

---

---

# Accés a fitxers XML amb DOM

Fluxes a fitxers

---

---

# Introducció

- Els fitxers XML es poden utilitzar per a proporcionar dades a una aplicació. Tenen el gran avantatge de ser arxius de text pla, el que permet exportar i recuperar dades d'una plataforma a una altra (només és necessari conèixer la codificació dels caràcters).
- Per llegir els fitxers XML s'utilitza un procesador XML o "parser". Els processadors més utilitzats són: **DOM** i SAX.
- Els processadors són independents del llenguatge de programació i existeixen versions particulars per a Java, Visual Basic / .NET, C, etc.

# Introducció

- DOM: Document Object Model.
- Es tracta d'una **API** per la manipulació de documents XML i HTML.
- Emmagatzema tota l'estructura d'un document en memòria en forma d'arbre; amb nodes pare, nodes fill i nodes finals (que no tenen descendents).
- Requereix una bona quantitat de recursos de memòria i temps sobretot si els fitxers XML a processar són bastant grans i complexos.

# Accés a fitxers amb DOM

- Per poder treballar amb DOM a Java necessitem les classes i interfícies que componen el [paquet org.w3c.dom](http://org.w3c.dom) i el paquet [javax.xml.parsers](http://javax.xml.parsers).
- Les dues classes més importants són [DocumentBuilderFactory](#) i [DocumentBuilder](#).
- Els programes Java que utilitzin DOM necessiten aquestes interfícies (entre altres):
  - **Document** → És un objecte que equival a un exemplar d'un document XML. Permet crear nous nodes en el document.
  - **Element** → Cada element del document XML té un equivalent en un objecte d'aquest tipus. Exposa propietats i mètodes per manipular els elements del document i els seus atributs.
  - **NodeList** → Conté una llista amb els nodes fills d'un node.
  - **Attr** → Permet accedir als atributs d'un node.
  - **Text** → Són les dades caràcter d'un element.

# Exemple



A continuació estudiarem un programa Java que crearà un fitxer XML a partir del fitxer aleatori "EmpleatAleatori.dat" que ja havíem creat anteriorment. El fitxer "EmpleatAleatori.dat" conté un registre per cada empleat en el qual s'utilitzen 36 bytes per a emmagatzemar les dades de l'empleat, el seu cognom, el seu departament i el seu salari.

```
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.*;
```

Paquets necessaris. Observeu que la jerarquia de noms no correspon amb la jerarquia de classes.

```
public class CrearEmpleatXml {
    public static void main (String args[]) throws IOException {
        File fitxer = new File ("EmpleatAleatori.dat");
        RandomAccessFile file = new RandomAccessFile(fitxer, "r");
        int id, dep, posicio=0;
        Double salari;
        char cognom[] = new char[10], aux;
```

Capçalera de la classe i instància del RandomAccessFile. També es defineixen les variables que recolliran els valors llegits.

# Exemple




Aquí a sota estem creant una instància de `DocumentBuilderFactory` per construir el `parser`. S'ha de tancar entre `try-catch` perquè es pot produir l'excepció `ParserConfigurationException`.

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

Creem un document buit de nom `document` amb el node arrel de nom `Empleats` i assignem la versió XML. La interfície `DOMImplementation` permet crear objectes `Document` amb node arrel.

```
DOMImplementation implementation = builder.getDOMImplementation();  
Document document = implementation.createDocument (null, "Empleats", null);  
document.setXmlVersion("1.0");  
for ( ; ; ){  
    file.seek(posicio); id = file.readInt();  
    for (int i = 0; i < cognom.length ; i++) {  
        aux =file.readChar();  
        cognom[i] = aux;  
    }  
  
    String cognoms = new String (cognom); dep = file.readInt();  
    salari = file.readDouble();
```



Això ja ho hem vist. No és més que la lectura byte a byte de l'arxiu dins d'un bucle "infinit".

