Lectura de un archivo XML

Ejemplo de archivo XML que utilizaremos

```
<concesionario>
  <coche id="1">
    <marca>Renault</marca>
    <modelo>Megane</modelo>
    <cilindrada>1.5</cilindrada>
  </coche>

  <coche id="2">
    <marca>Seat</marca>
    <modelo>León</modelo>
    <cilindrada>1.6</cilindrada>
  </coche>

  <coche id="3">
    <marca>Suzuki</marca>
    <modelo>Vitara</modelo>
    <cilindrada>1.9</cilindrada>
  </coche>
</concesionario>
```

A la hora de leer un archivo XML a través de DOM debemos instanciar una serie de clases antes de poder tratar el fichero. Primero utilizaremos la conocida clase File para cargar nuestro fichero.

```
File file = new File("coches.xml");
```

Posteriormente y ya dentro de un try/catch (para tratar los excepciones) parsearemos el fichero con estas clases: DocumentBuilderFactory, DocumentBuilder y Document.

```
try {
  DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
  DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
  Document doc = dBuilder.parse(file);
}

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

Una vez hecho todo esto ya podremos leer el archivo coches.xml además de usar otros métodos como los ejemplos de aquí abajo.

`getDocumentElement()`	Accede al nodo raíz del documento
`normalize()`	Elimina nodos vacíos y combina adyacentes en caso de que los hubiera

```
// estos métodos podemos usarlos combinados para normalizar el archivo XML
doc.getDocumentElement().normalize();
```

Siguiendo dentro del try/catch podemos utilizar la clase NodeList para almacenar el elemento que le indicaremos como parámetro.

```
// almacenamos los nodos para luego mostrar la
// cantidad de ellos con el método getLength()
NodeList nList = doc.getElementsByTagName("coche");
System.out.println("Número de coches: " + nList.getLength());
```

Una vez tenemos almacenados los datos del nodo “coche” podemos leer su contenido teniendo en cuenta que este código depende de que conozcamos la estructura y etiquetas utilizadas.

```
for(int temp = 0; temp < nList.getLength(); temp++) {
  Node nNode = nList.item(temp);

  if(nNode.getNodeType() == Node.ELEMENT_NODE) {
      Element eElement = (Element) nNode;
      System.out.println("\nCoche id: " + eElement.getAttribute("id"));
      System.out.println("Marca: "
                +
      eElement.getElementsByTagName("marca").item(0).getTextContent());
      System.out.println("Modelo: "
                +
      eElement.getElementsByTagName("modelo").item(0).getTextContent());
      System.out.println("Cilindrada: "
                +
      eElement.getElementsByTagName("cilindrada").item(0).getTextContent());
  }
}
```

Nos mostraría por consola:

```
Elemento Raiz: concesionario
Número de coches: 3
Coche id: 1
Marca: Renault
Modelo: Megane
Cilindrada: 1.5
Coche id: 2
Marca: Seat
Modelo: León
Cilindrada: 1.6
Coche id: 3
Marca: Suzuki
Modelo: Vitara
Cilindrada: 1.9
```

Si no conocemos la estructura podemos utilizar el código de más abajo para leer el archivo XML.

```
NodeList nList = doc.getElementsByTagName("coche");

for (int i = 0; i < nList.getLength(); i++) {
  Node node = nList.item(i);

  if (node.getNodeType() == Node.ELEMENT_NODE) {
    Element eElement = (Element) node;

    if(eElement.hasChildNodes()) {
      NodeList nl = node.getChildNodes();
      for(int j=0; j<nl.getLength(); j++) {
        Node nd = nl.item(j);
        System.out.println(nd.getTextContent());
      }
    }
  }
}
```

Escritura archivo XML

Si quisiéramos escribir un archivo XML siguiendo la misma estructura del concesionario deberíamos instanciar las clases DocumentBuilderFactory, DocumentBuilder y Document, definir toda la estructura del archivo (siempre dentro de un bloque try/catch) y por último instanciar las clases TransformerFactory, Transformer, DOMSource y StreamResult para crear el archivo.

```
public class Coche {
  private String id;
  private String marca;
  private String modelo;
  private String cilindrada;

  public Coches() {
    marca = "Renault";
    modelo = "Megane";
    cilindrada = "1.5";
  }

  public Coches(String id, String nombre, String creador, String ciudad) {
    this.id = id;
    this.marca = nombre;
    this.modelo = creador;
    this.cilindrada = ciudad;
  }

  @Override
  public String toString() {
    return this.marca.replace(' ', '_') + " " +
           this.modelo.replace(' ', '_') + " " +
           this.cilindrada.replace(' ', '_');
  }
}
```

Partiendo de la clase Coche, podríamos escribir lo siguiente:

```
Coche coche = new Coche(“001”, "Seat", "León", "1.5");

try {
  DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
  DocumentBuilder db = dbf.newDocumentBuilder();
  Document doc = db.newDocument();

  // definimos el elemento raíz del documento
  Element eRaiz = doc.createElement("concesionario");
  doc.appendChild(eRaiz);

  // definimos el nodo que contendrá los elementos
  Element eCoche = doc.createElement("coche");
  eRaiz.appendChild(eCoche);

  // atributo para el nodo coche
  Attr attr = doc.createAttribute("id");
  attr.setValue(coche.getId());
  eCoche.setAttributeNode(attr);

  // definimos cada uno de los elementos y le asignamos un valor
  Element eMarca = doc.createElement("marca");
  eMarca.appendChild(doc.createTextNode(coche.getMarca()));
  eCoche.appendChild(eMarca);
```

```
  Element eModelo = doc.createElement("modelo");
  eModelo.appendChild(doc.createTextNode(coche.getModelo()));
  eCoche.appendChild(eModelo);

  Element eCilindrada = doc.createElement("cilindrada");
  eCilindrada.appendChild(doc.createTextNode("1.5"));
  eCoche.appendChild(eCilindrada);

  // clases necesarias finalizar la creación del archivo XML
  TransformerFactory transformerFactory = TransformerFactory.newInstance();
  Transformer transformer = transformerFactory.newTransformer();
  DOMSource source = new DOMSource(doc);
  StreamResult result = new StreamResult(new File("ejercicio3.xml"));

  transformer.transform(source, result);
} catch(Exception e) {
  e.printStackTrace();
}
```