ESCRIBIR FICHEROS XML con DOM

1. Crear un DocumentFactory y un DocumentBuilder

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

2. Crear un Document a partir del builder. El document será nuestro XML

```
// Creo un document que será el xml
Document xml = builder.newDocument();
```

3. Crear el elemento root (el nodo raíz del XML) y añadirlo al document

```
// Creo el nodo raíz y lo añado al documento
Element root = xml.createElement("ejemplo");
xml.appendChild(root);
```

4. Crear los elementos (tags) que necesitemos

```
// Creo el resto de los nodos (Elementos)
Element nombre = xml.createElement("nombre");
Element dni = xml.createElement("dni");
```

5. Añadir los elementos a sus padres

```
// Anido unos elementos dentro de otros
root.appendChild(nombre);
root.appendChild(dni);
```

6. Añadir el texto a aquellos tags que lo tengan

```
nombre.setTextContent("Blas Blau Blau");
```

7. Añadir atributos a aquellos tags que lo tengan

```
// Pongo atributos a los nodo que los lleven
dni.setAttribute("tipo", "nacional");
```

8. Cuando tenga el XML terminado, tengo que exportarlo a un fichero. Lo hago usando un Transformer:

```
// Cuando termino el XML, lo quiero guardar en un fichero
TransformerFactory transformerFactory = TransformerFactory.newInstance();
Transformer transformer = transformerFactory.newTransformer();
DOMSource source = new DOMSource(xml);
StreamResult result = new StreamResult(new File("c:/temporal/ejemplo.xml"));
transformer.transform(source, result);
```

LEER FICHEROS XML con DOM

1. Crear un DocumentFactory y un DocumentBuilder

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

1. Creamos un Document a partir un File

```
// Creo un document que será el xml parseando el fichero donde está ese XML
Document xml = builder.parse(new File("c:/temporal/ejemplo.xml"));
```

2. Obtenemos el elemento root del xml

```
// Obtengo el nodo raíz
Element root = xml.getDocumentElement();
```

3. Vamos obteniendo el resto de elementos (tags) buscando por su nombre

```
Element nombre = (Element) root.getElementsByTagName("nombre").item(0);
System.out.println(nombre.getTextContent());
```

4. Si nos encontramos con más de uno porque es una lista, podemos recorrer la lista

```
NodeList listaDirecciones = direcciones.getElementsByTagName("direccion");
for (int i = 0; i < listaDirecciones.getLength(); i++) {
    Element direccion = (Element) listaDirecciones.item(i);
    Element id = (Element) direccion.getElementsByTagName("id").item(0);
    Element descripcion = (Element) direccion.getElementsByTagName("descripcion").item(0);
    System.out.println(id.getTextContent() + " - " + descripcion.getTextContent());
}
```

5. Obtenemos los atributos

```
System.out.println(dni.getAttribute("tipo"));
```

6. Si no sabemos cuántos atributos hay, podemos recorrerlo todos

```
NamedNodeMap atributos = dni.getAttributes();
for (int i = 0; i < atributos.getLength(); i++) {
    Node atributo = atributos.item(i);
    System.out.println(atributo.getNodeName() + ": " + atributo.getNodeValue());
}
```

7. Para usar los métodos getChildNodes, getFirstChild, getLastChild, getNextSibling debemos asegurarnos de que el nodo que vamos a tratar es un Element

```
// Al trabajar con nodos puede que no sean Element. Debemos asegurarnos antes.
NodeList nodos = direcciones.getChildNodes();
for (int i = 0; i < nodos.getLength(); i++) {
    Node nodo = nodos.item(i);
    if (nodo.getNodeType() != Node.ELEMENT_NODE) {
        continue;
    }
    Element direccion = (Element) nodo;
    Element id = (Element) direccion.getElementsByTagName("id").item(0);
    Element descripcion = (Element) direccion.getElementsByTagName("descripcion").item(0);
    System.out.println(id.getTextContent() + " - " + descripcion.getTextContent());
}
```

