

Acceso a ficheros XML con DOM

Introducción

- XML es un metalenguaje. Es decir, define lenguajes. Nos permite jerarquizar y estructurar la información.
- Los ficheros XML se pueden utilizar para proporcionar datos a una aplicación. Tienen la gran ventaja de ser archivos de texto lo que permite exportar y recuperar datos de una plataforma a otra (solo es necesario conocer la codificación de los caracteres)
- Para leer los ficheros XML se utiliza un procesador XML o “parser”. Los procesadores más utilizados son: DOM y SAX.
- Los procesadores son independientes del lenguaje de programación y existen versiones particulares para Java, VisualBasic, C, etc.

Sobre DOM

- DOM: Modelo de Objetos de Documento
- Se trata de una API para manipulación de documentos XML y HTML
- Almacena toda la estructura de un documento en memoria en forma de árbol; con nodos padre, nodos hijo y nodos finales (que no tienen descendientes)
- Requiere una buena cantidad de recursos de memoria y tiempo sobre todo si los ficheros XML a procesar son bastante grandes y complejos

Acceso a ficheros con DOM

- Para poder trabajar con DOM en Java necesitamos las clases e interfaces que componen el paquete **org.w3c.dom** y el paquete **javax.xml.parsers**.
- Las dos clases más importantes son **DocumentBuilderFactory** y **DocumentBuilder**.
- Los programas Java que utilicen DOM necesitan estas interfaces (entre otras):
 - **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
 - **NodeList** → Contiene una lista con los nodos hijos de un nodo
 - **Attr** → Permite acceder a los atributos de un nodo.
 - **Text** → Son los datos carácter de un elemento

/* A continuación vamos a estudiar un programa Java que va a crear un fichero XML a partir del fichero aleatorio "AleatorioEmpleado.dat" que ya habíamos creado anteriormente. El fichero "AleatorioEmpleado.dat" contiene un registro por cada empleado en el que se utilizan 36 bytes para almacenar el id del empleado, su apellido, su departamento y su salario. */

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

Paquetes necesarios. Observad que la jerarquía de nombres no corresponde con la jerarquía de clases.

```
public class CrearEmpleadoXml {
    public static void main (String args[]) throws IOException {
        File fichero = new File ("AleatorioEmpleado.dat");
        RandomAccessFile file = new RandomAccessFile(fichero, "r");
        int id, dep, posicion=0;
        Double salario;
        char apellido[] = new char[10], aux;
```

Cabecera de la clase e instancia del RandomAccessFile. También se definen las variables que recogerán los valores leídos.

/ Aquí estamos creando una instancia de DocumentBuilderFactory para construir el parser. Se debe encerrar entre try-catch porque se puede producir la excepción ParserConfigurationException*/*

```
DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();  
    try {  
        DocumentBuilder builder = factory.newDocumentBuilder();
```

*/*Creamos un documento vacío de nombre *document* con el nodo raíz de nombre *Empleados* y asignamos la versión XML. La interfaz DOMImplementation permite crear objetos Document con nodo raíz**

```
DOMImplementation implementation = builder.getDOMImplementation();  
Document document = implementation.createDocument (null,"Empleados", null);  
document.setXmlVersion("1.0");
```

```
for ( ; ;){  
    file.seek(posicion);  
    id = file.readInt();  
    for (int i = 0; i < apellido.length ; i++) {  
        aux =file.readChar();  
        apellido[i] = aux;  
    }  
    String apellidos = new String (apellido);  
    dep = file.readInt();  
    salario = file.readDouble();
```

Esto ya lo hemos visto. No es más que la lectura byte a byte del archivo “aleatorio” dentro de un bucle “infinito”

