Accés a fitxers XML amb SAX

Fluxes a fitxers

Consorci d'Educació de Barcelona
Generalitat de Catalunya
Ajuntament de Barcelona

Introducció

- SAX (API Simple per a XML) és un conjunt de classes i interfícies que ofereixen una eina molt útil pel processament de documents XML.
- Permet **analitzar documents** de manera **seqüencial** (és a dir, no carrega tot el fitxer en memòria com fa DOM), això implica:
  - Poc consum de memòria encara que els documents siguin de gran grandària.
  - Impedeix tenir una visió global del document.
  - L'accés és seqüencial NO aleatori.
  - SAX NO permet generar arxius XML.
- Per tant, SAX és útil quan volem cercar informació en fitxers grans XML o quan no tenim accés complet al document (accés mitjançant Stream).
- SAX és una API totalment escrita en Java i inclosa dins del JRE que ens permet crear el nostre propi parser de XML

Lectura del XML

La lectura d'un document XML produeix esdeveniments que ocasiona la crida a mètodes:

Document alumnes.xml	**Mètodes associats a esdeveniments del document**
<?xml version="1.0"?>	startDocument()
<llista_alumnes>	startElement()
<alumne>	startElement()
<nom>	startElement()
Joan	characters()
</nom>	endElement()
<edad>	startElement()

19	characters()
</edad>	endElement()
</alumne>	endElement()
<alumne>	startElement()
<nom>	startElement()
Isabel	characters()
</nom>	endElement()
<edad>	startElement()
20	characters()
</edad>	endElement()
</alumne>	endElement()
</llista_alumnes>	endElement()
	endDocument()

Lectura del XML

- Com s'observa en la taula:
  - L'etiqueta d'inici (<?xml versión = "1.0"?>) → provoca l'esdeveniment startDocument().
  - El final de document → provoca l'esdeveniment endDocument().
  - L'etiqueta d'inici d'un element → provoca l'esdeveniment startElement().
  - L'etiqueta de final d'un element → provoca l'esdeveniment endElement().
  - Els caràcters entre etiquetes → provoquen l'esdeveniment characters().

- Els objectes que posseeixen els mètodes que tractaran els esdeveniments són:
  - ContentHandler → rep les notificacions dels esdeveniments que ocorren en el document.
  - DTDHandler → recull esdeveniments relacionats amb un DTD.
  - ErrorHandler → defineix mètodes de tractaments d'errors.
  - EntityResolver → els seus mètodes es criden cada cop que es troba una referència a una entitat.
  - DefaultHandler → classe que proveeix una implementació per defecte per a tots els seus mètodes, el programador definirà els mètodes que seran utilitzats pel programa. Aquesta classe és la que estendrem per poder crear nostre parser de XML. En l'exemple que veurem a continuació, la classe es diu GestioContingut i es tracten només els esdeveniments bàsics: inici i fi de document, inici i fi d'etiqueta trobada, troba dades caràcter.

Exemple de lectura del XML

```
import java.io.*;
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 ProvaSax {
   public static void main (String [] args) throws FileNotFoundException, IOException, SAXException {
/* A continuació es crea un objecte processador XML-XMLReader-. Durant la creació d'aquest objecte es pot produir una excepció *SAXException. */
       XMLReader processadorXML = XMLReaderFactory.createXMLReader();
/* Tot seguit, mitjançant setContentHandler establim que la classe que gestiona els esdeveniments provocats per la lectura del XML serà GestioContingut */
       GestioContingut gestor = new GestioContingut();  processadorXML.setContentHandler(gestor);
/* Finalment, es defineix el fitxer que es va llegir mitjançant InputSource i es processa el document XML mitjançant el mètode parse() de XMLReader */
       InputSource fileXML = new InputSource ("alumnes.xml");  processadorXML.parse(fileXML);
   }
}
```

Exemple de lectura del XML

```
/* GestioContingut és la classe que implementa els mètodes necessaris per crear el nostre parser d'XML. És a dir,
definim els mètodes que seran cridats en provocar-se els esdeveniments comentats anteriorment: startDocument,
startElement, characters, etc. Si volguessim tractar més esdeveniments definiríem el mètode associat en aquesta classe.*/
class GestioContingut extends DefaultHandler {
   public GestioContingut(){
      super();              }
   public void startDocument(){
       System.out.println("Començament del document XML");     }
   public void endDocument(){
       System.out.println("Final del document XML");           }
   public void startElement (String uri, String nom, String nomC, Attributes atts) {
       System.out.printf("\tElement Inicial: %s %n", nom);     }
   public void endElement (String uri, String nom, String nomC){
       System.out.printf("\tElement final: %s %n",nom);        }
   public void characters(char[] ch, int inici, int longitud) throws SAXException {
       String car = new String (ch, inici, longitud);
       car = car.replaceAll("[\t\n]","");
       System.out.printf("\tCaràcters: %s %n", car);
   }
}
```

Activitat

Còpia l'exemple anterior i executa'l.

A1.- Crea un programa Java que llegeixi l'arxiu "discoteca.xml" (adjunt al moodle) i determini el número de discos llistats en l'arxiu. AJUDA: comptabilitzant el número de títols.

A2.- Crea un programa que retorni el número de discos registrats en "discoteca.xml" d'un determinat autor que li passem per la terminal. Si l'autor no té cap disc en l'arxiu, el programa retornarà un missatge de l'estil: "L'autor <xxxxxx> no apareix en l'arxiu.