# Exemple



A continuació, per a cada registre del fitxer, hem de crear un node Empleat que, al seu torn, tindrà 4 nodes-fill. Cada node fill tindrà el seu valor (per exemple: 1, FERNANDEZ, 10, 1000.45). Per a crear un element usem el mètode `createElement(String)` portant com a paràmetre el nom que es posa entre les etiquetes menor que i major que.

```
if (id>0) {  
    Element arrel = document.createElement("empleat");  
    document.getDocumentElement().appendChild(arrel);  
    CrearElement("id", Integer.toString(id), arrel, document);  
    CrearElement("cognom", cognoms.trim(), arrel, document);  
    CrearElement("dep", Integer.toString(dep), arrel, document);  
    CrearElement("salari", Double.toString(salari), arrel, document);  
}
```

`CrearElement` és un mètode que genera els nodes fill (id, cognom, dep i salari). Rep com a paràmetres els seus textos o valors que han d'estar en format `String`, el node al qual es va afegir (`arrel`) i el document (`document`). Després ho veiem amb més detall.

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

Aquí acaba el bucle infinit que hem iniciat abans. Prèviament es comprova que no hàgim arribat al final de l'arxiu.

# Exemple



L'especificació **DOM** no defineix cap mecanisme per generar un fitxer XML a partir d'un arbre DOM. Per a això usarem el paquet `javax.xml.transform` que permet especificar una font i un resultat. La font i el resultat poden ser fitxers, fluxos de dades o nodes **DOM** entre altres.

Primer crearem la font (**Source**) XML a partir del document (**document**); després crearem el resultat (**Result**) en el fitxer **Empleats.xml**. A continuació, s'obté un **TransformerFactory** i es realitza la transformació del document al fitxer.

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

Tanquem el try-catch i el fitxer .dat. Aquí acaba el main.



# Exemple



Faltarà comentar el mètode **CrearElement**. Aquest mètode, com hem vist anteriorment, genera els nodes-fill de cada node *<Empleat>*.

Aquest mètode és cridat 4 cops per cada *Empleat*. Genera els nodes *<id>*, *<cognom>*, *<dep>* i *<salari>*. Primer genera l'element **(Element)**, després el text o valor **(Text)** i els associa amb l'arrel o node pare.

```
static void CrearElement (String dadaEmpleat, String valor, Element arrel, Document document) {  
    Element elem = document.createElement (dadaEmpleat);  
    Text text = document.createTextNode(valor);  
    arrel.appendChild (elem);  
    elem.appendChild (text);  
}
```

Tot seguit trobareu el codi complet.

# Exemple



```
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 CrearEmpleatXml {
    public static void main (String args[]) throws IOException {
        File fitxer = new File ("EmpleatAleatori.dat");
        RandomAccessFile file = new RandomAccessFile(fitxer, "r");
        int id, dep, posicio=0;
        Double salari;
        char cognom[] = new char[10], aux;

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

# Exemple



```
DOMImplementation implementation = builder.getDOMImplementation();  
  
    Document document = implementation.createDocument (null, "Empleats", null);  
    document.setXmlVersion("1.0");  
  
    for ( ; ; ){  
        file.seek(posicio);  
        id = file.readInt();  
        for (int i = 0; i < cognom.length ; i++) {  
            aux =file.readChar();  
            cognom[i] = aux;  
        }  
        String cognoms = new String (cognom);  
        dep = file.readInt();  
        salari = file.readDouble();
```

# Exemple



```
if (id>0) {  
    Element arrel = document.createElement ("empleat");  
    document.getDocumentElement().appendChild(arrel);  
    CrearElement ("id", Integer.toString(id), arrel, document);  
    CrearElement ("cognom", cognoms.trim(), arrel, document);  
    CrearElement ("dep", Integer.toString(dep), arrel, document);  
    CrearElement ("salari", Double.toString(salari), arrel, document);  
}
```

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

