Acceso a Datos

Tema 1. Manejo de ficheros

Profesor: Juan Ignacio Contreras

ÍNDICE DE CONTENIDOS

OBJETIVOS

A) utilizar clases para la gestión de ficheros y directorios.

B) valorar las ventajas y los inconvenientes de las distintas formas de acceso.

C) utilizar clases para recuperar información almacenada en un fichero XML.

D)utilizar clases para almacenar información en un fichero XML.

E) utilizar clases para convertir a otro formato información contenida en un fichero XML.

F) gestionar las excepciones.

G) probar y documentar las aplicaciones desarrolladas.

1.1 Formas de acceso a un fichero

- En Java, en el paquete java.io, existen varias clases que dan soporte para trabajar con ficheros desde diferentes perspectivas:
  - Según el tipo de contenido:
    - Ficheros de caracteres.
    - Ficheros binarios.
  - Según el modo de acceso:
    - Ficheros secuenciales.
    - Ficheros aleatorios.

1.2 Gestión de flujos de datos

- En Java, el acceso a ficheros es tratado como un flujo (*stream*) de información entre el programa y el fichero.
- Para comunicar un programa con un origen o destino de cierta información (fichero) se usan flujos de información.
- Un flujo no es más que un objeto que hace de intermediario entre el programa y el origen o el destino de la información.
- Con independencia del tipo de flujo que maneje, todos los accesos se hacen más o menos de la misma manera:
  - Para leer.
  - Para escribir.

1.2.1 Clase FileWriter

El flujo **FileWriter** permite escribir caracteres en un fichero de modo secuencial. Esta clase hereda los métodos de la clase Writer. Los constructores principales son:

- **FileWriter** (***String*** *ruta,* ***boolean*** *añadir*).
- **FileWriter** (***File*** *fichero*).

donde ruta indica la localización del archivo en el sistema operativo.

Añadir igual a true indica que el fichero se usa para añadir datos a un fichero ya existente.

1.2.2 Clase FileReader

El flujo **FileReader** permite leer caracteres desde un fichero de modo secuencial. Esta clase hereda los métodos de la clase Reader. Los constructores principales son:

- **FileReader** (***String*** *ruta*)
- **FileReader** (***File*** *fichero*)

1.2.3 Clase FileOutputStream

- El flujo **FileOutputStream** permite escribir bytes en un fichero de manera secuencial.
- Sus constructores tienen los mismos parámetros que los mostrados para **FileWriter**: el fichero puede ser abierto vacío o listo para añadirle datos a los que ya contenga.
- Ya que este flujo está destinado a ficheros binarios (bytes) todas las escrituras se hacen a través de un buffer (array de bytes).

– *public void* **write** (*byte*[] b) **throws** *IOException*

1.2.4 Clase FileInputStream

- El flujo **FileInputStream** permite leer bytes en un fichero de manera secuencial. Sus constructores tienen los mismos parámetros que los mostrados para **FileReader**.
- El método más popular de FileInputStream es el método **read**() que acepta un array de bytes (buffer):
  - *public int* **read** *(byte[]* cbuf*)* ***throws*** *IOException*

1.2.5 Clase RandomAccessFile

- El flujo **RandomAccessFile** permite acceder directamente a cualquier posición dentro del fichero. Proporciona dos constructores básicos:
  - **RandomAccessFile** (***String*** *ruta,* ***String*** *modo*)
  - **RandomAccessFile** (***File*** *fichero,* ***String*** *modo*)

- El parámetro modo especifica para qué se abre el archivo: “r” solo lectura o “rw” lectura y escritura.

