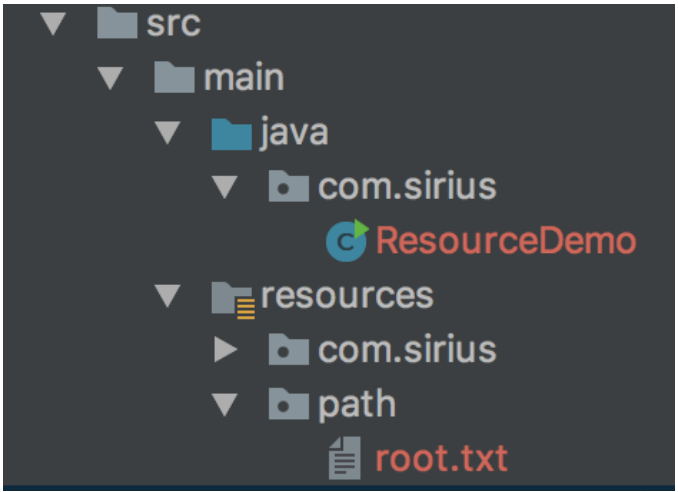
Работа с файлами

NIO. Получение ресурсов

Валерий Алексеевич Овчинников
valery.ovchinnikov@phystech.edu

Доступ к ресурсам

Доступ к ресурсам: Что такое?



Revolut МФТИ

Доступ к ресурсам: Где применяется

- В JEE контейнерах
- В многочисленных фреймворках типа Spring
- В собственном коде для конфигурации системы

Доступ к ресурсам: Расположение

Java ищет ресурсы в окружении исполняемого кода:

- Внутри всех jar-файлов в classpath
- Рядом с *.class файлами в classpath

Доступ к ресурсам: Пути к ресурсам

Абсолютный (относительно classpath):

```
"/path/to/resource"
```

Относительный (завсист от класса, из которого ищут):

```
"path/to/resource"
```

Доступ к ресурсам: Примеры. Class

1 и 3 эквивалентны

2 и 4 эквивалентны

```
// resolved into $classpath/path/to/resource
URL r = com.sirius.Resource.class.getResource("/path/to/resource");
// resolved into $classpath/com/sirius/path/to/resource
URL r = com.sirius.Resource.class.getResource("path/to/resource");

// resolved into $classpath/path/to/resource
URL r = this.getClass().getResource("/path/to/resource");
// resolved into $classpath/com/sirius/path/to/resource
URL r = this.getClass().getResource("path/to/resource");
```

Доступ к ресурсам: Примеры. ClassLoader

Только относительные пути!
Всегда относительно classpath

```
// resolved into $classpath/path/to/resource
URL r = Thread.currentThread().getContextClassLoader()
              .getResource("path/to/resource");
```

Доступ к ресурсам: Примеры. InputStream

```
InputStream is = Resource.class
                   .getResourceAsStream("/path/to/resource");
InputStream is = Resource.class
                   .getResourceAsStream("path/to/resource");
InputStream is = Thread.currentThread().getContextClassLoader()
                   .getResourceAsStream("path/to/resource");
```

File NIO

File NIO: New IO

Было: *Input/OutputStream*, *ByteStream*, *CharacterStream*
Стало: *Channels*, *Buffers*

Информация копируются из каналов в буферы при чтении и из буферов в каналы при записи

File NIO: New IO

Появились неблокирующие (non-blocking) вызовы
Появились Selectors

Теперь можно обрабатывать одновременно несколько каналов с данными в одном потоке (мультиплексирование)

File NIO: Основные типы каналов

- FileChannel
- DatagramChannel
- SocketChannel
- ServerSocketChannel

File NIO: Основные буферов

- ByteBuffer
- CharBuffer
- DoubleBuffer
- FloatBuffer
- IntBuffer
- LongBuffer
- ShortBuffer

File NIO: FileChannel

- FileStream.getChannel()
- RandomAccessFile.getChannel()
- FileChannel.open(path)

File NIO: FileChannel

- read(buffer)
- write(buffer)

File NIO: FileChannel

- read(buffer)
- write(buffer)
- size()
- position([position])
- truncate(size)

File NIO: FileChannel

- read(buffer)
- write(buffer)
- size()
- position([position])
- truncate(size)
- close()
- force(boolean flushMetaData)

File NIO: Пример. Чтение

```
ByteBuffer buf = ByteBuffer.allocate(48);
int bytesRead = inChannel.read(buf);
while (bytesRead != -1) {
    // NOTICE!!
    buf.flip();

    while (buf.hasRemaining()) {
        System.out.print((char) buf.get());
    }

    buf.clear(); // or buf.compact()
    bytesRead = inChannel.read(buf);
}
```

File NIO: Пример. Запись

```
ByteBuffer buf = ByteBuffer.allocate(bytes.length);
buf.put(bytes);

buf.flip();

while (buf.hasRemaining()) {
    outChannel.write(buf);
}
buf.clear();
```

File NIO: Пример. Позиционирование

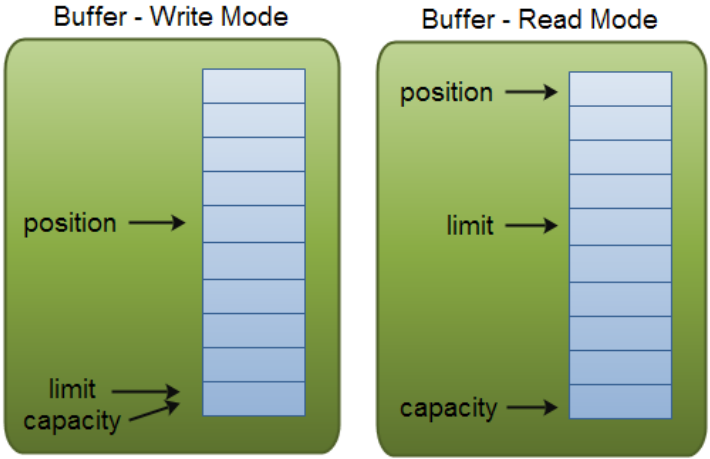
```
// write
outChannel.position(writeOffset);
while (buf.hasRemaining()) {
    outChannel.write(buf);
}

//read
inChannel.position(readOffest);
int bytesRead = inChannel.read(buf);
```

File NIO: Ужасы ByteBuffer

У ByteBuffer есть 3 параметра:

- Capacity (постоянное значение, общий размер буфера)
- Position (зависит от режима: конец записи/начало чтения)
- Limit (зависит от режима: capacity/конец чтения)

File NIO: Ужасы ByteBuffer



- Capacity
- Position
- Limit

File NIO: Ужасы ByteBuffer

rewind – вернет position в значение 0, можно будет снова читать сначала

```
buf.get(); // a
buf.get(); // b
buf.rewind();
buf.get(); // a
```

File NIO: Ужасы ByteBuffer

mark – запомнит текущую position reset – выставит помеченый position

```
buf.mark(); // position = 3
buf.get(); // position = 4
buf.get(); // position = 5
buf.reset(); // position = 3
```

File NIO: Ужасы ByteBuffer

clear – переводит в режим записи в буфер, position = 0, limit = capacity compact – переводит в режим записи в буфер, непрочитанное копирует в начало, position = n, limit = capacity

```
buf.get();
buf.compact();
```

File NIO: Path

Замена класса java.io.File

- get
- normalize
- toAbsolutePath

```
Path path = Paths.get("c:\\path\\to\\myfile.txt");
Path path = Paths.get("/path/to/myfile.txt");
Path path = Paths.get("/path", "to", "myfile.txt");
// relative to dir from which code is executed
Path path = Paths.get("relative/path/to/myfile.txt");
```

File NIO: Files

- exists
- createDirectory
- copy
- move
- delete
- walkFileTree

File NIO: Files

```
Path from;
Path to;
if (Files.exists(from)) {
    Files.delete(from);
} else {
    Files.createDirectory(newDirectory);
}
Files.copy(from, to);
Files.move(from, to, StandardCopyOption.REPLACE_EXISTING);
```

File NIO: Channel-to-Channel File Copy

```
from.transferTo(0, from.size(), to);
to.transferFrom(from, 0, from.size());
```

File NIO: AsynchronousFileChannel

```
Future<Integer> future = fileChannel.read(buf, position);
fileChannel.read(buffer, position, attachement,
        new CompletionHandler<Integer, ByteBuffer>() {
            @Override
            public void completed(Integer result,
                 ByteBuffer attachment) {}

            @Override
            public void failed(Throwable exc,
                 ByteBuffer attachment) {}
        });
```

Revolut МФТИ