[**自定义ClassLoader，用于加载用户JAR包**](http://obullxl.iteye.com/blog/651128)

**博客分类：**

* [Java](http://obullxl.iteye.com/category/103394)

[QQ](http://www.iteye.com/blogs/tag/QQ)[IBM](http://www.iteye.com/blogs/tag/IBM)[软件测试](http://www.iteye.com/blogs/tag/%E8%BD%AF%E4%BB%B6%E6%B5%8B%E8%AF%95)[.net](http://www.iteye.com/blogs/tag/.net)[thread](http://www.iteye.com/blogs/tag/thread)

最近在考虑C/S结构程序的软件自动升级的实现方式，比如QQ、飞信等都自动升级的功能。

自动升级模块虽然还没有编码完成，但是思路还是比较清晰的。

自动升级过程中，升级文件的JAR包是专门加载到程序中去的，因此，自定义一个ClassLoader，用于加载用户JAR包，就非常的重要了。

应用程序ClassLoader只提供了一个public Class<?> loadClass(String name) throws ClassNotFoundException 方法，没有提供加载JAR的方法。

URLClassLoader提供了一个protected void addURL(URL url)的方法，倒是可以加载JAR包，但苦于非public的。

经测试发现，AppClassLoader是URLClassLoader的子类。因此，我们完全可以利用URLClassLoader了哦。

**Java代码**

1. URLClassLoader system = (URLClassLoader) ClassLoader.getSystemClassLoader();

这样，我们可以通过反射得到addURL方法，在程序中加载我们自己的JAR包了。

整个源代码如下所示：

**Java代码**



1. /\*\*
2. \* Copyright (c) YMCN Team
3. \* All rights reserved.
4. \*/
5. **package** com.aboy.toolkit.util;
7. **import** java.io.File;
8. **import** java.lang.reflect.Method;
9. **import** java.net.URL;
10. **import** java.net.URLClassLoader;
11. **import** java.util.ArrayList;
12. **import** java.util.List;
14. /\*\*
15. \* @author obullxl
16. \*
17. \* email: obullxl@163.com  MSN: obullxl@hotmail.com  QQ: 303630027
18. \*
19. \* Blog: http://obullxl.iteye.com
20. \*/
21. **public** **final** **class** ClassLoaderUtil {
22. /\*\* URLClassLoader的addURL方法 \*/
23. **private** **static** Method addURL = initAddMethod();
25. /\*\* 初始化方法 \*/
26. **private** **static** **final** Method initAddMethod() {
27. **try** {
28. Method add = URLClassLoader.**class**
29. .getDeclaredMethod("addURL", **new** Class[] { URL.**class** });
30. add.setAccessible(**true**);
31. **return** add;
32. } **catch** (Exception e) {
33. e.printStackTrace();
34. }
35. **return** **null**;
36. }
38. **private** **static** URLClassLoader system = (URLClassLoader) ClassLoader.getSystemClassLoader();
40. /\*\*
41. \* 循环遍历目录，找出所有的JAR包
42. \*/
43. **private** **static** **final** **void** loopFiles(File file, List<File> files) {
44. **if** (file.isDirectory()) {
45. File[] tmps = file.listFiles();
46. **for** (File tmp : tmps) {
47. loopFiles(tmp, files);
48. }
49. } **else** {
50. **if** (file.getAbsolutePath().endsWith(".jar") || file.getAbsolutePath().endsWith(".zip")) {
51. files.add(file);
52. }
53. }
54. }
56. /\*\*
57. \* <pre>
58. \* 加载JAR文件
59. \* </pre>
60. \*
61. \* @param file
62. \*/
63. **public** **static** **final** **void** loadJarFile(File file) {
64. **try** {
65. addURL.invoke(system, **new** Object[] { file.toURI().toURL() });
66. System.out.println("加载JAR包：" + file.getAbsolutePath());
67. } **catch** (Exception e) {
68. e.printStackTrace();
69. }
70. }
72. /\*\*
73. \* <pre>
74. \* 从一个目录加载所有JAR文件
75. \* </pre>
76. \*
77. \* @param path
78. \*/
79. **public** **static** **final** **void** loadJarPath(String path) {
80. List<File> files = **new** ArrayList<File>();
81. File lib = **new** File(path);
82. loopFiles(lib, files);
83. **for** (File file : files) {
84. loadJarFile(file);
85. }
86. }
87. }

在程序中，只要使用上面最后两个方法，就可以加载自定义JAR包和一个目录中的所有JAR包了。

经本人测试，上面代码运行无误，能正常加载自定义JAR包。