	RAMA:	Informática	CICLO:	Desenvolvemento de Aplicacions Multiplataforma				
	MÓDULO:	Acceso a datos				CURSO:	2º	
	PROTOCOLO:	Apuntes clases		AVAL:	1	DATA:	2022/2023	
	UNIDAD		COMPETENCIA:					

Tema 2: Uso de ficheros XML en Java

Índice

1. Introducción.....	1
2. DOM	1
2.1. Uso de DOM en Java	3
2.2. Creación del árbol DOM	3
2.3. Obtención de información de un árbol DOM	4
2.4. Modificar un árbol DOM	8
2.5. Almacenar en disco un árbol DOM.....	9
3. SAX.....	10
3.1. Procesar ficheros XML usando SAX.....	11
3.2. Recorrer un documento XML con SAX	11
4. Anexo I: Trabajo con ficheros HTML.....	15
4.1. Comparativa de parsers HTML.....	19
5. Bibliografía	20

COLEXIO VIVAS S.L.	RAMA:	Informática	CICLO:	Desenvolvemiento de Aplicacions Multiplataforma				
	MÓDULO:	Acceso a datos				CURSO:	2º	
	PROTOCOLO:	Apuntes clases		AVAL:	1	DATA:	2022/2023	
	UNIDAD		COMPETENCIA:					

1. Introducción

Las clases vistas, en el tema pasado, para el tratamiento de ficheros de texto permiten trabajar con su contenido como una secuencia de caracteres pero sin ninguna información adicional sobre su contenido.

Existen, sin embargo, ficheros que no son simplemente una secuencia de caracteres sino que su contenido está estructurado y del cual se puede extraer información. Ejemplos pueden ser ficheros XML, JSON, HTML, CSS, ...

Este tema se centrará en el trabajo con ficheros XML puesto que son uno de los métodos más utilizados para el intercambio de información entre aplicaciones de software que pueden ser desarrolladas en lenguajes de programación distintos y ejecutadas sobre distintas plataformas.

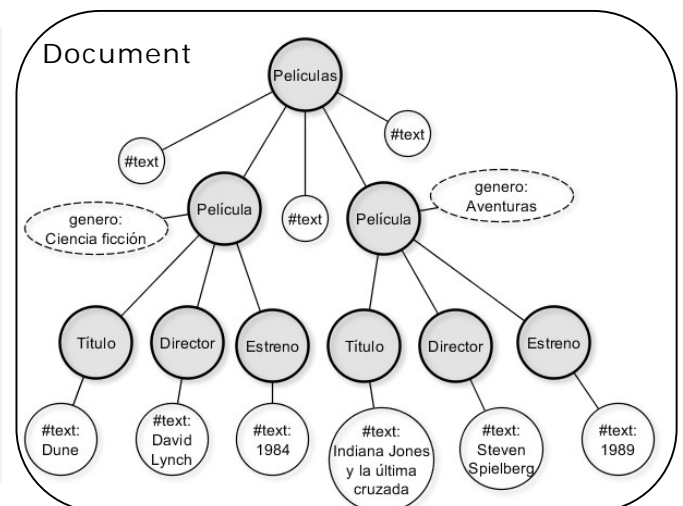
Se utilizará para ello dos tecnologías que, con enfoques diferentes, permiten conseguir, utilizando un parser¹, la información contenida en un fichero XML: DOM y SAX.

2. DOM²

DOM (Document Object Model) es una tecnología propuesta World Wide Web (W3C) con la cual se puede analizar un documento XML y crear un árbol invertido en memoria con la estructura del documento. Esta estructura permitirá navegar a través de los distintos nodos que genera.

Por ejemplo, el siguiente documento XML representan los datos de una serie de películas, cada una de ellas posee el atributo género y contiene los elementos título, director y estreno.


```
<Películas>
  <Película género="Ciencia ficción">
    <Título>Dune</Título>
    <Director>David Lynch</Director>
    <Estreno>1984</Estreno>
  </Película>
  <Película género="Aventuras">
    <Título>Indiana Jones y la última cruzada</Título>
    <Director>Steven Spielberg</Director>
    <Estreno>1989</Estreno>
  </Película>
</Películas>
```



El diagrama de la derecha es una representación, del que se han eliminado varios nodos #text, de la estructura del árbol DOM creado a partir del XML del izquierda.

¹ https://es.wikipedia.org/wiki/Analizador_sintáctico

² https://es.wikipedia.org/wiki/Document_Object_Model
https://en.wikipedia.org/wiki/Document_Object_Model


	RAMA:	Informática	CICLO:	Desenvolvemento de Aplicacions Multiplataforma				
	MÓDULO:	Acceso a datos					CURSO:	2º
	PROTOCOLO:	Apuntes clases		AVAL:	1	DATA:	2022/2023	
	UNIDAD COMPETENCIA:							

Tenemos los siguientes elementos:

- Cada nodo, los círculos en el diagrama, tiene que ser de un tipo³, los principales son:
 - Document: es un tipo de nodo que engloba todos los nodos y cuyo primer nodo es el nodo raíz.
<https://docs.oracle.com/javase/8/docs/api/org/w3c/dom/Document.html>
 - Element: el nodo es un elemento:
<https://docs.oracle.com/javase/8/docs/api/org/w3c/dom/Element.html>
 - Text: es un nodo de tipo texto.
<https://docs.oracle.com/javase/8/docs/api/org/w3c/dom/Text.html>
 - Comment: el nodo es un comentario:
<https://docs.oracle.com/javase/8/docs/api/org/w3c/dom/Comment.html>
 - Attr: el nodo es un atributo.
<https://docs.oracle.com/javase/8/docs/api/org/w3c/dom/Attr.html>
- Se pueden encontrar las siguientes clases de nodos.
 - Raíz: es el primer nodo del árbol y es el único obligatorio. En el ejemplo es películas.
 - Padre: es aquel nodo del que descienden otros nodos, es decir tiene hijos. En el ejemplo películas es el padre de dos nodos película que a su vez son padre de título, director y estreno.
 - Hijo: es un nodo que tiene un padre. Por ejemplo película, título, director y estreno.
 - Hermanos: son nodos que tienen el mismo padre. Por ejemplo título, director y estreno.
 - Hoja: es un nodo que no tiene hijos. Por ejemplo todos los nodos #text del ejemplo.
- Atributos: pertenecen a un elemento y representan propiedades del mismo. Un mismo elemento puede tener más de un atributo. Son pares clave-valor, en nuestro ejemplo los atributos género tienen los valores de ciencia ficción y aventuras. Se representan con una elipse punteada.

