 这个工具类的功能为：

* + - （1）可以压缩文件，也可以压缩文件夹
    - （2）同时支持压缩多级文件夹，工具内部做了递归处理
    - （3）碰到空的文件夹，也可以压缩
    - （4）可以选择是否保留原来的目录结构，如果不保留，所有文件跑压缩包根目录去了，且空文件夹直接舍弃。注意：如果不保留文件原来目录结构，在碰到文件名相同的文件时，会压缩失败。
    - （5）代码中提供了2个压缩文件的方法，一个的输入参数为文件夹路径，一个为文件列表，可根据实际需求选择方法。

        下面直接上代码

**一、代码**



1

package com.tax.core.util;

2

import java.io.File;

3

import java.io.FileInputStream;

4

import java.io.FileOutputStream;

5

import java.io.IOException;

6

import java.io.OutputStream;

7

import java.util.ArrayList;

8

import java.util.List;

9

import java.util.zip.ZipEntry;

10

import java.util.zip.ZipOutputStream;

11

12

/\*\*

13

\* ZipUtils

14

\* @author ZENG.XIAO.YAN

15

\* @date 2017年11月19日 下午7:16:08

16

\* @version v1.0

17

\*/

18

public class ZipUtils {

19

20

private static final int  BUFFER\_SIZE = 2 \* 1024;

21

22

/\*\*

23

\* 压缩成ZIP 方法1

24

\* @param srcDir 压缩文件夹路径

25

\* @param out   压缩文件输出流

26

\* @param KeepDirStructure 是否保留原来的目录结构,true:保留目录结构;

27

\* false:所有文件跑到压缩包根目录下(注意：不保留目录结构可能会出现同名文件,会压缩失败)

28

\* @throws RuntimeException 压缩失败会抛出运行时异常

29

\*/

30

public static void toZip(String srcDir, OutputStream out, boolean KeepDirStructure)

31

throws RuntimeException{

32

33

long start = System.currentTimeMillis();

34

ZipOutputStream zos = null ;

35

try {

36

zos = new ZipOutputStream(out);

37

File sourceFile = new File(srcDir);

38

compress(sourceFile,zos,sourceFile.getName(),KeepDirStructure);

39

long end = System.currentTimeMillis();

40

System.out.println("压缩完成，耗时：" + (end - start) +" ms");

41

} catch (Exception e) {

42

throw new RuntimeException("zip error from ZipUtils",e);

43

}finally{

44

if(zos != null){

45

try {

46

zos.close();

47

} catch (IOException e) {

48

e.printStackTrace();

49

}

50

}

51

}

52

53

}

54

55

/\*\*

56

\* 压缩成ZIP 方法2

57

\* @param srcFiles 需要压缩的文件列表

58

\* @param out       压缩文件输出流

59

\* @throws RuntimeException 压缩失败会抛出运行时异常

60

\*/

61

public static void toZip(List<File> srcFiles , OutputStream out)throws RuntimeException {

62

long start = System.currentTimeMillis();

63

ZipOutputStream zos = null ;

64

try {

65

zos = new ZipOutputStream(out);

66

for (File srcFile : srcFiles) {

67

byte[] buf = new byte[BUFFER\_SIZE];

68

zos.putNextEntry(new ZipEntry(srcFile.getName()));

69

int len;

70

FileInputStream in = new FileInputStream(srcFile);

71

while ((len = in.read(buf)) != -1){

72

zos.write(buf, 0, len);

73

}

74

zos.closeEntry();

75

in.close();

76

}

77

long end = System.currentTimeMillis();

78

System.out.println("压缩完成，耗时：" + (end - start) +" ms");

79

} catch (Exception e) {

80

throw new RuntimeException("zip error from ZipUtils",e);

81

}finally{

82

if(zos != null){

83

try {

84

zos.close();

85

} catch (IOException e) {

86

e.printStackTrace();

87

}

88

}

89

}

90

}

91

92

93

/\*\*

94

\* 递归压缩方法

95

\* @param sourceFile 源文件

96

\* @param zos zip输出流

97

\* @param name 压缩后的名称

98

\* @param KeepDirStructure 是否保留原来的目录结构,true:保留目录结构;

99

\* false:所有文件跑到压缩包根目录下(注意：不保留目录结构可能会出现同名文件,会压缩失败)

100

\* @throws Exception

101

\*/

102

private static void compress(File sourceFile, ZipOutputStream zos, String name,

103

boolean KeepDirStructure) throws Exception{

104

byte[] buf = new byte[BUFFER\_SIZE];

105

if(sourceFile.isFile()){

106

// 向zip输出流中添加一个zip实体，构造器中name为zip实体的文件的名字

107

zos.putNextEntry(new ZipEntry(name));

108

// copy文件到zip输出流中

109

int len;

110

FileInputStream in = new FileInputStream(sourceFile);

111

while ((len = in.read(buf)) != -1){

112

zos.write(buf, 0, len);

113

}

114

// Complete the entry

115

zos.closeEntry();

116

in.close();

117

} else {

118

File[] listFiles = sourceFile.listFiles();

