	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 ficheiros XML en Xava

### Índice

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

<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:					

## 1. Introducción

As clases vistas, no tema pasado, para o tratamento de ficheiros de texto permiten traballar co seu contido como unha secuencia de caracteres pero sen ningunha información adicional sobre o seu contido.

Existen, sin embargo, ficheiros que non son simplemente unha secuencia de caracteres senón que o seu contido está estruturado e do cal se pode extraer información. Exemplos poden ser ficheiros XML, JSON, HTML, CSS, ...

Este tema centrarase no traballo con ficheiros XML debido a que son un dos métodos máis utilizados para o intercambio de información entre aplicacións de software que poden ser desenvoltas en linguaxes de programación distintas e executadas sobre distintas plataformas.

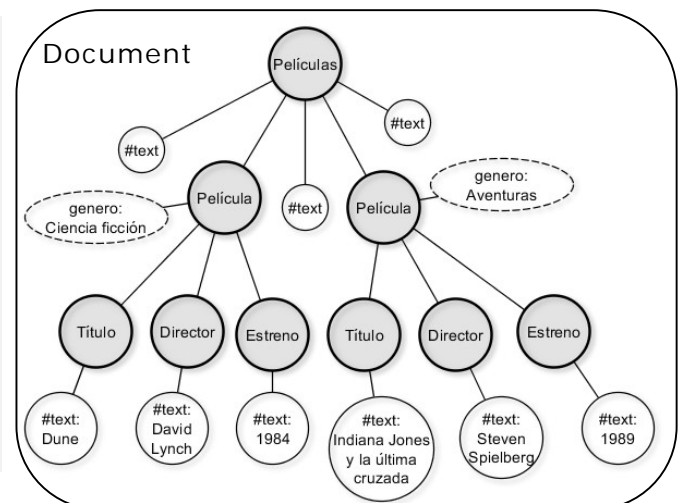
Utilizarase para iso dúas tecnoloxías que, con enfoques diferentes, permiten conseguir, utilizando un parser<sup>1</sup>, a información contida nun ficheiro XML: DOM e SAX.

## 2. DOM<sup>2</sup>

DOM (Document Object Model) é unha tecnoloxía proposta World Wide Web (W3C) coa cal se pode analizar un documento XML e crear unha árbore invertida en memoria coa estrutura do documento. Esta estrutura permitirá navegar a través dos distintos nodos que xera.

Por exemplo, o seguinte documento XML representan os datos dunha serie de películas, cada unha delas posúe o atributo xénero e contén os elementos título, director e estrea.

```
<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>
```



O diagrama da dereita é unha representación, do cal elimináronse algúns nodos #text, da estrutura da árbore DOM creado a partir do XML da esquerda. Pódense observar os seguintes elementos:

<sup>1</sup> [https://es.wikipedia.org/wiki/Analizador\\_sintáctico](https://es.wikipedia.org/wiki/Analizador_sintáctico)

<sup>2</sup> [https://es.wikipedia.org/wiki/Document\\_Object\\_Model](https://es.wikipedia.org/wiki/Document_Object_Model)  
[https://en.wikipedia.org/wiki/Document\\_Object\\_Model](https://en.wikipedia.org/wiki/Document_Object_Model)

<b>COLEXIO</b> <b>VIVAS</b> 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:					

- Cada nodo, os círculos no diagrama, ten que ser dun tipo<sup>3</sup>, os principais son:
  - Document: e un tipo de nodo que engloba todos os nodos e cuxo primeiro nodo é o nodo raíz.  
<https://docs.oracle.com/javase/8/docs/api/org/w3c/dom/Document.html>
  - Element: o nodo é un elemento.  
<https://docs.oracle.com/javase/8/docs/api/org/w3c/dom/Element.html>
  - Text: é un nodo de tipo texto.  
<https://docs.oracle.com/javase/8/docs/api/org/w3c/dom/Text.html>
  - Comment: o nodo é un comentario.  
<https://docs.oracle.com/javase/8/docs/api/org/w3c/dom/Comment.html>
  - Attr: o nodo é un atributo.  
<https://docs.oracle.com/javase/8/docs/api/org/w3c/dom/Attr.html>
- Pódense encontrar as seguintes clases de nodos.
  - Raíz: é o primeiro nodo da árbore e é o único obrigatorio. No exemplo é películas.
  - Pai: é aquel nodo do que descendan outros nodos, é dicir ten fillos. No exemplo películas é o pai de dous nodos película que á súa vez son pais de título, director e estrea.
  - Fillo: é un nodo que ten un pai. Por exemplo película, título, director e estrea.
  - Irmáns: son nodos que teñen o mesmo pai. Por exemplo título, director e estrea.
  - Folla: é un nodo que non ten fillos. Por exemplo todos os nodos #text do exemplo.
- Atributos: pertencen a un elemento e representan propiedades do mesmo. Un mesmo elemento pode ter máis dun atributo. Son pares chave-valor, no noso exemplo os atributos xénero teñen os valores de ciencia ficción e aventuras. Representáanse cunha elipse punteada

