



# Acceso a Datos

Tema 2 XML

José Luis González Sánchez



# Contenidos

¿Qué voy a aprender?



# Contenidos

1. XML
2. DOM
3. Librerías
4. CRUD

# XML

Estructurando la información



# XML

Los ficheros XML permiten el intercambio de información entre aplicaciones utilizando para ello un fichero de texto plano al que se le pueden añadir etiquetas para darle significado a cada uno de los valores que se almacenan.

XML es el acrónimo de Extensible Markup Language, es decir, es un lenguaje de marcado que define un conjunto de reglas para la codificación de documentos.

El lenguaje de marcado es un conjunto de códigos que se pueden aplicar en el análisis de datos o la lectura de textos creados por computadoras o personas. El lenguaje XML proporciona una plataforma para definir elementos para crear un formato y generar un lenguaje personalizado.

Un archivo XML se divide en dos partes: prolog y body. La parte **prolog** consiste en metadatos administrativos, como declaración XML, instrucción de procesamiento opcional, declaración de tipo de documento y comentarios. La parte del **body** se compone de dos partes: estructural y de contenido (presente en los textos simples).

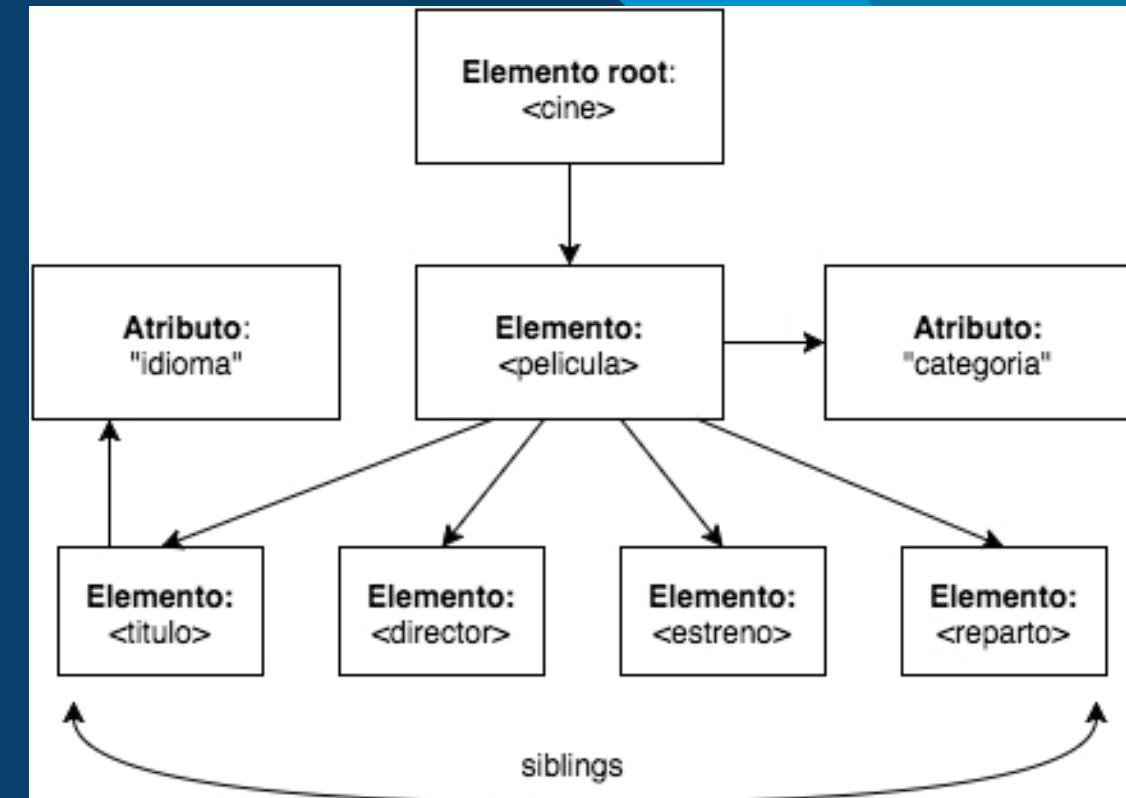
El diseño XML se centra en la simplicidad, la generalidad y la facilidad de uso y, por lo tanto, se utiliza para varios servicios web. Tanto es así que hay sistemas destinados a ayudar en la definición de lenguajes basados en XML, así como APIs que ayudan en el procesamiento de datos XML – que no deben confundirse con HTML o simplemente se puede usar para definir Interfaces de Usuario

