[zip4j -- Java处理zip压缩文件的完整解决方案](http://blog.csdn.net/djun100/article/details/18007099)

2014-01-08 18:21 22689人阅读 [评论](http://blog.csdn.net/djun100/article/details/18007099#comments)(9) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/djun100/article/details/18007099#report)

http://static.blog.csdn.net/images/category_icon.jpg 分类：

android（226） http://static.blog.csdn.net/images/arrow_triangle%20_down.jpg

maven依赖

|  |
| --- |
| <dependency>  <groupId>net.lingala.zip4j</groupId>  <artifactId>zip4j</artifactId>  <version>1.3.2</version>  </dependency> |

目录[(?)[+]](http://blog.csdn.net/djun100/article/details/18007099)

**前言**

一个多月前，因项目需要对[**Java**](http://lib.csdn.net/base/java)语言下的zip格式压缩文件的处理作了一些了解，尝试了多种开源项目并写了几篇博客做记录：

* [http://blog.csdn.net/zhangyihui1986/article/details/7724229](http://write.blog.csdn.net/postlist/1181273/all)
* <http://blog.csdn.net/zhangyihui1986/article/details/7723649>
* <http://blog.csdn.net/zhangyihui1986/article/details/7724616>

 ZIP4J，作为解决了我的问题的终极解决方案，本来一开始在[**搜索引擎**](http://lib.csdn.net/base/searchengine)上就看到了它的踪迹，但因天朝的网络环境问题，zip4j的官网一直无法访问，最终使我多走了好多冤枉路，期间试过JDK的zip包，试过Apache的zip解决方案，也试过如winzipaes等其它的开源框架，最终没有满足自己的需求，最后，我不得已挂了一下代理将zip4j下载了下来，试用了一下，果然威力无比，所到之处所向披靡...

闲话少说，如果需要可以到zip4j的官网下载该开源项目：

<http://www.lingala.net/zip4j/>

不过需要提醒的是可能无法直接访问，如果无法正常访问，请自行准备代理访问，如果各位嫌麻烦，可以到[这里下载](http://download.csdn.net/detail/zhangyihui1986/4418509)：

<http://download.csdn.net/detail/zhangyihui1986/4418509>

这是我的CSDN资源链接，下载需要3分，您如果分数不多，可以留言索取，呵呵...我也需要积分，请谅解！

官网上下载的资源好像是不带API帮助文档的，我利用其源码生成了一份，也一并打在我的资源文件中，希望能帮到大家。

**ZIP4J的官方说明**

（自己翻译了一下，英文不好，呵呵...）

**Key features(主要特性):**

* Create, Add, Extract, Update, Remove files from a Zip file  
  针对ZIP压缩文件创建、添加、抽出、更新和移除文件
* Read/Write password protected Zip files  
  (读写有密码保护的Zip文件)
* Supports AES 128/256 Encryption  
  (支持AES 128/256算法加密)
* Supports Standard Zip Encryption  
  (支持标准Zip算法加密)
* Supports Zip64 format  
  (支持zip64格式)
* Supports Store (No Compression) and Deflate compression method  
  (支持Store(非压缩)和Deflate压缩方法---不太明白)
* Create or extract files from Split Zip files (Ex: z01, z02,...zip)  
  (针对分块zip文件创建和抽出文件)
* Supports Unicode file names  
  (支持Unicode编码文件名)
* Progress Monitor  
  (进度监控)

从上面的主要特性可以看出，zip4j的功能是非常强大的，完全可以利用其写个类似于好压的zip文件管理软件，但我们用地最多的可能还是利用其作一些简单的解压和压缩操作，其它的功能极少触碰，我也一样，呵呵...

