Working with Files in Java Using the Java NIO API

ACCESSING FILES AND DIRECTORIES USING JAVA NIO PATH

José Paumard
PHD, JAVA CHAMPION, JAVA ROCK STAR

@JosePaumard https://github.com/JosePaumard

How to work with text files

- accessing files and directories
- reading and writing text files
- processing a CSV file
- copying, deleting, moving files

Based on the Java NIO API

This is a Java 7+ course

- basic knowledge of Java
- how to create and run a simple program
- basic knowledge of what is a file system

Agenda

First, let us see the Path interface

A Path is not a File!

How to create a Path

What can be done with a Path

Differentiating Files and Paths

File is a class from Java 1 that models files

Path is an interface from Java 7

A File is independent from the file system

It is created on a String

Where a Path is attached to a file system

It is created from a FileSystem

Accessing Files with Path Objects

Path is an interface from Java 7

It is used to access a file or a directory

A Path gives information on a path:

- its elements

- if is a symbolic link or not

The factory methods from Files give you more informations

Creating a Path

There are two patterns to create a Path

- from the Paths factory class

And, starting with Java 11:

- from the Path.of() factory methods

```java
Path path1 = Paths.get("c:/tmp/debug.log");
Path path2 = Paths.get("c:", "tmp", "debug.log");

URI uri = URI.create("file://c:/tmp/debug.log");
Path path3 = Paths.get(uri);

Path path3 = Path.of("c:/tmp/debug.log");
```

From the Paths factory class, two get() methods

- that takes a path as a String, or a vararg of paths

- or a URI as a String

Starting with Java 11: Path.of is available

Getting Information on Files from a Path

The Files factory class has the methods to check :

- if it exists or not
- if it is hidden
- if the path is a file or a directory
- if it is readable or writeable
- if it is executable

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

boolean exists = Files.exists(path);

boolean exists = Files.exists(path, LinkOption.NOFOLLOW_LINKS);


boolean sameFile = Files.isSameFile(path1, path2);
```

These methods may take a further argument: NOFOLLOW_LINKS

Meaning that the API can check if a path contains symbolic links

You can check if two path actually locate the same file

Module Wrap Up

What did you learn?

What a Path is

How to create it

And how to check for files and directories on your file system

Now let us read and write text files!