```
                        //creo el objeto donde voy a guardar los datos
leidos del disco
                        Alumnos alumno = (Alumnos) oi.readObject();
                        System.out.println(alumno);
                  }
            }catch(ClassNotFoundException cn){
                  System.out.println("Error en la clase alumnos");
            }catch(FileNotFoundException fn){
                  System.out.println("Error. Fichero no existe");
            }catch(EOFException eof){
                  System.out.println("Fin del la lectura");
            }catch(IOException io){
                  System.out.println("Error de lectura");
            }finally{
                  try{
                        if(fi != null){
                              fi.close();
                              oi.close();
                        }
                  }catch(IOException io){
                        System.out.println("Error al cerrar el fichero");
            }
      }// fin metod0 lecturaObjetos
}
```

Ficheros de Acceso Aleatorio

Los archivos de acceso secuencial son útiles para la mayoría de las aplicaciones, pero a veces son necesarios archivos de acceso aleatorio que permiten acceder a su contenido de forma secuencial o aleatoria.

La clase **`RandomAccessFile`** del paquete java.io implementa un archivo de acceso aleatorio. Puede ser usada tanto para la lectura como para la escritura de bytes. Dicha clase dispone de métodos para acceder al contenido de un fichero binario de forma aleatoria y para posicionarnos en una posición determinada del mismo. Esta clase no es parte de la jerarquía **InputStream** /**OutputStream**, ya que su comportamiento es totalmente distinto puesto que se puede avanzar y retroceder dentro de un fichero.

Cuando queramos abrir un fichero de acceso aleatorio tendremos que crear un objeto de tipo **`RandomAccessFile`** y en el constructor indicaremos la ruta del fichero y el modo de apertura: sólo lectura "**`r`**", o lectura/escritura "**`r/w`**".

Hay dos posibilidades para abrir un fichero de acceso aleatorio:

```
RandomAccessFile(String path, String modo); // Con el nombre
del fichero

RandomAccessFile(File fichero, String modo); // Con un objeto
File
```

Ejemplos:

```
RandomAccessFile fichero = new RandowAccessFile("Datos.txt", "rw");

RandomAccessFile fichero = new RandowAccessFile(File archivo, "rw");
```

Todo objeto, instancia de **`RandomAccessFile`** soporta el concepto de puntero que indica la posición actual dentro del archivo. Cuando en el fichero se crea el puntero se coloca en 0, apuntando al principio del mismo. Las sucesivas llamadas a los métodos read() y write() ajustan el puntero según la cantidad de bytes leídos o escritos.

Desplazamiento: cualquier operación de lectura/escritura de datos se realiza a partir de la posición actual del **"puntero"** del archivo.

Métodos:	**Función**
`long getFilePointer()`	Devuelve la posición actual del puntero del archivo.
`void seek( long k )`	Coloca el puntero del archivo en la posición indicada por **k** (los archivos empieza en la posición 0).
`int skipBytes( int n )`	Intenta saltar n bytes desde la posición actual.
`long length()`	Devuelve la longitud del archivo.
`void setLength( long t)`	Establece a **t** el tamaño del archivo.

Métodos Escritura:	**Función**
`void write(byte b[], int ini, int len)`	Escribe len caracteres del vector b.
`void write(int i )`	Escribe la parte baja de i (un byte) en el flujo.
`void writeXxx(xxx)`	Escribe el tipo indicado en xxx.