```
Source source = new DOMSource (document);  
Result result = new StreamResult (new java.io.File ("Empleats.xml"));  
Transformer transformer = TransformerFactory.newInstance().newTransformer();  
transformer.transform (source, result);  
}
```

# Exemple



```
catch (Exception e ) { System.err.println ("Error: " + e);}  
    file.close();  
}
```

```
static void CrearElement (String dadaEmpleat, String valor, Element arrel, Document document) {  
    Element elem = document.createElement (dadaEmpleat);  
    Text text = document.createTextNode(valor);  
    arrel.appendChild (elem);  
    elem.appendChild (text);  
}  
}
```

Copia el programa anterior i executa'l.

**A1.-** Realitzar exactament el mateix que en el programa anterior però en comptes d'utilitzar un fitxer d'accés aleatori (com era el cas de `EmpleatAleatori.dat`) utilitzar un fitxer de text amb les dades dels empleats. És a dir:

**a.** Crear manualment (o a través d'un programa Java) un arxiu de text que continga la informació de cada empleat. Cada línia de l'arxiu s'associarà al id, cognom, departament i salari de cada empleat. Cadascun d'aquests camps anirà separat pel caràcter ":" (Exemple:

1:`Fernandez`:10:1000.45). Anomenar al nou arxiu "Empleats.txt"

**b.** Generar el programa Java que crei l'arxiu XML corresponent a les dades contingudes en `Empleats.txt`

**AJUDA:** la dinàmica és exactament la mateixa que en l'exemple que hem vist però, en comptes de utilitzar un `RandomAccessFile`, cal utilitzar un `FileReader` o, millor, un `BufferedReader`. Per obtenir les dades es pot utilitzar el mètode `String.split(...)`.

# Lectura de fitxers XML amb DOM



- En primer lloc creem una instància de `DocumentBuilderFactory` per poder construir el `parser` i carreguem el document amb el mètode `parse()`.
- A continuació, obtenim la llista de nodes (`NodeList`) amb nom empleat de tot el document. Per a això utilitzarem el mètode `Document.getElementsByTagName("empleat")`.
- Finalment, es realitza un bucle per recórrer aquesta llista de nodes. Per cada node s'obtenen les seves etiquetes i els seus valors.
- Vegem el codi:

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

public class LecturaEmpleatXml {

    public static void main (String[] args) {

        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();

        try {

            DocumentBuilder builder = factory.newDocumentBuilder();

            Document document = builder.parse(new File ("Empleats.xml"));

            System.out.printf ("Element arrel : %s %n", document.getDocumentElement().getNodeName());

            NodeList empleats = document.getElementsByTagName("empleats");

            System.out.printf ("Nodes empleats a recorrer: %d %n", empleats.getLength());

            for (int i = 0; i < empleats.getLength(); i++) {

                Node emple = empleats.item(i);

                if (emple.getNodeType() == Node.ELEMENT_NODE) {

                    Element element = (Element) emple;

                    System.out.printf("ID = %s %n", element.getElementsByTagName("id").item(0).getTextContent());

                    System.out.printf(" * Cognom = %s %n",
element.getElementsByTagName("cognom").item(0).getTextContent());

                    System.out.printf(" * Departament = %s %n",
element.getElementsByTagName("dep").item(0).getTextContent());

                    System.out.printf(" * Salari = %s %n",
element.getElementsByTagName("salari").item(0).getTextContent());

                }

            }

        } catch (Exception e) {e.printStackTrace();}
```



Copia el programa anterior i executa'l.

**A2.-** Crea manualment un document XML que contingui **diversos** elements com el següent:

```
<?xml version="1.0" encoding="UTF-8" ?>
<album>
  <autor>Els pets</autor>
  <titol>Bon dia</titol>
  <format>MP3</format>
</album>
```

Crea un programa Java que llegeixi el document anterior i mostri per pantalla tota la informació que contingui.