Esta estrutura **créase en memoria** o que permite ter unha visión global de documento XML permitindo, usando os métodos dispoñibles en DOM, "navegar" a través da árbore. Por contrapartida documentos XML grandes poden ocupar moito en memoria.

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

<b>COLEXIO</b> <b>VIVAS</b> 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:					

Pódese destacar:

- Os elementos XML cun valor, como poden ser título, director ou estrea, represéntanse na árbore DOM como nodos Element pero o **seu valor non está no propio nodo**, senón que está no seu **primeiro nodo fillo de tipo #text**.
- Calquera conxunto de caracteres entre dous elementos convértese nun nodo #text. Isto inclúe saltos de liña, tabuladores, ... Por iso no diagrama anterior hai varios nodos #text que descenden de películas (terían que aparecer tamén os que descenden de película). Normalmente estes saltos de liña, espazos, tabuladores, ... introdúcense no documento para aumentar a súa lexibilidade por humanos pero para a información que contén o documento son totalmente superfluos e deberemos ignoralos.
- Aínda que por norma xeral traballárase con nodos de tipo Element e tipo #text, dentro do ficheiro XML pódense atopar outros tipos de elementos diferentes que, se non son do noso interese, poderanse omitir.

## 2.1. Uso de DOM en Java

O primeiro paso para traballar cun ficheiro XML é crear a súa estrutura en memoria. Esta se representa cun obxecto tipo Document e a partir del percorrelo na orde desexada.

Para procesar DOM utilizaranse as clases que pertencen aos seguintes paquetes:

- javax.xml.parsers<sup>4</sup>: conteñen os parsers DOM.
- org.w3c.dom<sup>5</sup>: é a implementación do modelo DOM que proporciona un conxunto de métodos e compoñentes: documentos, nodos, elementos, ...

## 2.2. Creación del árbore DOM

Para crear a árbore DOM en memoria, usando un arquivo ou unha URI, pódese usar o seguinte código:


### Exemplo 1: Creación da árbore 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("Erro xerando a árbore DOM: "+e.getMessage());
10    }
11    return doc;
12 }
```

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

<sup>5</sup> <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:					

Onde:

- Liña1: o parámetro entrada co documento XML pode ser tanto o path dun ficheiro almacenado en disco como a Uri dun arquivo XML na internet.
- Liña 2: defínese o obxecto Document que representa e almacena a estrutura de árbore creada a partir do ficheiro XML.
- Liña 4: créase unha nova instancia do obxecto DocumentBuilderFactory que permitirá, co parser seleccionado, crear a árbore DOM.
- Liña 5: ignóranse os posibles comentarios do documento ao realizar o 'parseo' do documento.
- Liña 6: créase, a partir do obxecto anterior, o construtor da árbore DOM.
- Liña 7: créase a árbore DOM, en memoria, almacenándose nun obxecto de tipo Document (variable doc creada ao principio do método).

Exemplos de chamada ao método creaArbol son:

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

Unha vez creado e almacenado a árbore xa se pode percorrer para obter a información que contén.

## 2.3. Obtención de información dunha árbore DOM

Para “navegar” a través da árbore DOM utilízanse principalmente catro interfaces das cales se poden destacar unha serie de métodos de especial interese, pero hai que ter presente que son só unha pequena parte das dispoñibles:


- Node<sup>6</sup>: representa un nodo xenérico dentro da árbore DOM.
  - getChildNodes(): lista, de tipo nodeList<sup>7</sup>, de nodos cos fillos dun nodo.
  - getFirstChild(): devolve o primeiro fillo dun nodo.
  - getNodeTypes<sup>8</sup>(): indica o tipo de nodo sendo os máis habituais: COMMENT\_NODE, ELEMENT\_NODE e TEXT\_NODE.
  - getNodeName(): devolve o nome do nodo.
  - getNodeValue(): devolve unha cadea co contido dun nodo. Nun nodo Element este valor é nulo xa que para obter o seu valor este está no seu primeiro nodo fillo de tipo #text.
  - getParentNode(): obtén o nodo pai do nodo actual.
  - getAttributes(): devolve un obxecto NamedNodeMap<sup>9</sup> cos atributos do nodo se o nodo é de tipo Element ou null noutro caso.
  - getNextSibling(): devolve o seguinte irmán dun nodo.
  - hasChildNodes(): indica se un nodo ten nodos fillos.

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

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

<sup>8</sup> Valores dos tipos de nodos en Java: <http://docs.oracle.com/javase/8/docs/api/constant-values.html#org.w3c>

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

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

- Document<sup>10</sup>: contén todos os nodos do arquivo XML. É á súa vez un nodo.
  - `getElementsByTagName(String tag)`: obtén unha lista de nodos cos elementos no que o seu nome coincida co parámetro `tag`.
  - `getElementById(String name)`: obtén unha lista de nodos cuxo atributo `id` ten o valor `name`.
- NodeList<sup>11</sup>: é unha lista de nodos. A primeira posición da lista é a 0.
  - `getLength()`: número de nodos na lista.
  - `item(int index)`: devolve o nodo da lista indicado por `index`.
- Element<sup>12</sup>: representa un elemento XML con atributos propios. Element deriva de nodo polo que é posible realizar un cast entre ambos.
  - `getAttribute(String name)`: obtén o valor dun atributo dun elemento polo seu nome.
  - `getElementsByTagName(String name)`: obtén unha lista de nodos cos elementos no que o seu nome coincida co parámetro `name`.
  - `getTagName()`: devolve o nome do elemento.
  - `hasAttribute(String name)`: devolve verdadeiro se o elemento ten o atributo especificado polo atributo `name`.

Coas interfaces e métodos anteriores pódese percorrer e visualizar o contido do ficheiro XML do punto 2.

#### Exemplo 2: Recorrido de un ficheiro XML

```

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

5      // Obtense o primeiro nodo do documento, o nodo raíz
6      raiz=doc.getFirstChild();
7      System.out.printf("O nodo visualizado e: %s%n",raiz.getNodeName());

8      // Obtéñense os fillos do nodo raíz como un obxecto NodeList.
9      peliculas=raiz.getChildNodes();
10     for (int i=0;i<peliculas.getLength();i++){ // Percórrense os nodos fillos
11         pelicula=peliculas.item(i); // Obtéñense a película i

12         // Procésase o nodo se é un nodo Element, os demais ignóranse
13         if (pelicula.getNodeType()==Node.ELEMENT_NODE){
14             System.out.println("-----");
15             datos=pelicula.getChildNodes(); // Obtéñense os fillos de película

```

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

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

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

<b>COLEXIO</b> <b>VIVAS</b> 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 fillo
17             nodoAux=datos.item(j);
18             // Solo Só procesáanse os nodos Element fillos de película, no noso
19             // caso o título, o director e a estrea.
20             if (nodoAux.getNodeType()==Node.ELEMENT_NODE){
21                 // Visualízase o nome do elemento e o seu valor, lembrar que o
22                 // valor dun elemento reside no seu primeiro fillo de tipo text.
23                 System.out.println(nodoAux.getNodeName()+"
24                                     "+nodoAux.getFirstChild().getNodeValue());
25             }
26         }
27
28         if (pelicula.hasAttributes()){ // Se un elemento película ten
29             atributos=pelicula.getAttributes(); // atributos visualízanse.
30             for (int k=0;k<atributos.getLength();k++){
31                 atributo=atributos.item(k);
32                 System.out.printf("Atributo: %s con valor %s%n",
33                                     atributo.getNodeName(), atributo.getNodeValue());
34             }
35         }
36     }
}

```

Se se queren acceder a un dato concreto dentro do ficheiro XML téñense dúas opcións:

- Realizar un percorrido de todo o ficheiro XML, como no exemplo anterior.
- Acceder a un nodo concreto dentro do ficheiro XML.