**使用注意点**

zip4j默认采用UTF-8编码，所以它支持中文，同时也支持密码，而且支持多种压缩[**算法**](http://lib.csdn.net/base/datastructure)，可以说功能强大，但使用起来却非常简单，当然，如果需求比较复杂，那就得好好去研究了。如果你仅仅是简单地解压一个zip压缩文件，那么只需要简单地几步即可：

**[java]** [view plaincopy](http://blog.csdn.net/zhyh1986/article/details/7921376)

1. **public** **static** **void** unzip(File zipFile, String dest, String passwd) **throws** ZipException {
2. ZipFile zFile = **new** ZipFile(zipFile);  // 首先创建ZipFile指向磁盘上的.zip文件
3. zFile.setFileNameCharset("GBK");       // 设置文件名编码，在GBK系统中需要设置
4. **if** (!zFile.isValidZipFile()) {   // 验证.zip文件是否合法，包括文件是否存在、是否为zip文件、是否被损坏等
5. **throw** **new** ZipException("压缩文件不合法,可能被损坏.");
6. }
7. File destDir = **new** File(dest);     // 解压目录
8. **if** (destDir.isDirectory() && !destDir.exists()) {
9. destDir.mkdir();
10. }
11. **if** (zFile.isEncrypted()) {
12. zFile.setPassword(passwd.toCharArray());  // 设置密码
13. }
14. zFile.extractAll(dest);      // 将文件抽出到解压目录(解压)
15. }

 当然将指定文件压缩成zip文件也是非常简单的事，此处不再贴代码，如有需要请参看下面的完整示例。

 提示：如果将要解压的压缩文件中的文件名含有中文，解压时需要注意一点，就是设置文件名字符集方法

**[java]** [view plaincopy](http://blog.csdn.net/zhyh1986/article/details/7921376)

1. zFile.setFileNameCharset("GBK");

这个方法的调用一定要靠前，要靠多前呢？其实最好在创建ZipFile之后就设置上，至少要在

**[java]** [view plaincopy](http://blog.csdn.net/zhyh1986/article/details/7921376)

1. zFile.isValidZipFile()