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <cine>
3      <pelicula categoria="accion">
4          <titulo idioma="ingles">Mad Max</titulo>
5          <director>George Miller</director>
6          <estreno>15 mayo 2015</estreno>
7          <reparto>Tom Hardy</reparto>
8          <reparto>Charlize Theron</reparto>
9          <reparto>Nicholas Hoult</reparto>
10     </pelicula>
11     <pelicula categoria="animacion">
12         <titulo idioma="ingles">Inside Out</titulo>
13         <director>Pete Docter</director>
14         <estreno>17 junio 2015</estreno>
15         <reparto>Amy Poehler</reparto>
16         <reparto>Phyllis Smith</reparto>
17         <reparto>Bill Hader</reparto>
18     </pelicula>
19 </cine>
```

# XML

- Los documentos XML forman una estructura de tipo árbol, comenzando desde la raíz (root), con ramas hacia las hojas (leaves).

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <cine>
3   <pelicula categoria="accion">
4     <titulo idioma="ingles">Mad Max</titulo>
5     <director>George Miller</director>
6     <estreno>15 mayo 2015</estreno>
7     <reparto>Tom Hardy</reparto>
8     <reparto>Charlize Theron</reparto>
9     <reparto>Nicholas Hoult</reparto>
10    </pelicula>
11    <pelicula categoria="animacion">
12      <titulo idioma="ingles">Inside Out</titulo>
13      <director>Pete Docter</director>
14      <estreno>17 junio 2015</estreno>
15      <reparto>Amy Poehler</reparto>
16      <reparto>Phyllis Smith</reparto>
17      <reparto>Bill Hader</reparto>
18    </pelicula>
19  </cine>
```





# XML

- Los documentos XML están formados por árboles de elementos. Cada árbol comienza con un elemento root, del cual surgen ramas (branches) a elementos hijo. Todos los elementos pueden tener elementos hijo.
- Se suelen emplear los términos parent, child y sibling para determinar el parentesco: padre, hijo y hermano.
- Todos los elementos pueden tener contenido textual ("Mad Max") y atributos (categoria="accion").



# XML. Reglas básicas

- Todos los elementos XML deben tener un elemento root que es padre de todos los demás.
- La primera línea del documento se llama prolog. <?xml version="1.0" encoding="UTF-8" ?> Es opcional, pero si se incluye ha de ir en la primera línea. UTF-8 es la codificación de caracteres por defecto y la recomendable.
- Todos los elementos han de tener su etiqueta de cierre.
- Las etiquetas XML son sensibles a mayúsculas y minúsculas.
- Las etiquetas han de ir correctamente anidadas.
- Los atributos XML deben ir entre comillas .
- XML tiene caracteres especiales que han de utilizarse de distinta forma para evitar conflicto, son 5:
  - \$lt; <
  - > >
  - & &
  - ' '
  - " "
- Sólo < y & están estrictamente prohibidos en XML, pero es recomendable utilizar las demás también.
- La sintaxis para escribir comentarios en XML es como en HTML: <!-- Esto es un comentario -->



