

LECTURA Y ESCRITURA DE FICHEROS CON JAVA

ADOLFO SANZ DE DIEGO

JUNIO DE 2011

1 FICHEROS EN JAVA

- Java es multiplataforma, e intenta aislarse del sistema de ficheros, aunque no consigue evitarlo del todo.
- Utiliza el clase **File** del paquete java.io:
 - Está pensada para realizar operaciones de **información** sobre archivos.
 - No proporciona métodos de acceso a los archivos.
 - Sólo proporciona operaciones a nivel de sistema de archivos:
 - listado de archivos,
 - crear carpetas,
 - borrar ficheros,
 - cambiar nombre,
 - etc.

2 USO DE LA CLASE FILE

- Utiliza como único argumento una cadena que representa una **ruta** (ya sea absoluta o relativa) en el sistema de archivos. El problema es que a veces se utiliza / otras \ etc.
- La clase File tiene variables estáticas que nos facilitan la tarea:
 - **separator**: El carácter separador de nombres de archivo y carpetas.
 - / para Linux
 - \ para Windows (que debe escribirse \\).
 - **pathSeparator**: El carácter separador de rutas de archivo que permite poner más de un archivo en una ruta.
 - : para Linux
 - ; para Windows.

3 MÉTODOS DE FILE (I)

boolean isDirectory()	Devuelve true si es una carpeta
----------------------------------	---------------------------------

boolean isFile()	Devuelve true si es un archivo
-------------------------	--------------------------------

boolean exists()	Devuelve true si existe la carpeta o archivo
-----------------------------	----------------------------------------------

boolean canRead()	Devuelve true si se puede leer
------------------------------	--------------------------------

boolean canWrite()	Devuelve true si se puede escribir
-------------------------------	------------------------------------

boolean isHidden()	Devuelve true si es oculto
-------------------------------	----------------------------

boolean
isAbsolute()

Devuelve true si la ruta es
absoluta

4 MÉTODOS DE FILE (II)

String getAbsolutePath()	Devuelve una cadena con la ruta absoluta
-------------------------------------------	------------------------------------------

File getAbsoluteFile()	Devuelve un objeto File con la ruta absoluta
-----------------------------------------	----------------------------------------------

String getName()	Devuelve una cadena con el nombre de la carpeta o archivo
-------------------------	-----------------------------------------------------------

String getParent()	Devuelve el nombre de la carpeta superior o null si no tiene
---------------------------	--------------------------------------------------------------

File
getParentFile()

Devuelve un objeto File la carpeta superior o null si no tiene

boolean
setReadOnly()

Activa el atributo de sólo lectura

boolean delete()

Devuelve true si borra la carpeta o archivo

5 MÉTODOS DE FILE (III)

boolean mkdir()	Devuelve true si consigue crear una carpeta
boolean mkdirs()	Devuelve true si consigue crear una carpeta y si hace falta toda la estructura
String[] list()	Devuelve un array con los nombres de los ficheros
File[] listfiles()	Devuelve un array de objetos File con los ficheros del directorio
static File[] listRoots()	Devuelve un array de objetos File con los ficheros de la carpeta raíz

6 MÉTODOS DE FILE (IV)

**boolean
renameTo(File
f2)**

Devuelve true si consigue
cambiar el nombre por el de f2

long length()

Devuelve el tamaño del archivo
en bytes

**boolean
createNewFile()**

Devuelve true si consigue crear
el archivo,

false si ya estaba creado, y
IOException si falla

**void
deleteOnExit()**

Borra el archivo cuando finaliza
la ejecución del programa

7 EJEMPLO LECTURA FICHERO

```
BufferedReader fichero = null;
StringBuilder texto = null;
try{
    fichero = new BufferedReader(new FileReader("texto.txt"));
    String linea = fichero.readLine();
    while (linea != null) {
        texto.append(linea);
        linea = fichero.readLine();
    }
} catch(Exception e){
    System.out.println("Error"+e.toString());
} finally {
    try {
        if (fichero != null) fichero.close();
    } catch(Exception e){
        System.out.println("Error"+e.toString());
    }
}
```

8 EJEMPLO ESCRITURA FICHERO

```
PrintWriter fichero = null;
try
{
    // si append == true  escribe al final del fichero
    // si append == false sobrescribe el fichero
    boolean append = false;
    fichero = new PrintWriter(new FileWriter("texto.txt", append));
    for (int i = 0; i < 10; i++) {
        fichero.println("Linea " + i);
    }
} catch (Exception e) {
    e.printStackTrace();
} finally {
    try {
        if (null != fichero) fichero.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

9 XML EN JAVA

- Existen numerosas librerías para procesar XML en Java.
- **XStream** nos permite parsear **POJO** (Plain Old Java Objects) a XML y viceversa.
- Características de XStream:
 - **Fácil de usar.**
 - **No requiere mapeos.**
 - **Alto rendimiento.**
 - **XML Limpio.**
 - **No requiere modificar los objetos.**
 - **Soporte completo para objetos complejos.**
 - **Integración con otras API de XML.**
 - **Estrategias de conversión personalizables.**
 - **Mensajes de error detallados.**
 - **Permite otros formatos de salida como JSON.**

10 EJEMPLO XSTREAM: POJO

```
public class PhoneNumber {  
    private int code;  
    private String number;  
    // ... constructores y métodos  
}  
public class Person {  
    private String firstname;  
    private String lastname;  
    private PhoneNumber phone;  
    private PhoneNumber fax;  
    // ... constructores y métodos  
}
```


11 EJEMPLO XSTREAM: XML

```
<person>
  <firstname>Joe</firstname>
  <lastname>Walnes</lastname>
  <phone>
    <code>123</code>
    <number>1234-456</number>
  </phone>
  <fax>
    <code>123</code>
    <number>9999-999</number>
  </fax>
</person>
```

12 EJEMPLO XSTREAM: CONVERSIÓN

- POJO to XML

```
XStream xstream = new XStream();  
xstream.alias("person", Person.class);  
Person joe = new Person("Joe", "Walnes");  
joe.setPhone(new PhoneNumber(123, "1234-456"));  
joe.setFax(new PhoneNumber(123, "9999-999"));  
String xml = xstream.toXML(joe);
```

- XML to POJO

```
Person newJoe = (Person) xstream.fromXML(xml);
```