

LES FICHIERS

Path et File

- Les instances de la classe **Path** représentent un chemin vers :
 - un fichier
 - un répertoire
 - (+ lien symbolique)
 - Manipulation facilitée avec la classe « helper » **Paths**
ex. `Path projectDir = Paths.get("");`
- Les instances de la classe **File** représentent un fichier ou dossier
 - Manipulation facilitée avec la classe « helper » **Files**
ex. `Files.delete(Paths.get("test.txt"));`

Listing de répertoire V1

```
// Recupération du repertoire de travail
Path workingDirPath = Paths.get("").toAbsolutePath() ;

File workingDir = workingDirPath.toFile() ;

if(workingDir.isDirectory()){

    // listing version collection
    System.out.println("Fichiers visibles de " + workingDirPath.normalize());

    List<Path> content = Files.list(workingDirPath).collect(Collectors.toList());
    for(Path p: content){
        if(!p.toFile().isHidden()) System.out.println(p.getFileName()) ;
    }
}
```

```
Fichiers visibles de /Users/Greg/Documents/Enseignements/Cours/Java/code/cours
cours.iml
out
src
test.txt
```

Listing de répertoire V2

```
// Recupération du repertoire de travail
Path workingDirPath = Paths.get("").toAbsolutePath() ;

File workingDir = workingDirPath.toFile() ;

if(workingDir.isDirectory()) {

    // listing version lambda
    System.out.println("Fichiers visibles de " + workingDirPath.normalize());

    Files.list(workingDirPath).forEach(p -> {
        if (!p.toFile().isHidden()) System.out.println(p.getFileName());
    });
}
```

```
Fichiers visibles de /Users/Greg/Documents/Enseignements/Cours/Java/code/cours
cours.iml
out
src
test.txt
```

Listing de répertoire V3

```
// Recupération du repertoire de travail
Path workingDirPath = Paths.get("").toAbsolutePath().normalize() ;

File workingDir = workingDirPath.toFile() ;

if(workingDir.isDirectory()) {

    // listing version lambda avec "method reference"
    System.out.println("Fichiers visibles de " + workingDirPath.normalize());

    Files.list(workingDirPath)
        .filter(p -> !p.toFile().isHidden()) // filtrage
        .map(Path::getFileName)             // mapping, equivalent à .map(p -> p.getFileName())
        .forEach(System.out::println);      // equivalent à .forEach(p -> System.out.println(p))
}
```

```
Fichiers visibles de /Users/Greg/Documents/Enseignements/Cours/Java/code/cours
cours.iml
out
src
test.txt
```

Accès aux informations

```
Path path = Paths.get(TestsFiles.class.getResource("lets_bang.txt").toURI()) ;
System.out.println("Informations sur " + path);

File file = path.toFile();
System.out.println("Nom court : " + file.getName());
System.out.println("Parent : " + file.getParent());
System.out.println("Caché : " + file.isHidden());
System.out.println("Répertoire : " + file.isDirectory());
System.out.println("Fichier : " + file.isFile());
System.out.println("Taille : " + file.length() + " octets");
System.out.println("Droit en lecture : " + file.canRead());
System.out.println("Droit en écriture : " + file.canWrite());
System.out.println("Droit en execution : " + file.canExecute());

System.out.println("\n!!! Changement de droits en exécution !!!\n");
file.setExecutable(true) ;
System.out.println("Droit en execution : " + file.canExecute());
```

Informations sur /Users/Greg/Documents/Enseignements/Cours/Java/code/cours/out/production/cours/files/lets_bang.txt

Nom court : lets_bang.txt

Parent : /Users/Greg/Documents/Enseignements/Cours/Java/code/cours/out/production/cours/files

Caché : false

Répertoire : false

Fichier : true

Taille : 1148 octets

Droit en lecture : true

Droit en écriture : true

Droit en execution : true

!!! Changement de droits en exécution !!!

Droit en execution : true

Lecture version Java 7

```
Path path = Paths.get(TestsFiles.class.getResource("lets_bang.txt").toURI()) ;

System.out.println("Contenu de " + path.getFileName());

StringBuilder content = new StringBuilder() ;

// java 7
List<String> lignes = Files.readAllLines(path, StandardCharsets.UTF_8);
for(String line: lignes){
    content.append(line) ;
    content.append("\n") ;
}

System.out.println(content);
```

Contenu de lets_bang.txt

Let's Bang (Shaka Ponk) :

Saturday night in a dancing house I say (yeah yeah yeah yeah)

Storming boys staring at my eyes I say (yeah yeah yeah yeah)

You're the one I like

I wanna share my story tonight

Big mama you're my kind

Gimme a part of your precious time

Lecture version Java 8

```
Path path = Paths.get(TestsFiles.class.getResource("lets_bang.txt").toURI()) ;

System.out.println("Contenu de " + path.getFileName());

StringBuilder content = new StringBuilder() ;

// java 8 avec lambda
Stream<String> lines = Files.lines(path, StandardCharsets.UTF_8);

lines.forEach(line -> {
    content.append(line) ;
    content.append("\n") ;
});

System.out.println(content);
```

Contenu de lets_bang.txt
Let's Bang (Shaka Ponk) :

Saturday night in a dancing house I say (yeah yeah yeah yeah)
Storming boys staring at my eyes I say (yeah yeah yeah yeah)
You're the one I like
I wanna share my story tonight
Big mama you're my kind
Gimme a part of your precious time

Ecriture dans un fichier

```
Path projectDir = Paths.get(""); ;
Path p = Paths.get(projectDir.toAbsolutePath().normalize().toString() + "/test.txt") ;

File f = p.toFile() ;
if(!f.exists()){
    System.out.println("Le fichier n'existe pas");
    // f = Files.createFile(p).toFile() ;
    // inutile ici car on a mis l'option StandardOpenOption.CREATE plus bas...
else System.out.println("Le fichier existe");

OutputStream out = Files.newOutputStream(p, StandardOpenOption.CREATE, StandardOpenOption.APPEND) ;

String msg = String.format("Nouvelle ligne ajoutée à %1$tT\n", Calendar.getInstance()) ;

out.write(msg.getBytes());

out.close();
```

Contenu de test.txt après plusieurs exécutions :

```
Nouvelle ligne ajoutée à 14:38:15
Nouvelle ligne ajoutée à 14:43:12
Nouvelle ligne ajoutée à 15:26:35
```