Esta estructura **se crea en memoria** lo que permite tener una visión global de documento XML permitiendo, usando los métodos disponibles en DOM, "navegar" a través del árbol. Por contrapartida documentos XML grandes pueden ocupar mucho en memoria.

³ <https://docs.oracle.com/javase/8/docs/api/org/w3c/dom/Node.html>

	RAMA:	Informática	CICLO:	Desenvolvemento de Aplicacions Multiplataforma				
	MÓDULO:	Acceso a datos					CURSO:	2º
	PROTOCOLO:	Apuntes clases		AVAL:	1	DATA:	2022/2023	
	UNIDAD		COMPETENCIA:					

Se puede destacar:

- Los elementos XML con un valor, como pueden ser título, director o estreno, se representan en el árbol DOM como nodos Element pero **su valor no está en el propio nodo**, sino que está en su **primer nodo hijo de tipo #text**.
- Cualquier conjunto de caracteres entre dos elementos se convierte en un nodo #text. Esto incluye saltos de línea, tabuladores, ... Por eso en el diagrama anterior hay varios nodos #text que descienden de películas (tendrían que aparecer también los que descienden de película). Normalmente estos saltos de línea, espacios, tabuladores, ... se introducen en el documento para aumentar su legibilidad por humanos pero para la información que contiene el documento son totalmente superfluos y deberemos ignorarlos.
- Aunque por norma general se trabajará con nodos de tipo Element y tipo #text, dentro del fichero XML se pueden encontrar otros tipos de elementos diferentes que, si no son de nuestro interés, se podrán omitir.

2.1. Uso de DOM en Java

El primer paso para trabajar con un fichero XML es crear su estructura en memoria. Esta se representa con un objeto tipo Document y a partir de él recorrerlo en el orden deseado.

Para procesar DOM se utilizarán las clases que pertenecen a los siguientes paquetes:

- javax.xml.parsers⁴: contienen los parsers DOM.
- org.w3c.dom⁵: es la implementación del modelo DOM que proporciona un conjunto de métodos y componentes: documentos, nodos, elementos, ...

2.2. Creación del árbol DOM

Para crear el árbol DOM en memoria, usando un archivo o una URI, se puede usar el siguiente código:


```

Ejemplo 1: Creación del árbol DOM
1  public Document creaArbol(String ruta) {
2      Document doc=null;
3      try {
4          DocumentBuilderFactory factoria = DocumentBuilderFactory.newInstance();
5          factoria.setIgnoringComments(true);
6          DocumentBuilder builder = factoria.newDocumentBuilder();
7          doc=builder.parse(ruta);
8      } catch (Exception e) {
9          System.out.println("Error generando el árbol DOM: "+e.getMessage());
10     }
11     return doc;
12 }

```

⁴ <http://docs.oracle.com/javase/8/docs/api/javax/xml/parsers/package-summary.html>

⁵ <http://docs.oracle.com/javase/8/docs/api/org/w3c/dom/package-summary.html>

	RAMA:	Informática	CICLO:	Desenvolvemento de Aplicacions Multiplataforma				
	MÓDULO:	Acceso a datos					CURSO:	2º
	PROTOCOLO:	Apuntes clases		AVAL:	1	DATA:	2022/2023	
	UNIDAD		COMPETENCIA:					

Donde:

- Línea 1: el parámetro entrada con el documento XML puede ser tanto el path de un fichero almacenado en disco como la Uri de un archivo XML en Internet.
- Línea 2: se define el objeto Document que representa y almacena la estructura de árbol creada a partir del fichero XML.
- Línea 4: se crea una nueva instancia del objeto DocumentBuilderFactory que permitirá, con el parser seleccionado, crear el árbol DOM.
- Línea 5: se ignoran los posibles comentarios del documento al realizar el 'parseo' del documento.
- Línea 6: se crea, a partir del objeto anterior, el constructor del árbol DOM.
- Línea 7: se crea el árbol DOM, en memoria, almacenándose en un objeto de tipo Document (variable doc creada al principio del método).

Ejemplos de llamada al método creaArbol son:

```
Document doc1=dom.creaArbol("c:/simple.xml");
Document doc2=dom.creaArbol("https://www.w3schools.com/xml/cd_catalog.xml");
```

Una vez creado y almacenado el árbol ya se puede recorrer para obtener la información que contiene.

2.3. Obtención de información de un árbol DOM

Para 'navegar' a través del árbol DOM se utilizan principalmente cuatro interfaces de las cuales se pueden destacar una serie de métodos de especial interés, pero hay que tener presente que son solo una pequeña parte de las disponibles:

- Node⁶: representa un nodo genérico dentro del árbol DOM.
 - getChildNodes(): lista, de tipo nodeList⁷, de nodos con los hijos de un nodo.
 - getFirstChild(): devuelve el primer hijo de un nodo.
 - getNode⁸(): indica el tipo de nodo siendo los más habituales: COMMENT_NODE, ELEMENT_NODE y TEXT_NODE
 - getNodeName(): devuelve el nombre del nodo.
 - getNodeValue(): devuelve una cadena con el contenido de un nodo. En un nodo Element este valor es nulo ya que para obtener su valor este está en su primer nodo hijo de tipo #text.
 - getParentNode(): obtiene el nodo padre del nodo actual.
 - getAttributes(): devuelve un objeto NamedNodeMap⁹ con los atributos del nodo si el nodo es de tipo Element o null en otro caso.
 - getNextSibling(): devuelve el siguiente hermano de un nodo.
 - hasChildNodes(): indica si un nodo tiene nodos hijos.

