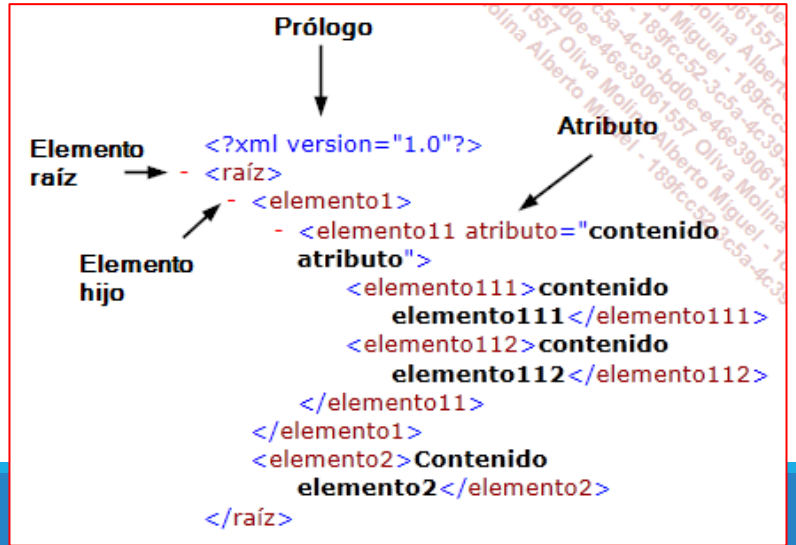
Acceso a Datos

UT2. MANEJO DE FICHEROS XML Y JSON
FICHEROS XML

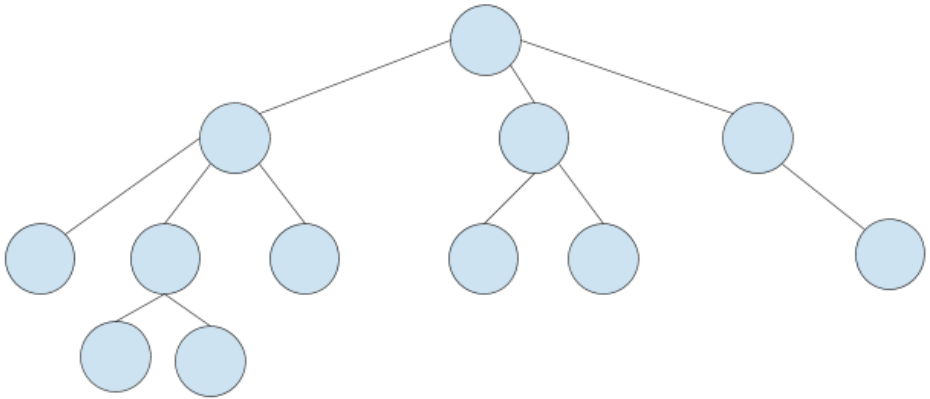
Trabajo con ficheros XML

- **XML** es un metalenguaje → Lenguaje para la definición de lenguajes de marcado.
- Nos permite jerarquizar y describir los contenidos dentro del propio documento.
- Los ficheros XML son ficheros donde la información se organiza forma secuencial y en orden jerárquico.
- Un fichero XML sencillo tiene la siguiente estructura:



Trabajo con ficheros XML

- En un documento XML la información se organiza de manera jerárquica, de manera que los elementos se relacionan entre sí mediante relaciones de padres, hijos, hermanos, ascendientes, descendientes, etc.



Trabajo con ficheros XML

- Los ficheros XML se pueden utilizar para:
  - Proporcionar datos a una BBDD
  - Almacenar información en BBDD especiales.
  - Ficheros de configuración
  - Intercambio de información en entornos web (SOAP)
  - Ejecución de comandos en servidores remotos.
  - Etc.

- Para realizar cualquiera de estas operaciones es necesario un lenguaje de programación que aporte funcionalidad a XML (XML no es un lenguaje Turing completo).
- Para acceder a los ficheros XML, leer su contenido o alterar su estructura se utiliza un **procesador de XML o XML parser.**
- Algunos de los parser más empleados son:
  - DOM
  - SAX

Trabajo con ficheros XML

Acceso a ficheros con DOM

- Los parser XML que trabajan con DOM almacenan toda la estructura jerarquica del documento en memoria principal.
- Es el estándar que definió W3C para trabajar con ficheros XML
- Este tipo de procesamiento necesita más recursos de memoria y tiempo que SAX.