O seguinte exemplo é unha mostra da segunda opción:

**Exemplo 3: Visualización dos títulos das 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 }

```

Onde:

- Liña 2: obtéñense todos os elementos con nome Título.
- Liñas 3 e 4: para obter os títulos só hai que percorrer o obxecto NodeList obtido na liña anterior.

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

O seguinte exemplo mostra como se poden obter os datos dunha película concreta identificada polo seu título:

#### Exemplo 4: Obtención dos datos dunha película

```

1  public void datosPelícula(Document doc, String película){
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(película)){
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             aux=padre.getElementsByTagName("Estreno");
18             if (aux.getLength()>0){
19                 System.out.println(aux.item(0).getFirstChild().getNodeValue());
20             }
21             break;
22         }
23     }
24 }

```

Onde:

- Liñas 4, 5 e 6: búscase todos os elementos cuxo título sexa igual ao parámetro película.
- Liña 7: cando atopo o título correcto obtense o pai, que é a película con ese título, para conseguir o resto dos datos (isto mesmo poderíase facer buscando os irmáns).
- Liñas 8, 12 e 16 desde o pai busco os elementos fillos co nome do elemento desexado. Neste caso título, director e "Estreno".
- Liñas 10, 14 e 18: visualizo o contido do elemento contido no seu primeiro fillo de tipo text.



<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:					


## 2.4. Modificar unha árbore DOM

Ademais de percorrer a árbore DOM tamén se pode modificar. Cambiando valores ou mesmo engadindo novos.

No exemplo seguinte móstrase como engadir unha nova película que sexa filla do elemento películas.

### Exemplo 5: Modificación dunha árbore DOM

```
public boolean addDOM(Document doc, String titulo, String director,
                      String estrea, String xenero) {
    try {
        // A partir do documento doc créase un nodo película que conterá os demais
        // elementos: titulo, director e estrea
        Element nodoPelícula = doc.createElement("Película");
        // Engádese o atributo xero ao elemento creado
        nodoPelícula.setAttribute("xenero", xenero);
        // Engádese a película un nodo tipo texto cun salto de liña (\n) para que
        // ao abrilo cun editor de texto cada nodo salga nun liña diferente.
        nodoPelícula.appendChild(doc.createTextNode("\n"));
        // Créanse os nodos fillos de película engadíndoo un nodo texto que conterá o
        // valor do elemento e engádenselle á película
        // Para titulo
        Element nodoTitulo = doc.createElement("Título");
        Text textNodoTitulo = doc.createTextNode(titulo);
        nodoTitulo.appendChild(textNodoTitulo);
        nodoPelícula.appendChild(nodoTitulo);
        // Engádese tamén un nodo text cun saldo de liña \n
        nodoPelícula.appendChild(doc.createTextNode("\n"));
        // Para director
        Element nodoDirector = doc.createElement("Director");
        Text textNodoDirector = doc.createTextNode(director);
        nodoDirector.appendChild(textNodoDirector);
        nodoPelícula.appendChild(nodoDirector);
        // Engádese tamén un nodo text cun saldo de liña \n
        nodoPelícula.appendChild(doc.createTextNode("\n"));
        // e para estrea
        Element nodoEstreno = doc.createElement("Estrea");
        Text textNodoEstreno = doc.createTextNode(estrea);
        nodoEstreno.appendChild(textNodoEstreno);
        nodoPelícula.appendChild(nodoEstreno);
        // Engádese a película a películas, que é o primeiro nodo do 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 unha árbore DOM<sup>13</sup>

Para conseguir a persistencia dun documento XML, e por tanto dos datos nel contidos, pódese almacenar o disco como un ficheiro XML. Existen diversas maneiras sendo unha delas a seguinte.