Métodos Lectura:	**Función**
`xxx readXxx();`	Lee y devuelve el tipo leído
`void readFully( byte b[] );`	Lee bytes del archivo y los almacena en un vector de bytes.
`void readFully( byte b[], int ini, int len )`	Lee len bytes del archivo y los almacena en un vector de bytes.
`String readUTF()`	Lee una cadena codificada con el formato UTF-8.

```
package ejemplos10FicherosAleatorios;

import java.io.EOFException;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.RandomAccessFile;
import java.io.BufferedReader;

/* crea un fichero de acceso directo
 * registro: clave (int), nombre (String 25), edad (int)
 * la clave es la posición del registro dentro del fichero
 */
public class Ej01ClaveEsDireccion {
      // en un fichero ramdow hay que establecer el tamaño del registro
      //recordar que un caracter son dos bytes en archivo
      private static final long tamanhoRegistro = 35;
      public static void main(final String[] args) {
```

```
		File f = new File("NombresEdades.DAT");
		String opcion;

		do{
			System.out.println("1.- Introducir nuevo registro");
			System.out.println("2.- Listado completo");
			System.out.println("3.- Buscar registro");
			System.out.println("4.- Modificar registro");
			System.out.println("5.- Salir");

			opcion = introduccionDatos("Introduce una opcion: ");
			try{
				switch (Integer.parseInt(opcion)){
				case 1:
					insertarRegistro(f);
					break;
				case 2:
					if(f.exists())
						listadoCompleto(f);
					else
						System.out.println("El fichero no
existe. Tiene que insertar datos");
					break;
				case 3:
					buscarRegistro(f);
					break;
				case 4:
					modificarRegistro(f);
					break;
				case 5:
					System.exit(0);
				default:
					System.out.println("Opcion erronea");
				}
			}catch (NumberFormatException e){
				System.out.println("La opcion tiene que ser un
numero");
			}
		}while(!opcion.equals("5"));
	}// fin main

	// métodos publicos y static
	/*-------------------------------------------------------------
	 * método para introducir los datos desde el teclado
	 */
	public static String introduccionDatos(final String s){
		try{
			System.out.println("-----------------------------------
------------------");
			System.out.print(s);
			return (new BufferedReader (new InputStreamReader
(System.in))).readLine();
```

```
            }catch (IOException ioe){
                  System.out.println("\nError interno en sistema de
entrada/salida\n");
            }
            return "";
      }// fin metodo introduccionDatos()
      /*-------------------------------------------------------------
       * Metodo que inserta los registrsos en el fichero
       * el campo clave determina la posición del registro dentro del
fichero
       */
      public static void insertarRegistro(File f){
            RandomAccessFile puntero = null;
            int clave=0;
            String nombre="";
            int edad=0;
            try{
                  // abriendo archivo, capturando y grabando datos
                  puntero = new RandomAccessFile (f,"rw");
                  String respuesta = null;
                  do{
                        // teclea los datos
                        clave =
Integer.parseInt(introduccionDatos("Introduce la clave: "));
                        //comprueba la longitud del nombre tecleado si es
menor que 25 lo rellena
                        // si es mayor lo acorta
                        nombre = introduccionDatos("Introduce el nombre:
");
                        if (nombre.length() < 25) {
                              for(int i=nombre.length(); i <25; i++)
                                    nombre=nombre+" ";
                        }
                        else {
                              nombre=nombre.substring(0,25);
                        }
                        edad =
Integer.parseInt(introduccionDatos("Introduce la edad: "));
                        //     grabando el registro en el archivo

                        //colocamos el puntero según la clave
                        puntero.seek((clave-1) * tamanhoRegistro);
                        puntero.writeInt(clave);
                        puntero.writeUTF(nombre);
                        puntero.writeInt(edad);

                        respuesta = introduccionDatos("Desea continuar
S/N");
                  }while(respuesta.equalsIgnoreCase("s"));
            }
            catch(NumberFormatException nfe){
                  System.out.println("Error al introducir los datos");
            }catch(FileNotFoundException fnf){
```

```
                System.out.println("Fichero inexistente");
            }catch (IOException ioe) {
                System.out.println(" Error al escribir en el fichero");
            }finally{
                try{
                    puntero.close();
                }
                catch(IOException e){
                    System.out.println(" Error al cerrar el fichero ");
                }
            }
        }
        //-------------------------------------------------------------
        public static void listadoCompleto(File f){
            int clave=0;
            String nombre="";
            int edad=0;
            RandomAccessFile puntero = null;
            long contadorRegistros = 0;
            try {
                // abriendo archivo, capturando datos
                puntero = new RandomAccessFile (f,"r");
                //calculando el numero de registros
                contadorRegistros = puntero.length()/tamanhoRegistro;
                System.out.println("Nº registros: " +contadorRegistros+"
Tamaño del fichero: " +puntero.length()+ "\n\n");

                for (int r=0; r < contadorRegistros; r++) {
                    puntero.seek(r* tamanhoRegistro);
                    clave=puntero.readInt();
                    nombre = puntero.readUTF();
                    edad=puntero.readInt();
                    if(clave != 0)
                        System.out.println(clave+" "+nombre+"
"+edad);
                }
            }catch(EOFException eof){
                System.out.println("Final del fichero ");
            }catch(FileNotFoundException fnf){
                System.out.println("Fichero inexistente");
            }catch (IOException ioe) {
                System.out.println("Error al leer el fichero ");
            }finally{
                try{
                    puntero.close();
                }
                catch(IOException e){
                    System.out.println(" Error al cerrar el fichero ");
                }
            }
        }
        //-------------------------------------------------------------
        public static void buscarRegistro(File f){
```

```
            int clave=0;
            String nombre="";
            int edad=0;
            RandomAccessFile puntero = null;
            try {
                  // abriendo archivo, capturando datos
                  puntero = new RandomAccessFile (f,"r");
                  clave = Integer.parseInt(introduccionDatos("Introducir la
clave. < 0 para Finalizar>: "));
                  while(clave != 0){
                        //colocamos el puntero según la clave
                        puntero.seek((clave-1) * tamanhoRegistro);
                        //leemos los campos del registro
                        clave=puntero.readInt();
                        nombre = puntero.readUTF();
                        edad=puntero.readInt();
                        //visualizamos los datos
                        System.out.println(clave+" "+nombre+" "+edad);
                        clave =
Integer.parseInt(introduccionDatos("Introducir la clave. < 0 para
Finalizar>: "));
                  }// fin while
            }catch(NumberFormatException nfe){
                  System.out.println("Error al introducir los datos");
            }catch (IOException ioe) {
                  System.out.println("Error de posicionamiento o lectura");
                  System.out.println(ioe.getMessage());
            }finally{
                  try{
                        puntero.close();
                  }catch(IOException e){
                        e.printStackTrace();
                  }
            }
      }
      /*-------------------------------------------------------------
       * Metodo que modificar datos de un registrsos en el fichero
       * el campo clave determina la posición del registro dentro del
fichero
       * hay que posicionarse para leer el registro y hay que volver a
posicionarse
       * antes de escribir
       */
      public static void modificarRegistro(File f){
            int clave=0;
            String nombre="";
            String respuesta = "";
            int edad=0;
            RandomAccessFile puntero = null;
            try {
                  // abriendo archivo, capturando datos
                  puntero = new RandomAccessFile (f,"rw");
```

```
                clave = Integer.parseInt(introduccionDatos("Introducir la
clave a modificar. < 0 para Finalizar>: "));
                while(clave != 0){
                      //colocamos el puntero según la clave para leer el
registro
                      puntero.seek((clave-1) * tamanhoRegistro);
                      //leemos los campos del registro
                      clave=puntero.readInt();
                      nombre = puntero.readUTF();
                      edad=puntero.readInt();
                      //visualizamos los datos
                      System.out.println(clave+" "+nombre+" "+edad);
                      respuesta = introduccionDatos("Desea modificar el
registro. S/N");
                      if (respuesta.compareToIgnoreCase("s") == 0){
                            //teclea los nuevos valores de los campos
                            nombre = introduccionDatos("Introduce el
nombre: ");
                            if (nombre.length() < 25) {
                                  for(int i=nombre.length(); i <25; i++)
                                        nombre=nombre+" ";
                            }
                            else {
                                  nombre=nombre.substring(0,25);
                            }
                            edad =
Integer.parseInt(introduccionDatos("Introduce la edad: "));
                            //     grabando al archivo
                            if (puntero.length()!= 0){
                                  puntero.seek( puntero.length() );
                            }
                            //recolocamos el puntero según la clave para
escribir
                            puntero.seek((clave-1) * tamanhoRegistro);
                            puntero.writeInt(clave);
                            puntero.writeUTF(nombre);
                            puntero.writeInt(edad);
                      }
                      clave =
Integer.parseInt(introduccionDatos("Introducir la clave. < 0 para
Finalizar>: "));
                }// fin while
          }catch(NumberFormatException nfe){
                System.out.println("Error al introducir los datos");
          }catch (IOException ioe) {
                System.out.println("Error de posicionamiento o lectura");
                System.out.println(ioe.getMessage());
          }finally{
                try{
                      puntero.close();
                }catch(IOException e){
                      e.printStackTrace();
                }
```

```
            }
      }// fin metodo modificarEdad()
}
```