- Para poder trabajar con DOM en Java vamos a hacer uso los paquetes
  - org.w3c.dom (Contenido en el JSDK)
  - javax.xml.parsers (Del API estándar de Java)
- Que proporcionan dos clases abstractas que debemos extender para trabajar con DOM.
  - **DocumentBuilderFactory**
  - **DocumentBuilder**

Trabajo con ficheros XML

Acceso a ficheros con DOM

- DOM no define ningún mecanismo para generar un fichero XML a partir de un árbol DOM.
  - Para ello usaremos el paquete javax.xml.transform. Este paquete permite especificar una fuente y un resultado:
    - Ficheros
    - Flujos de datos,
    - Nodos DOM
    - Etc.

Trabajo con ficheros XML

Acceso a ficheros con DOM

- Los programas en Java que utilicen DOM necesitan hacer uso de las siguientes interfaces:
  - **Document** → Es un objeto que representa el documento XML. Permite crear nuevos nodos.
  - **Element** → Representa cada uno de los elementos del documento.
  - **Node** → Representa a cualquier nodo del documento.
  - **NodeList** → Contiene una lista con los nodos hijos de un nodo determinado.
  - **Attr** → Permite acceder a los atributos de un nodo.
  - **Text** → Representa los datos caracter de un nodo.
  - **CharacterData** → Representa los datos caracter presentes en el documento.
  - **DocumentType** → Proporciona información contenida en la etiqueta <!DOCTYPE>

Trabajo con ficheros XML

Acceso a ficheros con DOM: Escritura

- A continuación vamos a crear un fichero XML de empleados a partir del fichero de empleados que usamos en el ejemplo anterior (Ficheros binarios).
- Lo primero que debemos hacer es importar los paquetes necesarios.

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

Trabajo con ficheros XML

Acceso a ficheros con DOM: Escritura

- Después crearemos el Document en el cual vamos a insertar a nuestros empleados.

```
/*
 * Crearemos un DocumentBuilder haciendo uso de la factoria
 * DocumentBuilderFactory
 */
DocumentBuilderFactory dbf= DocumentBuilderFactory.newInstance();
DocumentBuilder builder=dbf.newDocumentBuilder();

/*           * Creamos un documento vácio
 *  Nombre --> RegistroEmpleados
 *  Nodo Raíz --> Empleados
 */
DOMImplementation implementacion= builder.getDOMImplementation();
Document registroEmpleados = implementacion.createDocument(null, "empleados", null);
//Asignamos la versión de XML
registroEmpleados.setXmlVersion("1.0");
```

Trabajo con ficheros XML

Acceso a ficheros con DOM: Escritura

- E iremos creado cada uno de los empleados con sus datos

```
//Creamos un nodo empleado
Element empleado= registroEmpleados.createElement("empleado");
//Lo añadimos como hijo de empleados
registroEmpleados.getDocumentElement().appendChild(empleado);
//Creamos el nodo ID
Element id= registroEmpleados.createElement("id");
//Creamos el nodo texto con el valor de la ID
Text texto= registroEmpleados.createTextNode("01");
//Añadimos el valor al nodo ID
id.appendChild(texto);
//Añadimos el nodo ID a empleado
empleado.appendChild(id);
Element nombre= registroEmpleados.createElement("nombre");
texto= registroEmpleados.createTextNode("Antonio");
nombre.appendChild(texto);
empleado.appendChild(nombre);
Element apellidos= registroEmpleados.createElement("apellido");
texto= registroEmpleados.createTextNode("Morales");
apellidos.appendChild(texto);
empleado.appendChild(apellidos);
```

Trabajo con ficheros XML

Acceso a ficheros con DOM: Escritura

- Finalmente debemos escribir nuestro Document en disco.

```
/*
 * Finalmente, para guardar el documento en disco debemos:
 *
 * 1. Crear el origen de datos (Nuestro Document)
 * 2. Crear el resultado (El fichero de destino)
 * 3. Crear un TransformerFactory
 * 4. Realizar la transformación
 */

Source origen= new DOMSource(registroEmpleados);
Result resultado= new StreamResult(new File ("Empleados.xml"));
Transformer transformador = TransformerFactory.newInstance().newTransformer();
transformador.transform(origen, resultado);
```

Trabajo con ficheros XML

Acceso a ficheros con DOM: Escritura

- Del mismo modo que utilizamos la clase transform para enviar nuestro Document a fichero, podemos usarla para mostrarlo por la salida estándar

```
/*
 * Mostramos el resultado por la salida estándar
 */
Result salidaEstnadar = new StreamResult(System.out);
transformador.transform(origen, salidaEstnadar);
```