⁶ <https://docs.oracle.com/javase/8/docs/api/org/w3c/dom/Node.html>

⁷ <https://docs.oracle.com/javase/8/docs/api/org/w3c/dom/NodeList.html>

⁸ Valores de las constantes en Java: <http://docs.oracle.com/javase/8/docs/api/constant-values.html#org.w3c>

⁹ <https://docs.oracle.com/javase/8/docs/api/org/w3c/dom/NamedNodeMap.html>

COLEXIO VIVAS S.L.	RAMA:	Informática	CICLO:	Desenvolvemento de Aplicacions Multiplataforma				
	MÓDULO:	Acceso a datos					CURSO:	2º
	PROTOCOLO:	Apuntes clases		AVAL:	1	DATA:	2022/2023	
	UNIDAD		COMPETENCIA:					

- Document¹⁰: contiene todos los nodos del archivo XML. Es a su vez un nodo.
 - `getElementsByTagName(String tag)`: obtiene una lista de nodos con los elementos cuyo nombre coincida con el parámetro `tag`.
 - `getElementById(String name)`: obtiene una lista de nodos cuyo atributo `id` tiene el valor `name`.
- NodeList¹¹: es una lista de nodos. La primera posición de la lista es la 0.
 - `getLength()`: número de nodos en la lista.
 - `item(int index)`: devuelve el nodo de la lista indicado por `index`.
- Element¹²: representa un elemento XML con atributos propios. Element deriva de nodo por lo que es posibles realizar un cast entre ambos.
 - `getAttribute(String name)`: obtiene el valor de un atributo de un elemento por su nombre.
 - `getElementsByTagName(String name)`: obtiene una lista de nodos con los elementos cuyo nombre coincida con el parámetro `name`.
 - `getTagName()`: devuelve el nombre del elemento.
 - `hasAttribute(String name)`: devuelve verdadero si el elemento tiene el atributo especificado por el atributo `name`.

Con las interfaces y métodos anteriores se puede recorrer y visualizar el contenido del fichero XML del punto 2.

Ejemplo 2: Recorrido de un fichero XML

```

1 public void recorreDom(Document doc){
2     Node raiz, pelicula, nodoAux, atributo;
3     NodeList peliculas, datos;
4     NamedNodeMap atributos;

5     // Se obtiene el primer nodo del documento, el nodo raíz
6     raiz=doc.getFirstChild();
7     System.out.printf("El nodo visualizado es: %s%n",raiz.getNodeName());

8     // Se obtienen los hijos del nodo raíz como un objeto NodeList.
9     peliculas=raiz.getChildNodes();
10    for (int i=0;i<peliculas.getLength();i++){ // Se recorren los nodos hijos
11        pelicula=peliculas.item(i); // Se obtienen la película i

12        // Se procesa el nodo si es un nodo Element, los demás se ignoran
13        if (pelicula.getNodeType() == Node.ELEMENT_NODE){
14            System.out.println("-----");
15            datos=pelicula.getChildNodes(); // Se obtienen los hijos de película

```

¹⁰ <https://docs.oracle.com/javase/8/docs/api/org/w3c/dom/Document.html>

¹¹ <https://docs.oracle.com/javase/8/docs/api/org/w3c/dom/NodeList.html>

¹² <https://docs.oracle.com/javase/8/docs/api/org/w3c/dom/Element.html>

COLEXIO VIVAS S.L.	RAMA:	Informática	CICLO:	Desenvolvemento de Aplicacions Multiplataforma				
	MÓDULO:	Acceso a datos					CURSO:	2º
	PROTOCOLO:	Apuntes clases		AVAL:	1	DATA:	2022/2023	
	UNIDAD		COMPETENCIA:					

```

16         for (int j=0;j<datos.getLength();j++){ // Para cada hijo
17             nodoAux=datos.item(j);
18             // Solo se procesan los nodos Element hijos de película, en nuestro
19             // caso el título, el director y el estreno.
20             if (nodoAux.getNodeType()==Node.ELEMENT_NODE){
21                 // Se visualiza el nombre del elemento y su valor, recordar que el
22                 // valor de un elemento reside en su primer hijo de tipo text.
23                 System.out.println(nodoAux.getNodeName()+"
24                                     "+nodoAux.getFirstChild().getNodeValue());
25             }
26         }

27         if (pelicula.hasAttributes()){ // Si un elemento película tiene
28             atributos=pelicula.getAttributes(); // atributos se visualizan.
29             for (int k=0;k<atributos.getLength();k++){
30                 atributo=atributos.item(k);
31                 System.out.printf("Atributo: %s con valor %s%n",
32                                   atributo.getNodeName(), atributo.getNodeValue());
33             }
34         }
35     }
36 }

```

Si se quieren acceder a un dato concreto dentro del fichero XML se tienen dos opciones:

- Realizar un recorrido de todo el fichero XML, como en el ejemplo anterior.
- Acceder a un nodo concreto dentro del fichero XML.

El siguiente ejemplo es una muestra de la segunda opción:

Ejemplo 3: Visualización de los títulos de las películas

```

1 public void getTitulos(Document doc){
2     NodeList titulos=doc.getElementsByTagName("Título");
3     for (int i=0;i<titulos.getLength();i++){
4         System.out.println(titulos.item(i).getFirstChild().getNodeValue());
5     }
6 }

```

Donde:

- Línea 2: se obtienen todos los elementos con nombre Título.
- Líneas 3 y 4: para obtener los títulos solo hay que recorrer el objeto NodeList obtenido en la línea anterior.