这个方法调用之前调用，我在应用时因为这个问题耽误好久，最后查看源码才弄明白，好像是ZipFile在验证方法中就将编码设置好，以后就不再对文件名编码作改变了。

**完整示例**

下面提供一个自己写的例子，鄙人才疏学浅，天分也差，写的代码质量很差，斗胆贴上，希望能起到抛砖引玉的作用。

**[java]** [view plaincopy](http://blog.csdn.net/zhyh1986/article/details/7921376)

1. **package** com.ninemax.cul.util;
3. **import** java.io.File;
4. **import** java.util.ArrayList;
5. **import** java.util.Collections;
6. **import** java.util.List;
8. **import** org.apache.commons.lang3.StringUtils;
10. **import** net.lingala.zip4j.core.ZipFile;
11. **import** net.lingala.zip4j.exception.ZipException;
12. **import** net.lingala.zip4j.model.FileHeader;
13. **import** net.lingala.zip4j.model.ZipParameters;
14. **import** net.lingala.zip4j.util.Zip4jConstants;
16. /\*\*
17. \* ZIP压缩文件操作工具类
18. \* 支持密码
19. \* 依赖zip4j开源项目(http://www.lingala.net/zip4j/)
20. \* 版本1.3.1
21. \* @author ninemax
22. \*/
23. **public** **class** CompressUtil {
25. /\*\*
26. \* 使用给定密码解压指定的ZIP压缩文件到指定目录
27. \* <p>
28. \* 如果指定目录不存在,可以自动创建,不合法的路径将导致异常被抛出
29. \* @param zip 指定的ZIP压缩文件
30. \* @param dest 解压目录
31. \* @param passwd ZIP文件的密码
32. \* @return 解压后文件数组
33. \* @throws ZipException 压缩文件有损坏或者解压缩失败抛出
34. \*/
35. **public** **static** File [] unzip(String zip, String dest, String passwd) **throws** ZipException {
36. File zipFile = **new** File(zip);
37. **return** unzip(zipFile, dest, passwd);
38. }
40. /\*\*
41. \* 使用给定密码解压指定的ZIP压缩文件到当前目录
42. \* @param zip 指定的ZIP压缩文件
43. \* @param passwd ZIP文件的密码
44. \* @return  解压后文件数组
45. \* @throws ZipException 压缩文件有损坏或者解压缩失败抛出
46. \*/
47. **public** **static** File [] unzip(String zip, String passwd) **throws** ZipException {
48. File zipFile = **new** File(zip);
49. File parentDir = zipFile.getParentFile();
50. **return** unzip(zipFile, parentDir.getAbsolutePath(), passwd);
51. }
53. /\*\*
54. \* 使用给定密码解压指定的ZIP压缩文件到指定目录
55. \* <p>
56. \* 如果指定目录不存在,可以自动创建,不合法的路径将导致异常被抛出
57. \* @param zip 指定的ZIP压缩文件
58. \* @param dest 解压目录
59. \* @param passwd ZIP文件的密码
60. \* @return  解压后文件数组
61. \* @throws ZipException 压缩文件有损坏或者解压缩失败抛出
62. \*/
63. **public** **static** File [] unzip(File zipFile, String dest, String passwd) **throws** ZipException {
64. ZipFile zFile = **new** ZipFile(zipFile);
65. zFile.setFileNameCharset("GBK");
66. **if** (!zFile.isValidZipFile()) {
67. **throw** **new** ZipException("压缩文件不合法,可能被损坏.");
68. }
69. File destDir = **new** File(dest);
70. **if** (destDir.isDirectory() && !destDir.exists()) {
71. destDir.mkdir();
72. }
73. **if** (zFile.isEncrypted()) {
74. zFile.setPassword(passwd.toCharArray());
75. }
76. zFile.extractAll(dest);
78. List<FileHeader> headerList = zFile.getFileHeaders();
79. List<File> extractedFileList = **new** ArrayList<File>();
80. **for**(FileHeader fileHeader : headerList) {
81. **if** (!fileHeader.isDirectory()) {
82. extractedFileList.add(**new** File(destDir,fileHeader.getFileName()));
83. }
84. }
85. File [] extractedFiles = **new** File[extractedFileList.size()];
86. extractedFileList.toArray(extractedFiles);
87. **return** extractedFiles;
88. }
90. /\*\*
91. \* 压缩指定文件到当前文件夹
92. \* @param src 要压缩的指定文件
93. \* @return 最终的压缩文件存放的绝对路径,如果为null则说明压缩失败.
94. \*/
95. **public** **static** String zip(String src) {
96. **return** zip(src,**null**);
97. }
99. /\*\*
100. \* 使用给定密码压缩指定文件或文件夹到当前目录
101. \* @param src 要压缩的文件
102. \* @param passwd 压缩使用的密码
103. \* @return 最终的压缩文件存放的绝对路径,如果为null则说明压缩失败.
104. \*/
105. **public** **static** String zip(String src, String passwd) {
106. **return** zip(src, **null**, passwd);
107. }
109. /\*\*
110. \* 使用给定密码压缩指定文件或文件夹到当前目录
111. \* @param src 要压缩的文件
112. \* @param dest 压缩文件存放路径
113. \* @param passwd 压缩使用的密码
114. \* @return 最终的压缩文件存放的绝对路径,如果为null则说明压缩失败.
115. \*/
116. **public** **static** String zip(String src, String dest, String passwd) {
117. **return** zip(src, dest, **true**, passwd);
118. }
