EXPORTACIÓN E IMPORTACIÓN XML

Baltasar Rangel Pinilla
Jesús A. González Merín
Carlos de la Rosa García
Luis Manuel Becerra Álvarez

TIPOS

- SAX / SAXP
- EXCEL / CSV
- SQL
- JSON
- XSTREAM / XSERCES2
- DOM / JDOM

SAX / SAXP

Definición:

- SAX (Simple API XML). Originalmente una API, únicamente para el lenguaje de programación Java. Después se convirtió en la API estándar para usar XML en Java.
- API orientada a eventos (soportada por la mayoría de procesadores).

Diferencias con JDOM:

- No conlleva la generación de estructuras internas.
- SAX no fue diseñado pensando en Java, al contrario de JDOM.
- Es una herramienta más versátil, más veloz y menos potente que un analizador JDOM.

SAX / SAXP

Funcionamiento:

- Define una serie de interfaces, cuyos métodos son controladores de evento.
- Cuando se analiza un documento, el procesador notifica a los controladores de evento, los distintos sucesos que tienen lugar.
- Todas las operaciones de la aplicación se ejecutan durante la fase de análisis.

SAX / SAXP

Ventaja:

- No consume mucha memoria ya que va procesando el documento en serie.

Desventajas:

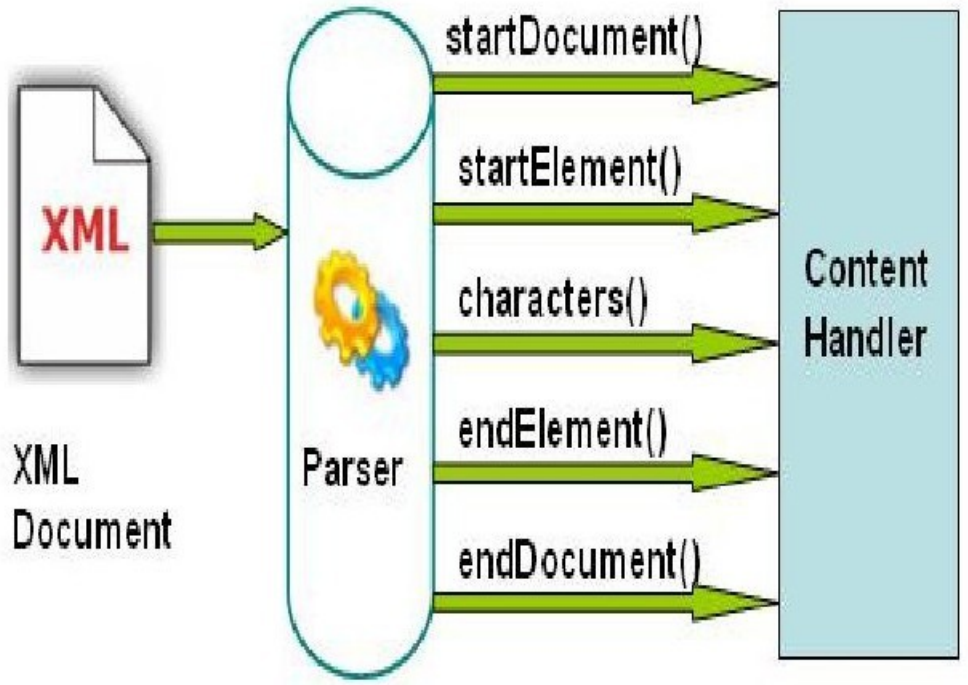
- Es poco intuitivo para el desarrollador.
- Requiere una serie de controles de estado adicionales que complican el procesado.
- No permite construir XML. Sólo permite parsearlos.
- No permite el acceso aleatorio al documento XML.

SAX / SAXP

Interfaces SAX:

Las más importantes son tres:

- DocumentHandler.
- DTDHandler.
- ErrorHandler.



EXCEL

Definición:

- Es un tipo de documento en formato abierto sencillo para representar datos en forma de tabla, en las que las columnas se separan con comas (o punto y coma donde la coma es el separador decimal) y las filas por salto de línea.

CSV

Definición:

- Es un tipo de documento en formato abierto sencillo para representar datos en forma de tabla, en las que las columnas se separan con comas (o punto y coma donde la coma es el separador decimal) y las filas por salto de línea.

