**Java对[zip格式压缩和解压缩](http://blog.csdn.net/lovingprince/article/details/1470887)**

**通过使用java的相关类可以实现对文件或文件夹的压缩，以及对压缩文件的解压。**

**1.1 ZIP和GZIP的区别**

**gzip是一种文件压缩工具（或该压缩工具产生的压缩文件格式），它的设计目标是处理单个的文件。gzip在压缩文件中的数据时使用的就是zlib。为了保存与文件属性有关的信息，gzip需要在压缩文件（\*.gz）中保存更多的头信息内容，而zlib不用考虑这一点。但gzip只适用于单个文件，所以我们在UNIX/Linux上经常看到的压缩包后缀都是\*.tar.gz或\*.tgz，也就是先用tar把多个文件打包成单个文件，再用gzip压缩的结果。   
 zip只是一种数据结构，跟rar同类型。zip是适用于压缩多个文件的格式（相应的工具有PkZip和WinZip等），因此，zip文件还要进一步包含文件目录结构的信息，比gzip的头信息更多。但需要注意，zip格式可采用多种压缩算法，我们常见的zip文件大多不是用zlib的算法压缩的，其压缩数据的格式与gzip大不一样。**

**1.2相关类与接口：**

**Checksum**接口：表示数据校验和的接口,被类Adler32和CRC32实现  
**Adler32** ：使用Alder32算法来计算Checksum数目  
**CRC32**：使用CRC32算法来计算Checksum数目

**CheckedInputStream**：InputStream派生类，可得到输入流的校验和Checksum,用于校验数据的完整性  
**CheckedOutputStream** ：OutputStream派生类，可得到输出流的校验Checksum， 用于校验数据的完整性

**DeflaterOutputStream** ：压缩类的基类。  
**ZipOutputStream**：DeflaterOutputStream的一个子类，把数据压缩成Zip文件格式  
**GZIPOutputStream** ：DeflaterOutputStream的一个子类，把数据压缩成GZip文件格式

I**nflaterInputStream**：解压缩类的基类  
**ZipInputStream**：InflaterInputStream的一个子类，能解压缩Zip格式的数据  
**GZIPInputStream**：InflaterInputStream的一个子类，能解压缩Zip格式的数据

**ZipEntry** 类：表示 ZIP 文件条目  
**ZipFile** 类：此类用于从 ZIP 文件读取条目

**1.3 压缩文件**

下面实例我们使用了apache的zip工具包（所在包为ant.jar ），因为java类型自带的不支持中文路径，不过两者使用的方式是一样的，只是apache压缩工具多了设置编码方式的接口，其他基本上是一样的。另外，如果使用org.apache.tools.zip.ZipOutputStream来压缩的话，我们只能使用org.apache.tools.zip.ZipEntry来解压，而不能使用java.util.zip.ZipInputStream来解压读取了，当然apache并未提供ZipInputStream类。

**public** **static** **void** compress(String srcFilePath, String destFilePath) {

File src = **new** File(srcFilePath);

**if** (!src.exists()) {

**throw** **new** RuntimeException(srcFilePath + "不存在");

}

File zipFile = **new** File(destFilePath);

**try** {

**FileOutputStream fos = new FileOutputStream(zipFile);**

**CheckedOutputStream cos = new CheckedOutputStream(fos, new CRC32());**

**ZipOutputStream zos = new ZipOutputStream(cos);**

**String baseDir = "";**

***compressbyType*(src, zos, baseDir);**

zos.close();

} **catch** (Exception e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

**private** **static** **void** compressbyType(File src, ZipOutputStream zos,

String baseDir) {

**if** (!src.exists())

**return**;

System.***out***.println("压缩" + baseDir + src.getName());

**if** (src.isFile()) {

*compressFile*(src, zos, baseDir);

} **else** **if** (src.isDirectory()) {

*compressDir*(src, zos, baseDir);