Segundo ejemplo en el que la clave es autoincrementable y la posición del registro en el fichero.

```
package ejemplos10FicherosAleatorios;
import java.io.EOFException;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.RandomAccessFile;
import java.io.BufferedReader;
/* crea un fichero de acceso directo
 * registro: clave (int), nombre (String 25), edad (int)
 * la clave es la posición del registro dentro del fichero
 */
public class Ej01ClaveEsDireccion {
      // en un fichero ramdow hay que establecer el tamaño del registro
      //recordar que un caracter son dos bytes en archivo
      private static final long tamanhoRegistro = 35;
      public static void main(final String[] args) {
            File f = new File("NombresEdades.DAT");
            String opcion;
            do{
                  System.out.println("1.- Introducir nuevo registro");
                  System.out.println("2.- Listado completo");
                  System.out.println("3.- Buscar registro");
                  System.out.println("4.- Modificar registro");
                  System.out.println("5.- Salir");
                  opcion = introduccionDatos("Introduce una opcion: ");
                  try{
                        switch (Integer.parseInt(opcion)){
                        case 1:
                              insertarRegistro(f);
                              break;
                        case 2:
                              if(f.exists())
                                    listadoCompleto(f);
                              else
                                    System.out.println("El fichero no
existe. Tiene que insertar datos");
                              break;
                        case 3:
                              buscarRegistro(f);
                              break;
                        case 4:
                              modificarRegistro(f);
                              break;
                        case 5:
                              System.exit(0);
```

```
                        default:
                              System.out.println("Opcion erronea");
                        }
                  }catch (NumberFormatException e){
                        System.out.println("La opcion tiene que ser un
numero");
                  }
            }while(!opcion.equals("5"));
      }// fin main

      // métodos publicos y static
      /*-------------------------------------------------------------
       * método para introducir los datos desde el teclado
       */
      public static String introduccionDatos(final String s){
            try{
                  System.out.println("-------------------------------------
-------------------");
                  System.out.print(s);
                  return (new BufferedReader (new InputStreamReader
(System.in))).readLine();
            }catch (IOException ioe){
                  System.out.println("\nError interno en sistema de
entrada/salida\n");
            }
            return "";
      }// fin metodo introduccionDatos()
      /*-------------------------------------------------------------
       * Metodo que inserta los registrsos en el fichero
       * el campo clave determina la posición del registro dentro del
fichero
       */
      public static void insertarRegistro(File f){
            RandomAccessFile puntero = null;
            int clave=0;
            String nombre="";
            int edad=0;
            try{
                  // abriendo archivo, capturando y grabando datos
                  puntero = new RandomAccessFile (f,"rw");
                  String respuesta = null;
                  do{
                        // teclea los datos
                        clave =
Integer.parseInt(introduccionDatos("Introduce la clave: "));
                        //comprueba la longitud del nombre tecleado si es
menor que 25 lo rellena
                        // si es mayor lo acorta
                        nombre = introduccionDatos("Introduce el nombre:
");
                        if (nombre.length() < 25) {
                              for(int i=nombre.length(); i <25; i++)
                                    nombre=nombre+" ";
```

```
                        }
                        else {
                              nombre=nombre.substring(0,25);
                        }

                        edad =
Integer.parseInt(introduccionDatos("Introduce la edad: "));
                        //     grabando el registro en el archivo
                        //colocamos el puntero según la clave
                        puntero.seek((clave-1) * tamanhoRegistro);
                        puntero.writeInt(clave);
                        puntero.writeUTF(nombre);
                        puntero.writeInt(edad);
                        respuesta = introduccionDatos("Desea continuar
S/N");
                  }while(respuesta.equalsIgnoreCase("s"));
            }
            catch(NumberFormatException nfe){
                  System.out.println("Error al introducir los datos");
            }catch(FileNotFoundException fnf){
                  System.out.println("Fichero inexistente");
            }catch (IOException ioe) {
                  System.out.println(" Error al escribir en el fichero");
            }finally{
                  try{
                        puntero.close();
                  }
                  catch(IOException e){
                        System.out.println(" Error al cerrar el fichero ");
                  }
            }
      }
      //-----------------------------------------------------------
      public static void listadoCompleto(File f){
            int clave=0;
            String nombre="";
            int edad=0;
            RandomAccessFile puntero = null;
            long contadorRegistros = 0;
            try {
                  // abriendo archivo, capturando datos
                  puntero = new RandomAccessFile (f,"r");
                  //calculando el numero de registros
                  contadorRegistros = puntero.length()/tamanhoRegistro;
                  System.out.println("Nº registros: " +contadorRegistros+"
Tamaño del fichero: " +puntero.length()+ "\n\n");
                  for (int r=0; r < contadorRegistros; r++) {
                        puntero.seek(r* tamanhoRegistro);
                        clave=puntero.readInt();
                        nombre = puntero.readUTF();
                        edad=puntero.readInt();
                        if(clave != 0)
```

```
                        System.out.println(clave+" "+nombre+"
"+edad);
                }
            }catch(EOFException eof){
                System.out.println("Final del fichero ");
            }catch(FileNotFoundException fnf){
                System.out.println("Fichero inexistente");
            }catch (IOException ioe) {
                System.out.println("Error al leer el fichero ");
            }finally{
                try{
                    puntero.close();
                }
                catch(IOException e){
                    System.out.println(" Error al cerrar el fichero ");
                }
            }
}
      //-------------------------------------------------------------
      public static void buscarRegistro(File f){
            int clave=0;
            String nombre="";
            int edad=0;
            RandomAccessFile puntero = null;
            try {
                // abriendo archivo, capturando datos
                puntero = new RandomAccessFile (f,"r");
                clave = Integer.parseInt(introduccionDatos("Introducir la
clave. < 0 para Finalizar>: "));
                while(clave != 0){
                    //colocamos el puntero según la clave
                    puntero.seek((clave-1) * tamanhoRegistro);
                    //leemos los campos del registro
                    clave=puntero.readInt();
                    nombre = puntero.readUTF();
                    edad=puntero.readInt();
                    //visualizamos los datos
                    System.out.println(clave+" "+nombre+" "+edad);
                    clave =
Integer.parseInt(introduccionDatos("Introducir la clave. < 0 para
Finalizar>: "));
                }// fin while
            }catch(NumberFormatException nfe){
                System.out.println("Error al introducir los datos");
            }catch (IOException ioe) {
                System.out.println("Error de posicionamiento o lectura");
                System.out.println(ioe.getMessage());
            }finally{
                try{
                    puntero.close();
                }catch(IOException e){
                    e.printStackTrace();
                }
```

```
            }
      }
      /*-------------------------------------------------------------
       * Metodo que modificar datos de un registrsos en el fichero
       * el campo clave determina la posición del registro dentro del
fichero
       * hay que posicionarse para leer el registro y hay que volver a
posicionarse
       * antes de escribir
       */
      public static void modificarRegistro(File f){
            int clave=0;
            String nombre="";
            String respuesta = "";
            int edad=0;
            RandomAccessFile puntero = null;
            try {
                  // abriendo archivo, capturando datos
                  puntero = new RandomAccessFile (f,"rw");
                  clave = Integer.parseInt(introduccionDatos("Introducir la
clave a modificar. < 0 para Finalizar>: "));
                  while(clave != 0){
                        //colocamos el puntero según la clave para leer el
registro
                        puntero.seek((clave-1) * tamanhoRegistro);
                        //leemos los campos del registro
                        clave=puntero.readInt();
                        nombre = puntero.readUTF();
                        edad=puntero.readInt();
                        //visualizamos los datos
                        System.out.println(clave+" "+nombre+" "+edad);
                        respuesta = introduccionDatos("Desea modificar el
registro. S/N");
                        if (respuesta.compareToIgnoreCase("s") == 0){
                              //teclea los nuevos valores de los campos
                              nombre = introduccionDatos("Introduce el
nombre: ");
                              if (nombre.length() < 25) {
                                    for(int i=nombre.length(); i <25; i++)
                                          nombre=nombre+" ";
                              }
                              else {
                                    nombre=nombre.substring(0,25);
                              }
                              edad =
Integer.parseInt(introduccionDatos("Introduce la edad: "));
                              //     grabando al archivo
                              if (puntero.length()!= 0){
                                    puntero.seek( puntero.length() );
                              }
                              //recolocamos el puntero según la clave para
escribir
                              puntero.seek((clave-1) * tamanhoRegistro);
```

```
                        puntero.writeInt(clave);
                        puntero.writeUTF(nombre);
                        puntero.writeInt(edad);
                  }
                  clave =
Integer.parseInt(introduccionDatos("Introducir la clave. < 0 para
Finalizar>: "));
            }// fin while
      }catch(NumberFormatException nfe){
            System.out.println("Error al introducir los datos");
      }catch (IOException ioe) {
            System.out.println("Error de posicionamiento o lectura");
            System.out.println(ioe.getMessage());
      }finally{
            try{
                  puntero.close();
            }catch(IOException e){
                  e.printStackTrace();
            }
      }
}// fin metodo modificarEdad()
}
```