Ventajas:

- En general, los formatos CSV ocupan la mitad de tamaño que el formato XML y JSON. Esta es la principal ventaja, puesto que ayuda a reducir el ancho de banda.

SQL

Definición:

- Es un lenguaje específico del dominio que da acceso a un sistema de gestión de base de datos relacionales que permite especificar diversos tipos de operaciones en ellos.

Ventajas:

- Una de sus características es el manejo de álgebra y cálculo relacional que permite efectuar consultas con el fin de recuperar, de forma sencilla, información en la base de datos, así cómo poder realizar cambios.

SQL

- Implementación flexible, dinámica y automatizada.
- Desarrollo ágil de bases de datos.

Desventajas:

- Falta de experiencia, o experiencia insuficiente por parte del personal.

JSON

Definición:

- JSON (JavaScript Object Notation). Es un formato ligero de intercambio de datos. Leerlo y escribirlo es simple para humanos, mientras que para las máquinas es simple interpretarlo y generarlo.

- Está basado en JavaScript.

- Es un formato de texto completamente independiente del lenguaje, pero utiliza convenciones que son ampliamente conocidos. Estas propiedades hacen que sea un lenguaje ideal para el intercambio de datos.

JSON

Ventajas:

- Es un formato de datos muy ligero.
- Independiente.
- Archivos de pequeño tamaño.

Desventajas:

- Su estructura es difícil de entender a simple vista.

JSON

Está constituido por dos estructuras:

- Una colección de pares de nombre/valor (objeto, registro, estructura).
- Una lista ordenada de valores (vectores, listas).

ejemplo.json

```
----------------------------------------
EJEMPLO REPRESENTACIÓN OBJETO EN JSON:
----------------------------------------
{
    "firstName": "John",
    "lastName": "Smith",
    "age": 25,
    "address": {
       "streetAddress": "21 2nd  Street",
       "city": "New York",
       "state": "NY",
       "postalCode": 10021
    },
    "phoneNumbers": [
       {
           "type": "home",
           "number": "212  555-1234"
       },
       {
           "type": "fax",
           "number": "646  555-4567"
       }
    ]
}
----------------------------------------
```

```
//Ejemplo de JSON para un objeto tipo Persona
{
"nombre":"Fulano Probencio",
"edad":27,
"nacionalidad":"Chileno",
"altura":"172 cm",
"peso":75
}
```

XSTREAM

Definición:

- Es una librería open-source que permite serializar objetos Java a un XML. También permite realizar la operación inversa.

- Ha ganado gran popularidad dentro de la comunidad debido a su eficiencia, simplicidad a la hora de utilizarla. En últimas versiones se ha añadido soporte para la interpretación y generación de JSON.

XSTREAM

Ventajas:

- No requiere modificación o creación de un objeto particular para el XML.
- Soporta anotaciones, lo cual facilita su configuración.
- La conversión de objetos a XML o viceversa es sencilla.
- Sencilla de aprender y utilizar.

Desventajas:

- No se pueden generar clases Java a partir de un esquema de datos.

XERCES2

Definición:

- Es una librería de Apache que nos permite el parseo de documentos XML mediante la implementación de los estándares de parseo SAX y DOM.

Ventajas:

- Introduce un nuevo componente llamado XNI (Xerces Native Interface). Framework que permite crear nuevos parsers o componentes a los ya existentes.
- Dado que Xerces2 es compatible con JAXP, podemos utilizarlo a través de dicha API. Con JAXP independizaríamos nuestra aplicación frente a los posibles cambios de versión del componente Xerces2.

DOM

Definición:

- DOM (Document Object Model). Es una interfaz de programación para los documentos HTML y XML facilita una representación estructurada del documento y define de que manera los programas puede acceder, al fin de modificar, tanto su estructura, estilo y contenido.

- Da una representación del documento cómo un grupo de nodos y objetos estructurados que tienen propiedades y métodos. Esencialmente, conecta las páginas Web a Scripts o lenguajes de programación.

JDOM

Definición:

- Es una biblioteca de código abierto para manipulaciones de datos XML optimizado y creado especialmente para Java.

- Proporciona un forma de representar un documento para una lectura, manipulación, redacción fácil y eficiente.