/* A continuación, por cada registro en el fichero AleatorioEmpleado.dat, debemos crear un nodo empleado que, a su vez, tendrá 4 nodos-hijo. Cada nodo hijo tendrá su valor (por ejemplo: 1, FERNANDEZ, 10, 1000.45). 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. */

```
if (id>0) {  
    Element raiz = document.createElement ("empleado");  
    document.getDocumentElement().appendChild(raiz);  
    CrearElemento ("id", Integer.toString(id), raiz, document);  
    CrearElemento ("apellido",apellidos.trim(), raiz, document);  
    CrearElemento ("dep", Integer.toString(dep), raiz, document);  
    CrearElemento ("salario", Double.toString(salario),raiz, document);  
}
```

CrearElemento es un método que genera los nodos-hijo (*id*, *apellido*, *dep* o *salario*). Recibe como parámetros sus textos o valores que deben estar en formato String, el nodo al que se va añadir (*raiz*) y el documento (*document*). Luego lo vemos con más detalle.


```
posicion = posicion + 36;  
if (file.getFilePointer() == file.length()) break;  
}
```

Aquí acaba el bucle infinito que hemos iniciado antes. Previamente se comprueba que no hayamos llegado al final del archivo "aleatorio".

/*La especificación 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.

Primero crearemos la fuente (**Source**) XML a partir del documento (**document**); después se crea el resultado (**Result**) en el fichero *Empleados.xml*. A continuación, se obtiene un **TransformerFactory** y se realiza la transformación del documento al fichero */

```
Source source = new DOMSource (document);
Result result = new StreamResult (new java.io.File ("Empleados.xml"));
Transformer transformer = TransformerFactory.newInstance().newTransformer();
transformer.transform (source, result);
} catch (Exception e ) { System.err.println ("Error: " + e);}
file.close();
}
```



Cerramos el try-catch y el fichero AleatorioEmpleado.dat. Aquí acaba el *main*

/* Faltará comentar el método CrearElemento. Este método, como hemos visto anteriormente, genera los nodos-hijo de cada nodo <Empleado>.

Este método es llamado 4 veces por cada Empleado. Genera los nodos <id>, <apellido>, <dep> y <salario>. Primero genera el elemento (**Element**), después el texto o valor (**Text**) y los asocia con la raíz o nodo padre */

```
static void CrearElemento (String datoEmpleado, String valor, Element raiz, Document document) {  
    Element elem = document.createElement (datoEmpleado);  
    Text text = document.createTextNode(valor);  
    raiz.appendChild (elem);  
    elem.appendChild (text);  
}  
}
```

A continuación, el código completo:

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

```
public class CrearEmpleadoXml {
    public static void main (String args[]) throws IOException {
        File fichero = new File ("AleatorioEmpleado.dat");
        RandomAccessFile file = new RandomAccessFile(fichero, "r");
        int id, dep, posicion=0;
        Double salario;
        char apellido[] = new char[10], aux;

        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
        try {
            DocumentBuilder builder = factory.newDocumentBuilder();
            DOMImplementation implementation = builder.getDOMImplementation();
            Document document = implementation.createDocument (null,"Empleados", null);
            document.setXmlVersion("1.0");
            for ( ; ){
                file.seek(posicion);
                id = file.readInt();
```

```

        for (int i = 0; i < apellido.length ; i++) {
            aux =file.readChar();
            apellido[i] = aux;
        }
        String apellidos = new String (apellido);
        dep = file.readInt();
        salario = file.readDouble();
        if (id>0) {
            Element raiz = document.createElement ("empleado");
            document.getDocumentElement().appendChild(raiz);
            CrearElemento ("id", Integer.toString(id), raiz, document);
            CrearElemento ("apellido",apellidos.trim(), raiz, document);
            CrearElemento ("dep", Integer.toString(dep), raiz, document);
            CrearElemento ("salario", Double.toString(salario),raiz, document);
        }
        posicion = posicion + 36;
        if (file.getFilePointer() == file.length()) break;
    }
    Source source = new DOMSource (document);
    Result result = new StreamResult (new java.io.File ("Empleados.xml"));
    Transformer transformer = TransformerFactory.newInstance().newTransformer();
    transformer.transform (source, result);
} catch (Exception e ) { System.err.println ("Error: " + e);}
file.close();
}