TRABAJO CON FICHEROS XML

XML (*eXtensible Markup Languaje – Lenguaje de Etiquetado Extensible*) es un metalenguaje, es decir; un lenguaje para la definición de lenguajes de marcado. Nos permite jerarquizar y estructurar la información así como describir los contenidos dentro del propio documento. Los ficheros XML son ficheros de texto escritos en XML donde la información está organizada de forma secuencial y en orden jerárquico. Existen una serie de marcas especiales como son los símbolos menor que < y mayor que >, que se usan para delimitar las marcas que dan la estructura al documento. Cada marca tiene un nombre y puede tener 0 o más atributos.

Un fichero XML sencillo tiene la siguiente estructura:

```
<Empleado>
      <empleado>
            <id>1</id>
            <apellido>GARCIA</apellido>
            <departamento>25</departamento>
            <salario>1826.25</salario>
      </empleado>
      <empleado>
            <id>2</id>
            <apellido>FERNANDEZ</apellido>
            <departamento>2</departamento>
            <salario>1256.25</salario>
      </empleado>
      <empleado>
            <id>3</id>
            <apellido>ALVAREZ</apellido>
            <departamento>10</departamento>
```

```
            <salario>2356.25</salario>
        </empleado>
</Empleado>
```

Los ficheros XML se pueden utilizar:

- Para proporcionar datos a una base de datos o para almacenar copias de partes del contenido de la base de datos.
- Para escribir ficheros de configuración de programas
- En el protocolo SOAP (Simple Object Access Protocol), para ejecutar comandos en servidores remotos; la información enviada al servidor remoto y el resultado de la ejecución del comando se envían en ficheros XML.

Para leer los ficheros XML y acceder a su contenido y estructura, se utiliza un procesador de XML o **parser**. El procesador lee los documentos y proporciona acceso a su contenido y estructura. Algunos de los procesadores más empleados son: **DOM**: *Modelo de Objetos de Documento* y **SAX**: *API Simple para XML*.

- **DOM**: un procesador XML que utilice este planteamiento almacena la estructura del documento en memoria en forma de árbol con nodos padre, nodos hijo y nodos finales (que son aquellos que no tienen descendientes). Una vez creado el árbol, se van recorriendo los diferentes nodos (de arriba abajo y también se puede volver atrás) y se analiza a qué tipo particular pertenecen. Podemos modificar cualquier nodo del árbol. Tiene su origen en el W3C[1]. Este tipo de procesamiento necesita más recursos de memoria y tiempo sobre todo si los ficheros XML a procesar son bastante grandes y complejos.

  Con ficheros XML pequeños no tendremos problemas, pero si tuviéramos un árbol muy muy grande entonces tendríamos una falta de **heap space**[2].

- **SAX**: un procesador que utilice este planteamiento lee un fichero XML de forma secuencial y produce una secuencia de eventos (comienzo/fin del documento, comienzo/fin de una etiqueta, etc.) en función de los resultados de la lectura. Cada evento invoca a un método definido por el programador. Este tipo de procesamiento prácticamente no consume memoria, pero por otra parte, impide tener una visión global del documento por el que navegar.

  Al contrario que con DOM, al procesar en SAX no vamos a tener la representación completa del árbol en memoria, pues SAX funciona con eventos. Esto implica:

  - Al no tener el árbol completo no puede volver atrás, pues va leyendo secuencialmente.
  - La modificación de un nodo y la inserción de nuevos nodos son mucho más complejas.
  - Como no tiene el árbol en memoria es mucho más **memory friendly**, de modo que es la única opción viable para casos de ficheros muy grandes, pero demasiado complejo para ficheros pequeños.

[1] El Word Wide Web Consortium (W3C) es una comunidad internacional que desarrolla estándares que aseguran el crecimiento de la Web a largo plazo.
[2] Espacio de memoria que utilizan los objetos.

- Al ser orientado a eventos, el procesado se vuelve bastante complejo.

Acceso a Ficheros XML con DOM

Para poder trabajar con **DOM** en Java necesitamos las clases e interfaces que componen el paquete ***org.w3c.dom*** (contenido en el JSDK) y el paquete ***javax.xml.parsers*** del API estándar de Java que proporciona un par de clases abstractas que toda implementación **DOM** para Java debe extender. Estas clases ofrecen métodos para cargar documentos desde una fuente de datos (fichero, InputStream, etc.) Contiene dos clases fundamentales: ***DocumentBuilderFactory*** y ***DocumentBuilder***.

DOM no define ningún mecanismo para generar un fichero XML a partir de un árbol DOM. Para eso usaremos el paquete ***javax.xml.transform***, que permite especificar una fuente y un resultado. La fuente y el resultado pueden ser ficheros, flujos de datos o nodos **DOM** entre otros.

Los programas Java que utilicen **DOM** necesitan interfaces, algunas son:

Interface	Función
Document	Es un objeto que equivale a un ejemplar de un documento XML. Permite crear nuevos nodos en el documento.
Element	Cada elemento del documento XML tiene un equivalente en un objeto de este tipo. Expone propiedades y métodos para manipular los elementos del documento y sus atributos
Node	Representa cualquier nodo del documento
NodeList	Contiene una lista con los nodos hijos de un nodo.
Att	Permite acceder a los atributos de un nodo
Text	Son los datos carácter de un elemento
CharacterData	Representa los datos carácter presentes en el documento. Proporciona atributos y métodos para manipular los datos de caracteres.
DocumentType	Proporciona información contenida en la etiqueta <!DOCTYPE>

Ejemplo de creación de un fichero XML, en el ejemplo vamos a partir del fichero Personas.txt.