120. /\*\*
121. \* 使用给定密码压缩指定文件或文件夹到指定位置.
122. \* <p>
123. \* dest可传最终压缩文件存放的绝对路径,也可以传存放目录,也可以传null或者"".<br />
124. \* 如果传null或者""则将压缩文件存放在当前目录,即跟源文件同目录,压缩文件名取源文件名,以.zip为后缀;<br />
125. \* 如果以路径分隔符(File.separator)结尾,则视为目录,压缩文件名取源文件名,以.zip为后缀,否则视为文件名.
126. \* @param src 要压缩的文件或文件夹路径
127. \* @param dest 压缩文件存放路径
128. \* @param isCreateDir 是否在压缩文件里创建目录,仅在压缩文件为目录时有效.<br />
129. \* 如果为false,将直接压缩目录下文件到压缩文件.
130. \* @param passwd 压缩使用的密码
131. \* @return 最终的压缩文件存放的绝对路径,如果为null则说明压缩失败.
132. \*/
133. **public** **static** String zip(String src, String dest, **boolean** isCreateDir, String passwd) {
134. File srcFile = **new** File(src);
135. dest = buildDestinationZipFilePath(srcFile, dest);
136. ZipParameters parameters = **new** ZipParameters();
137. parameters.setCompressionMethod(Zip4jConstants.COMP\_DEFLATE);           // 压缩方式
138. parameters.setCompressionLevel(Zip4jConstants.DEFLATE\_LEVEL\_NORMAL);    // 压缩级别
139. **if** (!StringUtils.isEmpty(passwd)) {
140. parameters.setEncryptFiles(**true**);
141. parameters.setEncryptionMethod(Zip4jConstants.ENC\_METHOD\_STANDARD); // 加密方式
142. parameters.setPassword(passwd.toCharArray());
143. }
144. **try** {
145. ZipFile zipFile = **new** ZipFile(dest);
146. **if** (srcFile.isDirectory()) {
147. // 如果不创建目录的话,将直接把给定目录下的文件压缩到压缩文件,即没有目录结构
148. **if** (!isCreateDir) {
149. File [] subFiles = srcFile.listFiles();
150. ArrayList<File> temp = **new** ArrayList<File>();
151. Collections.addAll(temp, subFiles);
152. zipFile.addFiles(temp, parameters);
153. **return** dest;
154. }
155. zipFile.addFolder(srcFile, parameters);
156. } **else** {
157. zipFile.addFile(srcFile, parameters);
158. }
159. **return** dest;
160. } **catch** (ZipException e) {
161. e.printStackTrace();
162. }
163. **return** **null**;
164. }
166. /\*\*
167. \* 构建压缩文件存放路径,如果不存在将会创建
168. \* 传入的可能是文件名或者目录,也可能不传,此方法用以转换最终压缩文件的存放路径
169. \* @param srcFile 源文件
170. \* @param destParam 压缩目标路径
171. \* @return 正确的压缩文件存放路径
172. \*/
173. **private** **static** String buildDestinationZipFilePath(File srcFile,String destParam) {
174. **if** (StringUtils.isEmpty(destParam)) {
175. **if** (srcFile.isDirectory()) {
176. destParam = srcFile.getParent() + File.separator + srcFile.getName() + ".zip";
177. } **else** {
178. String fileName = srcFile.getName().substring(0, srcFile.getName().lastIndexOf("."));
179. destParam = srcFile.getParent() + File.separator + fileName + ".zip";
180. }
181. } **else** {
182. createDestDirectoryIfNecessary(destParam);  // 在指定路径不存在的情况下将其创建出来
183. **if** (destParam.endsWith(File.separator)) {
184. String fileName = "";
185. **if** (srcFile.isDirectory()) {
186. fileName = srcFile.getName();
187. } **else** {
188. fileName = srcFile.getName().substring(0, srcFile.getName().lastIndexOf("."));
189. }
190. destParam += fileName + ".zip";
191. }
192. }
193. **return** destParam;
194. }
196. /\*\*
197. \* 在必要的情况下创建压缩文件存放目录,比如指定的存放路径并没有被创建
198. \* @param destParam 指定的存放路径,有可能该路径并没有被创建
199. \*/
200. **private** **static** **void** createDestDirectoryIfNecessary(String destParam) {
201. File destDir = **null**;
202. **if** (destParam.endsWith(File.separator)) {
203. destDir = **new** File(destParam);
204. } **else** {
205. destDir = **new** File(destParam.substring(0, destParam.lastIndexOf(File.separator)));
206. }
207. **if** (!destDir.exists()) {
208. destDir.mkdirs();
209. }
210. }
212. **public** **static** **void** main(String[] args) {
213. zip("d:\\test\\cc", "d:\\test\\cc.zip", "11");
214. //      try {
215. //          File[] files = unzip("d:\\test\\汉字.zip", "aa");
216. //          for (int i = 0; i < files.length; i++) {
217. //              System.out.println(files[i]);
218. //          }
219. //      } catch (ZipException e) {
220. //          e.printStackTrace();
221. //      }
222. }
223. }