Licencia:

- Está disponible bajo una licencia de código abierto de estilo Apache.
  - Esta licencia se encuentra entre las menos restrictivas disponibles, lo que permite a los desarrolladores utilizar JDOM en la creación de nuevos productos sin que tengan que lanzar sus propios productos como de código abierto.

JDom

JDOM

¿Qué querían conseguir?

- Querían ofrecer una solución para usar XML en Java que sea tan simple y se comporte como tal, usando sus colecciones.

- JDOM se integra bien con los estándares existentes como la API para XML (SAX) y el **M**odelo de **O**bjeto de **D**ocumentos(DOM), no es una capa de abstracción o una mejora de esas API.

JDom

JDOM

¿Cómo funciona con DOM y SAX?

- Los documentos JDOM pueden construirse a partir de archivos XML, árboles DOM, eventos SAX o cualquier otra fuente.

- Se pueden convertir a archivos XML, árboles DOM, eventos SAX o cualquier otro destino. Esta capacidad es útil cuando se integra con un programa que espera eventos SAX.

JDom

JDOM

Ventajas:

- Tiene una API directa. Es liviana, rápida y está optimizada para programar en Java. Es una alternativa a DOM y SAX aunque se integra bien con estas dos.

Diferencias:

- Es muy similar a DOM, pero éste fue creado para ser un lenguaje neutral e inicialmente usado para la manipulación de páginas HTML con JavaScript.

- JDOM se creó específicamente para Java.

JDom

JDOM

Como configurar el proyecto para implementar JDOM:

- Descargar librería JDOM (2.0.6). Implementarla al proyecto.
- Crear la base de datos.
- Crear una clase JDOM.
- Implementación de métodos importar y exportar en la clase creada anteriormente.

JDom

JDOM

MÉTODO IMPORTACIÓN:

- Pasaremos por parámetro la ruta del archivo.
- Creamos los Arrays correspondientes a las tablas existentes en la base de datos.

```
public class JDOM {

    public List<ArrayList> importarXml(String path) {
        List<ArrayList> listaArrays = new ArrayList<>();

        //Creamos las listas en donde vamos a importar los datos para pasarselas al método que las pida.
        ArrayList<Curso> ListaCompletaCursos = new ArrayList();
        ArrayList<Alumno> listaCompletaAlumnos = new ArrayList();
        ArrayList<CursoAlumno> ListaCompletaCursosAlumnos = new ArrayList();

        //Se crea un SAXBuilder para poder parsear el archivo.
        SAXBuilder builder = new SAXBuilder();
        File xmlFile = new File(path);
```

JDom

JDOM

- Creamos el documento a través del archivo que hemos pasado por parámetros. En este caso, cuando se haga la llamada desde la interfaz, se abrirá un gestor de archivos.

- Obtenemos la raíz academia del XML.

- Recorremos la lista de la raíz Alumnos.

- Mediante un bucle, recorremos la lista.

- Por cada elemento alumno existente, se guarda en el Array creado anteriormente.

```
SAXBuilder builder = new SAXBuilder();
File xmlFile = new File(path);
try {
    //Se crea el documento a traves del archivo
    Document document = (Document) builder.build(xmlFile);

    //----------------------ALUMNOS-------------------
    //Se obtiene la raiz 'academia'.
    Element rootNode = document.getRootElement();

    //Se obtiene la lista de hijos de la raiz 'academia'.
    List listaAlumnos = rootNode.getChildren("Alumnos");

    //Se recorre la lista de hijos de 'Alumnos'.
    for (int i = 0; i < listaAlumnos.size(); i++) {
        //Se obtiene el elemento 'alumnos'
        Element alumnos = (Element) listaAlumnos.get(i);

        //Se obtiene la lista de hijos del tag 'alumnos'.
        List lista_campos = alumnos.getChildren();

        //Se recorre la lista de campos.
        for (int j = 0; j < lista_campos.size(); j++) {
            //Se obtiene el elemento 'alumno'.
            Element campo = (Element) lista_campos.get(j);

            //Se obtienen los valores que estan entre los tags '<alumno></alumno>'.
            //Se obtiene el valor que esta entre los tags '<nombre></nombre>'.
            Integer id = Integer.parseInt(campo.getAttributeValue("id"));
            Integer matricula = Integer.parseInt(campo.getChildTextTrim("matricula"));
            String nombre = campo.getChildTextTrim("nombre");
            String apellido1 = campo.getChildTextTrim("apellido1");
            String apellido2 = campo.getChildTextTrim("apellido2");

            Alumno alumnoTemp = new Alumno();
            alumnoTemp.setId(id);
            alumnoTemp.setMatricula(matricula);
            alumnoTemp.setNombre(nombre);
            alumnoTemp.setApellido1(apellido1);
            alumnoTemp.setApellido2(apellido2);

            listaCompletaAlumnos.add(alumnoTemp);
        }

    }
```

