java中动态的为某个类生成属性和方法

通过程序修改 .class 文件！  
  
下面这个是用 ASM 工具为 Student 类添加一个 public String 类型的 address 属性：  
  
**1，需要添加属性的原始类：Student.java**

Java code

[?](http://bbs.csdn.net/topics/250014721#clipboardWindow)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | public class Student {  private int age;  private String name;  public int getAge() {  return age;  }  public void setAge(int age) {  this.age = age;  }  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  } |

**2，添加属性的适配器：AddFieldAdapter.java**

Java code

[?](http://bbs.csdn.net/topics/250014721#clipboardWindow)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37 | import org.objectweb.asm.ClassAdapter;  import org.objectweb.asm.ClassVisitor;  import org.objectweb.asm.FieldVisitor;    public class AddFieldAdapter extends ClassAdapter {  private int accessModifier;  private String name;  private String desc;  private boolean isFieldPresent;    public AddFieldAdapter(ClassVisitor cv, int accessModifier, String name, String desc) {  super(cv);  this.accessModifier = accessModifier;  this.name = name;  this.desc = desc;  }    @Override  public FieldVisitor visitField(int access, String name, String desc,  String signature, Object value) {  if (name.equals(this.name)) {  isFieldPresent = true;  }  return cv.visitField(access, name, desc, signature, value);  }    @Override  public void visitEnd() {  if (!isFieldPresent) {  FieldVisitor fv = cv.visitField(accessModifier, name, desc, null, null);  if (fv != null) {  fv.visitEnd();  }  }  cv.visitEnd();  }  } |

**3，添加属性的工具 AddField.java**

Java code

[?](http://bbs.csdn.net/topics/250014721#clipboardWindow)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82 | import java.io.File;  import java.io.FileOutputStream;  import java.io.IOException;  import java.io.OutputStream;    import org.objectweb.asm.ClassAdapter;  import org.objectweb.asm.ClassReader;  import org.objectweb.asm.ClassWriter;  import org.objectweb.asm.Opcodes;    public class AddField {    private Class clazz = null;  private ClassReader cr = null;  private ClassWriter cw = null;  private ClassAdapter ca = null;  private File classFile = null;    private final static String CLASS\_FILE\_SUFFIX = ".class";    public AddField(Class clazz) {  this.clazz = clazz;  }    /\*\*  \* 添加一个 public 的类成员  \* @param fieldName 类成员名  \* @param fieldDesc 类成员类型描述  \*/  public void addPublicField(String fieldName, String fieldDesc) {  if(cr == null) {  try {  cr = new ClassReader(clazz.getCanonicalName());  } catch (IOException e) {  e.printStackTrace();  }  cw = new ClassWriter(cr, ClassWriter.COMPUTE\_MAXS);  }  if(ca == null) {  ca = new AddFieldAdapter(cw, Opcodes.ACC\_PUBLIC, fieldName, fieldDesc);  } else {  ca = new AddFieldAdapter(ca, Opcodes.ACC\_PUBLIC, fieldName, fieldDesc);  }  }    /\*\*  \* 将字节码写入类的 .class 文件  \*  \*/  public void writeByteCode() {  cr.accept(ca, ClassReader.SKIP\_DEBUG);  byte[] bys = cw.toByteArray();  OutputStream os = null;  try {  os = new FileOutputStream(getFile());  os.write(bys);  } catch (IOException e) {  e.printStackTrace();  } finally {  try {  os.close();  } catch (IOException e) {  e.printStackTrace();  }  }  }    /\*\*  \* 获得类文件的 File 对象  \* @return  \*/  private File getFile() {  if(classFile == null) {  StringBuffer sb = new StringBuffer();  sb.append(clazz.getResource("/"))  .append(clazz.getCanonicalName().replace(".", File.separator))  .append(CLASS\_FILE\_SUFFIX);  classFile = new File(sb.substring(6));  }  return classFile;  }  } |

**4，字节码处理：PreCompileProcess.java**

Java code

[?](http://bbs.csdn.net/topics/250014721#clipboardWindow)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | public class PreCompileProcess {    public static void main(String[] args) {    // 为 Student 添加字段  AddField add = new AddField(Student.class);    // 添加一个名为 address，类型为 java.lang.String 的 public 字段  add.addPublicField("address", "Ljava/lang/String;");    // 再增加一个名为 tel，类型为 int 的 public 方法  add.addPublicField("tel", "I");    // 重新生成 .class 文件  add.writeByteCode();  }  } |

**5，测试类：Test.java**

Java code

[?](http://bbs.csdn.net/topics/250014721#clipboardWindow)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | public class Test {    public static void main(String[] args) {  Student stu = new Student();  stu.setAge(10);  stu.setName("Tom");  stu.address = "Beijing"; // 新增加的 address 字段  stu.tel = 56; // 新增加的 tel 字段  System.out.println("Name: " + stu.getName());  System.out.println("Age: " + stu.getAge());  System.out.println("Address: " + stu.address);  System.out.println("Tel: " + stu.tel);  }  } |

上面的代码需要 ASM 工具才能进行编译，ASM 可以到 <http://asm.objectweb.org/> 上面去下载。  
  
执行顺序：  
  
javac Student.java  
javac -cp .;lib/asm-all-3.1.jar PreCompileProcess.java  
java -cp .;lib/asm-all-3.1.jar PreCompileProcess  
javac Test.java  
java Test  
  
在 Student.java 没有修改的情况下，PreCompileProcess 只需要执行一次就可以了。