[一个自定义类加载器ClassLoader示例](http://blog.csdn.net/chenjiazhan/article/details/37714401)

标签： [classloader](http://www.csdn.net/tag/classloader)[类加载器](http://www.csdn.net/tag/%e7%b1%bb%e5%8a%a0%e8%bd%bd%e5%99%a8)[java](http://www.csdn.net/tag/java)

2014-07-12 01:34 1098人阅读 [评论](http://blog.csdn.net/chenjiazhan/article/details/37714401#comments)(1) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/chenjiazhan/article/details/37714401#report)

版权声明：本文为博主原创文章，未经博主允许不得转载。

我们的自定义类加载器

**[java]** [view plain](http://blog.csdn.net/chenjiazhan/article/details/37714401) [copy](http://blog.csdn.net/chenjiazhan/article/details/37714401)

1. **package** cn.gd.cjz.class\_loader;
3. **import** java.io.\*;
4. **import** java.nio.ByteBuffer;
5. **import** java.nio.channels.FileChannel;
7. /\*\*
8. \* 自定义类加载器
9. \*/
10. **public** **class** CustomClassLoader **extends** ClassLoader {
11. /\*\*类名\*\*/
12. **private** String name;
13. /\*\*通过构造方法设置父类加载器和要热加载的类名\*\*/
14. **public** CustomClassLoader(ClassLoader parent , String name) {
15. **super**(parent);
16. **if**(name == **null** || name.length() <= 0)
17. **throw** **new** NullPointerException();
19. **this**.name = name;
20. }
22. @Override
23. **protected** Class<?> loadClass(String name, **boolean** resolve) **throws** ClassNotFoundException {
24. Class<?> clazz = **null**;
25. /\*\*如果是我们想要热加载的类则调用我们重写的findClass方法来加载\*\*/
26. **if**(**this**.name.equals(name) && !"java".equals(name)){
27. /\*\*先看看要热加载的类之前是否已经加载过了，因为一个类加载器只能加载一个类一次，加载多次会报异常\*\*/
28. clazz = findLoadedClass(name);
29. /\*\*clazz==null说明之前没有加载过\*\*/
30. **if**(clazz == **null**)
31. clazz = findClass(name);
33. /\*\*
34. \* 类的生命周期包括：加载、验证、准备、解析、初始化、使用、卸载。其中验证、准备、解析统称为连接
35. \* 如果要连接类
36. \*/
37. **if**(resolve)
38. resolveClass(clazz);//如果类已连接过，resolveClass方法会直接返回
39. **return** clazz;
40. }
41. **return** **super**.loadClass(name , resolve);
42. }
44. @Override
45. **protected** Class<?> findClass(String name) **throws** ClassNotFoundException {
46. String fileName = c2f(name);
47. **byte**[] bytes = f2b(fileName);
48. **return** defineClass(name, bytes, 0, bytes.length);
49. }
51. /\*\*
52. \* 类名转为文件名
53. \* @param name
54. \* @return
55. \*/
56. **private** String c2f(String name){
57. /\*\*编译后的class文件存放的目录\*\*/
58. String baseDir = "F:\\idea\_workspace\\Test\\target\\classes\\";
59. name = name.replace("." , File.separator);
60. name = baseDir + name + ".class";
61. **return** name;
62. }
64. /\*\*
65. \* 读取文件byte数组
66. \* @param fileName
67. \* @return
68. \*/
69. **private** **byte**[] f2b(String fileName){
70. RandomAccessFile file = **null**;
71. FileChannel channel = **null**;
72. **byte**[] bytes = **null**;
73. **try** {
74. /\*\*随机存取文件对象，只读取模式\*\*/
75. file = **new** RandomAccessFile(fileName , "r");
76. /\*\*NIO文件通道\*\*/
77. channel = file.getChannel();
78. /\*\*NIO字节缓冲\*\*/
79. ByteBuffer buffer = ByteBuffer.allocate(1024);
80. **int** size = (**int**) channel.size();
81. bytes = **new** **byte**[size];
82. **int** index = 0;
83. /\*\*从NIO文件通道读取数据\*\*/
84. **while** (channel.read(buffer) > 0){
85. /\*\*字节缓冲从写模式转为读取模式\*\*/
86. buffer.flip();
87. **while** (buffer.hasRemaining()){
88. bytes[index] = buffer.get();
89. ++index;
90. }
91. /\*\*字节缓冲的readerIndex、writerIndex置零\*\*/
92. buffer.clear();
93. }
94. } **catch** (FileNotFoundException e) {
95. e.printStackTrace();
96. } **catch** (IOException e) {
97. e.printStackTrace();
98. }**finally** {
99. **if** (channel != **null**) {
100. **try** {
101. channel.close();
102. } **catch** (IOException e) {
103. e.printStackTrace();
104. }
105. }
106. **if** (file != **null**) {
107. **try** {
108. file.close();
109. } **catch** (IOException e) {
110. e.printStackTrace();
111. }
112. }
113. }
114. **return** bytes;
115. }
117. /\*\*
118. \* 热加载类
119. \* @return
120. \*/
121. **public** Class<?> loadClass(){
122. **try** {
123. **return** loadClass(name);
124. } **catch** (ClassNotFoundException e) {
125. e.printStackTrace();
126. **return** **null**;
127. }
128. }
129. }

