[运用URLClassLoader加载外部jar包的java类，生成Class文件](http://blog.csdn.net/wan368500/article/details/8215668)

标签： [class](http://www.csdn.net/tag/class)[Class](http://www.csdn.net/tag/Class)[classloader](http://www.csdn.net/tag/classloader)[java](http://www.csdn.net/tag/java)[Java](http://www.csdn.net/tag/Java)[JAVA](http://www.csdn.net/tag/JAVA)

2012-11-23 11:18 7827人阅读 [评论](http://blog.csdn.net/wan368500/article/details/8215668#comments)(1) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/wan368500/article/details/8215668#report)

版权声明：本文为博主原创文章，未经博主允许不得转载。

先来一段网上copy的话术：

      //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

很多时候 我们写的java程序是分模块的，有很好的扩展机制，即我们可以为我们自己的java类添加插件，来运行将来某天我们可能开发出来的类，以下称这些类为插件类。

下边是一种简单的实现方法：

Class A 作为程序的主入口，其中包含了程序的执行入口（main）函数。然后在main函数中通过外部的配置文件，然后通过外部的配置文件，我们可以获得插件类的信息（位于哪个jar包，jar包的具体路径），然后获得jar包中某一个类的实例，来完成相应的工作。这个jar包很可能是外部的jar包，是我们自己写好的，那么我们放到哪里，他才能自己找到呢？我尝试过很多次，除非将其具体目录，放到class\_path中才可以成功执行，否则报的异常只有一个ClassNotFoundException，就是找不到类。不过还有一种方法，就是将该jar包解压到运行jar包所在的目录，这样就可以通过class\_path中的.来获得相应的类了。不过这样会显得很不专业，java写出来的东西都是jar包啊，自己感觉的。放到claspath中，不能每次写出新的jar包都配置一遍吧！

如此出现了如下的解决办法：

想了解解决办法的含义，首先要了解java的类加载机制。众所周知，程序若想执行，必须加载到内存当中才能成功执行。java程序并不是可执行文件，由许多独立的类文件来完成。所以java中加载程序是以类为单外来完成的。这也就需要我们来简单了解一下java的class loader加载机制。

java程序开始执行，遇到的第一个classloader是bootstrap classloader，这个classloader是用c++语言编写，通过他来完成加载java中的核心类。第二个classloader是extension classloader，加载的是jre/lib目录中的ext目录中的jar包。然后第三个是system classloader，也被称为应用加载器，主要负责完成加载-classpath 或者系统中的全局变量ClassPath中的类。System.out.println(System.getProperty(“java.class.path”));可以获得classpath的配置，也就是system classloader 加载的类，第四个class loader可能是用户自定义的加载器，来自定义加载类。通常一个类的加载过程是这样的通过当前的类加载器的父加载器尝试查找，如果没有再找其父加载器尝试加载，直到最终的bootstrap classloader为止，如果还没有找到，那么就开始从上往下加载类。这样做的目的是防止自定义的类来覆盖系统中的类，如果没有这种机制很容易出现这种笑话，自己写了一个String类，然后new string的时候是自己写的String类，这样就比较好玩了。

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

    下面是我最近做项目的实际代码，可以作为参考：

1.自己定义URLClassLoader对象加载外部jar包，针对jar包里面不再出现别的jar包的情况，即只解析.class文件：

**[java]** [view plain](http://blog.csdn.net/wan368500/article/details/8215668) [copy](http://blog.csdn.net/wan368500/article/details/8215668)

1. **private** **static** **void** test1() {
2. String path = "D:\\test.jar";//外部jar包的路径
3. Set<Class<?>> classes = **new** LinkedHashSet<Class<?>>();//所有的Class对象
4. Map<Class<?>, Annotation[]> classAnnotationMap = **new** HashMap<Class<?>, Annotation[]>();//每个Class对象上的注释对象
5. Map<Class<?>, Map<Method, Annotation[]>> classMethodAnnoMap = **new** HashMap<Class<?>, Map<Method,Annotation[]>>();//每个Class对象中每个方法上的注释对象
6. **try** {
7. JarFile jarFile = **new** JarFile(**new** File(path));
8. URL url = **new** URL("file:" + path);
9. ClassLoader loader = **new** URLClassLoader(**new** URL[]{url});//自己定义的classLoader类，把外部路径也加到load路径里，使系统去该路经load对象
10. Enumeration<JarEntry> es = jarFile.entries();
11. **while** (es.hasMoreElements()) {
12. JarEntry jarEntry = (JarEntry) es.nextElement();
13. String name = jarEntry.getName();
14. **if**(name != **null** && name.endsWith(".class")){//只解析了.class文件，没有解析里面的jar包
15. //默认去系统已经定义的路径查找对象，针对外部jar包不能用
16. //Class<?> c = Thread.currentThread().getContextClassLoader().loadClass(name.replace("/", ".").substring(0,name.length() - 6));
17. Class<?> c = loader.loadClass(name.replace("/", ".").substring(0,name.length() - 6));//自己定义的loader路径可以找到
18. System.out.println(c);
19. classes.add(c);
20. Annotation[] classAnnos = c.getDeclaredAnnotations();
21. classAnnotationMap.put(c, classAnnos);
22. Method[] classMethods = c.getDeclaredMethods();
23. Map<Method, Annotation[]> methodAnnoMap = **new** HashMap<Method, Annotation[]>();
24. **for**(**int** i = 0;i<classMethods.length;i++){
25. Annotation[] a = classMethods[i].getDeclaredAnnotations();
26. methodAnnoMap.put(classMethods[i], a);
27. }
28. classMethodAnnoMap.put(c, methodAnnoMap);
29. }
30. }
31. System.out.println(classes.size());
32. } **catch** (IOException e) {
33. e.printStackTrace();
34. } **catch** (ClassNotFoundException e) {
35. e.printStackTrace();
36. }
37. }