JDom

JDOM

- Por cada elemento curso, se guarda en el Array creado anteriormente.

```
//------------------CURSOS------------------

//Se obtiene la raiz 'academia'.
//Se obtiene la lista de hijos de la raiz 'academia'.
List listaCursos = rootNode.getChildren("Cursos");

//Se recorre la lista de hijos de 'Cursos'.
for (int i = 0; i < listaCursos.size(); i++) {
    //Se obtiene el elemento 'Cursos'.
    Element cursos = (Element) listaCursos.get(i);

    //Se obtiene la lista de hijos del tag 'Cursos'.
    List lista_camposCursos = cursos.getChildren();

    //Se recorre la lista de cursos.
    for (int j = 0; j < lista_camposCursos.size(); j++) {

        //Se obtiene el elemento 'Curso'.
        Element campo = (Element) lista_camposCursos.get(j);

        //Se obtienen los valores que estan entre los tags '<curso></curso>'.
        //Se obtiene el valor que esta entre los tags '<descripcion></descripcion>'.
        Integer id = Integer.parseInt(campo.getAttributeValue("id"));
        String codCurso = campo.getChildTextTrim("codCurso");
        String descripcion = campo.getChildTextTrim("descripcion");

        Curso cursoTemp = new Curso();
        cursoTemp.setId(id);
        cursoTemp.setCodCurso(codCurso);
        cursoTemp.setDescripcion(descripcion);

        ListaCompletaCursos.add(cursoTemp);

    }

}
```

JDom

JDOM

- Por cada elemento cursoAlumno existente, se guarda en el Array creado anteriormente.

```
        //------------------CURSOSALUMNOS------------------
        //Se obtiene la raiz 'academia'.
        //Se obtiene la lista de hijos de la raiz 'academia'.
        List listaCursosAlumnos = rootNode.getChildren("CursosAlumnos");

        //Se recorre la lista de hijos de 'CursosAlumnos'.
        for (int i = 0; i < listaCursosAlumnos.size(); i++) {
            //Se obtiene el elemento 'cursoAlumno'.
            Element cursoAlumno = (Element) listaCursosAlumnos.get(i);

            //Se obtiene la lista de hijos del tag 'CursosAlumnos'.
            List lista_camposCursosAlumnos = cursoAlumno.getChildren();

            //Se recorre la lista de campos.
            for (int j = 0; j < lista_camposCursosAlumnos.size(); j++) {

                //Se obtiene el elemento 'cursoAlumno'.
                Element campo = (Element) lista_camposCursosAlumnos.get(j);

                //Se obtienen los valores que estan entre los tags '<cursoAlumno></cursoAlumno>'.
                //Se obtiene el valor que esta entre los tags '<idCurso></idCurso>'.
                String idCurso = campo.getChildTextTrim("idCurso");
                String idAlumno = campo.getChildTextTrim("idAlumno");

                CursoAlumno cursoAlumnoTemp = new CursoAlumno();
                cursoAlumnoTemp.setIdAlumno(Integer.parseInt(idAlumno));
                cursoAlumnoTemp.setIdCurso(Integer.parseInt(idCurso));
                ListaCompletaCursosAlumnos.add(cursoAlumnoTemp);

            }

        }

    } catch (IOException io) {
        System.out.println(io.getMessage());
    } catch (JDOMException jdomex) {
        System.out.println(jdomex.getMessage());
    }

    listaArrays.add(ListaCompletaCursos);//get(0)
    listaArrays.add(ListaCompletaCursosAlumnos);//get(1)
    listaArrays.add(listaCompletaAlumnos);//get(2)

    return listaArrays;
}
```

JDom

JDOM

MÉTODO EXPORTACIÓN:

- Pasamos por parámetro la ruta del fichero, al igual que en el método de importación.