Lo primero que tenemos que hacer es importar los paquetes necesarios:

```
import java.io.*;
import javax.xml.parsers.*;
import javax.xml.transform.*;
import org.w3c.dom.*;
```

Después crearemos una instancia de ***DocumentBuilderFactory*** para construir el parser (necesario para crear el árbol), se debe encerrar entre **try-catch** porque se puede producir la excepción ***ParserConfigurationException***.

```
DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
     try{
          DocumentBuilder db = dbf.newDocumentBuilder();
……..
```

Creamos un documento vacío de nombre ***documento*** con el nodo raíz de nombre Personas y asignamos la versión XML:

```
DOMImplementation implementacion = db.getDOMImplementation();
Document documento = implementacion.createDocument(null, "Personas", null);
documento.setXmlVersion("1.0"); // asignamos la versión de nuestro XML
```

```
DOMImplementation implementacion = db.getDOMImplementation();
Document documento = implementacion.createDocument(null, "Personas",
null);// crea el documento con el nodo raíz de nombre Personas
```

El siguiente paso será recorrer el fichero con los datos de las personas y por cada registro crear un nodo persona con 3 hijos (clave, nombre, edad). Cada nodo hijo tendrá su valor (por ejemplo: 1. Isabel, 42). Para crear un elemento usamos el método ***createElement(String)*** llevando como parámetro el nombre que se pone entre las etiquetas menor que y mayor que. El siguiente código crea y añade el nodo <persona> al documento:

```
Element raiz = documento.createElement("persona");    //creamos el
nodo persona
documento.getDocumentElement().appendChild(raiz);// lo pegamos a la
raíz del documento
```

A continuación se añaden los hijos de ese nodo (raíz), estos se añaden en la función *CrearElemento():*

```
CrearElemento("clave", Integer.toString(clave),raiz,documento); //
añadir clave
CrearElemento("nombre", nombre.trim(),raiz,documento); // añadir
nombre
CrearElemento("edad",       Integer.toString(edad),raiz,documento);//
añadir edad
```

La función recibe el nombre del nodo hijo (clave, nombre, edad) y sus textos o valores que tienen que estar en formato String (1. Isabel, 42), el nodo al que se va a añadir (raíz) y el documento (documento). Para crear el nodo hijo (<clave> o <nombre> o <edad>) se escribe:

```
Element elemento = documento.createElement(datoPersona); // creamos
hijo
```

Para añadir su valor o texto se usa el método ***createTextNode(String)***:

```
Text texto = documento.createTextNode(valor); // damos valor
```

A continuación se añade el nodo hijo a la raíz (persona) y su texto o valor al nodo hijo:

```
raiz.appendChild(elemento); // pegamos el elemento hijo a la raiz
elemento.appendChild(texto); // pegamos el valor
```

Al final se generaría algo similar a esto:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Personas>
     <persona>
          <clave>3</clave>
          <nombre>Beatriz</nombre>
          <edad>25</edad>
```

```
	</persona>
	<persona>
		<clave>5</clave>
		<nombre>Isabel</nombre>
		<edad>30</edad>
	</persona>
	<persona>
		<clave>7</clave>
		<nombre>Iciar</nombre>
		<edad>16</edad>
	</persona>
</Personas>
```

La función para crear el fichero es la siguiente:

```
// metodo de insercion de los datos de la persona
static void CrearElemento(final String datoPersona, final String
valor, final Element raiz, final Document documento){
	Element elemento = documento.createElement(datoPersona); //
creamos hijo
	Text texto = documento.createTextNode(valor); // damos valor
	raiz.appendChild(elemento); // pegamos el elemento hijo a la
raiz
	elemento.appendChild(texto); // pegamos el valor
}// fin del método
```

En los últimos pasos se crea la fuente XML a partir del documento:

```
	Source fuente = new DOMSource(documento);
```

Se crea el resultado en el fichero **Personas.xml**:

```
	Result resultado = new StreamResult(new
java.io.File("Personas.xml"));
```

Se obtiene su TransformerFactory:

```
	Transformer transformer =
TransformerFactory.newInstance().newTransformer();
```

Se realiza la transformación del documento a fichero.

```
	transformer.transform(fuente, resultado);
```

Para mostrar el documento por pantalla, podemos especificar como resultado el canal de salida ***System.out***:

```
	Result consola = new StreamResult(System.out);
	transformer.transform(fuente, consola);
```

El código completo es:

```
package ejemplos11FicherosXML;

import java.io.File;
import java.io.IOException;
```

```
import java.io.RandomAccessFile;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.transform.Result;
import javax.xml.transform.Source;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;

import org.w3c.dom.DOMImplementation;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Text;

public class CrearFicheroDOM {
	private static final long tamanhoRegistro = 35;
	public static void main(final String[] args) throws IOException {
		File fichero = new File ("NombresEdades.dat");
		RandomAccessFile raf = new RandomAccessFile(fichero, "r");
		int clave, edad;
		long posicion = 0; // para situarnos al principio del fichero
		String nombre, aux;

		DocumentBuilderFactory dbf =
DocumentBuilderFactory.newInstance();
		try{
			DocumentBuilder db = dbf.newDocumentBuilder();
			DOMImplementation implementacion =
db.getDOMImplementation();
			Document documento = implementacion.createDocument(null,
"Personas", null);// crea el documento con el nodo raíz de nombre Personas
			documento.setXmlVersion("1.0"); // asignamos la versión
de nuestro XML

			for(;;){
				raf.seek(posicion); // nos posicionamos al comienzo
del fichero
				clave=raf.readInt();    // leemos los datos del
fichero
				nombre = raf.readUTF();
				edad=raf.readInt();

				if(clave > 0){    // clave valida
					Element raiz =
documento.createElement("persona");//creamos el  nodo persona

	documento.getDocumentElement().appendChild(raiz);// lo pegamos a la
raíz del documento
					CrearElemento("clave",
Integer.toString(clave),raiz,documento); // añadir clave
```

```
                              CrearElemento("nombre",
nombre.trim(),raiz,documento); // añadir nombre
                              CrearElemento("edad",
Integer.toString(edad),raiz,documento);// añadir edad

                              // el método trim() elimina los espacios en
blanco al pricipio y al final de la cadena
                        }// fin if clave

                        posicion = posicion+tamanhoRegistro; // se
posiciona para el siguiente registro
                        if(raf.getFilePointer() == raf.length())
                              break;
                  }// fin del for que recorre el fichero

                  // recorremos el fichero XML para ver su contenido
                  Source fuente = new DOMSource(documento);
                  Result resultado = new StreamResult(new
java.io.File("Personas.xml"));
                  Transformer transformer =
TransformerFactory.newInstance().newTransformer();
                  transformer.transform(fuente, resultado);
                  // para mostrar el documento por pantalla,podemos
especificar como resultado el canal de salida System.out
                  Result consola = new StreamResult(System.out);
                  transformer.transform(fuente, consola);
            }catch(Exception e){
                  System.err.println("Error: " +e);
            }
            raf.close();
      }// fin del main

      // metodo de insercion de los datos de la persona
      static void CrearElemento(final String datoPersona, final String
valor, final Element raiz, final Document documento){
            Element elemento = documento.createElement(datoPersona); //
creamos hijo
            Text texto = documento.createTextNode(valor); // damos valor
            raiz.appendChild(elemento); // pegamos el elemento hijo a la
raiz
            elemento.appendChild(texto); // pegamos el valor
      }// fin del metodo
}// fin de la clase
```

Leer un documento XML con DOM

Para leer un documento XML, creamos una instancia de **DocumentBuilderFactory** para construir el parser y cargamos el documento con el método **parse().**