以上的这种情况可以在别的project项目里写test方法，是平时最常用的，如果当.class文件里有依赖别的jar包里的对象的时候，就要把该jar包拷贝到写此测试方法的project并buildPath,不然的话运行的时候会报找不到Class对象的异常。

2.第二种情况是针对加载jar包里面的jar包的Class对象，还有读取某一个properties文件的方法。

**[java]** [view plain](http://blog.csdn.net/wan368500/article/details/8215668) [copy](http://blog.csdn.net/wan368500/article/details/8215668)

1. **private** **static** **void** test2() {
2. String path = "D:\\test.jar";//此jar包里还有别的jar包
3. **try** {
4. JarFile jarfile = **new** JarFile(**new** File(path));
5. Enumeration<JarEntry> es = jarfile.entries();
6. **while** (es.hasMoreElements()) {
7. JarEntry je = es.nextElement();
8. String name = je.getName();
9. **if**(name.endsWith(".jar")){//读取jar包里的jar包
10. File f = **new** File(name);
11. JarFile j = **new** JarFile(f);
12. Enumeration<JarEntry> e = j.entries();
13. **while** (e.hasMoreElements()) {
14. JarEntry jarEntry = (JarEntry) e.nextElement();
15. System.out.println(jarEntry.getName());
16. //.........接下去和上面的方法类似
17. }
18. }
19. //    System.out.println(je.getName());
20. **if**(je.getName().equals("entity\_pk.properties")){
21. InputStream inputStream = jarfile.getInputStream(je);
22. Properties properties = **new** Properties();
23. properties.load(inputStream);
24. Iterator<Object> ite = properties.keySet().iterator();
25. **while** (ite.hasNext()) {
26. Object key = ite.next();
27. System.out.println(key + " : " +properties.get(key));
28. }
29. }
30. }
31. } **catch** (IOException e) {
32. e.printStackTrace();
33. }
34. }

3.第三种情况是在该项目下获取某个包的Class对象，当然了，测试方法是在该项目下写的（这样classLoader就直接可以知道对象了，不需要再自定义URLClassLoader了，用Thread.currentThread().getContextClassLoader().loadClass(.....)就可以直接获得Class对象了，回去ClassPath下找，System.out.print(System.getProperty("java.class.path"))就可以找到classPath路径）。

**[java]** [view plain](http://blog.csdn.net/wan368500/article/details/8215668) [copy](http://blog.csdn.net/wan368500/article/details/8215668)

1. **private** **static** Set<Class<?>>  getclass() {
2. Set<Class<?>> classes = **new** LinkedHashSet<Class<?>>();
3. **boolean** flag = **true**;//是否循环迭代
5. String packName = "com.yk.framework.db";
6. //  String packName = "org.jdom";
7. String packDir = packName.replace(".", "/");
8. Enumeration<URL> dir;
9. **try** {
10. dir = Thread.currentThread().getContextClassLoader().getResources(packDir);
11. **while**(dir.hasMoreElements()){
12. URL url = dir.nextElement();
13. System.out.println("url:\*\*\*" + url);
14. String protocol = url.getProtocol();//获得协议号
15. **if**("file".equals(protocol)){
16. System.err.println("file类型的扫描");
17. String filePath = URLDecoder.decode(url.getFile(), "UTF-8");
18. System.out.println("filePath :" + filePath);
19. findAndAddClassesInPackageByFile(packName, filePath,flag,classes);
20. }**else** **if**("jar".equals(protocol)){
21. System.err.println("jar类型扫描");
22. JarFile jar;
23. jar = ((JarURLConnection)url.openConnection()).getJarFile();
24. Enumeration<JarEntry> entries = jar.entries();
25. **while**(entries.hasMoreElements()){
26. JarEntry entry = entries.nextElement();
27. String name = entry.getName();
28. System.out.println(">>>>:" + name);
29. //......
30. }
32. }
34. }
36. } **catch** (IOException e) {
37. e.printStackTrace();
38. }
39. System.out.println(classes.size());
40. **return** classes;
42. }

**[java]** [view plain](http://blog.csdn.net/wan368500/article/details/8215668) [copy](http://blog.csdn.net/wan368500/article/details/8215668)