- Creamos las listas correspondientes.

```
public void exportarXml(String path) {
    AlumnosDao alumnosDao = new AlumnosDao();
    List<Alumno> listaAlumnos = alumnosDao.listAll();
    CursosDao cursosDao = new CursosDao();
    List<Curso> listaCursos = cursosDao.listAll();
    CursosAlumnoDao cursosAlumnosDao = new CursosAlumnoDao();
    List<CursoAlumno> listaCursosAlumnos = cursosAlumnosDao.listAll();

    try {

        //Creamos el elemento 'academia'.
        Element academia = new Element("academia");
        Document doc = new Document(academia);

        //doc.setRootElement(academia);
        //--------------------------ALUMNOS-------------------------
        //Creamos el elemento Alumnos.
        Element alumnos = new Element("Alumnos");

        //Recorremos la lista de alumnos y recogemos su atributo.
        for (int i = 0; i < listaAlumnos.size(); i++) {
            //Creamos el elemento alumno.
            Element alumno = new Element("Alumno");
            //Creamos el atributo id para recoger la id del alumno.
            Attribute idalumno = new Attribute("id", String.valueOf(listaAlumnos.get(i).getId()));
            //añadimos a la "Lista" de alumnos el contenido de cada alumno.
            alumnos.addContent(alumno);
            //Establecemos el atributo a cada alumno, que se correspondería con la id.
            alumno.setAttribute(idalumno);
            //Añadimos los contenidos...
            alumno.addContent(new Element("matricula").setText(String.valueOf(listaAlumnos.get(i).getMatricula())));
            alumno.addContent(new Element("nombre").setText(listaAlumnos.get(i).getNombre()));
            alumno.addContent(new Element("apellido1").setText(listaAlumnos.get(i).getApellido1()));
            alumno.addContent(new Element("apellido2").setText(listaAlumnos.get(i).getApellido2()));

        }
        //Esta es la parte donde se añade(sacarlo del bucle).
        doc.getRootElement().addContent(alumnos);
```

JDom

JDOM

- Exportación cursos.

```
//---------------------------CURSOS--------------------------
Element cursos = new Element("Cursos");

for (int i = 0; i < listaCursos.size(); i++) {
    Element curso = new Element("Curso");
    Attribute idCurso = new Attribute("id", String.valueOf(listaCursos.get(i).getId()));
    cursos.addContent(curso);
    curso.setAttribute(idCurso);
    curso.addContent(new Element("codCurso").setText(String.valueOf(listaCursos.get(i).getCodCurso())));
    curso.addContent(new Element("descripcion").setText(listaCursos.get(i).getDescripcion()));

}

//Esta es la parte donde se añade(sacarlo del bucle).
doc.getRootElement().addContent(cursos);
```

JDOM

- Exportación cursos-Alumnos.

```
        //---------------------------CURSOSALUMNOS---------------------------
        Element cursosAlumnos = new Element("CursosAlumnos");

        for (int i = 0; i < listaCursosAlumnos.size(); i++) {
            Element cursoAlumno = new Element("cursoAlumno");
            cursosAlumnos.addContent(cursoAlumno);
            cursoAlumno.addContent(new Element("idCurso").setText(String.valueOf(listaCursosAlumnos.get(i).getIdCurso())));
            cursoAlumno.addContent(new Element("idAlumno").setText(String.valueOf(listaCursosAlumnos.get(i).getIdAlumno())));

        }


        //Esta es la parte donde se añade(sacarlo del bucle).
        doc.getRootElement().addContent(cursosAlumnos);

        XMLOutputter xmlOutput = new XMLOutputter();

        //Establece el formato .
        xmlOutput.setFormat(Format.getPrettyFormat());
        //Elige el documento en el que va a guardar la información.
        xmlOutput.output(doc, new FileWriter(path));

        System.out.println("Archivo Exportado!");
    } catch (IOException ex) {
        System.out.println(ex.getMessage());
    }
}
```