<div> <div>COLEXIO</div> <div>VIVAS</div> <div>S.L.</div> </div>	RAMA:	Informática	CICLO:	Desenvolvimento de Aplicacions Multiplataforma		
	MÓDULO:	Acceso a datos				CURSO: 2º
	PROTOCOLO:	Apuntes clases	AVAL:	1	DATA:	2022/2023
	UNIDAD	COMPETENCIA:				

El siguiente ejemplo muestra cómo se pueden obtener los datos de una película concreta identificada por su título:

Ejemplo 4: Obtención de los datos de una película

```

1  public void datosPelicula(Document doc, String pelicula){
2      Element padre;
3      NodeList aux;
4      NodeList titulos=doc.getElementsByTagName("Título");
5      for (int i=0;i<titulos.getLength();i++){
6          if (titulos.item(i).getFirstChild().getNodeValue().equals(pelicula)){
7              padre=(Element)titulos.item(i).getParentNode();
8              aux=padre.getElementsByTagName("Título");
9              if (aux.getLength()>0) {
10                 System.out.println(aux.item(0).getFirstChild().getNodeValue());
11             }
12
13             aux=padre.getElementsByTagName("Director");
14             if (aux.getLength()>0){
15                 System.out.println(aux.item(0).getFirstChild().getNodeValue());
16             }
17
18             aux=padre.getElementsByTagName("Estreno");
19             if (aux.getLength()>0){
20                 System.out.println(aux.item(0).getFirstChild().getNodeValue());
21             }
22             break;
23         }
24     }
25 }

```

Donde:

- Líneas 4, 5 y 6: se buscan todos los elementos cuyo título sea igual al parámetro película.
- Línea 7: cuando encuentro el título correcto se obtiene el padre, que es la película con ese título, para conseguir el resto de los datos (esto mismo se podría hacer buscando los hermanos).
- Líneas 8, 12 y 16: desde el padre busco los elementos hijos con el nombre del elemento deseado. En este caso título, director y estreno.
- Líneas 10, 14 y 18: visualizo el contenido del elemento contenido en su primer hijo de tipo text.

<div>COLEXIO</div> <div>VIVAS</div> <div>S.L.</div>	RAMA:	Informática	CICLO:	Desenvolvemiento de Aplicacions Multiplataforma				
	MÓDULO:	Acceso a datos					CURSO:	2º
	PROTOCOLO:	Apuntes clases		AVAL:	1	DATA:	2022/2023	
	UNIDAD		COMPETENCIA:					


2.4. Modificar un árbol DOM

Además de recorrer el árbol DOM también se puede modificar. Cambiando valores o incluso añadiendo nuevos.

En el ejemplo siguiente se muestra como añadir una nueva película que sea hija del elemento películas.

Ejemplo 5: Modificación de un árbol DOM

```
public boolean addDOM(Document doc, String titulo, String director,
                      String estreno, String genero) {
    try {
        // A partir del documento doc se crea un nodo película que contendrá los demás
        // elementos: titulo, director y estreno
        Element nodoPelícula = doc.createElement("Película");
        // Se añade el atributo genero al elemento creado
        nodoPelícula.setAttribute("genero", genero);
        // Se añade a película un nodo tipo texto con un salto de línea (\n) para que
        // al abrirlo con un editor de texto cada nodo salga en un línea diferente.
        nodoPelícula.appendChild(doc.createTextNode("\n"));
        // Se crean los nodos hijos de película añadiéndolo un nodo texto que contendrá el
        // valor del elemento y se le añaden a la película
        // Para titulo
        Element nodoTitulo = doc.createElement("Título");
        Text textNodoTitulo = doc.createTextNode(titulo);
        nodoTitulo.appendChild(textNodoTitulo);
        nodoPelícula.appendChild(nodoTitulo);
        // Se añade también un nodo text con un saldo de línea \n
        nodoPelícula.appendChild(doc.createTextNode("\n"));
        // Para director
        Element nodoDirector = doc.createElement("Director");
        Text textNodoDirector = doc.createTextNode(director);
        nodoDirector.appendChild(textNodoDirector);
        nodoPelícula.appendChild(nodoDirector);
        // Se añade también un nodo text con un saldo de línea \n
        nodoPelícula.appendChild(doc.createTextNode("\n"));
        // y para estreno
        Element nodoEstreno = doc.createElement("Estreno");
        Text textNodoEstreno = doc.createTextNode(estreno);
        nodoEstreno.appendChild(textNodoEstreno);
        nodoPelícula.appendChild(nodoEstreno);
        // Se añade la película a películas, que es el primer nodo del documento
        Node raiz = doc.getFirstChild();
        raiz.appendChild(nodoPelícula);
        return true;
    } catch (DOMException e) {
        return false;
    }
}
```

	RAMA:	Informática	CICLO:	Desenvolvemento de Aplicacions Multiplataforma				
	MÓDULO:	Acceso a datos				CURSO:	2º	
	PROTOCOLO:	Apuntes clases		AVAL:	1	DATA:	2022/2023	
	UNIDAD		COMPETENCIA:					

2.5. Almacenar en disco un árbol DOM¹³

Para conseguir la persistencia de un documento XML, y por lo tanto de los datos en él contenidos, se puede almacenar el disco como un fichero XML. Existen diversas maneras siendo una de ellas la siguiente.

