Archives

Борислав Капукаранов и Владимир Панов
Ноември, 2014

Public

Поддържани архивни типове от Java 7

ZIP

JAR

.GZ

RAR

TAR

Поддържани архивни типове от Java 7

ZIP

JAR

.GZ

RAR

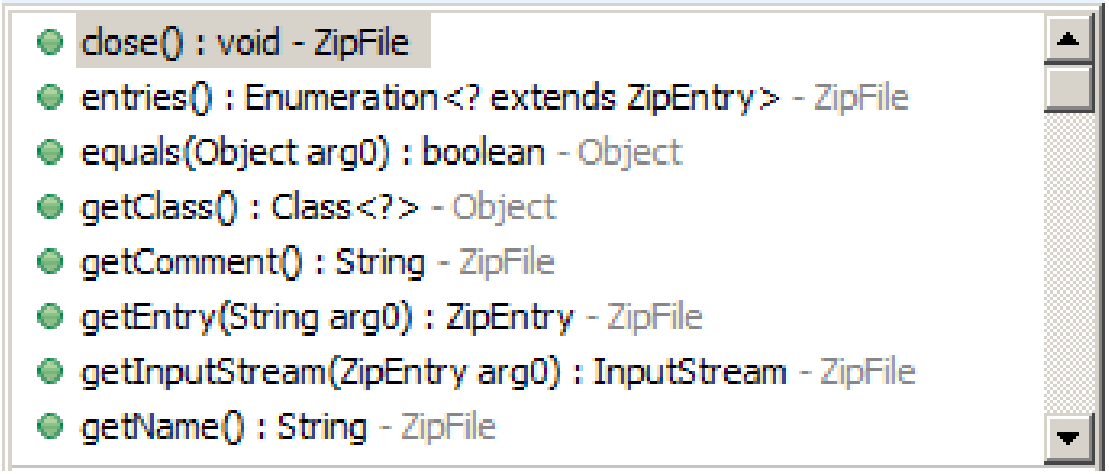
TAR

Поне не и в Java SE, open source решава този проблем.

ZipFile и ZipEntry

ZipFile

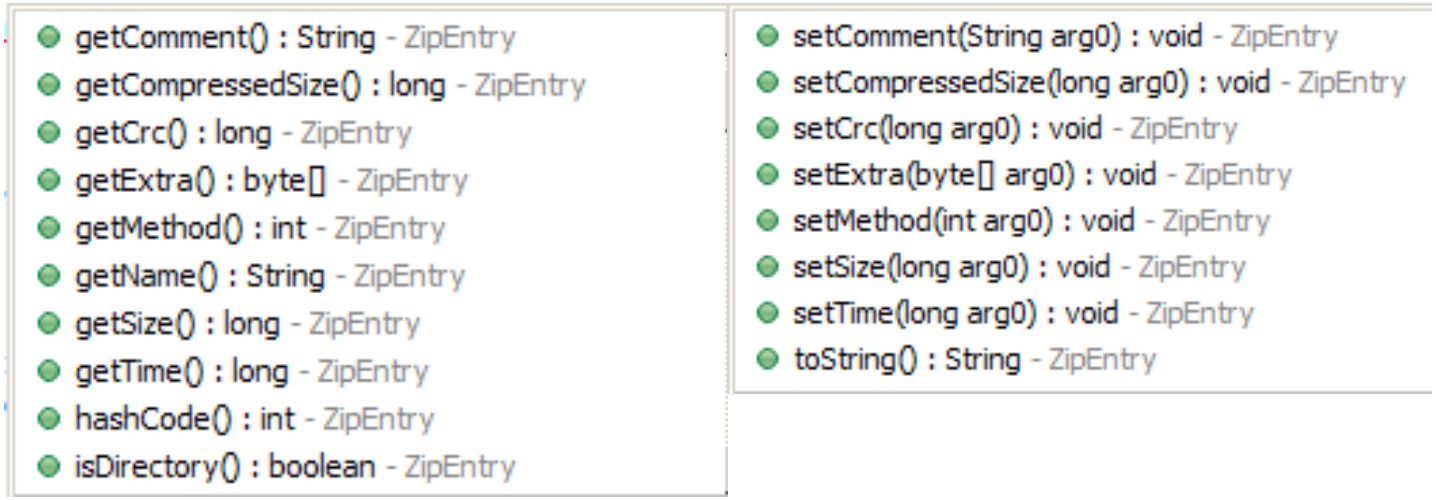
- **Основния клас за четене на ZipFile-ове.**
- **НЕ е наследник на класът File.**
- **Просто API, но не много мощно.**