# XML. Elementos

- **Un elemento XML es cualquier cosa que esté incluída entre la etiqueta inicial y la final:**
  - <director>George Miller</director>
- Puede contener texto, atributos, otros elementos o una mezcla de los anteriores.
- También es posible que no contenga nada.
- **Reglas para los elementos:**
  - Son sensibles a mayúsculas y minúsculas
  - Deben empezar con una letra o barra baja
  - No pueden empezar con las letras xml
  - Pueden contener letras, dígitos, guiones, barras bajas y puntos
  - No pueden contener espacios
  - Lo ideal es mantener unos nombres sencillos, evitando caracteres como guiones y puntos.
  - Si los documentos XML tienen una base de datos correspondiente, lo normal es usar los nombres de la base de datos para los elementos.

```
 1  <?xml version="1.0" encoding="UTF-8"?>
 2  <cine>
 3   <pelicula categoria="accion">
 4     <titulo idioma="ingles">Mad Max</titulo>
 5     <director>George Miller</director>
 6     <estreno>15 mayo 2015</estreno>
 7     <reparto>Tom Hardy</reparto>
 8     <reparto>Charlize Theron</reparto>
 9     <reparto>Nicholas Hoult</reparto>
10   </pelicula>
11   <pelicula categoria="animacion">
12     <titulo idioma="ingles">Inside Out</titulo>
13     <director>Pete Docter</director>
14     <estreno>17 junio 2015</estreno>
15     <reparto>Amy Poehler</reparto>
16     <reparto>Phyllis Smith</reparto>
17     <reparto>Bill Hader</reparto>
18   </pelicula>
19 </cine>
```



# XML. Atributos

- Los elementos XML pueden tener atributos. Están diseñados para contener datos relacionados con un elemento específico. Han de ir siempre encerrados en comillas, ya sean simples o dobles:
  - <pelicula categoria="accion"> .... </pelicula>
- Los atributos de alguna forma están limitados. Al contrario que los elementos, los atributos:
  - No pueden contener múltiples valores
  - No pueden contener estructuras en árbol
  - No se pueden expandir fácilmente (para futuros cambios)
- A menudo los atributos se utilizan para asignar ID's a elementos.
- Lo ideal es guardar los metadatos (datos sobre datos) como atributos, y los datos en sí mismos como elementos.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <cine>
3      <pelicula categoria="accion">
4          <titulo idioma="ingles">Mad Max</titulo>
5          <director>George Miller</director>
6          <estreno>15 mayo 2015</estreno>
7          <reparto>Tom Hardy</reparto>
8          <reparto>Charlize Theron</reparto>
9          <reparto>Nicholas Hoult</reparto>
10     </pelicula>
11     <pelicula categoria="animacion">
12         <titulo idioma="ingles">Inside Out</titulo>
13         <director>Pete Docter</director>
14         <estreno>17 junio 2015</estreno>
15         <reparto>Amy Poehler</reparto>
16         <reparto>Phyllis Smith</reparto>
17         <reparto>Bill Hader</reparto>
18     </pelicula>
19 </cine>
```



# XML. Namespaces

- Los espacios de nombres o namespaces proporcionan una forma de evitar conflictos de nombres entre elementos. Los nombres de elementos pueden coincidir entre diferentes documentos XML.
  - La solución es emplear un prefijo.
  - Pero para emplear un prefijo, hay que establecer un espacio de nombres. El espacio de nombres se define con el atributo xmlns en la etiqueta inicial del elemento. La sintaxis es: xmlns:prefijo="URI".
  - Cuando se define un namespace para un elemento, todos los elementos hijo con el mismo prefijo están asociados al mismo namespace.
  - También pueden declararse namespaces en el elemento root.
  - La URI no es para buscar información en esa URL, sino que se utiliza para darle un nombre único al namespace. Se suele emplear una URL porque las empresas a veces proporcionan información sobre dicho namespace en esa URL.
  - URI viene de Uniform Resource Identifier, y es un string de caracteres que identifican una Internet Resource. La URI más común es la Uniform Resource Locator (URL) que identifica un dominio. Otro no tan común es el Universal Resource Name (URN).