Ejemplo 6: Guardar un documento DOM a disco

```
public void grabarDOM(Document document, String ficheroSalida) throws
    ClassNotFoundException, InstantiationException,
    IllegalAccessException, FileNotFoundException {

    DOMImplementationRegistry registry = DOMImplementationRegistry.newInstance();
    DOMImplementationLS ls=(DOMImplementationLS)
        registry.getDOMImplementation("XML 3.0 LS 3.0");

    // Se crea un destino vacio
    LSOutput output = ls.createLSOutput();
    output.setEncoding("UTF-8");


    // Se establece el flujo de salida
    output.setByteStream(new FileOutputStream(ficheroSalida));
    //output.setByteStream(System.out);

    // Permite escribir un documento DOM en XML
    LSSerializer serializer = ls.createLSSerializer();

    // Se establecen las propiedades del serializador
    serializer.setNewLine("\r\n");
    serializer.getDomConfig().setParameter("format-pretty-print", true);

    // Se escribe el documento ya sea en un fichero o en una cadena de texto
    serializer.write(document, output);
    // String xmlCad=serializer.writeToString(document);
}
```

¹³ <https://www.w3.org/TR/2004/REC-DOM-Level-3-LS-20040407/>

	RAMA:	Informática	CICLO:	Desenvolvemento de Aplicacions Multiplataforma				
	MÓDULO:	Acceso a datos					CURSO:	2º
	PROTOCOLO:	Apuntes clases		AVAL:	1	DATA:	2022/2023	
	UNIDAD		COMPETENCIA:					

3. SAX¹⁴

SAX (Simple API for XML), al igual que DOM, tiene como objetivo procesar documentos XML para obtener información de ellos. Sin embargo SAX funciona de una forma totalmente diferente a DOM.

Se caracteriza por:

- SAX no crea estructuras en memoria sino que recorre el fichero de forma secuencial desde el principio hasta el final, o hasta que encuentra un error. Esto provoca un bajo consumo en memoria, en contraposición a DOM, más marcado sobre todo en documentos de gran tamaño.
- El programador no tiene control sobre el recorrido del fichero por lo que solo puede acceder al elemento que el parser está accediendo en un momento dado. Esto implica que no tiene una visión global del documento ni puede “navegar” por él.
- A medida que el recorrido avanza sobre el fichero se generan eventos que provoca que se invoquen métodos callback de SAX encargados de tratarlos.

Estos eventos se producen cuando SAX detecta:

- El comienzo del documento XML.
- El comienzo de un elemento XML.
- El contenido de un elemento como una cadena de caracteres.
- El final del documento XML.
- El final de un elemento XML.
- Que detecte un error en el procesamiento del fichero XML.

Para utilizar SAX se deberá implementar una clase que actúe como parser y que debe heredar de la clase `DefaultHandler`¹⁵. Esta clase incorpora los métodos callback que son invocados de forma automática al encontrarse alguno de los eventos indicados anteriormente.


Es en la clase definida y sobrescribiendo los métodos callback¹⁶ donde se puede personalizar el comportamiento del parser.

¹⁴ https://es.wikipedia.org/wiki/Simple_API_for_XML

https://en.wikipedia.org/wiki/Simple_API_for_XML

¹⁵ <http://docs.oracle.com/javase/8/docs/api/org/xml/sax/helpers/DefaultHandler.html>

¹⁶ [https://es.wikipedia.org/wiki/Callback_\(inform%C3%A1tica\)](https://es.wikipedia.org/wiki/Callback_(inform%C3%A1tica))

	RAMA:	Informática	CICLO:	Desenvolvemento de Aplicacions Multiplataforma				
	MÓDULO:	Acceso a datos					CURSO:	2º
	PROTOCOLO:	Apuntes clases		AVAL:	1	DATA:	2022/2023	
	UNIDAD		COMPETENCIA:					

3.1. Procesar ficheros XML usando SAX

De forma análoga a DOM se usarán las clases SAXParserFactory¹⁷ y SAXParser¹⁸ para procesar el fichero XML mediante SAX. Un ejemplo de invocación del parser SAX es la siguiente:

Ejemplo 7: Recorrido de un fichero XML con SAX

```

1 public void getSax(String entradaXML) throws
2     ParserConfigurationException, SAXException, IOException {
3     SAXParserFactory factory = SAXParserFactory.newInstance();
4     SAXParser parser = factory.newSAXParser();
5     ParserSAX parserSax = new ParserSAX(); // ParserSAX es la clase que se deberá
6     parser.parse(entradaXML, parserSax); // implementar y que hereda de DefaultHandler
7 }

```

Donde:

- Línea 1: el parámetro entrada puede ser tanto el path de un fichero almacenado en disco como la URI de un archivo XML en Internet.
- Línea 3: se crea una nueva instancia de la factoría que permite crear el parser XML.
- Línea 4: se crea el objeto parser que permite interpretar el documento XML.
- Línea 5: se crea una instancia de la clase creada por nosotros que define las acciones a llevar a cabo cuando se produzca un evento y se invoquen un método callback.
- Línea 6: se procesa el documento XML de entrada mediante el parser definido en el punto anterior.

3.2. Recorrer un documento XML con SAX

Como se comenté en el punto anterior para personalizar el comportamiento del parser se deberá sobrescribir los métodos callback de la clase que hereda de DefaultHandler. Solo se deben sobrescribir aquellos métodos que se deseen personalizar.

Entre otros se pueden encontrar los siguientes métodos callback que se invocan cuando:

- startDocument: se detecta el principio de un documento XML.
- startElement: se detecta el principio de un elemento XML. Es en este método donde se pueden obtener los atributos¹⁹ de un elemento.
- endDocument: se detecta el fin de un documento XML.
- endElement: se detecta el fin de un elemento XML.
- characters: se encuentra un valor de un elemento como una cadena de caracteres.

¹⁷ <https://docs.oracle.com/javase/8/docs/api/javax/xml/parsers/SAXParserFactory.html>

¹⁸ <https://docs.oracle.com/javase/8/docs/api/javax/xml/parsers/SAXParser.html>

¹⁹ <http://www.saxproject.org/apidoc/org/xml/sax/Attributes.html>

COLEXIO VIVAS S.L.	RAMA:	Informática	CICLO:	Desenvolvemento de Aplicacions Multiplataforma				
	MÓDULO:	Acceso a datos					CURSO:	2º
	PROTOCOLO:	Apuntes clases		AVAL:	1	DATA:	2022/2023	
	UNIDAD		COMPETENCIA:					

El siguiente documento XML permite mostrar la secuencia de invocación de los métodos callback a medida que se recorre, en orden, el fichero XML y se van detectando los eventos indicados en el punto 3.