### Exemplo 6: Gardar 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");

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


    // Establécese o fluxo de saída
    output.setByteStream(new FileOutputStream(ficheroSalida));
    //output.setByteStream(System.out);

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

    // Establécense as propiedades do serializador
    serializer.setNewLine("\r\n");
    serializer.getDomConfig().setParameter("format-pretty-print", true);

    // Escríbese o documento xa sexa nun ficheiro ou nunha cadea de texto
    serializer.write(document, output);
    // String xmlCad=serializer.writeToString(document);
}
```

<sup>13</sup> <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. SAX<sup>14</sup>

SAX (Simple API for XML), do mesmo xeito que DOM, ten como obxectivo procesar documentos XML para obter información deles. Con todo SAX funciona dunha forma totalmente diferente a DOM.

Caracterízase por:

- SAX non crea estruturas en memoria senón que percorre o ficheiro de forma secuencial desde o principio ata o final, ou ata que atopa un erro. Isto provoca un baixo consumo en memoria, en contraposición a DOM, máis marcado sobre todo en documentos de gran tamaño.
- O programador non ten control sobre o percorrido do ficheiro polo que só pode acceder ao elemento que o parser está accedendo nun momento dado. Isto implica que non ten unha visión global do documento nin pode “navegar” por el.
- A medida que o percorrido avanza sobre o ficheiro xéranse eventos que provoca que se invoquen métodos callback de SAX encargados de tratalos.

Estes eventos prodúcense cando SAX detecta:

- O comezo do documento XML.
- O comezo dun elemento XML.
- O contido dun elemento como unha cadea de caracteres.
- O final do documento XML.
- O final dun elemento XML.
- Que detecte un erro no procesamento do ficheiro XML.

Para utilizar SAX deberase implementar unha clase que actúe como parser e que debe herdar da clase `DefaultHandler`<sup>15</sup>. Esta clase incorpora os métodos callback<sup>16</sup> que son invocados de forma automática ao atoparse algún dos eventos indicados anteriormente.


É na clase definida e sobrescribindo os métodos callback onde se pode personalizar o comportamento do parser.

<sup>14</sup> [https://es.wikipedia.org/wiki/Simple\\_API\\_for\\_XML](https://es.wikipedia.org/wiki/Simple_API_for_XML)

[https://en.wikipedia.org/wiki/Simple\\_API\\_for\\_XML](https://en.wikipedia.org/wiki/Simple_API_for_XML)

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

<sup>16</sup> [https://es.wikipedia.org/wiki/Callback\\_\(informática\)](https://es.wikipedia.org/wiki/Callback_(informática))

	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 ficheiros XML usando SAX

De forma análoga a DOM se usarán as clases SAXParserFactory<sup>17</sup> e SAXParser<sup>18</sup> para procesar o ficheiro XML mediante SAX. Un exemplo de invocación do parser SAX é a seguinte:

Exemplo 7: Percorrido dun ficheiro 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 é a clase que se deberá
6     parser.parse(entradaXML, parserSax); // implementar e que hereda de DefaultHandler
7 }
```

Onde:

- Liña 1: o parámetro entrada pode ser tanto o path dun ficheiro almacenado en disco como a URI dun arquivo XML na internet.
- Liña 3: créase unha nova instancia da factoría que permite crear o parser XML.
- Liña 4: créase o obxecto parser que permite interpretar o documento XML.
- Liña 5: créase unha instancia da clase creada por nós que define as accións para levar a cabo cando se produza un evento e invóquense un método callback.
- Liña 6: procésase o documento XML de entrada mediante o parser definido no punto anterior.

### 3.2. Percorrer un documento XML con SAX

Como se comento no punto anterior para personalizar o comportamento do parser deberase sobrescribir os métodos callback da clase que herda de DefaultHandler. Só débense sobrescribir aqueles métodos que se desexen personalizar.

Entre outros se poden atopar os seguintes métodos callback que se invocan cando:

- startDocument: detéctase o principio dun documento XML.
- startElement: detéctase o principio dun elemento XML. É neste método onde se poden obter os atributos<sup>19</sup> dun elemento.
- endDocument: detéctase o fin dun documento XML.
- endElement: detéctase o fin dun elemento XML.
- characters: atópase un valor dun elemento como unha cadea de caracteres.