```
DocumentBuilder db = dbf.newDocumentBuilder();
Document documento = db.parse(new File("Personas.xml"));
```

Obtenemos la lista de nodos con nombre *personas* de todo el documento:

```
NodeList personas = documento.getElementsByTagName("persona");
```

Se realiza un bucle para recorrer la lista de nodos. Por cada nodo se obtienen sus etiquetas y sus valores llamando a la función **getNodo().**

El código es el siguiente:

```
package ejemplos11FicherosXML;

import java.io.File;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;

public class LecturaFicheroDOM {

	public static void main(String[] args) {
		DocumentBuilderFactory dbf =
DocumentBuilderFactory.newInstance();
		try{
			DocumentBuilder db = dbf.newDocumentBuilder();
			Document documento = db.parse(new File("Personas.xml"));
			//el método parse devuelve el documento DOM que se va a
crear para el fichero XML
			documento.getDocumentElement().normalize();
			/*
			 * el método normalize transforma el texto Unicode que
nosotros le enviemos en texto
			 * normalizado o bien, texto bajo una misma norma
estándar, basado en la estandarización
			 * de normalización Unicode descrita en Unicode Standard
Annex #15 – Unicode Normalization
			 *  Forms.
			 */
			System.out.println("Elemento raiz: "
+documento.getDocumentElement().getNodeName());
			// getNodeName() imprime el nombre de la raiz

			//crea una lista con todos los nodos de personas
			NodeList personas =
documento.getElementsByTagName("persona");
			/*
			 * Por medio de este método lo que se selecciona es una
lista de nodos cuyo elemento es
			 * el especificado como parámetro; a cada uno de los
nodos se le asigna un índice,
			 * de acuerdo al orden en el que aparecen en el marcado
del documento.
			 */

			// recorre la lista
```

```
                for(int i=0; i< personas.getLength(); i++){
                        Node persona = personas.item(i); // obtener un nodo
                        if(persona.getNodeType() == Node.ELEMENT_NODE){ //
tipo de nodo
                                Element elemento = (Element) persona;
//obtener los elementos del nodo

                                System.out.println("Clave: "
+getNodo("clave", elemento));
                                System.out.println("Nombre: "
+getNodo("nombre", elemento));
                                System.out.println("Edad: " +getNodo("edad",
elemento));
                        }// fin if
                }// fin for
            }catch(Exception e){
                e.printStackTrace();
            }
        }// fin main

        // obtener la información de un  nodo
        private static String getNodo(final String etiqueta, final Element
elemento){
            NodeList nodo =
elemento.getElementsByTagName(etiqueta).item(0).getChildNodes();
            /*
             * getChildNodes() para obtener una lista de los nodos hijos de
un elemento
             */
            Node valorNodo = nodo.item(0);
            //metodo item() devuelve el valor del nodo indicado como
parametro
            return valorNodo.getNodeValue(); // devuelve el valor del nodo
        }
}// fin clase
```

Acceso a Ficheros XML con SAX

SAX (API Simple para XML) es un conjunto de clases e interfaces que ofrecen una herramienta muy útil para el procesamiento de documentos XML. Permite analizar los documentos de forma secuencial (es decir, no carga todo el fichero en memoria como hace DOM), esto implica poco consumo de memoria aunque los documentos sean de gran tamaño, en contraposición, impide tener una visión global del documento que se va a analizar. SAX es más complejo de programar que DOM, es un API totalmente escrita en Java e incluida dentro de JRE que nos permite crear nuestro propio parser XML.

La lectura de un documento XML produce eventos que ocasiona la llamada a métodos, los eventos son encontrar la etiqueta de inicio y fin de documento (***startDocument()*** y ***endDocument()***), la etiqueta de inicio y fin de un elemento (***startElement()*** y ***endElement()***), los caracteres entre etiquetas (***characters()***), etc.

Documento XML (Personas.xml)	Métodos asociados a eventos del documento
<?xml version="1.0" encoding="UTF-8" standalone= "no"?>	startDocument()
<Personas>	startElement()
<persona>	startElement()
<clave>	startElement()
3	characters()
</clave>	endElement()
</persona>	endElement()
<persona>	startElement()
<clave>	startElement()
5	characters()
</clave>	endElement()
</persona>	endElement()
. . . . .	. . . . .
</Personas>	endElement()
	endDocument()

Ejemplo en Java en el que se muestran los pasos básicos necesarios para hacer que se puedan tratar los eventos.

En primer lugar se incluyen las clases e interfaces de SAX:

```
import org.xml.sax.Attributes;
import org.xml.sax.InputSource;
import org.xml.sax.SAXException;
import org.xml.sax.XMLReader;
import org.xml.sax.helpers.DefaultHandler;
import org.xml.sax.helpers.XMLReaderFactory;
```

Se crea un objeto procesador de XML, es decir un ***XMLReader***[3], durante la creación de este objeto se puede producir una excepción (***SAXException***) que es necesario capturar:

```
XMLReader procesadorXML = XMLReaderFactory.createXMLReader();
```

A continuación, hay que indicar al ***XMLReader*** qué objetos poseen los métodos que tratarán los eventos. Estos objetos serán normalmente implementaciones de las siguientes interfaces:

Interface	Función
ContentHandler	Recibe las notificaciones de los eventos que ocurren en el documento.
DTDHandler[4]	Recoge eventos relacionados con la DTD (Declaración de tipo de documento
ErrorHandler	Define métodos de tratamiento de errores
EntityResolver	Sus métodos se llaman cada vez que se encuentra una referencia a una entidad.
DefaultHandler	Clase que provee una implementación por defecto para todos sus métodos, el programador definirá los métodos que sean utilizados por el programa. Esta clase es de la que extenderemos para poder crear nuestro parser de XML.

[3] La extensión XMLReader es un analizador de XML. El lector actúa como un cursor yendo hacia delante en el flujo del documento y deteniéndose en cada nodo del camino.

[4] Definición de tipo de documento

DefaultHandler es una clase que va a procesar cada evento que lance el procesador SAX. Basta con heredar del manejador por defecto de **SAX *DefaultHandler*** y sobreescribir los métodos correspondientes a los eventos deseados. Los más comunes son:

- ***startDocument***: se produce al comenzar el procesado del documento xml.
- ***endDocument***: se produce al finalizar el procesado del documento xml.
- ***startElement***: se produce al comenzar el procesado de una etiqueta xml. Es aquí donde se leen los atributos de las etiquetas.
- ***endElement***: se produce al finalizar el procesado de una etiqueta xml.
- ***characters***: se produce al encontrar una cadena de texto.

Para indicar al procesador XML los objetos que realizarán el tratamiento se utiliza alguno de los siguientes métodos incluidos dentro de los objetos ***XMLReader***: ***setContentHandler(), setDTDHandler(), setEntilyResolver(), setErrorHandler()***; cada uno trata un tipo de evento y está asociado con una interfaz determinada.