<pre> <?xml version="1.0" encoding="UTF-8"?> <Películas> <Película genero="Ciencia ficción"> <Título> Dune </Título> <Director> David Lynch </Director> <Estreno> 1984 </Estreno> </Película> <Película genero="Aventuras"> <Título> Indiana Jones y la última cruzada </Título> <Director> Steven Spielberg </Director> <Estreno> 1989 </Estreno> </Película> </Películas> </pre>	<pre> startDocument startElement startElement characters endElement startElement characters endElement startElement characters endElement endElement startElement startElement characters endElement startElement characters endElement startElement characters endElement endElement endElement endDocument </pre>
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Por lo que para obtener, mediante SAX, los títulos de las películas se podría implementar la siguiente clase:

Ejemplo 8: Obtención de los títulos de la películas


```

1 public class ParserSAX extends DefaultHandler{
2     boolean esTitulo=false;

3     @Override
4     public void startElement(String uri, String localName, String qName,
5                             Attributes attributes) throws SAXException {
6         if (qName.equals("Título")) this.esTitulo=true;
7     }

8     @Override
9     public void characters(char[] ch, int start, int length) throws SAXException {
10        if (esTitulo) {
11            String titulo=new String(ch, start, length);
12            System.out.println(titulo);
13            esTitulo=false;
14        }
15    }
16 }

```

	RAMA:	Informática	CICLO:	Desenvolvemento de Aplicacions Multiplataforma				
	MÓDULO:	Acceso a datos					CURSO:	2º
	PROTOCOLO:	Apuntes clases		AVAL:	1	DATA:	2022/2023	
	UNIDAD		COMPETENCIA:					

Donde:

- Línea 4: se detecta el comienzo de un elemento.
- Línea 6: mediante el parámetro qName se conoce el nombre del elemento al que se está accediendo. Se utiliza para detectar el comienzo de los elementos título.
- Líneas 9: el método characters es el utilizado para obtener los valores de los elementos en SAX. Tiene tres parámetros:
 - ch: contiene todo el documento XML de entrada como un array de caracteres.
 - start: indica el índice de comienzo del valor del elemento actual dentro del array.
 - length: indica la longitud de caracteres del valor del elemento actual.
- Línea 10: si se está en un título es que los caracteres que viene a continuación es el valor de un elemento título, es decir, el valor buscado.
- Línea 11: se crea una nueva cadena con el tamaño del título.
- Línea 13: se pone el título a falso para que no vuelva a acceder hasta que encuentre un nuevo elemento título.

En el siguiente ejemplo se muestra como recorrer el documento XML mostrando su contenido

Ejemplo 9: Recorrido de un fichero XML con SAX

```

1  public class ParserSAXB extends DefaultHandler{
2      String qName="";
3
4      @Override
5      public void startDocument() throws SAXException {
6          System.out.println("Comienzo del documento XML");
7      }
8
9      @Override
10     public void endDocument() throws SAXException {
11         System.out.println("Fin del documento XML");
12     }
13
14     @Override
15     public void startElement(String uri, String localName, String qName,
16                             Attributes attributes) throws SAXException {
17         this.qName=qName;
18         if (qName.equals("Película")) {
19             for (int i=0;i<attributes.getLength();i++)
20                 System.out.printf("Atributo %s con valor %s\n",
21                                 attributes.getLocalName(i),attributes.getValue(i));
22         }
23     }
24 }
```

<div>COLEXIO</div> <div>VIVAS</div> <div>S.L.</div>	RAMA:	Informática	CICLO:	Desenvolvemento de Aplicacions Multiplataforma				
	MÓDULO:	Acceso a datos					CURSO:	2º
	PROTOCOLO:	Apuntes clases		AVAL:	1	DATA:	2022/2023	
	UNIDAD		COMPETENCIA:					

```

21     @Override
22     public void endElement(String uri, String localName, String qName)
23                             throws SAXException {
24         if (qName.equals("Película")) System.out.println("-----");
25         this.qName="";
26     }

27     @Override
28     public void characters(char[] ch, int start, int length) throws SAXException{
29         String cad=new String(ch, start, length);
30         if (this.qName.equals("Título")) System.out.println("Título: "+cad+ "");
31         else if (this.qName.equals("Director")) System.out.println("Direct: "+cad);
32         else if (this.qName.equals("Estreno")) System.out.println("Año: "+cad);
33     }

34     @Override
35     public void warning(SAXParseException e) throws SAXException {
36         System.out.println("Aviso: "+e.getMessage());
37     }


38     @Override
39     public void error(SAXParseException e) {
40         System.out.println("Error: "+e.getMessage());
41     }

42     @Override
43     public void fatalError(SAXParseException e) {
44         System.out.println("Error fatal: "+e.getMessage());
45     }
46 }

```

Donde:

- Líneas 4 y 8: Se muestra un mensaje descriptivo al comienzo y al final del recorrido del documento XML.
- Línea 14: establezco un valor al atributo qName para conocer, en el método characters, el elemento que se está procesando en este momento.
- Líneas 15, 16, 17 y 18: si el elemento que se está procesando es una película se muestran sus atributos.
- Línea 24: cuando se termina de procesar una película se muestra un separador.
- Línea 25: se elimina el valor de qName para evitar que se muestre información no deseada en el método characters.
- Línea 29: se crea una cadena con el valor del elemento.
- Líneas 30, 31 y 22: se muestra la información de los elementos deseados.
- Líneas 35, 38 y 42: se gestionan posibles avisos y errores ocurridos durante el recorrido del documento XML.