ZipFile и ZipEntry

ZipEntry

- **Класът, който репрезентира компресиран файл в архива.**
- **Използва се основно за наблюдаване на качествата на компресирания файл.**



Четене преди Java 7…

Пример за четене на архив със ZipFile:

```
final ZipFile file = new ZipFile( "myFile.zip" );
try {
    final Enumeration<? extends ZipEntry> entries = file.entries();
    while ( entries.hasMoreElements() ) {
        final ZipEntry entry = entries.nextElement();
     System.out.println( entry.getName() );
        //use entry input stream:
        readInputStream( file.getInputStream( entry ) );
    }
}
finally {
    file.close();
}
private static int readInputStream( final InputStream is ) throws IOException {…}
```

Четене след Java 7…

Пример за четене на архив със ZipFile:

```
try (ZipFile file = new ZipFile( "myFile.zip" )) {
    final Enumeration<? extends ZipEntry> entries = file.entries();
    while ( entries.hasMoreElements() ) {
        final ZipEntry entry = entries.nextElement();
        System.out.println( entry.getName() );
        //use entry input stream:
        readInputStream( file.getInputStream( entry ) );
    }
}

private static int readInputStream( final InputStream is ) throws IOException {…}
```

Новата try-with-resources конструкция изчиства записа.

Създаване и променяне на архиви

ZipInputStream, ZipOutputStream

- **Основните подходи за създаване и модифициране на архиви преди Java 7.**
- **Държат се като нормални потоци**
  - Наследници на абстрактните InputStream и OutputStream, тоест може да очаквате консистентно API поведение
- **Надграждат потоците със свои абстракции:**

getNextEntry()	ZIS
createZipEntry(String name)	ZIS
putNextEntry(ZipEntry e)	ZOS
setLevel(int level)	ZOS
setMethod(int method)	ZOS
closeEntry()	both

Създаване на архив със ZipOutputStream

Пример:

```
byte[] buffer = new byte[1024];
try (FileInputStream in = new FileInputStream("fileToBeCompressed.log");
FileOutputStream fos = new FileOutputStream("New.zip");
ZipOutputStream zos = new ZipOutputStream(fos)) {

     ZipEntry ze = new ZipEntry("fileToBeCompressed.log");
     zos.putNextEntry(ze);
     int len;
     while ((len = in.read(buffer)) > 0) {
          zos.write(buffer, 0, len);
     }
     zos.closeEntry();

}
```

Разархивиране със ZipInputStream

Този пример ще разархивира цялата вътрешна структура на един Zip файл в указана изходна директория.

```
byte[] buffer = new byte[1024];
try (ZipInputStream zis = new ZipInputStream(new FileInputStream(zipFile))) {
    ZipEntry ze = zis.getNextEntry();

    while (ze != null) {
        String fileName = ze.getName();
        String outputPathStr = outputFolder + File.separator + fileName;
        //тук пресъздаваме вътрешната структура в изходната директория
        new File(outputPathStr).getParentFile().mkdirs();
        //тук пишем разкомпресирания файл
        try (FileOutputStream fos = new FileOutputStream(outputPathStr)) {
            int len;
            while ((len = zis.read(buffer)) > 0) {
                fos.write(buffer, 0, len);
            }
        }
        ze = zis.getNextEntry();
    }
    zis.closeEntry();
}
```

JarFile

- **JAR = Java Archive**
  - това е стандартния начин за пакетиране на Java приложения
  - JAR файловете могат да бъдат изпълними
    >java –jar myApp.jar
  - използва ZIP компресия (deflate)
- **Наследник на класа ZipFile**
- **Държи се по същия начин, добавяйки абстракция за manifest файл**
- **Manifest файла се използва за описание на конфигурация за съответното Java приложение**
  - От гледна точка на компресия, това е просто един компресиран файл в архива
  - Има уникален път: META-INF/MANIFEST.MF
    Не може да има два манифеста в един JarFile
  - Има си собствен клас, който се грижи за операциите свързани с манифеста
    java.util.jar.Manifest

