Manipulating Files with File and Path

José Paumard
PHD, Java Champion, JavaOne RockStar

@JosePaumard https://github.com/JosePaumard

Agenda

Before reading and writing data

You need to know how to deal with files

There two APIs

- File: a class from Java I/O
- Path: an interface from Java NIO 2

Accessing Files with the File Class

Demo

Discovering File Objects in the IDE

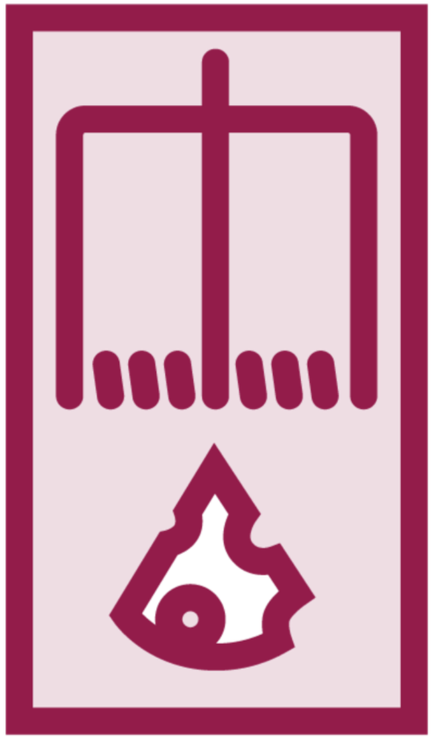
A file object is independent
of the file system
It can be used to access it

Several key points for a File:

- create and delete files
- create directories
- analyze the path
- get the real path to a file or directory

With Files, you can:

- copy and move files



One trap to avoid:

- creating a File object does not create a file

Accessing Files with the Path Interface

Why do you need a second model for a path?

- the File class is here to stay!

- a File is independent of any file system

- it is not possible to access features specific of Windows, Mac or Linux

The solution is to create an interface

Along with specific implementations

To access specific features
of each type of file system

This is why the Path interface was created

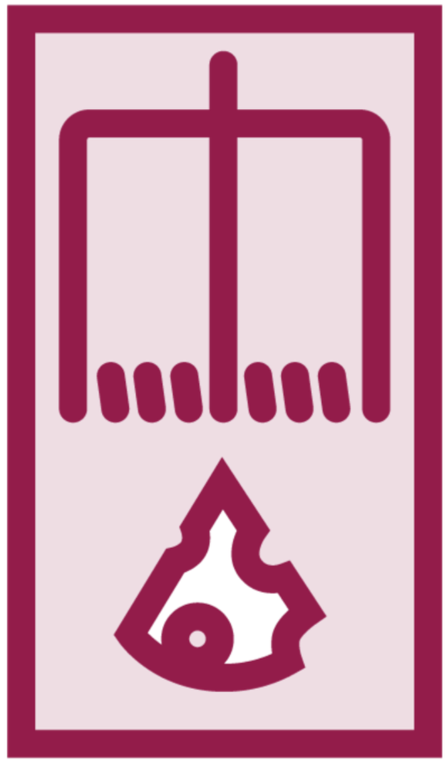
Demo

Discovering the Path interface in action

Several key points for a Path:

- parent: everything before the last /
- root: may not exist, file system dependent
- absolute path: identified by the file system

A file with a root may not be absolute



Several traps to avoid:

- a File is a class, a Path is an interface
- a File is not linked to a File System
- a Path is created from a File System
- a Path with a root may not be absolute

More key points:

- a Path object is linked to a file system
- resolve()
- resolveSibling()
- relativize()
- normalize()

resolve(other)

If other is:

- empty, then it returns this
- absolute, it returns other
- a root path then the result is undefined

Otherwise, if concatenates other with this

resolveSibling(other)

If other is:

- absolute, it returns other

- empty, it returns the parent of this

If this does not have a parent

- empty

Otherwise, joins the parent of this with other

relativize()

If both this and other have a root

- build a relative path from this to other

If none have a root

- build a relative path from this to other

If this or other does not have a parent

- throws an exception

Symbolic links? Implementation dependent

normalize()

Implementation dependent!

Removes redundancies (. and ..)

toRealPath()

Implementation dependent!

Creates an absolute path with no links

Module Wrap Up

What did you learn?

Files and Paths!

Up Next: Reading and Writing Characters