[根据一个接口返回该接口的所有实现类](http://blog.csdn.net/struggle_lyh/article/details/7800785)

标签： [classloader](http://www.csdn.net/tag/classloader)[exception](http://www.csdn.net/tag/exception)[file](http://www.csdn.net/tag/file)[string](http://www.csdn.net/tag/string)[resources](http://www.csdn.net/tag/resources)[date](http://www.csdn.net/tag/date)

2012-07-30 10:00 6721人阅读 [评论](http://blog.csdn.net/struggle_lyh/article/details/7800785#comments)(3) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/struggle_lyh/article/details/7800785#report)

 分类：



Java从头学（5）



版权声明：本文为博主原创文章，未经博主允许不得转载。

**[java]** [view plain](http://blog.csdn.net/struggle_lyh/article/details/7800785) [copy](http://blog.csdn.net/struggle_lyh/article/details/7800785)

1. **package** com.pattern.factory;
3. **import** java.io.File;
4. **import** java.io.IOException;
5. **import** java.net.URL;
6. **import** java.util.ArrayList;
7. **import** java.util.Enumeration;
8. **import** java.util.List;
10. **public** **class** ClassUtil {
12. /\*\*
13. \* @Description: 根据一个接口返回该接口的所有类
14. \* @param c 接口
15. \* @return List<Class>    实现接口的所有类
16. \* @author LiYaoHua
17. \* @date 2012-4-5 上午11:22:24
18. \*/
19. @SuppressWarnings("unchecked")
20. **public** **static** List<Class> getAllClassByInterface(Class c){
21. List returnClassList = **new** ArrayList<Class>();
22. //判断是不是接口,不是接口不作处理
23. **if**(c.isInterface()){
24. String packageName = c.getPackage().getName();  //获得当前包名
25. **try** {
26. List<Class> allClass = getClasses(packageName);//获得当前包以及子包下的所有类
28. //判断是否是一个接口
29. **for**(**int** i = 0; i < allClass.size(); i++){
30. **if**(c.isAssignableFrom(allClass.get(i))){
31. **if**(!c.equals(allClass.get(i))){
32. returnClassList.add(allClass.get(i));
33. }
34. }
35. }
36. } **catch** (Exception e) {
37. // TODO: handle exception
38. }
39. }
40. **return** returnClassList;
41. }
43. /\*\*
44. \*
45. \* @Description: 根据包名获得该包以及子包下的所有类不查找jar包中的
46. \* @param pageName 包名
47. \* @return List<Class>    包下所有类
48. \* @author LiYaoHua
49. \* @date 2012-4-5 上午11:26:48
50. \*/
51. **private** **static** List<Class> getClasses(String packageName) **throws** ClassNotFoundException,IOException{
52. ClassLoader classLoader = Thread.currentThread().getContextClassLoader();
53. String path = packageName.replace(".", "/");
54. Enumeration<URL> resources = classLoader.getResources(path);
55. List<File> dirs = **new** ArrayList<File>();
56. **while**(resources.hasMoreElements()){
57. URL resource = resources.nextElement();
58. dirs.add(**new** File(resource.getFile()));
59. }
60. ArrayList<Class> classes = **new** ArrayList<Class>();
61. **for**(File directory:dirs){
62. classes.addAll(findClass(directory, packageName));
63. }
64. **return** classes;
65. }
67. **private** **static**  List<Class> findClass(File directory, String packageName)
68. **throws** ClassNotFoundException{
69. List<Class> classes = **new** ArrayList<Class>();
70. **if**(!directory.exists()){
71. **return** classes;
72. }
73. File[] files = directory.listFiles();
74. **for**(File file:files){
75. **if**(file.isDirectory()){
76. **assert** !file.getName().contains(".");
77. classes.addAll(findClass(file, packageName+"."+file.getName()));
78. }**else** **if**(file.getName().endsWith(".class")){
79. classes.add(Class.forName(packageName+"."+file.getName().substring(0,file.getName().length()-6)));
80. }
81. }
82. **return** classes;
83. }
84. }