LEER FICHEROS XML con SAX

1. Crear un Handler. Será una clase que extienda de DefaultHandler. Podremos sobrescribir estos métodos:

    a. startDocument y endDocument → se llaman cuando el xml empieza o termina.

    b. startElement → se llama cuando empieza un nuevo tag. En el parámetro qName se indica el nombre del tag que acaba de comenzar. En el parámetro attributes vendrían los atributos de ese nodo.

    c. endElement → se llama cuando termina un tag. En el atributo qName se indica el nombre del tag que acaba de terminar.

    d. characters → se llama cada vez que se procesa texto (incluido el texto de los propios tags). Este texto estaría contenido en el array que se pasa por parámetro como buffer empezando en start y con un tamaño de length.

2. Si necesitamos leer el texto contenido en los tags, de algún modo habrá que poner una alerta para saber en qué tag estoy para guardar ese texto, o saber si me interesa.

3. Una vez creado el Handler, para usarlo, primero tendremos que crear un SaxFactory y un SaxParser

```
SAXParserFactory factory = SAXParserFactory.newInstance();
SAXParser saxParser = factory.newSAXParser();
```

4. Luego instanciamos nuestro Handler

```
XMLHandler handler = new XMLHandler();
```

5. Por último, invocamos al método parse indicando el File que queremos procesar y nuestro handler

```
File file = new File("c:/temporal/ejemplo2.xml");
saxParser.parse(file, handler);
```

ESCRIBIR/LEER FICHEROS XML con JACKSON (y JSON)

Jackson es una librería que nos permite hacer una conversión automática entre un objeto y un XML. Esta conversión se llama serialización o marshall (sentido objeto → cadena) o deserialización o unmarshall (sentido cadena → objeto)

Existen muchas otras librerías que permiten hacer esto. Tanto para XML como para JSON. Algunas de las más conocidas son:

- Jackson (para json y xml)
- GSon (para json)
- Jaxb (para xml)

¿Cómo se usa Jackson para serializar y deserializar?

1. En ambos casos necesitamos un modelo que sea fiel reflejo al XML que vamos a leer o escribir. Por tanto, lo primero será crear las clases. Si no hay una correspondencia idéntica, tendremos que incluir anotaciones en el modelo. Las más usuales:
   - @JsonProperty → para cambiar el nombre del tag respecto al atributo
   - @JsonIgnore → para ignorar un atributo (que no aparezca en el XML)
   - @JacksonXmlRootElement → para cambiar el nombre del tag root del xml
   - @JacksonXmlElementWrapper → para indicar si las listas van en un tag padre.
   - @JacksonXmlProperty → para indicar si el atributo de la clase es un atributo del tag
   - @JacksonXmlText → nos permite indicar que el atributo de la clase irá como texto del tag principal, no en un nuevo tag
   - Ejemplos de otras anotaciones: https://www.baeldung.com/jackson-annotations

2. Incluimos la librería de Maven:

```xml
<dependency>
    <groupId>com.fasterxml.jackson.dataformat</groupId>
    <artifactId>jackson-dataformat-xml</artifactId>
    <version>2.11.1</version>
</dependency>
```

3. Nos creamos un objeto XMLMapper

```
XmlMapper xmlMapper = new XmlMapper();
```

4. Nos creamos un objeto File con el fichero que vamos a leer o escribir.

5. Para leer: Hacemos un readValue del mapper pasándole el file y la clase del modelo.

```
Asignatura asignatura = xmlMapper.readValue(xmlFile, Asignatura.class);
```

6. Para escribir: Hacemos un writeValue del mapper pasándole el file y el objeto.

```
xmlMapper.writeValue(new File("c:/temporal/asignatura-2.xml"), objeto);
```

¿Cómo se usa Jackson para serializar y deserializar JSON?

Si en lugar de XML queremos serializar o deserializar un JSON, basta con usar un ObjectMapper en lugar de XMLMapper