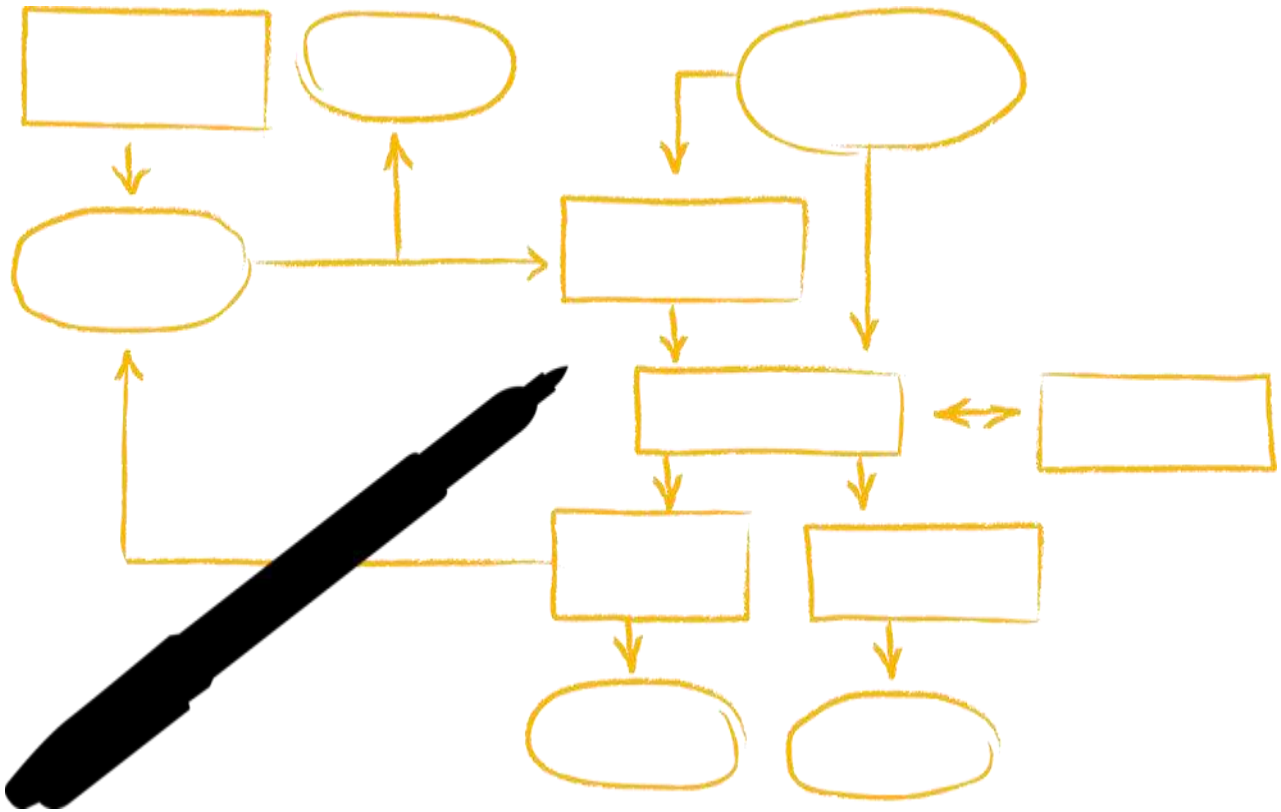
Файлов вход/изход в Java

Владимир Панов,
Кирил Гавраилов,
Иван Ст. Иванов

Public

Ноември, 2016



Съдържание

Операции с файловите метаданни

Операции с файл като едно цяло

Произволен (непоследователен) достъп до файловите данни

Архиви и компресирани файлове

Метаданни

Пътища

Файлова система
- Йерархична структура
- Състои се от директории и файлове
- Главна директория (корен)
- Текуща (работна) директория

Пътища
- Относителни
- Абсолютни

Символни връзки

java.nio.file.Path
Създаване

Класът **java.nio.file.Path** служи за представяне на път.

- Може да сочи към файл или директория.
- Може да сочи към съществуващ или несъществуващ обект.