```
GestionContenido gestor = new GestionContenido();
procesadorXML.setContentHandler(gestor);
```

A continuación se define el fichero XML que se va a leer mediante un objeto ***InputSource***:

```
InputSource ficheroXML = new InputSource("Personas.xml");
```

Por último, se procesa el documento XML mediante el método ***parse()*** del objeto XMLReader, le pasaremos un objeto InputSource:

```
procesadorXML.parse(ficheroXML);
```

El ejemplo completo sería:

```
package ejemplos11FicherosXML;
/*
 * Ejemplo que crea un fichero XML con SAX a partir del fichero aleatorio
Personas.txt
 */
import java.io.IOException;

import org.xml.sax.Attributes;
import org.xml.sax.InputSource;
import org.xml.sax.SAXException;
import org.xml.sax.XMLReader;
import org.xml.sax.helpers.DefaultHandler;
import org.xml.sax.helpers.XMLReaderFactory;

public class LeerFicheroXMLconSAX {

	public static void main(final String[] args) {

		try{/*
		 * La extensión XMLReader es un analizador de XML.
		 * El lector actúa como un cursor yendo hacia delante en el
flujo del documento y
		 * deteniéndose en cada nodo del camino.
```

```
            */
                  XMLReader procesadorXML =
XMLReaderFactory.createXMLReader();

                  // esta clase es la que extiende a DefaultHandler
                  GestionContenido gestor = new GestionContenido();

                  // le pasa al procesador XML los eventos que van pasando
en el fichero XML
                  procesadorXML.setContentHandler(gestor);

                  /*
                   * InputSource
                   * Esta clase permite una aplicación SAX para encapsular
información acerca de una fuente de entrada
                   * en un solo objeto, que puede incluir un identificador
público, un identificador de sistema,
                   * un flujo de bytes (posiblemente con una codificación
especificada), y / o un flujo de caracteres.
                   * El analizador SAX utilizará el objeto InputSource para
determinar cómo leer la entrada XML
                   * según sean caracteres, bytes.
                   */
                  InputSource ficheroXML = new InputSource("Personas.xml");
                  //le pasamos al procesador el fichero a leer
                  procesadorXML.parse(ficheroXML);
            }catch(SAXException se){
                  System.out.println("Error SAX");
            }catch(IOException io){
                  System.out.println("Error de L/E");
            }

      }// fin main
}// fin clase EjemploFicheroXMLSAX01

class GestionContenido extends DefaultHandler{
      public GestionContenido(){
            super();
      }
      @Override
      public void startDocument(){
            System.out.println("Comienzo del documento XML");
      }
      @Override
      public void endDocument(){
            System.out.println("Final del documento XML");
      }
      public void startElement(final String uri, final String nombre, final
String nombreC, final Attributes aatts){
            System.out.println("Principio Elemento: " +nombre);
      }
      @Override
```

```
	public void endElement(final String uri, final String nombre, final
String nombreC){
		System.out.println("Fin Elemento: " +nombre);
	}
	@Override
	public void characters(final char[] ch, final int inicio, final int
longitud){
		String car = new String(ch, inicio, longitud);
		car = car.replaceAll("[\t\n]", ""); //quitar saltos de linea
		System.out.println("\tTexto: " +car);
	}
}
```

Serialización de Objetos XML

Para serializar objetos Java a XMl o viceversa necesitamos la librería **XStream**. Para poder utilizarla debemos descargar los fichero JAR desde la web: http://xstream.codehaus.org/download.html, para el ejemplo descargaremos el fichero ***Binary distribution*** (*xstream-distribution-1.4-2-bin.zip*) lo descomprimimos y buscamos el fichero JAR ***xstream-1.4.7.jar*** que está en la carpeta ***lib.*** También necesitamos el fichero ***kxml2-2.3.0.jar*** que se puede descargar desde el apartado ***Optional Dependencies***.

Partimos del fichero *"Alumnos.Dat"* que contiene objetos *Alumno*. El proceso es crear una lista de objetos *Alumno* y la convertiremos en un fichero de datos XML. Necesitamos la clase *Alumno* y la clase *ListaAlumno* en la que definimos una lista de objetos *Alumno* que pasaremos al fichero XML:

```
package ejemplos12SerializacionObjetosXML01;
/*
 * Ejemplo que recorre el fichero Alumnos.Dat para crear una lista de
alumnos
 * que después se insertarán en el fichero Alumnos.xml
 */

import java.io.EOFException;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import com.thoughtworks.xstream.XStream;
public class CrearFicheroXMLconObjetos {
	public static void main(final String[] args) throws IOException,
ClassNotFoundException {
		File fichero = new File("Alumnos.DAT");
		FileInputStream lectura = new FileInputStream(fichero); //
flujo de entrada
		// conecta el flujo de bytes al flujo de datos
		ObjectInputStream datos = new ObjectInputStream(lectura);
		System.out.println("Comienza el proceso de creación del fichero
XML....");

		// Creamos un objeto Lista de alumnos
```

```
            ListaAlumnos listaalu = new ListaAlumnos();
            try{
                  while(true){ // lectura del fichero
                        Alumno alumno = (Alumno)datos.readObject();// leer
un alumno
                        listaalu.add(alumno); //añadir un alumno a la lista
                  }// fin while
            }catch(EOFException eo){}
            datos.close();
            try{
                  XStream xstream = new XStream();
                  //cambiar de nombre a las etiquetas XML
                  xstream.alias("ListadoAlumnos", ListaAlumnos.class);
                  xstream.alias("DatosAlumno", Alumno.class);
                  //quitar etiqueta lista (atributo de la clase ListaAlumno
                  xstream.addImplicitCollection(ListaAlumnos.class,
"lista");
                  //Insertar los objetos en el XML
                  xstream.toXML(listaalu, new
FileOutputStream("Alumnos.xml"));
                  System.out.println("Creado el fichero xml");
            }catch(Exception e){
                  e.printStackTrace();
            }
      }// fin main
}// fin clase
```

El fichero generado tiene el siguiente aspecto:

```
<Listado Alumnos>
      <DatosAlumno>
            <dni>11111A</dni>
            <nombre>Marta Aguirre</nombre>
            <telefono>986141414</telefono>
      </DatosAlumno>
       <DatosAlumno>
            <dni>222222B</dni>
            <nombre>Ana Sánchez</nombre>
             <telefono>627323232</telefono>
      </DatosAlumno>
      <DatosAlumno>
            <dni>333333C</dni>
            <nombre>Pedro García</nombre>
            <telefono>615545454</telefono>
      </DatosAlumno>
</Listado Alumnos>
```

En primer lugar para utilizar ***XStream*** simplemente creamos una instancia de la clase ***XStream***:

```
XStream xstream = new XStream();
```

En general las etiquetas XML se corresponden con el nombre de los atributos de la clase, pero pueden cambiarse utilizando el método ***alias***(). En el ejemplo se ha dado un alias a la clase ListaAlumnos que en el fichero XML aparecerá con el nombre Listado Alumnos.