JarInputStream и JarOutputStream

- **Наследници на съответните Zip Stream класове**
- **Добавят абстракция за Manifest файл**
  JIS : getManifest()
  JOS : JarOutputStream(OutputStream out, Manifest man)
- **Запазват същото очаквано за streams поведение, но**
- **Особености**
- JarOutputStream
  - Имената на директориите трябва да завършват на /
  - Всички пътища трябва да използват /, не \

Пример – създаване на JAR файл(1/3)

```
public void createJar() throws IOException {

    Manifest manifest = new Manifest();

    manifest.getMainAttributes()
       .put(Attributes.Name.MANIFEST_VERSION, "1.0");

    try (JarOutputStream target = new JarOutputStream(
                new FileOutputStream("output.jar"), manifest)) {

        add(new File("inputDirectory"), target);
    }

}
```

Пример – създаване на JAR файл(2/3)

```
private void add(File source, JarOutputStream target) throws IOException {

    try (BufferedInputStream in = new BufferedInputStream(
                                                new FileInputStream(source))) {
        if (source.isDirectory()) {
            String dirName = source.getPath().replace("\\", "/");
            if (!dirName.endsWith("/")) dirName += "/";

            JarEntry entry = new JarEntry(dirName);
            entry.setTime(source.lastModified());
            target.putNextEntry(entry);
            target.closeEntry();

            //продължаваме да добавяме файловете от входната директорията
            for (File nestedFile : source.listFiles()) {
                add(nestedFile, target);
            }
            return;
        }

...
```

Пример – създаване на JAR файл(3/3)

```
...
    //когато достигнем файл го записваме в архива
    JarEntry entry = new JarEntry(source.getPath().replace("\\", "/"));
    entry.setTime(source.lastModified());
    target.putNextEntry(entry);

    byte[] buffer = new byte[1024];
    while (true) {
        int count = in.read(buffer);
        if (count == -1) break;

        target.write(buffer, 0, count);
    }

    target.closeEntry();
}
}
```

GZip

По-слаба поддръжка, но достатъчна.

GZIPInputStream
GZIPOutputStream

Държат се подобно на Zip Streams, но не ги наследяват.

Модифицирани за работа с gzip формата и неговите специфики

Java 7

Досега се възползвахме от

- **try-with-resources**

Има и още, значително по-мощни благини…

ZipFileSystemProvider

ZipFileSystemProvider

Създаване

- **с URI:**
  ```
  URI uri = URI.create("jar:file:/codeSamples/zipfs/zipfstest.zip");
  FileSystem fs = FileSystems.newFileSystem(uri, env);
  ```

- **с Path:**
  ```
  Path zipfile = Paths.get("/codeSamples/zipfs/zipfstest.zip");
  FileSystem fs = FileSystems.newFileSystem(zipfile, env, null);
  ```

  опции като createIfNotExisting или encoding се подават през env, което е от тип java.util.Map

ZipFileSystemProvider

Веднъж като имаме ZIP файлова система можем да извършваме всички операции валидни за файловите системи:

- Копиране
- Местене
- Преименуване
- Модифициране на файлови атрибути

С разликата че всичко това работи с автоматичен преход между компресирано и не-компресирано състояние

ZipFileSystemProvider

Пример за създаване на Zip чрез копиране на файл вътре:

```
Map<String, String> env = new HashMap<>();
env.put("create", "true");

URI uri = URI.create("jar:file:/codeSamples/zipfs/zipfstest.zip");

try (FileSystem zipfs = FileSystems.newFileSystem(uri, env)) {

    Path externalTxtFile = Paths.get("/codeSamples/zipfs/SomeTextFile.txt");
    Path pathInZipfile = zipfs.getPath("/SomeTextFile.txt");

    // копираме файл в архива, компресирайки го автоматично
    Files.copy(externalTxtFile, pathInZipfile,
                                        StandardCopyOption.REPLACE_EXISTING);

}
```

ZipFileSystemProvider

По подобен начин може да се изчита и разархивира даден zip файл.

Още примери как се използва ZipFileSystemProvider:

http://fahdshariff.blogspot.de/2011/08/java-7-working-with-zip-files.html

Това е най-лесният начин за работа със zip файлове в Java 7.

Липси

RAR и TAR поддръжка

- За щастие има достатъчно open source проекти, които вършат достатъчно добра работа

Благодаря за вниманието!