Създаване – чрез **java.nio.file.Paths**:

```
Path absolutePath = Paths.get("/home", "joe", "file.txt");
Path absolutePath = Paths.get("c:\\Users\\joe", "file.txt");
Path relativePath = Paths.get("documents", "FileIO.odp");
Path relativePath = Paths.get("documents/FileIO.odp");
```

java.nio.file.Path
Извличане на информация

Метод	... връща под UNIX	... връща под Windows
`toString()`	/home/joe/foo	C:\home\joe\foo
`getFileName()`	foo	foo
`getName(0)`	home	home
`getNameCount()`	3	3
`subpath(0,2)`	home/joe	home\joe
`getParent()`	/home/joe	\home\joe
`getRoot`	/	C:\

java.nio.file.Path
Преобразувания

`Path toAbsolutePath()` - преобразува пътя до абсолютен (започващ от главната директория).
`Path toRealPath()` - преобразува пътя до реален:

- абсолютен
- с премахнати символни връзки (заменени от своите цели)
- с премахнати **.** и **..**

`Path resolve(Path anotherPath)` – добавя относителен път:

```
Path p1 = Paths.get("/home/joe/foo");
Path p2 = p1.resolve("bar"); // резултатът е /home/joe/foo/bar

Path p3 = Paths.get("whatever");
Path p4 = p3.resolve("/home/joe"); // резултатът е /home/joe
```

java.nio.file.Path
Сравняване

```
boolean equals(Path anotherPath)
boolean startsWith(Path anotherPath)
boolean endsWith(Path anotherPath)
```

java.nio.file.Files
Метаданни

```
long size(Path path)
boolean isRegularFile(Path path)
boolean isDirectory(Path path)
boolean isSymbolicLink(Path path)
boolean isHidden(Path path)
FileTime getLastModifiedTime(Path path)
FileTime setLastModifiedTime(Path path, FileTime time)
UserPrincipal getOwner(Path path)
Object getAttribute(Path path, String attribute)
Path setAttribute(Path path, String attribute, Object value)
```

java.nio.file.Files
Други проверки

```
boolean exists(Path path)
boolean notExists(Path path)
boolean isReadable(Path path)
boolean isWritable(Path path)
boolean isExecutable(Path path)
boolean isSameFile(Path path1, Path path2)
```

java.io.File

Класът **java.io.File**

- Дескриптор за файлове преди Java 7
- Присъства все още в много API-та
- Притежава същите свойства като **java.nio.file.Path**

Създаване:

```
File absoluteFile = new File("c:\\Users\\joe\\file.txt");

File absoluteFile = new File("c:\\Users\\joe", "file.txt");

File parentFile = new File("c:\\Users\\joe");
File absoluteFile = new File(parentFile, "file.txt");

URI fileUri = URI.create("file:///home/joe/file.txt");
File absoluteFile = new File(fileUri);
```

Конвертиране от и към към **java.nio.file.Path**:

```
Path absolutePath = absoluteFile.toPath();
File absoluteFile = absolutePath.toFile();
```

java.io.File
Операции

```
boolean exists()
boolean isFile()
boolean isDirectory()

boolean canRead()    / boolean setReadable(boolean)
boolean canWrite()   / boolean setWritable(boolean)
boolean canExecute() / boolean setExecutable(boolean)

String getAbsolutePath()
String getCannonicalPath()
File getParentFile()

File[] listFiles()
File[] listFiles(FilenameFilter)

boolean mkdirs()
boolean delete()
```

Файлове като едно цяло

java.nio.file.Files
Изтриване

`void delete(Path path)`

- ако **`path`** не съществува, хвърля **`NoSuchFileException`**
- ако **`path`** е непразна директория, хвърля **`DirectoryNotEmptyException`**