```
xstream.alias("ListadoAlumnos", ListaAlumnos.class);
```

También se ha dado un alias a la clase *Alumno*, en el XML aparecerá con el nombre *DatosAlumno*

```
xstream.alias("DatosAlumno", Alumno.class);
```

Para que no aparezca el atributo *lista* de la clase *ListaAlumnos* en el XML generado se utiliza el método ***addImplicitCollection();***

```
xstream.addImplicitCollection(ListaAlumnos.class, "lista");
```

Por último, para generar el fichero *Alumnos.xml* a partir de la lista de objetos se utiliza el método ***toXML(objeto, OutputStream):***

```
xstream.toXML(listaalu, new FileOutputStream("Alumnos.xml"));
```

El proceso para leer del fichero XML generado es el siguiente:

```
package ejemplos12SerializacionObjetosXML01;
/*
 * Lectura de un fichero XML a objetos
 */
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import com.thoughtworks.xstream.XStream;

public class LeerFicheroXML {
	public static void main(final String[] args) throws
FileNotFoundException {

		//crear una instancia de la clase XStream
		XStream xstream = new XStream();
		//cambiar de nombre a las etiquetas XML
		xstream.alias("ListadoAlumnos", ListaAlumnos.class);
		xstream.alias("DatosAlumno", Alumno.class);
		//quitar etiqueta lista (atributo de la clase ListaAlumnos
		xstream.addImplicitCollection(ListaAlumnos.class, "lista");
		ListaAlumnos listadoTodas = (ListaAlumnos)
				xstream.fromXML(new
FileInputStream("Alumnos.xml"));
		System.out.println("Número de alumnos: "
+listadoTodas.getListaAlumno());
		List<Alumno> listaAlumnos = new ArrayList<Alumno>();
		listaAlumnos = listadoTodas.getListaAlumno();
		Iterator iterator = listaAlumnos.listIterator(); //recorrer los
elementos
		while(iterator.hasNext()){
```

```
                Alumno alu = (Alumno) iterator.next(); //obtenemos el
elemento
                System.out.println("DNI: "+alu.getDni() +"\tNombre: "
+alu.getNombre() +"\tTeléfono: "
                                +alu.getTelefono());
            }// fin del while
            System.out.println("\n\nFin del listado... ");
      }// fin main
}//fin clase
```

Se deben de utilizar los métodos ***alias***() y ***addImplicitCollection()*** para leer el XML ya que se utilizaron para hacer la escritura del mismo. Para obtener el objeto con la lista de personas o lo que es lo mismo para deserializar el objeto utilizamos el método ***fromXML(InputStream)***.

```
ListaAlumnos listadoTodas = (ListaAlumnos)
xstream.fromXML(new FileInputStream("Alumnos.xml"));
```

Api XStream:
http://xstream.codehaus.org/javadoc/com/thoughtworks/xstream/XStream.html

Conversión de Ficheros XML a otro formato

XSL (*Extensible Stylesheet Language*) son recomendaciones del Word Wide Web Consortium (http://www.w3.org/Style/XSL/) para expresar hojas de estilo en lenguaje XML. Una hoja de estilo **XSL** describe el proceso de presentación a través de un pequeño conjunto de elementos XML. Esta hoja puede contener elementos de reglas que representan a las reglas de construcción y elementos de reglas de estilo que representan a las reglas de mezclas de estilos.

En el ejemplo vamos a ver como a partir de un fichero XML que contiene datos y otro XSL que contiene la presentación de esos datos se puede generar un fichero HTML usando el lenguaje Java.

LIBROS XML

```
<?xml version="1.0" ?>
<ListaLibrosBiblioteca>
  <DatosLibro>
    <titulo>El Quijote</titulo>
    <autor>Cervantes</autor>
    <precio>32</precio>
  </DatosLibro>
  <DatosLibro>
    <titulo>Patria</titulo>
    <autor>Fernando Aranjuren</autor>
    <precio>27</precio>
  </DatosLibro>
  <DatosLibro>
    <titulo>Los documentos electronicos</titulo>
    <autor>Jorde Serra Serra</autor>
    <precio>15</precio>
  </DatosLibro>
</ListaLibrosBiblioteca>
```

LIBROSPLANTILLA.XLS

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
	xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
	<xsl:template match="/">
		<html> <xsl:apply-templates/> </html>
	</xsl:template>
	<xsl:template match="ListaLibrosBiblioteca">
		<head><title>LISTADO DE LIBROS</title></head>
		<body>
			<h1>LISTADO DE LIBROS</h1>
			<table border="1" width="50%">
				<tr><th>Título</th><th>Autor</th><th>Precio</th></tr>
				<xsl:apply-templates select='DatosLibro'/>
			</table>
		</body>
	</xsl:template>
	<xsl:template match='DatosLibro'>
		<tr><xsl:apply-templates /></tr>
	</xsl:template>
	<xsl:template match='titulo|autor|precio'>
		<td><xsl:apply-templates /></td>
	</xsl:template>
</xsl:stylesheet>
```

Para realizar la transformación se necesita obtener un objeto ***Transformer*** que se obtiene creando una instancia de ***TransformerFactory*** y aplicando el método ***newTransformer*** a la fuente XSL que vamos a utilizar para aplicar la transformación del fichero de datos XML, o lo que es lo mismo para aplicar la hoja de estilos XSL al fichero XML:

```
Transformer transformer =
TransformerFactory.newInstance().newTransformer(estilos);
```

La transformación se consigue llamando al método transform(), pasándole los datos (fichero XML) y el stream de salida (el fichero HTML)

```
transformer.transform(datos, resultado);
```

El programa completo podría ser:

```
package ej09ConervsionFicherosXML;

import org.w3c.dom.*;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import javax.xml.transform.Result;
import javax.xml.transform.Source;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.stream.StreamResult;
import javax.xml.transform.stream.StreamSource;
public class Ej01ConvertirFicherosXML {
```

```
	public static void main(String[] args) {
		String hojaEstilo = "librosPlantilla.xsl";
		String datosLibros = "Libros.xml";

		File paginaHTML = new File("mipagina.html");
		FileOutputStream foi = null;

		// fuentes XSL "librosPlantilla.xsl";
		Source estilos = null;
		// fuente XML "Libros.xml";
		Source datos = null;
		//resultado de la transformacion
		Result resultado = null;

		// hacer la transformacion
		Transformer transformer = null;
		try {
			//crea el fichero HTML "mipagina.html"
			foi = new FileOutputStream(paginaHTML);

			// fuentes XSL "librosPlantilla.xsl";
			estilos = new StreamSource(hojaEstilo);

			// fuente XML "Libros.xml";
			datos = new StreamSource(datosLibros);

			//resultado de la transformacion
			resultado = new StreamResult(foi);

			//Transformamos el resultado
			transformer =
TransformerFactory.newInstance().newTransformer(estilos);
			//obtiene el HTML
			transformer.transform(datos, resultado);
		} catch (FileNotFoundException e) {
					e.printStackTrace();
		}catch(Exception e){
			e.printStackTrace();
		}
		try {
			foi.close();
		} catch (IOException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
	}

}
```

El fichero HTML generado es el siguiente: