Reading and Writing Text Files Using the Java NIO API

José Paumard
PHD, JAVA CHAMPION, JAVA ROCK STAR

@JosePaumard https://github.com/JosePaumard

Agenda

- Why are text and binary files separated?
- How to read the content of a text file
- How to write text to a file
- What is a CharSet and why is it needed

Separating Text Files and Binary Files

Binary files are about storing bytes

Your application makes sense of them!

Text files are about storing characters

And there are many ways to encode them!

ASCII, ISO 8859, UTF 8, Unicode, ...

Java Separates Text and Binary Files

Text Files

Reader, Writer

Binary Files

InputStream, OutputStream

Java Separates Text and Binary Files

Text Files

Reader, Writer

Binary Files

InputStream, OutputStream

Reading from a Text File

Getting a Reader from a Path

One simple pattern:

- get a BufferedReader
- use it to read your file line by line

```
Path path =
    Paths.get("c:/tmp/sonnet.txt");

BufferedReader reader =
    Files.newBufferedReader(path);

String line = reader.readLine();
```

1) create a path

2) get a BufferedReader on the corresponding file

3) read a line (returns null if there is none)

Dealing with Errors!

What can possibly go wrong?

- the file may not be there

- the file may not have the right encoding!

It leads to subtle errors...

```java
Path path =
    Paths.get("c:/tmp/sonnet.txt");

BufferedReader reader =
    Files.newBufferedReader(path);

String line = reader.readLine();
```

ANSI

```
From fairest creatures we desire increase,
That thereby beauty's rose might never die,
But as the riper should by time decease,
His tender heir might bear his memory:
But thou contracted to thine own bright eyes,
Feed'st thy light's flame with self-substantial fuel,
Making a famine where abundance lies,
Thy self thy foe, to thy sweet self too cruel:
Thou that art now the world's fresh ornament,
And only herald to the gaudy spring,
Within thine own bud buriest thy content,
And, tender churl, mak'st waste in niggarding:
Pity the world, or else this glutton be,
To eat the world's due, by the grave and thee.
```

It’s all good, reading the file will not give any error

Despite the fact that the reader reads UTF-8 by default

```
Path path =
    Paths.get("c:/tmp/sonnet.txt");

BufferedReader reader =
    Files.newBufferedReader(path);

String line = reader.readLine();
```

José Paumard

ANSI

Then the following Exception is raised:

```
Exception in thread "main" java.nio.charset.MalformedInputException: Input length = 1
    at java.base/java.nio.charset.CoderResult.throwException(CoderResult.java:274)
```

```
Path path =
    Paths.get("c:/tmp/sonnet.txt");

BufferedReader reader =
    Files.newBufferedReader(path, StandardCharsets.ISO_8859_1);

String line = reader.readLine();
```

Passing the CharSet fixes the exception

The StandardCharSet class has a set of predefined charsets

Writing to a Text File

Getting a Writer from a Path

One simple pattern:

- get a BufferedWriter
- use it to write to your file

```
Path path =
    Paths.get("c:/tmp/sonnet.txt");

BufferedWriter writer =
    Files.newBufferedWriter(path);

writer.write("Hello world!");
```

1) create a path

2) get a BufferedWriter on the corresponding file

3) write a String

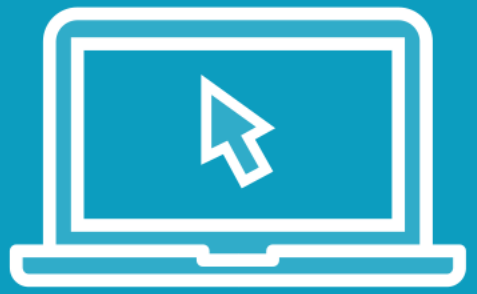
Dealing with Errors!

What can possibly go wrong?

- the file may already be there

- flush the buffer before closing it!

Demo



Let us write some code!

You will see simple examples

How to read and write to a text file

How to use the try with resources pattern

With errors!

Module Wrap Up

What did you learn?

Readers and Writers!

How to deal with charsets

How to deal with the closing of buffers

Dealing with real text files? CSV!