[Java中动态加载jar文件和class文件](http://blog.csdn.net/mousebaby808/article/details/31788325)

2014-06-17 17:57 7962人阅读 [评论](http://blog.csdn.net/mousebaby808/article/details/31788325#comments)(3) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/mousebaby808/article/details/31788325#report)

 分类：



JavaSE（3）



版权声明：本文为博主原创文章，未经博主允许不得转载。

目录[(?)[+]](http://blog.csdn.net/mousebaby808/article/details/31788325)

**概述**

　　诸如tomcat这样的服务器，在启动的时候会加载应用程序中lib目录下的jar文件以及classes目录下的class文件，另外像spring这类框架，也可以根据指定的路径扫描并加载指定的类文件，这个技术可以实现一个容器，容纳各类不同的子应用。

　　Java类由于需要加载和编译字节码，动态加载class文件较为麻烦，不像C加载动态链接库只要一个文件名就可以搞定，但JDK仍提供了一整套方法来动态加载jar文件和class文件。

**动态加载jar文件**

**[java]** [view plain](http://blog.csdn.net/mousebaby808/article/details/31788325) [copy](http://blog.csdn.net/mousebaby808/article/details/31788325)

1. // 系统类库路径
2. File libPath = **new** File(jar文件所在路径);
4. // 获取所有的.jar和.zip文件
5. File[] jarFiles = libPath.listFiles(**new** FilenameFilter() {
6. **public** **boolean** accept(File dir, String name) {
7. **return** name.endsWith(".jar") || name.endsWith(".zip");
8. }
9. });
11. **if** (jarFiles != **null**) {
12. // 从URLClassLoader类中获取类所在文件夹的方法
13. // 对于jar文件，可以理解为一个存放class文件的文件夹
14. Method method = URLClassLoader.**class**.getDeclaredMethod("addURL", URL.**class**);
15. **boolean** accessible = method.isAccessible();     // 获取方法的访问权限
16. **try** {
17. **if** (accessible == **false**) {
18. method.setAccessible(**true**);     // 设置方法的访问权限
19. }
20. // 获取系统类加载器
21. URLClassLoader classLoader = (URLClassLoader) ClassLoader.getSystemClassLoader();
22. **for** (File file : jarFiles) {
23. URL url = file.toURI().toURL();
24. **try** {
25. method.invoke(classLoader, url);
26. LOG.debug("读取jar文件[name={}]", file.getName());
27. } **catch** (Exception e) {
28. LOG.error("读取jar文件[name={}]失败", file.getName());
29. }
30. }
31. } **finally** {
32. method.setAccessible(accessible);
33. }
34. }

**动态加载class文件**

**[java]** [view plain](http://blog.csdn.net/mousebaby808/article/details/31788325) [copy](http://blog.csdn.net/mousebaby808/article/details/31788325)

1. // 设置class文件所在根路径
2. // 例如/usr/java/classes下有一个test.App类，则/usr/java/classes即这个类的根路径，而.class文件的实际位置是/usr/java/classes/test/App.class
3. File clazzPath = **new** File(**class**文件所在根路径);
5. // 记录加载.class文件的数量
6. **int** clazzCount = 0;
8. **if** (clazzPath.exists() && clazzPath.isDirectory()) {
9. // 获取路径长度
10. **int** clazzPathLen = clazzPath.getAbsolutePath().length() + 1;
12. Stack<File> stack = **new** Stack<>();
13. stack.push(clazzPath);
15. // 遍历类路径
16. **while** (stack.isEmpty() == **false**) {
17. File path = stack.pop();
18. File[] classFiles = path.listFiles(**new** FileFilter() {
19. **public** **boolean** accept(File pathname) {
20. **return** pathname.isDirectory() || pathname.getName().endsWith(".class");
21. }
22. });
23. **for** (File subFile : classFiles) {
24. **if** (subFile.isDirectory()) {
25. stack.push(subFile);
26. } **else** {
27. **if** (clazzCount++ == 0) {
28. Method method = URLClassLoader.**class**.getDeclaredMethod("addURL", URL.**class**);
29. **boolean** accessible = method.isAccessible();
30. **try** {
31. **if** (accessible == **false**) {
32. method.setAccessible(**true**);
33. }
34. // 设置类加载器
35. URLClassLoader classLoader = (URLClassLoader) ClassLoader.getSystemClassLoader();
36. // 将当前类路径加入到类加载器中
37. method.invoke(classLoader, clazzPath.toURI().toURL());
38. } **finally** {
39. method.setAccessible(accessible);
40. }
41. }
42. // 文件名称
43. String className = subFile.getAbsolutePath();
44. className = className.substring(clazzPathLen, className.length() - 6);
45. className = className.replace(File.separatorChar, '.');
46. // 加载Class类
47. Class.forName(className);
48. LOG.debug("读取应用程序类文件[class={}]", className);
49. }
50. }
51. }
52. }

　　完成上述两步操作后，即可使用Class.forName来加载jar中或.class文件包含的Java类了。