下面上第二段代码

**[java]** [view plain](http://blog.csdn.net/wan368500/article/details/8215668) [copy](http://blog.csdn.net/wan368500/article/details/8215668)

1. **private** **static** **void** findAndAddClassesInPackageByFile(String packName, String filePath, **final** **boolean** flag, Set<Class<?>> classes) {
2. File dir = **new** File(filePath);
3. **if**( !dir.exists() || !dir.isDirectory()){
4. System.out.println("此路径下没有文件");
5. **return**;
6. }
7. File[] dirfiles = dir.listFiles(**new** FileFilter(){
8. @Override
9. **public** **boolean** accept(File pathname) {
10. **return** flag && pathname.isDirectory() || pathname.getName().endsWith(".class");
11. }
12. });
13. **for** (File file : dirfiles) {
14. **if**(file.isDirectory()){//如果是目录，继续扫描
15. findAndAddClassesInPackageByFile(packName + "." + file.getName(),file.getAbsolutePath(),flag,classes);
16. }**else**{//如果是文件
17. String className = file.getName().substring(0,file.getName().length() - 6);
18. System.out.println("类名：" +className);
19. **try** {
20. classes.add(Thread.currentThread().getContextClassLoader().loadClass(packName + "." + className));
21. } **catch** (ClassNotFoundException e) {
22. e.printStackTrace();
23. }
24. }
25. }
26. }

好了，大致就是这样，好好学习，天天向上！哈哈~~

补充：

4.读取jar包中的entity\_pk.properties键值对到项目本地entity\_pk.properties文件中

**[java]** [view plain](http://blog.csdn.net/wan368500/article/details/8215668) [copy](http://blog.csdn.net/wan368500/article/details/8215668)

1. /\*\*
2. \* 读取jar包中的entity\_pk.properties键值对到项目本地entity\_pk.properties文件中
3. \*/
4. **private** **static** **void** test2() {
5. String path = "D:\\test.jar";
6. **try** {
7. JarFile jarFile = **new** JarFile(**new** File(path));
8. Enumeration<JarEntry> es = jarFile.entries();
9. **while** (es.hasMoreElements()) {
10. JarEntry jarEntry = (JarEntry) es.nextElement();
11. String name = jarEntry.getName();
12. **if**(name.equals("entity\_pk.properties")){
13. InputStream inputStream = **new** FileInputStream(name);
14. Properties prop = **new** Properties();
15. prop.load(inputStream);//先load本地的entity\_pk.propertoes文件
16. InputStream inputStream2 = jarFile.getInputStream(jarEntry);//获得jar包entity\_pk.propertoes文件流
17. Properties prop2 = **new** Properties();
18. prop2.load(inputStream2);
19. Enumeration<Object> ks = prop2.keys();
20. OutputStream out = **new** FileOutputStream(name);
21. **while** (ks.hasMoreElements()) {
22. String key =  (String)ks.nextElement();
23. System.out.println(key + " : " +prop2.getProperty(key));
24. prop.setProperty(key, prop2.getProperty(key));//把jar包entity\_pk.properties键值对放入本地
25. }
26. prop.store(out,"");
27. }
28. }
30. } **catch** (IOException e) {
31. e.printStackTrace();
32. }
34. }

 5.读写db\_config.xml文件

**[java]** [view plain](http://blog.csdn.net/wan368500/article/details/8215668) [copy](http://blog.csdn.net/wan368500/article/details/8215668)

1. /\*\*
2. \* 读写db\_config.xml文件
3. \*/
4. **private** **static** **void** test3() {
5. String path = "D:\\test.jar";
6. **try** {
7. JarFile jarFile = **new** JarFile(**new** File(path));
8. Enumeration<JarEntry> es = jarFile.entries();
9. **while** (es.hasMoreElements()) {
10. JarEntry jarEntry = (JarEntry) es.nextElement();
11. String name = jarEntry.getName();
12. **if**(name.equals("db\_config.xml")){
13. InputStream inputStream = jarFile.getInputStream(jarEntry);
14. SAXBuilder builder = **new** SAXBuilder();
15. Document doc = **null**;
16. **try** {
17. doc = builder.build(inputStream);
18. **if**(doc != **null**){
19. Element e = doc.getRootElement();
20. Element proxool = e.getChild("proxool");
21. List<Element> children = proxool.getChildren();
22. **for** (Element element : children) {
23. System.out.println(element.getText());
24. }
25. Element s = **new** Element("test");
26. s.addContent("test");
27. proxool.addContent(s);
28. XMLOutputter outputter = **new** XMLOutputter();
29. outputter.output(doc, **new** FileOutputStream("config/db\_config.xml"));
30. }
31. } **catch** (JDOMException e) {
32. e.printStackTrace();
33. }**finally**{
34. **if**(inputStream != **null**){
35. inputStream.close();
36. }
37. }
39. }
41. }
43. } **catch** (IOException e) {
44. e.printStackTrace();
45. }
46. }