Acceso a ficheros XML con DOM

- Se necesitan las clases e interfaces de los paquetes org.w3c.dom y javax.xml.parsers
- Además para generar ficheros XML a partir de un árbol DOM, se utilizará el paquete javax.xml.transform
- Algunas de las interfaces que se utilizan son:
  - Document. Es un objeto que representa un documento XML.
  - Element. Cada uno de los objetos que representa cada elemento del documento XML. Tiene propiedades y métodos para manipular los elementos del documento.
  - Node. Representa a cualquier nodo del documento.
  - NodeList. Contiene una lista de los nodos hijos de un nodo.
  - Attr. Permite acceder a los atributos de un nodo.
  - Text. Son los datos carácter de un elemento.
  - CharacterData. Representa a los datos carácter de un documento, con atributos y métodos para manipularlos.
  - DocumentType. Proporciona información de la etiqueta <!DOCTYPE>

Acceso a ficheros XML con DOM: escritura

- Para empezar a crear un fichero XML se suele partir de los datos almacenados en algún fichero, o en alguna colección de datos.
- Lo primero es construir el parser:

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

- Lo siguiente es crear un documento XML vacío:

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

Acceso a ficheros XML con DOM: escritura

- Ahora hay que ir creando los elementos anidados. Por cada registro, u objeto, o elemento similar, se crea un nodo, por ejemplo un empleado, y para cada empleado sus atributos:

```
Element raiz = document.createElement("empleado");
document.getDocumentElement().appendChild(raiz);

//se crea cada atributo del nodo
Element id = document.createElement("id");
Text textid = document.createTextNode(Integer.toString(emp.getId()));
raiz.appendChild(id); //se une el elemento con su padre
id.appendChild(textid); // se añade el valor del elemento

Element nombre = document.createElement("nombre");
Text textnombre = document.createTextNode(emp.getNombre());
raiz.appendChild(nombre);
nombre.appendChild(textnombre);
...
```

Acceso a ficheros XML con DOM: escritura

- Por último se genera el fichero XML, a partir del árbol DOM creado, utilizando transform, que permite especificar una fuente y un destino:

```
//se crea la fuente XML  partir del documento
Source source = new DOMSource(document);

//se crea el fichero de texto Empleados.xml
Result result = new StreamResult(new java.io.File("Empleados.xml"));

// se crea un TransformerFactory
Transformer transformer = TransformerFactory.newInstance().newTransformer();

//se transforma el árbol al fichero
transformer.transform(source, result);

//se visualiza el documento por pantalla
Result console= new StreamResult(System.out);
transformer.transform(source, console);
```

Acceso a ficheros XML con DOM: lectura

- Para leer un fichero XML se crea una instancia de *DocumentBuilderFactory* para construir el parser
- Se crea una instancia del documento, a partir de los datos del fichero XML, mediante el parser

```
public void leerDOM() throws ParseException {
DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
        try {
            //Pasamos un fichero  XML a un árbol DOM usando el DocumentBuilder
            DocumentBuilder builder = factory.newDocumentBuilder();

            //Se crea un Java DOM XML parser(analizador sintáctico)
            Document document = builder.parse("empleados.xml");
            document.getDocumentElement().normalize();

            //Se visualiza el nombre del elemento raíz
            System.out.println("Elemento raiz:" +
document.getDocumentElement().getNodeName());
```

Acceso a ficheros XML con DOM: lectura

- A partir de este momento se pueden cargar todos los elementos de un tipo determinado en una lista de nodos para después procesarlos. Por ejemplo:

```
NodeList listaempleados =
document.getElementsByTagName("empleado");
```

- Podemos ir sacando los elementos de esa lista y hacer el tipo de tratamiento que necesitemos, por ejemplo visualizarlos por pantalla:

Acceso a ficheros XML con DOM: lectura

```
for (int i = 0; i < listaempleados.getLength(); i++) {
      Node empleado = listaempleados.item(i); //obtiene un nodo
      if (empleado.getNodeType() == Node.ELEMENT_NODE) {
         Element elemento = (Element) empleado;
         System.out.println("Id: "+getNodo("id", elemento)));
         System.out.println("Nombre: "+ getNodo("nombre", elemento));
         System.out.println("Apell1: "+ getNodo("apell1", elemento);
         System.out.println("Apell2: "+ getNodo("apell2", elemento);
          System.out.println("Salario:"+ getNodo("salario", elemento);
             }
```

- Podemos ir sacando los elementos de esa lista y hacer el tipo de tratamiento que necesitemos, por ejemplo visualizarlos por pantalla:

Acceso a ficheros XML con DOM: lectura

- Para obtener la información de un nodo hemos creado el método:

```
private static String getNodo(String tag, Element elem) {
        NodeList nodo =
elem.getElementsByTagName(tag).item(0).getChildNodes();
        Node valornodo = (Node) nodo.item(0);
        return valornodo.getNodeValue();
    }
```