Trabajo con ficheros XML

Acceso a ficheros con DOM: Lectura

- Para leer un documento XML haciendo uso de DOM debemos crear una instancia de DocumentBuilderFactory para construir el parser y, a través de él cargar el documento.

```
//Creamos el DocumentBuilder para poder obtener el Document
DocumentBuilderFactory dbf= DocumentBuilderFactory.newInstance();
DocumentBuilder builder=dbf.newDocumentBuilder();
//Leemos el Document desde el fichero
Document registroEmpleados= builder.parse(new File("Empleados.xml"));
//Normalizamos el documento para evitar errores de lectura.
registroEmpleados.getDocumentElement().normalize();
```

Trabajo con ficheros XML

Acceso a ficheros con DOM: Lectura

- Después creamos una lista como todos los elementos empleado usando la clase NodeList

```
//Mostramos el nombre del elemento raíz
System.out.println("El elemento raíz es "+
registroEmpleados.getDocumentElement().getNodeName());

//Creamos una lista de todos los nodos empleado
NodeList empleados= registroEmpleados.getElementsByTagName("empleado");
//Mostramos el nº de elementos empleado que hemos encontrado
System.out.println("Se han encontrado "+empleados.getLength()+" empleados");
```

Trabajo con ficheros XML

Acceso a ficheros con DOM: Lectura

- Finalmente, usando un bucle recorremos el NodeList y mostramos su contenido.

```
//Recorremos la lista.
for(int i=0; i<empleados.getLength();i++) {
    //Obtenemos el primer nodo de la lista
    Node emple= empleados.item(i);
    //En caso de que ese nodo sea un Elemento
    if(emple.getNodeType()==Node.ELEMENT_NODE) {
        //Creamos el elemento empleado y leemos su información
        Element empleado= (Element)emple;
        System.out.print("ID: " + empleado.getElementsByTagName("id").
                item(0).getTextContent() );
        System.out.print("\tNombre: " + empleado.getElementsByTagName("nombre").
                item(0).getTextContent()  );
        System.out.println("\tApellido: " + empleado.getElementsByTagName("apellido").
                item(0).getTextContent()  );
    }
}
```

Trabajo con ficheros XML

Acceso a ficheros con SAX

- SAX (Simple API for XML) es un conjunto de clases e interfaces que ofrecen una herramienta para el procesamiento de documentos XML.
- Permite analizar los documentos de forma secuencial (No carga todo el documento en memoria, a diferencia de DOM).
- La API de SAX está totalmente incluida dentro de JRE.
- SAX es más complejo de programar que DOM, sin embargo nos ofrece un menor consumo de memoria, por lo que es más adecuada para el tratamiento de ficheros XML de gran tamaño.

Trabajo con ficheros XML

Acceso a ficheros con SAX

- La lectura de un XML con SAX produce eventos que ocasionan la llamada a diferentes métodos:
  - **startDocuement() / endDocument()** → Encuentra las etiquetas de inicio y fin del documento.
  - **startElement() / endElement()** → Encuentra las etiquetas de apertura y cierre de un elemento.
  - **characters()** → Accede al contenido (Caracteres) de un elemento.

```
<?xml version="1.0" encoding="UTF-8"?><!--startDocument()-->
<alumnos><!--startElement()-->
    <alumno><!--startElement() -->
        <nombre><!--startElement()-->
            Juan <!--characters-->
        </nombre><!--endElement()-->
    </alumno><!--endElement() -->
    <alumno><!--startElement() -->
        <nombre><!--startElement()-->
            Marta <!--characters-->
        </nombre><!--endElement()-->
    </alumno><!--endElement() -->
</alumnos><!--endElement() -->
                                    <!--endDocument()-->
```

Trabajo con ficheros XML

Acceso a ficheros con SAX

- Vamos a ver un pequeño ejemplo en Java.
  - Lo primero que haremos será importar todas las clases e interfaces necesarias.

```
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;

import org.xml.sax.InputSource;
import org.xml.sax.SAXException;
import org.xml.sax.XMLReader;
```

- A continuación crearemos un objeto XMLReader (Objeto XML)

```
SAXParserFactory parserFactory = SAXParserFactory.newInstance();
SAXParser parser = parserFactory.newSAXParser();
XMLReader procesadorxml = parser.getXMLReader();
```