```
1 <root xmlns:h="http://www.w3.org/TR/html5/"  
2   xmlns:m="http://www.ejemplo.com/muebles">  
3   <h:table>  
4     <h:tr>  
5       <h:td>León</h:td>  
6       <h:td>Tigre</h:td>  
7     </h:tr>  
8   </h:table>  
9   <m:table >  
10    <m:name>Mesa de café</m:name>  
11    <m:colour>Marrón</m:colour>  
12  </m:table>  
13 </root>
```



# XML. XSLT

- XSLT son las siglas de eXtensible Stylesheet Language Transformations, y es el lenguaje de hojas de estilo de XML.
- XSLT es más sofisticado que CSS. Con XSLT se pueden añadir/eliminar elementos y atributos del archivo de salida. Se pueden también reordenar elementos, hacer tests y decidir qué elementos mostrar o quitar, entre otras muchas cosas. XSLT utiliza XPath para encontrar información en un documento XML (explicado en la siguiente sección).

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <?xml-stylesheet type="text/xsl" href="hojaxslt.xsl"?>
3
4  <cine>
5      <pelicula categoria="accion">
6          <titulo idioma="ingles">Mad Max</titulo>
7          <director>George Miller</director>
8          <estreno>15 mayo 2015</estreno>
9          <reparto>Tom Hardy</reparto>
10         <reparto>Charlize Theron</reparto>
11         <reparto>Nicholas Hoult</reparto>
12     </pelicula>
13     <pelicula categoria="animacion">
14         <titulo idioma="ingles">Inside Out</titulo>
15         <director>Pete Docter</director>
16         <estreno>17 junio 2015</estreno>
17         <reparto>Amy Poehler</reparto>
18         <reparto>Phyllis Smith</reparto>
19         <reparto>Bill Hader</reparto>
20     </pelicula>
21 </cine>
```

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <html xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
3      <xsl:for-each select="cine/pelicula">
4          <div style="background-color:#000;color:white;padding:6px">
5              <xsl:value-of select="titulo"/>
6          - 
7              <xsl:value-of select="director"/>
8          </div>
9          <div style="margin-left:20px;margin-bottom:1em;font-size:14pt">
10             <p>
11                 Reparto:
12                 <xsl:for-each select="reparto">
13                     <p>
14                         <xsl:value-of select="."/>
15                     </p>
16                 </xsl:for-each>
17                 </p>
18             </div>
19             <xsl:for-each>
20                 </p>
21             </xsl:for-each>
22         </body>
23     </html>
```

## Mad Max - George Miller

Reparto:

*Tom Hardy*

*Charlize Theron*

*Nicholas Hoult*

## Inside Out - Pete Docter

Reparto:

*Amy Poehler*

*Phyllis Smith*

*Bill Hader*



# XML. Xpath

- **XPath es un lenguaje de ruta de XML para encontrar información en un documento XML. XPath es una sintaxis para definir partes de un documento XML, fundamental para las hojas XSLT. XPath utiliza expresiones de ruta para navegar por el documento, y contiene una librería con funciones estándar. XPath es una recomendación W3C y se utiliza también en XQuery, XSLT y XLink.**
- XPath utiliza expresiones de ruta para seleccionar nodos y conjuntos de nodos en un documento XML. Estas rutas son muy parecidas a las que se utilizan en los sistemas de ordenadores tradicionales.
- Las expresiones XPath pueden utilizarse en JavaScript, Java, XML Schema, PHP, Python, C, C++...
- Sobre nuestro ejemplo:
  - /cine/pelicula[1] Selecciona el primer elemento pelicula del elemento cine
  - /cine/pelicula[last()] Selecciona el último elemento pelicula del elemento cine
  - /cine/pelicula[last()-1] Selecciona el penúltimo elemento pelicula del elemento cine
  - /cine/pelicula[position()<3] Selecciona los dos primeros elementos pelicula del elemento cine
  - //titulo[@idioma] Selecciona todos los elementos titulo que tienen el atributo idioma
  - //titulo[@idioma='ingles'] Selecciona todos los elementos titulo que tienen el atributo idioma con el valor inglés
  - /cine/pelicula[ticket>8] Selecciona todos los elementos pelicula del elemento cine que tienen un elemento ticket mayor que 8
  - /cine/pelicula[ticket>8]/titulo Selecciona todos los elementos titulo de los elementos pelicula que tienen un elemento ticket cuyo valor es mayor que 8



# XML. XLink

- **XLink se utiliza para crear hiperenlaces en documentos XML.**  
**Cualquier elemento en un documento XML puede comportarse como un enlace, y con XLink los enlaces pueden definirse desde fuera de los archivos enlazados.**
- Para poder acceder a las características de XLink se ha de incluir el namespace XLink: "http://www.w3.org/1999/xlink".
- Los atributos xlink:type y xlink:href vienen del namespace XLink.
- xlink:type="simple" crea un enlace normal de HTML. xlink:href indica la URL a la que dirigirse.
- El atributo xlink:show="new" especifica que el enlace se debe abrir en una nueva ventana.
- Si por ejemplo en el atributo xlink:show ponemos el valor "embed", el resource se procesará en la misma página. Si este resource es otro documento XML, se puede crear una jerarquía de documentos XML.
- También es posible especificar cuándo debe aparecer el resource, con el atributo xlink:actuate.