}

}

/\*\*

\* 压缩文件

\*

\*/

**private** **static** **void** compressFile(File file, ZipOutputStream zos,

String baseDir) {

**if** (!file.exists())

**return**;

**try** {

**BufferedInputStream bis = new BufferedInputStream(**

**new FileInputStream(file));**

**ZipEntry entry = new ZipEntry(baseDir + file.getName());**

**zos.putNextEntry(entry);**

**int count;**

**byte[] buf = new byte[*BUFSIZE*];**

**while ((count = bis.read(buf)) != -1) {**

**zos.write(buf, 0, count);**

**}**

bis.close();

} **catch** (Exception e) {

// **TODO**: handle exception

}

}

/\*\*

\* 压缩文件夹

\*

\*/

**private** **static** **void** compressDir(File dir, ZipOutputStream zos,

String baseDir) {

**if** (!dir.exists())

**return**;

File[] files = dir.listFiles();

**if(files.length == 0){**

**try {**

**zos.putNextEntry(new ZipEntry(baseDir + dir.getName()**

**+ File.*separator*));**

**} catch (IOException e) {**

**e.printStackTrace();**

**}**

**}**

**for** (File file : files) {

*compressbyType*(file, zos, baseDir + dir.getName() + File.***separator***);

}

}

总结步骤：

1. 创建压缩到的文件File zipFile = **new** File(destFilePath);
2. 根据zipFile生成ZipOutputStream用于写入即将被压缩的文件

**FileOutputStream fos = new FileOutputStream(zipFile);**

**CheckedOutputStream cos = new CheckedOutputStream(fos, new CRC32());**

**ZipOutputStream zos = new ZipOutputStream(cos);**

1. 循环遍历源文件，首先需要创建ZipEntry用于标记压缩文件中含有的条目

**ZipEntry entry = new ZipEntry(baseDir + file.getName());**

然后将条目增加到ZipOutputStream中，

**zos.putNextEntry(entry);**

最后再调用要写入条目对应文件的输入流读取文件内容写入到压缩文件中。

**BufferedInputStream bis = new BufferedInputStream(**

**new FileInputStream(file));**

**ZipEntry entry = new ZipEntry(baseDir + file.getName());**

**zos.putNextEntry(entry);**

**int count;**

**byte[] buf = new byte[*BUFSIZE*];**

**while ((count = bis.read(buf)) != -1) {**

**zos.write(buf, 0, count);**

**}**

**注意：**如果是空目录**直接zos.putNextEntry(new ZipEntry(baseDir + dir.getName()+ File.*separator*))**并不用写入文件内容，**其中最主要的涉及到目录的压缩的，就是这一句话  out.putNextEntry(new ZipEntry(base + "/")); //放入一级目录 (防止空目录被丢弃)**

**1.4 解压zip文件**

**public** **static** **void** decompress(String srcPath, String dest) **throws** Exception {

File file = **new** File(srcPath);

**if** (!file.exists()) {

**throw** **new** RuntimeException(srcPath + "所指文件不存在");

}

ZipFile zf = **new** ZipFile(file);

Enumeration entries = zf.getEntries();

ZipEntry entry = **null**;

**while** (entries.hasMoreElements()) {

entry = (ZipEntry) entries.nextElement();

System.***out***.println("解压" + entry.getName());

**if** (entry.isDirectory()) {

String dirPath = dest + File.***separator*** + entry.getName();

File dir = **new** File(dirPath);

dir.mkdirs();

} **else** {

// 表示文件

File f = **new** File(dest + File.***separator*** + entry.getName());

**if** (!f.exists()) {

String dirs = FileUtils.*getParentPath*(f);

File parentDir = **new** File(dirs);

parentDir.mkdirs();

}

f.createNewFile();

// 将压缩文件内容写入到这个文件中

InputStream is = zf.getInputStream(entry);

FileOutputStream fos = **new** FileOutputStream(f);

**int** count;

**byte**[] buf = **new** **byte**[8192];

**while** ((count = is.read(buf)) != -1) {

fos.write(buf, 0, count);

}

is.close();

fos.close();

}

}

}