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

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

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

<b>COLEXIO</b> <b>VIVAS</b> 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:					

O seguinte documento XML permite mostrar a secuencia de invocación dos métodos callback a medida que se percorre, en orde, o ficheiro XML e vanse detectando os eventos indicados no punto 3.


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

Polo que para obter, mediante SAX, os títulos das películas poderíase implementar a seguinte clase:

```

Exemplo 8: Obtención dos títulos da películas
1 public class ParserSAX extends DefaultHandler{
2     boolean esTitulo=false;
3
4     @Override
5     public void startElement(String uri, String localName, String qName,
6                             Attributes attributes) throws SAXException {
7         if (qName.equals("Título")) this.esTitulo=true;
8     }
9
10    @Override
11    public void characters(char[] ch, int start, int length) throws SAXException {
12        if (esTitulo) {
13            String titulo=new String(ch, start, length);
14            System.out.println(titulo);
15            esTitulo=false;
16        }
17    }
18 }

```

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

Onde:

- Liña 4: detéctase o comezo dun elemento.
- Liña 6: mediante o parámetro qName coñécese o nome do elemento ao que se está accedendo. Utilízase para detectar o comezo dos elementos título.
- Liñas 9: o método characters é o utilizado para obter os valores dos elementos en SAX. Ten tres parámetros:
  - ch: contén todo o documento XML de entrada como un array de caracteres.
  - start: indica o índice de comezo do valor do elemento actual dentro do array.
  - length: indica a lonxitude de caracteres do valor do elemento actual.
- Liña 10: se se está nun título é que os caracteres que vén a continuación é o valor dun elemento título, é dicir, o valor buscado.
- Liña 11: créase unha nova cadea co tamaño do título.
- Liña 13: ponse o título a falso para que non volva acceder ata que atope un novo elemento título.

No seguinte exemplo móstrase como percorrer o documento XML mostrando o seu contido

```

Exemplo 9: Percorrido dun ficheiro 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("Comenzo do documento XML");
7      }
8
9      @Override
10     public void endDocument() throws SAXException {
11         System.out.println("Fin do 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("Ano: "+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 }

```

Onde:

- Liñas 4 e 8: Móstrase unha mensaxe descritiva ao comezo e ao final do percorrido do documento XML.
- Liña 14: establezo un valor ao atributo qName para coñecer, no método characters, o elemento que se está procesando neste momento.
- Liñas 15, 16, 17 e 18: se o elemento que se está procesando é unha película móstranse os seus atributos.
- Liña 24: cando se termina de procesar unha película móstrase un separador.
- Liña 25: elimínase o valor de qName para evitar que se mostre información non desexada no método characters.
- Liña 29: créase unha cadea co valor do elemento.
- Liñas 30, 31 e 22: móstrase a información dos elementos desexados.
- Liñas 35, 38 e 42: xestiónanse posibles avisos e erros ocorridos durante o percorrido do 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: Traballo con ficheiros HTML

Unha páxina web, do mesmo xeito que un ficheiro XML, segue o modelo DOM pero non ten por que estar ben formado polo que os parsers usados en XML non adoitan funcionar cunha páxina web.

Entre a serie de parsers específicos para ficheiros HTML (ver punto 4.1) nós centrarémonos en JSoup<sup>20</sup>.

JSoup é unha librería de Xava que proporciona unha API para extraer e manipular arquivos HTML. Entre outras posibilidades permítenos extraer toda a información dun sitio web (Web scraping)<sup>21</sup>.

Antes de usar JSoup deberemos obter o JAR<sup>22</sup> que contén a librería JSoup e incluílo no CLASSPATH ou engadilo ao noso IDE, por exemplo en Eclipse pulsamos no nome do proxecto co botón dereito do rato e seleccionamos Build Paths → Add External Archives e engadimos o ficheiro JAR coa librería.

Do mesmo xeito que DOM JSoup xera unha árbore en memoria polo cal podemos navegar. Esta árbore será un obxecto Document de tipo org.jsoup.nodes.Document.