	RAMA:	Informática	CICLO:	Desenvolvemento de Aplicacions Multiplataforma				
	MÓDULO:	Acceso a datos				CURSO:	2º	
	PROTOCOLO:	Apuntes clases		AVAL:	1	DATA:	2022/2023	
	UNIDAD		COMPETENCIA:					

4. Anexo I: Trabajo con ficheros HTML

Una página web, al igual que un fichero XML, sigue el modelo DOM pero no tiene por qué estar bien formado por lo que los parsers usados en XML no suelen funcionar con una página web.

Entre la serie de parsers específicos para ficheros HTML (ver punto 4.1) nosotros nos centraremos en JSoup²⁰.

JSoup es una librería de Java que proporciona una API para extraer y manipular archivos HTML. Entre otras posibilidades nos permite extraer toda la información de un sitio web (Web scraping)²¹.

Antes de usar JSoup deberemos obtener el JAR²² que contiene la librería JSoup e incluirlo en el CLASSPATH o añadirlo a nuestro IDE, por ejemplo en Eclipse pulsamos en el nombre del proyecto con el botón derecho del ratón y seleccionamos Build Paths → Add External Archives y añadimos el fichero JAR con la librería.

Al igual que DOM JSoup genera un árbol en memoria por el cual podemos navegar. Este árbol será un objeto Document de tipo org.jsoup.nodes.Document.

Para generar el árbol disponemos del método connect de Jsoup que entre dispone de los siguientes métodos:

- UserAgent: Establecer el userAgent de la petición.
- Referrer: establecer la url desde la cual se hace la petición.
- Timeout: establecer el tiempo máximo usado para establecer la conexión.

Un ejemplo de su uso para obtener el árbol JSoup es:

```
Document doc=Jsoup.connect("http://www.google.es/search?1=hola")
.referrer("http://www.google.es").
.userAgent("Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:90.0)")
.timeout(10000)
.get();
```


Ejemplos de métodos disponibles en JSoup para trabajar con el árbol generado son:

- Encontrar elementos:
 - getElementById
 - getElementsByTagName
 - getElementsByClass
 - getElementsByAttribute
 - siblingElements, firstElementSibling, nextElementSibling, previousElementSibling
 - parent, children, child

²⁰ Página web oficial: <http://jsoup.org> y javadoc: <http://jsoup.org/apidocs/>

²¹ Conseguir información de una página web: http://es.wikipedia.org/wiki/Web_scraping

²² Descargar de: <http://jsoup.org/download>

	RAMA:	Informática	CICLO:	Desenvolvemento de Aplicacions Multiplataforma				
	MÓDULO:	Acceso a datos				CURSO:	2º	
	PROTOCOLO:	Apuntes clases		AVAL:	1	DATA:	2022/2023	
	UNIDAD		COMPETENCIA:					

- Elemento data:
 - attr(String key) pasa conseguir el atributo key y attr(String key, String value) para establecer el atributo key con el valor value.
 - attributes para obtener todos los atributos.
 - text(): Consigue el valor del texto que posee un nodo.
 - html() y html(String value): consigue el código html asociado al nodo.
 - tag() y tagName(): consigue el nombre del nodo
- Manipular:
 - append(String html), prepend(String html).
 - appendText(String text), prependText(String text).
 - appendElement(String tagName), prependElement(String tagName).

Ejemplo 10: Obtener todos los enlaces una pagina

```
import java.io.IOException;
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;
import org.jsoup.select.Elements;


public class Html {

    public void getEnlaces(String url) throws IOException {
        Document doc = Jsoup.connect(url).get();
        // Document doc=Jsoup.connect(url).userAgent("Mozilla/5.0").get();

        Elements links= doc.getElementsByTag("a"); // Consigo todos los enlaces html
        for (Element link : links) {
            System.out.println("\nlink : "+link.attr("href"));
            System.out.println("text : "+link.text());
            System.out.println("text : "+link.html());
            System.out.println("text : "+link.outerHtml());
        }
    }
}
```

Aparte de los métodos estándar para encontrar elementos JSoup soporta la misma sintaxis que CSS para encontrar elementos.

Usaremos el método select junto un selector. Está disponible en Document, Element y en Elements. Devuelve una lista de elementos.

	RAMA:	Informática	CICLO:	Desenvolvemento de Aplicacions Multiplataforma				
	MÓDULO:	Acceso a datos					CURSO:	2º
	PROTOCOLO:	Apuntes clases		AVAL:	1	DATA:	2022/2023	
	UNIDAD		COMPETENCIA:					

Ejemplos de selectores son:

➤ Sencillos:

- `tagname`: encuentra elementos por tag, por ejemplo `a`.
- `#id`: encuentra elementos por ID, por ejemplo `#logo`.
- `.class`: encuentra elementos por class name, por ejemplo `.masthead`.
- `[attribute]`: encuentra elementos con el atributo attribute, por ejemplo `[href]`.
- `[^attr]`: encuentra elementos con el prefijo prefix en el nombre.
- `[attr=value]`: elementos con el atributo value, por ejemplo `[width=500]`.
- `[attr^=value]`, `[attr$=value]`, `[attr*=value]`: elemento con el atributo que empiece con, termina con o contiene el valor value, por ejemplo `[href*= /path/]`
- `[attr~=regex]`: encuentra elementos que el atributo cumpla con la expresión regular regex.; por ejemplo `img[src~=(?i)\.(png|jpe?g)]`.
- `*`: encuentre todos los elementos.

➤ Combinación de selectores.

- `el#id`: elementos con ID, por ejemplo `div#logo`.
- `el.class`: elementos con clase, por ejemplo `div.masthead`.
- `el[attr]`: elementos con atributo, por ejemplo `a[href]`.
- Cualquier combinación, por ejemplo `a[href].highlight`.
- `antecesor hijo`: Elementos hijos que descienden de antecesor, por ejemplo `.body p` encuentra elementos `p` dentro de bloques con la class "body".
- `padre > hijo`: Elementos hijo que descienden directamente de padre, por ejemplo `body > *` encuentra todos los hijos que descienden directamente de `body`.
- `el1, el2, el3`: un grupo de múltiples selectores. Encuentra elementos que casa con alguno de los selectores del grupo; por ejemplo `div.masthead, div.logo`.