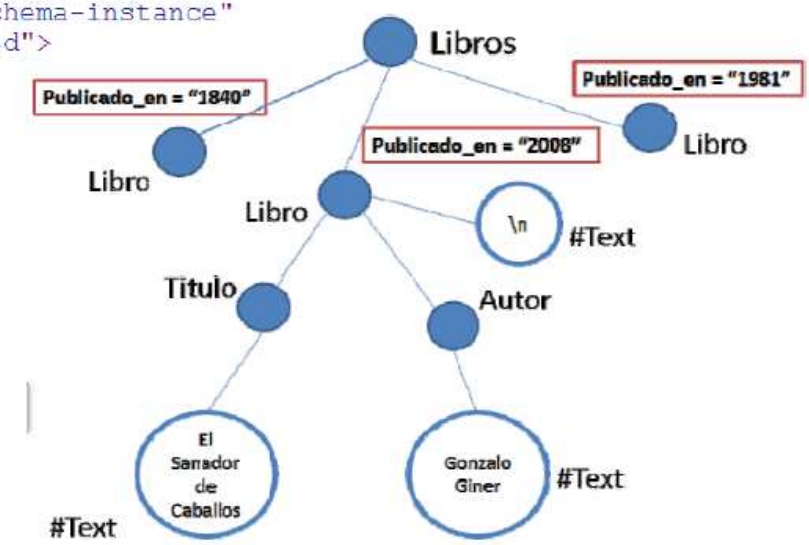
1.3 Ficheros XML (eXtended Markup Language)

- **XML** (eXtended Markup Language – Lenguaje de marcas extendido) ofrece la posibilidad de representar la información de forma neutra, independiente del lenguaje de programación y del sistema operativo empleado.
- Desde un punto de vista a “bajo nivel” XML no es otra cosa que un fichero de texto.
- Sin embargo, desde un punto de vista a “alto nivel” XML no es un fichero de texto cualquiera.
- A continuación se describe el acceso a datos con las tecnologías más extendidas: **DOM**, **SAX** y **JAXB**.

1.4 Acceso a datos con DOM (Document Object Model)

DOM (*Document Object Model*) es una interfaz de programación que permite analizar y manipular dinámicamente y de manera global el contenido, el estilo y la estructura de un documento.

```
▼<Libros xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="LibrosEsquema.xsd">
  ▼<Libro publicado_en="1840">
     <Titulo>El Capote</Titulo>
     <Autor>Nikolai Gogol</Autor>
   </Libro>
  ▼<Libro publicado_en="2008">
     <Titulo>El Sanador de Caballos</Titulo>
     <Autor>Gonzalo Giner</Autor>
   </Libro>
  ▼<Libro publicado_en="1981">
     <Titulo>El Nombre de la Rosa</Titulo>
     <Autor>Umberto Eco</Autor>
   </Libro>
 </Libros>
```



1.4.1 DOM y Java

- DOM ofrece una manera de acceder a documentos XML tanto para ser leído como para ser modificado.
- Una aplicación Java que desea manipular documentos XML accede a la interfaz JAXP (Java API for XML Processing) porque le ofrece una manera transparente de utilizar los diferentes *parsers* que hay para manejar XML.
- Algunos métodos que facilita para manipular un árbol DOM son:
  - *getFirstChild, getNextSibling, getNodeType, getAttributes, getNodeName, getNodeValue*

1.5 Acceso a datos con SAX (Simple Api for XML)

- Es una tecnología para poder acceder a XML desde lenguajes de programación.
- Lee documentos XML de forma secuencial.
- A diferencia de DOM, no carga el documento en memoria, sino que lo lee directamente desde el fichero.

1.5 Acceso a datos con SAX (Simple Api for XML)

- Sigue los siguientes pasos básicos:

  - Se le dice al parser SAX, que fichero quiere que sea leído.
  - El documento XML es traducido a eventos.
  - Los eventos generados pueden controlarse con métodos callbacks.
  - Para implementar los callbacks basta con implementar la interfaz ContentHandler.

1.6 Acceso a datos con JAXB (BINDING)

- **JAXB** (no confundir con la interfaz de acceso JAXP) es una librería de (Un)-Marshalling.
- El concepto de Serialización o Marshalling que ya ha sido introducido, es el proceso de almacenar un conjunto de objetos en un fichero.
- Unmarshaling es justo el proceso contrario: convertir en objetos el contenido de un fichero.
- JAXB es capaz de obtener de un esquema XML una estructura de clases que le da soporte en Java.
- En las siguientes secciones se muestra cómo se puede implementar el uso de JAXB en Java.

1.6.1 ¿Cómo crear clases Java de esquemas XML

• Partiendo de un esquema XML, el proceso de crear la estructura de clases Java que le dé soporte es muy sencillo con JAXB.

- Con JDK 1.7.0 se puede obtener las clases asociadas a un esquema XML con la aplicación xjc.
- Con el IDE eclipse se puede obtener las clases asociadas a un esquema XML.