`boolean deleteIfExists(Path path)`

- ако **`path`** не съществува, просто връща **`false`**
- ако **`path`** е непразна директория, хвърля **`DirectoryNotEmptyException`**

java.nio.file.Files
Копиране

`Path copy(Path source, Path target, CopyOption... options)`

- съдържанието на директория не се копира
- **`options`** може да бъде една или повече от:
  - **`StandardCopyOption.COPY_ATTRIBUTES:`** копират се и атрибутите на файла/директорията, напр. време на последна промяна
  - **`StandardCopyOption.REPLACE_EXISTING:`** ако **`target`** е съществуващ файл, то той се надписва вместо да се хвърля **`FileAlreadyExistsException`**

java.nio.file.Files
Преместване (преименуване)

`Path move(Path source, Path target, CopyOption... options)`

- директория може да се премести само на същата файлова система
- **`options`** може да бъде една или повече от:
  - **`StandardCopyOption.ATOMIC_MOVE:`** преместването е атомарно; ако това не е възможно, се хвърля **`AtomicMoveNotSupportedException`**
  - **`StandardCopyOption.REPLACE_EXISTING:`** ако **`target`** е съществуващ файл, то той се надписва вместо да се хвърля изключение

java.nio.file.Files
Създаване на директория

`Path createDirectory(Path dir)` – създава директория с път **`dir`**, ако родителската директория на този път вече съществува

`Path createDirectories(Path dir)` - създава директория с път **`dir`**, създавайки и родителските директории, ако има нужда

java.nio.file.Files
Обхождане на директория

`DirectoryStream<Path> newDirectoryStream(Path dir)` – създава поток, от който могат да бъдат прочетени файловете и поддиректориите на директорията с път **`dir`** като инстанции на **`Path`**

- **`DirectoryStream`** прилича повече на колекция (защото е наследник на **`Iterable`**), отколкото на входно-изходен поток.

```
Path dir = ...;
try (DirectoryStream<Path> stream =
        Files.newDirectoryStream(dir)) {

    for (Path fileOrSubDir: stream) {
        // употреба на fileOrSubDir
    }
} catch (IOException | DirectoryIteratorException x) {
    // обработка на изключенията
}
```

java.nio.file.Files

Обхождане на директория с глоб (жокери, wild cards)

`DirectoryStream<Path> newDirectoryStream(Path dir, String glob)` – създава поток, който съдържа само файловете и поддиректориите, които отговарят на глоба **`glob`**

java.nio.file.Files
Глоб

`*` – замества произволен брой символи (вкл. николко) в рамките на едно име от пътя (т.е. не може да замести пътния разделител - / или \)

`**` – като *, но пресича границите на имената в пътя

`?` – замества точно един символ

`{subGlob1, ..., subGlobN}` – замества някой от подглобовете

`[c1...cN]` – замества някой от символите

`[c1-c2]` – замества някой от символите в диапазона от c1 до c2

`\` - използва се за escape на специалните символи, вкл. \

всеки друг символ - замества себе си

java.nio.file.Files
Примерни глобове

`/home/*/file` - замества **`/home/joe/file`** (и други), но не и **`/home/joe/work/file`**

`/home/**/file` - замества **`/home/joe/work/file`** (и други)

`{temp*, tmp*}` - замества всички имена, започващи с **`temp`** или **`tmp`**

`[fmi]` – замества една от буквите **`f`**, **`m`**, и **`i`**

`[a-z]` – замества една от малките латински букви

`[fF]` – замества малко или голямо **`F`**

`\\` - замества обратно наклонена черта

`alabala` – замества **`alabala`**

`[[]` - замества **`[`**; в **`[]`** всички специални символи с изключение на **`]`**, **`-`** и **`\`** може да не се escape-ват

java.nio.file.Files
Временнни файлове и директории

`Path createTempFile(Path dir, String prefix, String suffix)` – създава нов файл в директорията **`dir`** и с име, започващо с **`prefix`** и завършващо със **`suffix`**

`Path createTempFile(String prefix, String suffix)` – създава нов файл в системната временна директория и с име, започващо с **`prefix`** и завършващо със **`suffix`**

`Path createTempDirectory(Path dir, String prefix)` – създава нова поддиректория в директорията **`dir`** и с име, започващо с **`prefix`**

`Path createTempDirectory(String prefix)` – създава нова поддиректория в системната временна директория и с име, започващо с **`prefix`**

java.nio.file.Files
Обработване на съдържанието на файл наведнъж

Следните методи са подходящи за малки файлове:

```
byte[] readAllBytes(Path file)
List<String> readAllLines(Path file, Charset cs)
Path write(Path file, byte[] bytes, OpenOption... options)
Path write(Path path, Iterable<? extends CharSequence> lines,
Charset cs, OpenOption... options)
```

За големи файлове се използват потоци (за байтове) и четци/записвачи (за символи).