➤ Pseudo selectors

- `:lt(n)`: find elements whose sibling index (i.e. its position in the DOM tree relative to its parent) is less than n; por ejemplo `td:lt(3)`.
- `:gt(n)`: find elements whose sibling index is greater than n; por ejemplo `div p:gt(2)`.

<div>COLEXIO</div> <div>VIVAS S.L.</div>	RAMA:	Informática	CICLO:	Desenvolvemento de Aplicacions Multiplataforma				
	MÓDULO:	Acceso a datos					CURSO:	2º
	PROTOCOLO:	Apuntes clases		AVAL:	1	DATA:	2022/2023	
	UNIDAD		COMPETENCIA:					

- `:eq(n)`: find elements whose sibling index is equal to n; por ejemplo form input: `eq(1)`.
- `:has(selector)`: find elements that contain elements matching the selector; por ejemplo `div:has(p)`.
- `:not(selector)`: find elements that do not match the selector; por ejemplo `div:not(.logo)`.
- `:contains(text)`: find elements that contain the given text. The search is case-insensitive; por ejemplo `p:contains(jsoup)`.
- `:containsOwn(text)`: find elements that directly contain the given text.
- `:matches(regex)`: find elements whose text matches the specified regular expression; por ejemplo `div:matches((?i)login)`.
- `:matchesOwn(regex)`: find elements whose own text matches the specified regular expression.
- Note that the above indexed pseudo-selectors are 0-based, that is, the first element is at index 0, the second at 1, etc.

Ejemplo 13: Obtener todos los enlaces una pagina

```
public void getEnlacesConSelectores(String url) throws IOException {
    Document doc = Jsoup.connect(url).get();

    Elements links = links = doc.select("a[href]");
    for (Element link : links) {
        System.out.println("\nlink : " + link.attr("href"));
        System.out.println("text : " + link.text());
        System.out.println("text : " + link.html());
        System.out.println("text : " + link.outerHtml());
    }
}
```

COLEXIO VIVAS S.L.	RAMA:	Informática	CICLO:	Desenvolvimento de Aplicacions Multiplataforma		
	MÓDULO:	Acceso a datos				CURSO: 2º
	PROTOCOLO:	Apuntes clases	AVAL:	1	DATA:	2022/2023
	UNIDAD	COMPETENCIA:				

4.1. Comparativa de parsers HTML.²³


El cuadro siguiente muestra una comparativa de algunos parsers HTML.

Parser	Licencia	Lenguaje	HTML Parsing	Limpia HTML*	Actualiza HTML**
Html Agility Pack	Microsoft Public License	C#	Si	No	?
Beautiful Soup	Python S. F. L.	Python	Si	?	?
Gumbo	Apache License 2.0	C	Si	?	?
html5lib	MIT License	Python and PHP	Si	Si	No
HTML::Parser	Perl license	Perl	No	?	?
htmlPurifier	GNU Lesser GPL	PHP	No	Si	Si
HTML Tidy	W3C license	ANSI C	Si	Si	?
HtmlCleaner	BSD License	Java	No	Si	?
Hubbub	MIT License	C	Si	?	?
Jaunt API	Jaunt Beta License	Java	Si	Si	No
Jericho HTML Parser	Eclipse Public License	Java	No?	?	?
jsdom	MIT license	JavaScript	No	?	?
jsoup	MIT license	Java	Si	Si	Si
JTidy	JTidy License	Java	Si	Si	?
libxml2 HTMLparser	MIT License	C	Si	?	?
NekoHTML	Apache License 2.0	Java	No	?	?
TagSoup	Apache License 2.0	Java	No	?	?
Validator.nu HTML Parser	MIT license	Java	Si	?	?

* Satinizar (generación de la página Web estándar) y limpiar (eliminar las etiquetas de presentación excedentes, eliminar el código XSS, etc) código HTML.

** Actualizaciones HTML4.X a XHTML o HTML5, convertir etiquetas en desuso (ej. CENTER) en etiquetas válidas (ej. DIV with style="text-align:center;").

²³ Fuente: http://en.wikipedia.org/wiki/Comparison_of_HTML_parsers

	RAMA:	Informática	CICLO:	Desenvolvemento de Aplicacions Multiplataforma				
	MÓDULO:	Acceso a datos				CURSO:	2º	
	PROTOCOLO:	Apuntes clases	AVAL:	1	DATA:	2022/2023		
	UNIDAD		COMPETENCIA:					

5. Bibliografía

1. [Analizador sintáctico](#)
2. [Modelos](#)
3. DOM
 1. [DOM en Wikipedia castellano](#)
 2. [DOM en Wikipedia Ingles](#)
 3. [Document](#)
 4. [Nodo](#)
 5. [Element](#)
 6. [Text](#)
 7. [Comment](#)
 8. [Attr](#)
 9. [NodeList](#)
 10. [NamedNodeMap](#)
 11. [Valores dos tipos de nodos en Java](#)
 12. [Document Object Model \(DOM\) Level 3 Load and Save Specification](#)
4. SAX
 1. [SAX en Wikipedia castellano](#)
 2. [SAX en Wikipedia Ingles](#)
 3. [Métodos callback](#)
 4. [Clase DefaultHandler utilizada para crear el parser SAX](#)
 5. [SAXParserFactory](#)
 6. [SAXParser](#)
 7. [Atributos SAX](#)
5. JSOUP
 1. [Página web](#)
 2. [API](#)
 3. [Descargar Jar](#)
 4. [Conseguir información dunha páxina web Web Scraping](#)
 5. [Comparativa de parsers HTML](#)