将要被热加载的类接口

**[java]** [view plain](http://blog.csdn.net/chenjiazhan/article/details/37714401) [copy](http://blog.csdn.net/chenjiazhan/article/details/37714401)

1. **package** cn.gd.cjz.class\_loader;
3. /\*\*
4. \* 测试类接口
5. \*/
6. **public** **interface** IPrinter {
7. **public** **void** print();
8. }

将要被热加载的类

**[java]** [view plain](http://blog.csdn.net/chenjiazhan/article/details/37714401) [copy](http://blog.csdn.net/chenjiazhan/article/details/37714401)

1. **package** cn.gd.cjz.class\_loader;
3. /\*\*
4. \* 测试类
5. \*/
6. **public** **class** Printer **implements** IPrinter {
7. @Override
8. **public** **void** print() {
9. System.out.println("彪悍的人生不需要解释是谁说的？");
10. }
11. }

好了，写个测试类来测试一下

**[java]** [view plain](http://blog.csdn.net/chenjiazhan/article/details/37714401) [copy](http://blog.csdn.net/chenjiazhan/article/details/37714401)

1. **package** cn.gd.cjz.class\_loader;
3. **import** java.io.BufferedReader;
4. **import** java.io.IOException;
5. **import** java.io.InputStreamReader;
7. /\*\*\*
8. \* 自定义类加载器测试类
9. \*/
10. **public** **class** CustomClassTest {
11. **public** **static** **void** main(String[] args) {
12. /\*\*要进行热加载的类名\*\*/
13. String name = "cn.gd.cjz.class\_loader.Printer";
14. BufferedReader reader = **new** BufferedReader(**new** InputStreamReader(System.in));
15. IPrinter printer = **null**;
16. **while** (**true**) {
17. System.out.println("输入任意字符进行热加载，直接敲回车键退出程序");
18. **try** {
19. String line = reader.readLine();
20. **if**(line != **null** && line.length() > 0){
21. CustomClassLoader loader = **new** CustomClassLoader(Thread.currentThread().getContextClassLoader() , name);
22. Class<?> clazz = loader.loadClass();
23. /\*\*
24. \* 被子加载器加载的类拥有被父加载器加载的类的可见性
25. \* Printer是由自定义类加载器加载的，
26. \* 而它的父类IPrinter是由系统类加载器加载的，
27. \* 因此IPrinter对于Printer具有可见性，
28. \* 因此转型成功，并不会因为类加载器不同导致ClassCastException异常
29. \*/
30. printer = (IPrinter) clazz.newInstance();
31. /\*\*看看是否热加载成功了\*\*/
32. printer.print();
33. }**else**{
34. **break**;
35. }
36. } **catch** (IOException e) {
37. e.printStackTrace();
38. } **catch** (InstantiationException e) {
39. e.printStackTrace();
40. } **catch** (IllegalAccessException e) {
41. e.printStackTrace();
42. }
43. }
44. }
45. }

首先运行测试类，输入任意字符看看控制台输出了什么？再把Printer类改成下面的代码，编译后再在控制台输入任意字符，看看控制台又输出了什么？

**[java]** [view plain](http://blog.csdn.net/chenjiazhan/article/details/37714401) [copy](http://blog.csdn.net/chenjiazhan/article/details/37714401)

1. **package** cn.gd.cjz.class\_loader;
3. /\*\*
4. \* 测试类
5. \*/
6. **public** **class** Printer **implements** IPrinter {
7. @Override
8. **public** **void** print() {
9. System.out.println("锤子老罗说的。");
10. }
11. }

以下是我的测试结果：