Trabajo con ficheros XML

Acceso a ficheros con SAX

- Lo siguiente que haremos será indicarle a nuestro XMLReader que objetos poseen los métodos que trataran los eventos que se produzcan en la lectura.
  - **ContentHandler** → Recibe las notificaciones de los eventos que se producen en el documento.
  - **DTDHandler** → Recoge eventos relacionados con el DTD.
  - **ErrorHandler** → define métodos de tratamiento de errores.
  - **EntityResolver** → sus métodos se llaman cada vez que se encuentra una referencia a una entidad.
  - **DefaultHandler** → Esta clase provee la implementación por defecto para todos sus métodos, el programador debe definir aquellos métodos que vayan a ser utilizados por la aplicación. **Esta es la clase de la que extenderemos para poder crear nuestro Parser XML.**

Trabajo con ficheros XML

Acceso a ficheros con SAX

- Por tanto, lo primero que debemos hacer antes de continuar será crearnos una clase que extienda de DefaultHandler para poder manejar los eventos de lectura.

```
public class AlumnosReader extends DefaultHandler {

    public AlumnosReader() {
        super();
    }
    @Override
    public void startDocument() throws SAXException {
        super.startDocument();
        System.out.println("Inicio del documento");
    }
    @Override
    public void endDocument() throws SAXException {
        super.endDocument();
        System.out.println("Fin del documento");
    }
    @Override
    public void startElement(String uri, String localName, String qName, Attributes attributes)throws SAXException {
        super.startElement(uri, localName, qName, attributes);
        System.out.printf("\t Inicio del Elemento: %s \n", qName);
    }
    @Override
    public void endElement(String uri, String localName, String qName) throws SAXException {
        super.endElement(uri, localName, qName);
        System.out.printf("\t Final del Elemento: %s \n", qName);
    }
    @Override
    public void characters(char[] ch, int start, int length) throws SAXException {
        super.characters(ch, start, length);
        String car = new String(ch, start, length);
        car = car.replaceAll("[\t\n]", "");
        System.out.printf("\t Valor del Elemento: %s \n\n", car);
    }
}
```

Trabajo con ficheros XML

Acceso a ficheros con SAX

- Una vez que tenemos nuestro propio manejador, debemos indicarle a nuestro XML Parser que utilice una instancia del mismo para poder leer el fichero.
- Para indicar al procesador XML que objetos realizarán el tratamiento se utiliza alguno de los siguientes métodos incluidos dentro de los objetos XMLReader:
  - setContentHandler()
  - setDTDHandler()
  - setEntityResolver
  - setErrorResolver()
- Cada uno de ellos trata un evento diferente y está asociada a una interfaz determinada.
- En nuestro caso usaremos un ContentHandler, por tanto:

```
procesadorxml.setContentHandler(new AlumnosReader());
```

Trabajo con ficheros XML

Acceso a ficheros con SAX

- Finalmente debemos definir la fuente de la que vamos a leer (El fichero) usando un objeto de la clase InputSource.

```
InputSource xmlFile= new InputSource("./alumnos.xml");
```

- Una vez tengamos la fuente creada y el ContentHandler asignado vamos a proceder a leer el fichero simplemente haciendo uso del método Parse de nuestro procesador XML.

```
procesadorxml.parse(xmlFile);
```

Trabajo con ficheros XML

Acceso a ficheros con SAX

- Código completo de la clase principal.

```
public class ReaderSAX {

    public static void main(String[] args) {
        try {

            SAXParserFactory parserFactory = SAXParserFactory.newInstance();
            SAXParser parser = parserFactory.newSAXParser();
            XMLReader procesadorxml = parser.getXMLReader();

            procesadorxml.setContentHandler(new AlumnosReader());
            InputSource xmlFile= new InputSource("./alumnos.xml");
            procesadorxml.parse(xmlFile);

        } catch (SAXException | IOException e) {
            e.printStackTrace();
        } catch (ParserConfigurationException e) {
            e.printStackTrace();
        }
    }
}
```

Trabajo con ficheros XML

Acceso a ficheros con SAX

- Por tanto, a modo de resumen, para poder leer un fichero usando SAX debemos:
  1. Creamos una clase que extienda de DefaultHandler e implementamos los métodos necesarios para el tratamiento.
  2. Creamos una nueva clase que será la encargada de la ejecución principal de nuestra aplicación (MAIN) en la que:
     a. Creamos un XMLReader (Nuestro parser)
     b. Asignamos a nuestro XMLReader un objeto de la clase DefaultHander que acabamos de crear.
     c. Creamos un objeto InputSource indicando el origen de datos.
     d. Usamos el método parse de nuestro XMLReader para leer el fichero.

Dudas y preguntas