```

```
static void CrearElemento (String datoEmpleado, String valor, Element raiz, Document document) {  
    Element elem = document.createElement (datoEmpleado);  
    Text text = document.createTextNode(valor);  
    raiz.appendChild (elem);  
    elem.appendChild (text);  
}  
}
```

Problemas AccesoDOM

1. Copia el programa anterior y ejecútalo
2. Vas a realizar exactamente lo mismo que en el programa anterior pero en vez de utilizar un fichero de acceso aleatorio (como era el caso de AleatorioEmpleado.dat) vas a utilizar un fichero de texto con los datos de los empleados. Es decir:
 - a. Generar un archivo de texto que contendrá la información de cada empleado. Cada línea del archivo se asociará al id, apellido, departamento y salario de cada empleado. Cada uno de estos campos irá separado por el carácter “:” (Ejemplo: 1:Fernandez:10:1000.45). Lllamarás al archivo “Empleados.txt”
 - b. Generar el programa Java que genere el archivo XML correspondiente a los datos contenidos en Empleados.txt

AYUDA: la dinámica es exactamente la misma que en el ejemplo que hemos visto pero, en vez de utilizar un RandomAccessFile, hay que utilizar un FileReader o, mejor, un BufferedReader. Para obtener los datos se puede utilizar el método String.split(...)

Lectura archivos XML con DOM

- En primer lugar creamos una instancia de **DocumentBuilderFactory** para poder construir el parser y cargamos el documento con el método **parse()**
- A continuación, obtenemos la lista de nodos (**NodeList**) con nombre *empleado* de todo el documento. Para ello utilizaremos el método **Document.getElementsByTagName (“empleado”)**.
- Por último, se realiza un bucle para recorrer esa lista de nodos. Por cada nodo se obtienen sus etiquetas y sus valores.
- Veamos el código:

```

import java.io.File;
import javax.xml.parsers.*;
import org.w3c.dom.*;
public class LecturaEmpleadoXml {
    public static void main (String[] args) {
        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
        try {
            DocumentBuilder builder = factory.newDocumentBuilder();
            Document document = builder.parse(new File ("Empleados.xml"));
            System.out.printf ("Elemento raíz : %s %n", document.getDocumentElement().getNodeName());
            NodeList empleados = document.getElementsByTagName("empleado");
            System.out.printf ("Nodos empleado a recorrer: %d %n", empleados.getLength());
            for (int i = 0; i < empleados.getLength(); i++) {
                Node emple = empleados.item(i);
                if (emple.getNodeType() == Node.ELEMENT_NODE){
                    Element elemento = (Element) emple;
                    System.out.printf("ID = %s %n", elemento.getElementsByTagName("id").item(0).getTextContent());
                    System.out.printf(" * Apellido = %s %n",
                        elemento.getElementsByTagName("apellido").item(0).getTextContent());
                    System.out.printf(" * Departamento = %s %n",
                        elemento.getElementsByTagName("dep").item(0).getTextContent());
                    System.out.printf(" * Salario = %s %n",
                        elemento.getElementsByTagName("salario").item(0).getTextContent());
                }
            }
        } catch (Exception e) {e.printStackTrace();}
    }
}

```

Problemas AccesoDOM

3. Copia el programa anterior y ejecútalo.
4. Crea un documento XML que contenga varios elementos como el siguiente:

```
<?xml version="1.0" encoding="UTF-8" ?>
<album>
  <autor>Joaquín Sabina</autor>
  <titulo>Nos sobran los motivos</titulo>
  <formato>MP3</formato>
</album>
```

5. Crea un programa Java que lea el documento anterior y muestre toda la información que contenga.