```
● ● ●
1  <?xml version="1.0" encoding="UTF-8"?>
2  <cine xmlns:xlink="http://www.w3.org/1999/xlink">
3      <pelicula title="Mad Max">
4          <description xlink:type="simple" xlink:href="/imagenes/madmax.gif" xlink:show="new">
5              Descripción de la película Mad Max
6          </description>
7      </pelicula>
8      <pelicula title="Inside Out">
9          <description xlink:type="simple" xlink:href="/imagenes/insideout.gif" xlink:show="new">
10             Descripción de la película Inside Out
11         </description>
12     </pelicula>
```



# XML. XLink

- **Referencia de atributos XLink:**
- **xlink:actuate**
  - onLoad, onRequest, other, none
  - Define cuando se ha de leer y mostrar el resource enlazado
- **xlink:href**
  - URL
  - Especifica la URL a la que enlazar
- **xlink:show**
  - embed, new, replace, other, none
  - Especifica donde abrir el enlace. Por defecto es "replace"
- **xlink:type**
  - simple, extended, locator, arc, resource, title, none
  - Especifica el tipo de enlace

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <cine xmlns:xlink="http://www.w3.org/1999/xlink">
3      <pelicula title="Mad Max">
4          <description xlink:type="simple" xlink:href="/imagenes/madmax.gif" xlink:show="new">
5              Descripción de la pelicula Mad Max
6          </description>
7      </pelicula>
8      <pelicula title="Inside Out">
9          <description xlink:type="simple" xlink:href="/imagenes/insideout.gif" xlink:show="new">
10             Descripción de la pelicula Inside Out
11         </description>
12     </pelicula>
```



# XML. XPointer

- **XPointer permite a los enlaces apuntar a partes específicas de un documento XML. XPointer utiliza expresiones XPath para navegar por el documento XML.**
- En lugar de enlazar al documento entero como con XLink, XPointer permite enlazar partes específicas. Para ello se añade una almohadilla # y una expresión XPointer después de la URL en el atributo xlink:href

```
● ● ●  
1 <cine>  
2   <mifavorita>  
3     <imagen url="http://ejemplo.com/madmax.gif" />  
4     <enlace xlink:type="simple" xlink:href="http://ejemplo.com/peliculas.xml#34">  
5       Enlace a mi película favorita</enlace>  
6   </mifavorita>  
7 </cine>
```



# XML. DTD

- **Document Type Definition, que son reglas que definen los elementos y atributos permitidos.**  
**Un DTD permite que un grupo de personas acuerden utilizar un DTD estándar para el intercambio de datos y así poder asegurar una uniformidad.**
- **Si se cumple, se dice que el XML es bien formado.**
  - !DOCTYPE pelicula define que el elemento root del documento es pelicula.
  - !ELEMENT pelicula define que el elemento pelicula debe contener los elementos titulo, estreno, director y reparto.
  - !ELEMENT elemento (#PCDATA) define el tipo de elemento. #PCDATA significa datos de texto interpretables.
  - Un DTD puede emplearse también para definir caracteres especiales en un documento.



