TRATAMIENTO DE FICHEROS

JAVA 8

Profesor: Alonso Rebolo

Curso 2019 - 2020

Stream y Buffer

Cuando los ficheros son leídos, o se desea escribir en ellos, la información pasa a ser un flujo de datos.
La interfaz que regula ese flujo es:

Interface Stream<T>

<T> hace referencia al tipo de datos, normalmente será String

Esta interfaz está en: java.util.stream

Un Buffer es un contenedor intermedio entre los periféricos de I/O y el fichero donde se acumula información hasta llenarse o hasta que se fuerze a vaciarse de información que será transferida al fichero si estamos escribiendo o hacia un periférico de salida si estamos leyendo.

Las clases BufferedReader y BufferedWriter controlan la creación de buffers para leer y escribir, respectivamente. Están en el paquete: java.io

FileSystems

La clase FileSystems obtiene el sistema de ficheros que se está utilizando por nuestro sistema operativo y que es accesible por la Máquina Virtual de Java (JVM).

Esta clase está en java.nio.file

El método más utilizado es getDefault(). Este método es estático y devuelve un FileSystem que hace la veces de interface con el sistema de ficheros y que también está en el mismo paquete que FileSystems.

```
FileSystem ficheros = FileSystems.getDefault();
```

Path

Con la interface *Path* se pueden hacer operaciones sobre rutas como obtener la ruta absoluta de un *Path* relativo o el *Path* relativo de una ruta absoluta, de cuantos elementos se compone la ruta, obtener el *Path* padre o una parte de una ruta.

La interface *Path* se encuentra en java.nio.file

Path camino = ficheros.getPath(“c:\\carpeta1\\carpeta2\\...\\fichero.txt

Nota: ficheros es un objeto de la clase FileSystem.

Files

La clase Files regula aspectos que afectan a los ficheros. Se encuentra en el paquete java.nio.files

```
Files.lines(camino, Charset.forName("UTF-8"))
```

El método lines es estático y devuelve un Stream<String>. Con este método se leen todas las líneas de un fichero.

Nota: Camino es un objeto de Path.

BufferedWriter

La clase BufferedWriter manda un flujo de caracteres a un archivo.
Se puede especificar el tamaño del buffer o utilizar el que tiene por defecto.

Se encuentra en java.io

```
BufferedWriter br = Files.newBufferedWriter(camino, Charset.forName("UTF-8"),
StandardOpenOption.APPEND)
```

Esta clase implementa AutoCloseable por lo que puede utilizarse en try-with-resources

```
try(BufferedWriter br = Files.newBufferedWriter(camino, Charset.forName("UTF-8"), StandardOpenOption.APPEND))
```

StandardOpenOption

Enum Constant and Description

APPEND
If the file is opened for WRITE access then bytes will be written to the end of the file rather than the beginning.

CREATE
Create a new file if it does not exist.

CREATE_NEW
Create a new file, failing if the file already exists.

DELETE_ON_CLOSE
Delete on close.

READ
Open for read access.

TRUNCATE_EXISTING
If the file already exists and it is opened for WRITE access, then its length is truncated to 0.

WRITE
Open for write access.

Otros métodos de File

Borrar un fichero: delete(Path path) -> void

Borrar un fichero si existe: deleteIfExists(Path path) -> Boolean

Comprobar si existe un fichero: exists(Path path, LinkOption... options) -> Boolean

Comprobar si es un directorio: isDirectory(Path path, LinkOption... options) -> Boolean

Comprobar si es ejecutable: isExecutable(Path path) -> Boolean

Comprobar si es un fichero oculto: isHidden(Path path) -> Boolean

Comprobar si se puede leer: isReadable(Path path) -> Boolean

Lee todas las líneas de un fichero: lines(Path path, Charset cs) -> Stream<String>

Lee el contenido de un directorio: list(Path dir) ->Stream<path>

Tamaño del fichero en bytes: size(Path path) -> long

Escribe bytes al fichero: write(Path path, byte[] bytes, OpenOption... options) -> Path