



做最棒的软件开发交流社区



[首页](#) [新闻](#) [论坛](#) [问答](#) [专栏](#) [博客](#) [文摘](#) [圈子](#) [招聘](#) [服务](#) [搜索](#)

[Java](#) [Web](#) [Ruby](#) [Python](#) [敏捷](#) [MySQL](#) [润乾报表](#) [图书](#)

[论坛首页](#) → [Java编程和Java企业应用版](#) → [OO](#) → 使用javassist动态注入代码

[全部](#) [Hibernate](#) [Spring](#) [Struts](#) [iBATIS](#) [企业应用](#) [设计模式](#) [DAO](#) [领域模型](#) [OO](#) [Tomcat](#) [SOA](#) [JBoss](#) [Swing](#) [Java综合](#)

浏览 3176 次

锁定老贴子 主题：使用javassist动态注入代码

精华帖 (0) :: 良好帖 (0) :: 新手帖 (0) :: 隐藏帖 (0)

作者

正文

zyl

发表时间：2007-02-10

等级：☆☆☆☆☆



文章：91

积分：468



猎头职位：安徽：合肥,杭州,苏州：诚聘java架构师

关于java字节码的处理，目前有很多工具，如bcel，asm。不过这些都需要直接跟虚拟机指令打交道。如果你不想了解虚拟机指令，可以采用javassist。javassist是jboss的一个子项目，其主要的优点，在于简单，而且快速。直接使用java编码的形式，而不需要了解虚拟机指令，就能动态改变类的结构，或者动态生成类。

下面通过一个简单的例子，通过javassist来实现如何动态注入代码。

假设，存在类A,如下：

java 代码

```
1. public class A {
2.     public void method() {
3.         for (int i = 0; i < 1000000; i++) {
4.         }
5.         System.out.println("method1");
6.     }
7. }
```

测试类B如下：

java 代码

相关文章：

- [基于javassist实现对接口的动态代理引擎](#)
- [Javassite:使字节码工程变得简单\(译\)](#)
- [如何把数据库表中一条记录变成一个Java对象](#)

推荐圈子： [Pipboy](#)

[更多相关推荐](#)

```

1. public class B {
2.     public static void main(String[] args) {
3.         A a = new A();
4.         a.method();
5.     }
6. }

```

现在想统计一下method的执行时间，
默认的实现是修改method：

java 代码

```

1. public void method() {
2.     long start = System.currentTimeMillis();
3.     for (int i = 0; i < 1000000; i++) {
4.     }
5.     System.out.println("method1");
6.     long end = System.currentTimeMillis();
7.     System.out.println(end - start);
8. }

```

如果A的方法很多，统计方法的执行时间的代码就会相应的增加。为了减少工作量，通过动态注入代码的形式来实现。
修改B的main方法：

java 代码

```

1. public static void main(String[] args) throws Exception {
2.     //用于取得字节码类，必须在当前的classpath中，使用全称
3.     CtClass ctClass = ClassPool.getDefault().get("org.esoft.A");
4.     //需要修改的方法名称
5.     String mname = "method";
6.     CtMethod mold = ctClass.getDeclaredMethod(mname);
7.     //修改原有的方法名称
8.     String nname = mname + "$impl";
9.     mold.setName(nname);
10.    //创建新的方法，复制原来的方法
11.    CtMethod mnew = CtNewMethod.copy(mold, mname, ctClass, null);
12.    //主要的注入代码
13.    StringBuffer body = new StringBuffer();
14.    body.append("{\nlong start = System.currentTimeMillis();\n");
15.    //调用原有代码，类似于method();($$)表示所有的参数
16.    body.append(nname + "($$);\n");
17.    body.append("System.out.println(\"Call to method \"

```

```
18.         + mname
19.         + " took \" +\n (System.currentTimeMillis()-start) + "
20.         + "\" ms.\");\n");
21.
22.     body.append("}");
23.     //替换新方法
24.     mnew.setBody(body.toString());
25.     //增加新方法
26.     ctClass.addMethod(mnew);
27.     //类已经更改, 注意不能使用A a=new A();, 因为在同一个classloader中, 不允许装载同一个类两次
28.     A a=(A)ctClass.toClass().newInstance();
29.     a.method();
30. }
```

这只是简单的一个应用。javassist还提供了很多的功能,用于更改类结构。有兴趣的可以参考相关文档

声明: JavaEye 文章版权属于作者,受法律保护。没有作者书面许可不得转载。

推荐链接




www.itany.com

Google 提供的广告

[返回顶楼](#)

jamesby

等级: 



文章: 484

积分: 845

发表时间: 2007-02-11

Tapestry 好象用到了javassist,当时还不知道做什么的,感谢楼主的这篇文章,收藏!

[返回顶楼](#)

回帖地址

0

0 请登录后投票

[论坛首页](#) → [Java编程和Java企业应用版](#) → [OO](#)

跳转论坛:

[北京: Hoolai游戏诚聘【社交游戏】JAVA开发工程师/](#)

□□: □□□□□□□□□□ java□□□□□

□□: □□□□□□□□□□ Java□□□□□

□ □ : □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

□□: □□□□□ JAVA □□□□□

广告服务 | [JavaEye黑板报](#) | [关于我们](#) | [联系我们](#) | [友情链接](#)

© 2003-2010 JavaEye.com. 上海炯耐计算机软件有限公司版权所有 [沪ICP备05023328号]