如下：

[java] view plaincopyprint?

```

123. begin!
124. yy
125. 1
126. over!

```

通过 XJad 反编译结果如下：

[java] view plaincopyprint?

```

127. // Decompiled by Jad v1.5.8e2. Copyright 2001 Pavel Kouznetsov.
128. // Jad home page: http://kpdus.tripod.com/jad.html
129. // Decompiler options: packimports(3) fieldsfirst ansi space
130. // Source File Name:   Emp.java
131.
132. package com.study.javassist;
133.
134. import java.io.PrintStream;
135.
136. public class Emp
137. {
138.
139.     private String ename;
140.     private int eno;
141.

```

```

142.     public String getEname()
143.     {
144.         return ename;
145.     }
146.
147.     public void setEname(String s)
148.     {
149.         ename = s;
150.     }
151.
152.     public int getEno()
153.     {
154.         return eno;
155.     }
156.
157.     public void setEno(int i)
158.     {
159.         eno = i;
160.     }
161.
162.     public Emp()
163.     {
164.         ename = "yy";
165.         eno = 1;
166.     }
167.
168.     public void printInfo()
169.     {
170.         System.out.println("begin!");
171.         System.out.println(ename);
172.         System.out.println(eno);
173.         System.out.println("over!");
174.     }
175. }
176. // Decompiled by Jad v1.5.8e2. Copyright 2001 Pavel Kouznetsov.
177. // Jad home page: http://kpdus.tripod.com/jad.html
178. // Decompiler options: packimports(3) fieldsfirst ansi space
179. // Source File Name:  Emp.java
180.
181. package com.study.javassist;
182.
183. import java.io.PrintStream;
184.
185. public class Emp
186. {
187.
188.     private String ename;
189.     private int eno;
190.
191.     public String getEname()
192.     {
193.         return ename;
194.     }
195.
196.     public void setEname(String s)
197.     {
198.         ename = s;
199.     }
200.
201.     public int getEno()
202.     {
203.         return eno;
204.     }
205.
206.     public void setEno(int i)
207.     {

```

```
208.         eno = i;
209.     }
210.
211.     public Emp()
212.     {
213.         ename = "yy";
214.         eno = 1;
215.     }
216.
217.     public void printInfo()
218.     {
219.         System.out.println("begin!");
220.         System.out.println(ename);
221.         System.out.println(eno);
222.         System.out.println("over!");
223.     }
224. }
225.
```

可见，通过 `javassist` 可以动态的生成类。