Непоследователен достъп

java.nio.file.Files

Файлове с произволен достъп - отваряне

Позволяват непоследователен (произволен) достъп до файловото съдържание (за разлика от потоците).

`SeekableByteChannel newByteChannel(Path file, OpenOption... options)`

- **`options`** може да бъде една или повече от:
  - **`StandardOpenOption.WRITE`**
  - **`StandardOpenOption.APPEND`**
  - **`StandardOpenOption.TRUNCATE_EXISTING`**
  - **`StandardOpenOption.CREATE_NEW`**
  - **`StandardOpenOption.CREATE`**
  - **`StandardOpenOption.DELETE_ON_CLOSE`**
  - **`StandardOpenOption.SYNC`**

java.nio.channels.SeekableByteChannel

Файлове с произволен достъп

`long position()` - връща текущата позиция на канала

`SeekableByteChannel position(long newPosition)` - променя текущата позиция на канала на **`newPosition`**

`int read(ByteBuffer dst)` - прочита данни от текущата позиция на канала в **`dst`**

`int write(ByteBuffer src)` - записва данни от **`src`** на текущата позиция в канал

`SeekableByteChannel truncate(long size)` - отрязва канала до размер **`size`**

Архиви и компресирани файлове

Архиви и компресирани файлове

Компресиран файл: файл, чийто размер е намален чрез компресиращ алгоритъм (напр. код на Хафмън (Huffman))

- Последователно четене: бавно
- Непоследователно четене: много бавно
- Запис: изключително бавно

Използват се за файлове, които само се четат, и то рядко.

Архив: файл, който съдържа в себе си директории и файлове.

- Удобни за създаване на резервни копия
- Бърз трансфер към и от латентни (отдалечени) файлови системи
- (защото се прехвърлят метаданните само на един файл)

Компресиран архив

java.util.zip.ZipFile
Четене на компресиран ZIP архив

`ZipFile(String name)` – отваря ZIP архив по име на файл (**`java.nio.file.Path.toString()`**)

`Enumeration<? extends ZipEntry> entries()` - връща списък с метаданните на всички членове (директории и файлове) на архива

`int size()` - връща броя на членовете на архива

`ZipEntry getEntry(String name)` – връща метаданните за член по име

`InputStream getInputStream(ZipEntry entry)` – връща входен поток, от който могат да се прочетат данните на даден файл в декомпресиран вид

java.util.zip.ZipEntry

Метаданни за член (директория или файл) на ZIP архив

`String getName()`
`boolean isDirectory()`
`long getTime()`
`long getSize()`
`long getCompressedSize()`
`void setMethod(int method)` – задава метод на компресия при създаване на член: *STORED* (без компресия) или *DEFLATED*
`int getMethod()`

`size / compressedSize` се нарича коефициент на компресия.

java.util.zip.ZipFile
Пример за четене на компресиран ZIP архив

```
try (ZipFile zipFile = new ZipFile("alabala.zip")) {
    Enumeration<? extends ZipEntry> entriesEnum =
                        zipFile.entries();
    while (entriesEnum.hasMoreElements()) {
        ZipEntry entry = entriesEnum.nextElement();
        System.out.println(entry.getName());
        if (!entry.isDirectory()) {
            InputStream entryInputStream =

      zipFile.getInputStream(entry);
            processEntryData(entry, entryInputStream);
        }
    }
}
```

java.util.zip.ZipInputStream
Последователно изчитане на на ZIP архив

`ZipInputStream(InputStream in)` – **`in`** е поток, представляващ компресиран ZIP архив