JDom

JDOM

- Para probar el funcionamiento, haremos uso de la clase Test.

```
public class test {
      public static void main(String[] args) {
        //Instanciamos la clase JDOM.
        JDOM jdom =new JDOM();

        //Probamos a importar un archivo XML, creado previamente.
        jdom.importarXml("academiaExportado.xml");

        //Probamos a exportar un archivo XML.
        jdom.exportarXml("academiaExportado02.xml");
    }
}
```

JDOM

- Documento XML formado con la estructura deseada.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<academia>
    <Alumnos>
        <Alumno id="1">
            <matricula>1</matricula>
            <nombre>Carlos</nombre>
            <apellido1>De La Rosa</apellido1>
            <apellido2>García</apellido2>
        </Alumno>
        <Alumno id="2">
            <matricula>2</matricula>
            <nombre>Baltasar</nombre>
            <apellido1>Rangel</apellido1>
            <apellido2>Pinilla</apellido2>
        </Alumno>
        <Alumno id="3">
            <matricula>3</matricula>
            <nombre>Jesús</nombre>
            <apellido1>González</apellido1>
            <apellido2>Merín</apellido2>
        </Alumno>
        <Alumno id="4">
            <matricula>4</matricula>
            <nombre>Luis Manuel</nombre>
            <apellido1>Becerra</apellido1>
            <apellido2>Álvarez</apellido2>
        </Alumno>
        <Alumno id="5">
            <matricula>5</matricula>
            <nombre>Paco</nombre>
            <apellido1>Pérez</apellido1>
            <apellido2>Pérez</apellido2>
        </Alumno>
    </Alumnos>
    <Cursos>
        <Curso id="1">
            <codCurso>DAM1</codCurso>
            <descripcion>Desarrollo de Aplicaciones Multiplataforma</descripcion>
        </Curso>
        <Curso id="2">
            <codCurso>DAM2</codCurso>
            <descripcion>Desarrollo de Aplicaciones Multiplataforma</descripcion>
        </Curso>
        <Curso id="3">
            <codCurso>SMR1</codCurso>
            <descripcion>Sistemas Microinformáticos y Redes</descripcion>
        </Curso>
        <Curso id="4">
            <codCurso>SMR2</codCurso>
            <descripcion>Sistemas Microinformáticos y Redes</descripcion>
        </Curso>
    </Cursos>
    <CursosAlumnos>
        <cursoAlumno>
            <idCurso>1</idCurso>
            <idAlumno>2</idAlumno>
        </cursoAlumno>
        <cursoAlumno>
            <idCurso>1</idCurso>
            <idAlumno>3</idAlumno>
        </cursoAlumno>
        <cursoAlumno>
            <idCurso>2</idCurso>
            <idAlumno>2</idAlumno>
        </cursoAlumno>
        <cursoAlumno>
            <idCurso>3</idCurso>
            <idAlumno>1</idAlumno>
        </cursoAlumno>
        <cursoAlumno>
            <idCurso>3</idCurso>
            <idAlumno>2</idAlumno>
        </cursoAlumno>
        <cursoAlumno>
            <idCurso>4</idCurso>
            <idAlumno>1</idAlumno>
        </cursoAlumno>
        <cursoAlumno>
            <idCurso>4</idCurso>
            <idAlumno>2</idAlumno>
        </cursoAlumno>
    </CursosAlumnos>
</academia>
```

JDom

BIBLIOGRAFÍA

https://www.w3schools.com/js/js_json_intro.asp

http://xerces.apache.org/xerces2-j/

https://www.w3.org/2005/03/DOM3Core-es/introduccion.html

https://www.w3schools.com/js/js_htmldom.asp

https://www.osmosislatina.com/xml/domsax.htm

http://www.jdom.org/docs/faq.html#a0000
http://www.studytrails.com/java/xml/jdom2/java-xml-jdom2-introduction/

http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/227

http://www.oracle.com/technetwork/es/articles/java/api-java-para-json-2251318-esa.htmls

http://www.blog.teraswap.com/xstream-introduccion/

https://geekytheory.com/json-i-que-es-y-para-que-sirve-json/

https://empleotoday.com/technology/csv-vs-xml-vs-json-cual-es-el-mejor-formato-de-datos-de-respuesta/

FIN

Baltasar Rangel Pinilla
Jesús A. González Merín
Carlos de la Rosa García
Luis Manuel Becerra Álvarez