Para xerar a árbore dispoñemos do método connect de Jsoup que entre dispón dos seguintes métodos:

- UserAgent: Establecer o userAgent da petición.
- Referrer: establecer a url desde a cal se fai a petición.
- Timeout: establecer o tempo máximo usado para establecer a conexión..

Un exemplo do seu uso para obter a árbore JSoup é:

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

Exemplos de métodos dispoñibles en JSoup para traballar coa árbore xerada son:

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

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

<sup>21</sup> Conseguir información dunha páxina web: [http://es.wikipedia.org/wiki/Web\\_scraping](http://es.wikipedia.org/wiki/Web_scraping)

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



<b>COLEXIO</b> <b>VIVAS</b> 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:					

- Elemento data:
  - attr(String key) pasa conseguir o atributo key e attr(String key, String value) para establecer o atributo key co valor value.
  - attributes para obter todos os atributos..
  - text(): Consegue o valor do texto que posúe un nodo.
  - html() e html(String value): consegue o código html asociado o nodo.
  - tag() e tagName(): consegue o nome do 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 os 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());
        }
    }
}
```

Á parte dos métodos estándar para atopar elementos JSoup soporta a mesma sintaxe que CSS para atopar elementos.

Usaremos o método select xunto un selector. Está dispoñible en Document, Element e en Elements. Devolve unha 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:					

Exemplos de selectores son:

➤ Sinxelos:


- `tagname`: atopa elementos por etiqueta, por exemplo `a`.
- `#id`: atopa elementos por ID, por exemplo `#logo`.
- `.class`: atopa elementos por class name, por exemplo `masthead`.
- `[attribute]`: atopa elementos co atributo attribute, por exemplo `[href]`.
- `[^attr]`: atopa elementos co prefixo prefix no nome.
- `[attr=value]`: elementos co atributo value, por exemplo `[width=500]`.
- `[attr^=value]`, `[attr$=value]`, `[attr*=value]`: elemento co atributo que empece con, termina con ou conten o valor value, por exemplo `[href*= /path/]`
- `[attr~=regex]`: atopa elementos no que o atributo cumpra con la expresión regular regex. Por exemplo `img[src~=(?i)\.(png|jpe?g)]`.
- `*`: atopa todos los elementos.

➤ Combinación de selectores.

- `el#id`: elementos con ID, por exemplo `div#logo`.
- `el.class`: elementos con clase, por exemplo `div.masthead`.
- `el[attr]`: elementos con atributo, por exemplo `a[href]`.
- calquera combinación, por exemplo `a[href].highlight`.
- `antecesor fillo`: elementos fillo que descendan directamente de antecesor, por exemplo `.body p` encontra elementos `p` dentro de bloques con la class "body".
- `padre > fillo`: Elementos fillo que descendan directamente de pai, por exemplo `body > *` encontra todos os fillos que descendan directamente de `body`.
- `el1, el2, el3`: un grupo de múltiples selectores. Atopa elementos que casa con algún dos selectores do grupo; por exemplo `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 exemplo `td:lt(3)`.
- `:gt(n)`: find elements whose sibling index is greater than n; por exemplo `div p:gt(2)`.
- `:eq(n)`: find elements whose sibling index is equal to n; por exemplo `form input:eq(1)`.

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

- `:has(selector)`: find elements that contain elements matching the selector; por exemplo `div:has(p)`.
- `:not(selector)`: find elements that do not match the selector; por exemplo `div:not(.logo)`.
- `:contains(text)`: find elements that contain the given text. The search is case-insensitive; por exemplo `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 exemplo `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.

#### Exemplo 13: Obter todas as ligazóns unha 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());
    }
}
```

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

#### 4.1. Comparativa de parsers HTML.<sup>23</sup>


O cadro seguinte mostra unha comparativa dalgúns parsers HTML.

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

\* Satinizar (xeración da páxina Web estándar) e limpar (eliminar as etiquetas de presentación excedentes, eliminar o código XSS, etc) código HTML.

\*\* Actualizacións HTML4.X a XHTML ou HTML5, converter etiquetas en desuso (ex. CENTER) en etiquetas válidas (ex. DIV with style=" text-align: center;").

<sup>23</sup> Fonte: [http://en.wikipedia.org/wiki/Comparison\\_of\\_HTML\\_parsers](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](#)