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE pelicula SYSTEM "pelicula.dtd">
```



```
1 <!DOCTYPE pelicula
2 [
3 <!ELEMENT pelicula (titulo,estreno,director,reparto)
4 <!ELEMENT titulo (#PCDATA)>
5 <!ELEMENT estreno (#PCDATA)>
6 <!ELEMENT director (#PCDATA)>
7 <!ELEMENT reparto (#PCDATA)>
8 ]>
```



# XML. XML Schemas

- **Un esquema XML o XML Schema describe la estructura de un documento XML, como un DTD. Es una alternativa a DTD basada en XML**
- Sobre el ejemplo:
  - define el elemento llamado pelicula.
  - el elemento pelicula es del tipo complex.
  - el tipo complex es una secuencia de elementos.
  - define que el elemento "elemento" ha de ser del tipo string.
- Los XML Schemas son más potentes que los DTD porque están escritos en XML, se pueden extender y soportan tipos de datos y namespaces.



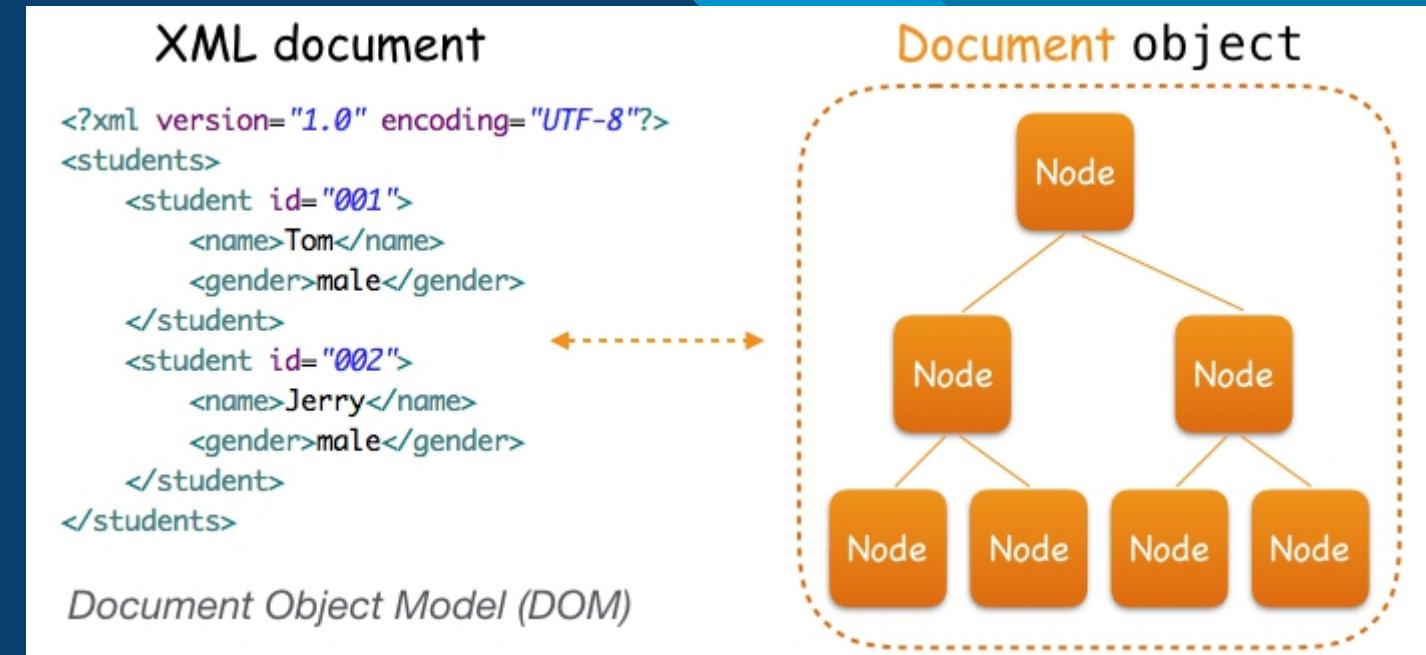
```
1 <xs:element name="pelicula">
2   <xs:complexType>
3     <xs:sequence>