需要学习的东西太多，没太多时间（或许只是借口）去研究它，上面的例子仅是简单地解压和压缩操作；但在使用中可以发现Zip4J功能比较完备，如果需要更多地支持，那就真要好好去研究一下它，也许它真的不会使您失望。。。

**补充**

**删除压缩文件中的目录**

看到有朋友在问如何删除压缩文件中的目录，在这里补充一下。

利用zip4j删除压缩文件中的目录，查阅API后很容易想到这样的方式：

**[java]** [view plaincopy](http://blog.csdn.net/zhyh1986/article/details/7921376)

1. ZipFile zipFile = **new** ZipFile("d:\\FeiQ-V2.5.zip");
2. zipFile.setFileNameCharset("GBK");
3. zipFile.removeFile("sounds/");      // sounds是zip文件中的一个目录

但这种直接删除压缩文件中非空目录的方式是不会成功的，你会看到zip文件丝毫没有变化，虽然目录对应的FileHeader已被删除（表现就是如果这时再将目录下的所有文件删除，则该目录随之消失） ；因此我们需要将该目录下所有的文件都删除掉，最后再将目录删除，根据这个思路，我们很容易形成如下的代码：

**[java]** [view plaincopy](http://blog.csdn.net/zhyh1986/article/details/7921376)

1. **void** removeDirFromZipArchive(String file, String removeDir) **throws** ZipException {
2. // 创建ZipFile并设置编码
3. ZipFile zipFile = **new** ZipFile(file);
4. zipFile.setFileNameCharset("GBK");
6. // 给要删除的目录加上路径分隔符
7. **if** (!removeDir.endsWith(File.separator)) removeDir += File.separator;
9. // 如果目录不存在, 直接返回
10. FileHeader dirHeader = zipFile.getFileHeader(removeDir);
11. **if** (**null** == dirHeader) **return**;
13. // 遍历压缩文件中所有的FileHeader, 将指定删除目录下的子文件删除
14. List allHeaders = zipFile.getFileHeaders();
15. **for**(**int** i=0, len = allHeaders.size(); i<len; i++) {
16. FileHeader subHeader = (FileHeader) allHeaders.get(i);
17. **if** (subHeader.getFileName().startsWith(dirHeader.getFileName())
18. && !subHeader.getFileName().equals(dirHeader.getFileName())) {
19. zipFile.removeFile(subHeader);
20. }
21. }
22. // 最后删除指定目录
23. zipFile.removeFile(dirHeader);
24. }

这样仍然解决不了问题，如果你这样做了，那么你将会得到一个java.lang.IndexOutOfBoundsException异常，那么看似正常的代码为什么会报索引越界异常呢？其实我们通过zipFile.getFileHeaders()方法得到的List会随遍历中的删除操作而发生变化，也就是说我们删除了某个FileHeader，将会反映到该List中。每成功删除一个FileHeader，List长度就减1，而i一直在0至List的初始长度之间递增，反复几次后就可能出现越界异常。

为了避免这种情况发生，我们可以多做一些操作，比如可以在遍历中暂不进行删除操作，而只是将要删除的文件记录下来，遍历结束后再统一删除，最后将目录删除，经[**测试**](http://lib.csdn.net/base/softwaretest)，这个思路可以解决问题。

简单示例代码：

**[java]** [view plaincopy](http://blog.csdn.net/zhyh1986/article/details/7921376)

1. **void** removeDirFromZipArchive(String file, String removeDir) **throws** ZipException {
2. // 创建ZipFile并设置编码
3. ZipFile zipFile = **new** ZipFile(file);
4. zipFile.setFileNameCharset("GBK");
6. // 给要删除的目录加上路径分隔符
7. **if** (!removeDir.endsWith(File.separator)) removeDir += File.separator;
9. // 如果目录不存在, 直接返回
10. FileHeader dirHeader = zipFile.getFileHeader(removeDir);
11. **if** (**null** == dirHeader) **return**;
13. // 遍历压缩文件中所有的FileHeader, 将指定删除目录下的子文件名保存起来
14. List headersList = zipFile.getFileHeaders();
15. List<String> removeHeaderNames = **new** ArrayList<String>();
16. **for**(**int** i=0, len = headersList.size(); i<len; i++) {
17. FileHeader subHeader = (FileHeader) headersList.get(i);
18. **if** (subHeader.getFileName().startsWith(dirHeader.getFileName())
19. && !subHeader.getFileName().equals(dirHeader.getFileName())) {
20. removeHeaderNames.add(subHeader.getFileName());
21. }
22. }
23. // 遍历删除指定目录下的所有子文件, 最后删除指定目录(此时已为空目录)
24. **for**(String headerNameString : removeHeaderNames) {
25. zipFile.removeFile(headerNameString);
26. }
27. zipFile.removeFile(dirHeader);
28. }

[http://static.blog.csdn.net/images/save_snippets.png](javascript:;)

也许还有其它的办法来解决此问题，如果您有需要，就留待您来解决了。

转自：<http://blog.csdn.net/zhyh1986/article/details/7921376>