119

if(listFiles == null || listFiles.length == 0){

120

// 需要保留原来的文件结构时,需要对空文件夹进行处理

121

if(KeepDirStructure){

122

// 空文件夹的处理

123

zos.putNextEntry(new ZipEntry(name + "/"));

124

// 没有文件，不需要文件的copy

125

zos.closeEntry();

126

}

127

128

}else {

129

for (File file : listFiles) {

130

// 判断是否需要保留原来的文件结构

131

if (KeepDirStructure) {

132

// 注意：file.getName()前面需要带上父文件夹的名字加一斜杠,

133

// 不然最后压缩包中就不能保留原来的文件结构,即：所有文件都跑到压缩包根目录下了

134

compress(file, zos, name + "/" + file.getName(),KeepDirStructure);

135

} else {

136

compress(file, zos, file.getName(),KeepDirStructure);

137

}

138

139

}

140

}

141

}

142

}

143

144

public static void main(String[] args) throws Exception {

145

/\*\* 测试压缩方法1 \*/

146

FileOutputStream fos1 = new FileOutputStream(new File("c:/mytest01.zip"));

147

ZipUtils.toZip("D:/log", fos1,true);

148

149

/\*\* 测试压缩方法2 \*/

150

List<File> fileList = new ArrayList<>();

151

fileList.add(new File("D:/Java/jdk1.7.0\_45\_64bit/bin/jar.exe"));

152

fileList.add(new File("D:/Java/jdk1.7.0\_45\_64bit/bin/java.exe"));

153

FileOutputStream fos2 = new FileOutputStream(new File("c:/mytest02.zip"));

154

ZipUtils.toZip(fileList, fos2);

155

}

156

}

**二、注意事项**

    写该工具类时，有些注意事项说一下：

        （1）支持选择是否保留原来的文件目录结构，如果不保留，那么空文件夹直接不用处理。

        （1）碰到空文件夹时，如果需要保留目录结构，则直接添加个ZipEntry就可以了，不过就是这个entry的名字后面需要带上一斜杠（/）表示这个是目录。

        （2）递归时，不需要把zip输出流关闭，zip输出流的关闭应该是在调用完递归方法后面关闭

        （3）递归时，如果是个文件夹且需要保留目录结构，那么在调用方法压缩他的子文件时，需要把文件夹的名字加一斜杠给添加到子文件名字前面，这样压缩后才有多级目录。

**三、如何在javaWeb项目中使用该工具类**

    这个工具类在web项目中的使用场景就是多文件下载，我就简单说个下载多个excel表格的案例吧。

    代码中的步骤为：

        （1）创建一个临时文件夹

        （2）将要下载的文件生成至该临时文件夹内

        （3）当所有文件生成完后，获取HttpServletResponse获取设置下载的header

        （4）调用工具类的方法，传入上面生成的临时文件夹路径及response获取的输出流；这样就下载出来zip包了

        （5）递归删除掉上面生成的临时文件夹和文件

    下面为一个示例代码的代码片段，不是完整代码，简单看一下代码中的步骤

1

if(userList.size() > 0){

2

/\*\* 下面为下载zip压缩包相关流程 \*/

3

HttpServletRequest request = ServletActionContext.getRequest();

4

FileWriter writer;

5

/\*\* 1.创建临时文件夹 \*/

6

String rootPath = request.getSession().getServletContext().getRealPath("/");

7

File temDir = new File(rootPath + "/" + UUID.randomUUID().toString().replaceAll("-", ""));

8

if(!temDir.exists()){

9

temDir.mkdirs();

10

}

11

12

/\*\* 2.生成需要下载的文件，存放在临时文件夹内 \*/

13

// 这里我们直接来10个内容相同的文件为例，但这个10个文件名不可以相同

14

for (int i = 0; i < 10; i++) {

15

dataMap.put("userList", userList);

16

Map<String, String> endMap = new HashMap<>();

17

endMap.put("user", "老王");

18

endMap.put("time", "2017-10-10 10:50:55");

19

dataMap.put("endMap", endMap);

20

Configuration cfg = new Configuration(Configuration.VERSION\_2\_3\_22);

21

cfg.setServletContextForTemplateLoading(ServletActionContext.getServletContext(), "/ftl");

22

Template template = cfg.getTemplate("exportExcel.ftl");

23

writer = new FileWriter(temDir.getPath()+"/excel"+ i +".xls");

24

template.process(dataMap, writer);

25

writer.flush();

26

writer.close();

27

}

28

29

/\*\* 3.设置response的header \*/

30

HttpServletResponse response = ServletActionContext.getResponse();

31

response.setContentType("application/zip");

32

response.setHeader("Content-Disposition", "attachment; filename=excel.zip");

33

34

/\*\* 4.调用工具类，下载zip压缩包 \*/

35

// 这里我们不需要保留目录结构

36

ZipUtils.toZip(temDir.getPath(), response.getOutputStream(),false);

37

38

/\*\* 5.删除临时文件和文件夹 \*/

39

// 这里我没写递归，直接就这样删除了

40

File[] listFiles = temDir.listFiles();

41

for (int i = 0; i < listFiles.length; i++) {

42

listFiles[i].delete();

43

}

44

temDir.delete();

45

}