`ZipEntry getNextEntry()` - връща метаданните за следващия член на архива и позиционира потока в началото на декомпресираните му данни

java.util.zip.ZipInputStream
Пример за четене на компресиран ZIP архив

```
try (ZipInputStream zis =
       new ZipInputStream(new FileInputStream("alabala.zip")))
{
    ZipEntry entry;
    while ((entry = zis.getNextEntry()) != null) {
        System.out.println(entry.getName());
        if (!entry.isDirectory()) {
            processEntryData(entry, zis);
        }
    }
}
```

java.util.zip.ZipOutputStream
Създаване на на ZIP архив

`ZipOutputStream(OutputStream out)` – **`out`** е поток, в който ще бъде записан компресираният ZIP архив

`void putNextEntry(ZipEntry e)` – добавя нов член в потока на архива; последващ запис в потока ще записва в този член

`void closeEntry()` - затваря текущо записвания член

java.util.zip.ZipOutputStream
Пример за създаване на частично компресиран ZIP архив

```
try (ZipOutputStream zos =
new ZipOutputStream(new FileOutputStream("alabala.zip"))) {
    ZipEntry entry1 = new ZipEntry("file1.txt");
    entry1.setMethod(ZipEntry.STORED);  // без компресия
    zos.putNextEntry(entry1);
    writeFirstFileData(zos);
    zos.closeEntry();

    ZipEntry entry2 = new ZipEntry("file2.txt");
    entry2.setMethod(ZipEntry.DEFLATED);  // с компресия
    zos.putNextEntry(entry2);
    writeSecondFileData(zos);
    zos.closeEntry();
}
```

JAR файлове

JAR е файлов формат за разпространение на артефакти на Java програми.
JAR всъщност е ZIP архив с допълнителен дескриптор:
`META-INF/MANIFEST.MF`
В JAR файлове най-често се пакетират клас-файлове и малки прилежащи им ресурси.

- Вместо да се разпространяват много файлове, пръснати в цяло директорно дърво, се разпространява само един файл.
- Веднъж създаден, един JAR файл повече не се променя.
- Всеки един от файловете в архива се прочита най-много веднъж (при първа употреба след стартиране на JVM).

Работа с файлови системи

java.nio.file.spi.FileSystemProvider
Доставчици на файлови системи

Доставчик (provider) на файлова система – библиотека, даваща възможност за достъп до файловете и директориите на конкретна файлова система по унифициран начин.

Подразбиращият се доставчик дава достъп до файловите системи, предоставени от ядрото на ОС.

В стандартната библиотека има доставчици за:

- Файлова система в паметта (съдържанието се загубва при прекратяване на JVM)
- Файлова система в ZIP архив

Всеки може да напише доставчик за своята файлова система. Тогава тя ще може да се използва от съществуващи програми без те да трябва да се модифицират.

java.nio.file.FileSystems
Инстанциране на файлова система

`FileSystem newFileSystem(URI uri, Map<String,?> env)` – инстанциира нова файлова система; ако доставчикът не може да бъде намерен, хвърля **`ProviderNotFoundException`**

`uri` – указва кой доставчик трябва да се ползва, както и къде се намира самата файлова система

Пр. за uri:

```
"jar:file:/data/archive.zip"
"memory:///?name=logfs"
```

`env` – опции (напр. за създване на файловата система, ако тя не съществува)

Пр. за опции:

```
Map<String, String> env = new HashMap<>();
env.put("create", "true");            // за jar:
env.put("capacity", "16G");    // за memory:
```

Доставчик на файлова система в ZIP архив
Пример

```
Map<String, String> env = new HashMap<>();
env.put("create", "true");
URI uri = URI.create("jar:file:/tmp/alabala.zip");

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

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

    Files.copy(externalTxtFile, pathInZipfile,

       StandardCopyOption.REPLACE_EXISTING);
}
```

Допълнителни материали

Java tutorial за файлов вход/изход:

https://docs.oracle.com/javase/tutorial/essential/io/fileio.html

… или просто потърсете следните ключови думи:

Java tutorial file io

Благодарим за вниманието!