4       <xs:element name="titulo" type="xs:string" />
5       <xs:element name="director" type="xs:string" />
6       <xs:element name="reparto" type="xs:string" />
7     </xs:sequence>
8   </xs:complexType>
9 </xs:element>
```

# DOM

Modelo de Objetos del Documento

# DOM

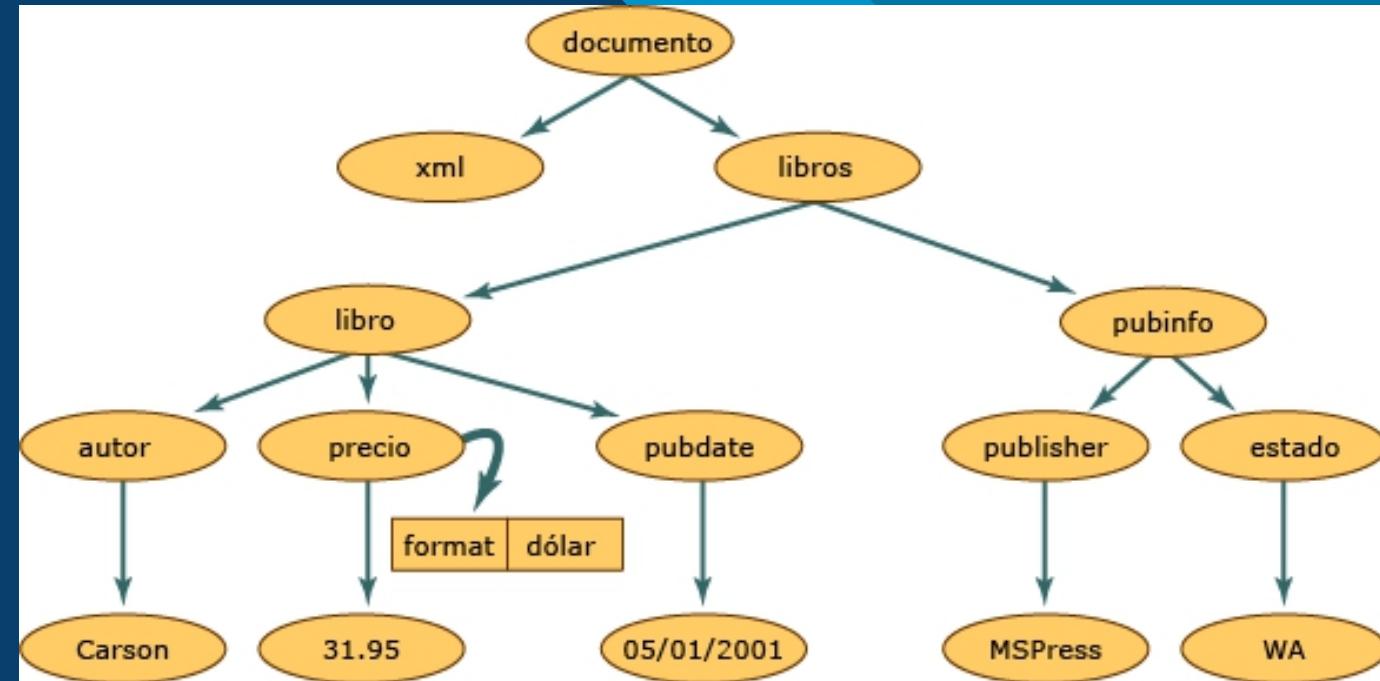
- Document Object Model o DOM ('Modelo de Objetos del Documento' o 'Modelo en Objetos para la Representación de Documentos')
- Es esencialmente una interfaz de plataforma que proporciona un conjunto estándar de objetos para representar documentos HTML, XHTML y XML, un modelo estándar sobre cómo pueden combinarse dichos objetos, y una interfaz estándar para acceder a ellos y manipularlos.
- A través del DOM, los programas pueden acceder y modificar el contenido, estructura y estilo de los documentos HTML y XML, que es para lo que se diseñó principalmente.
- El responsable del DOM es el World Wide Web Consortium (W3C).
- El DOM permite el acceso dinámico a través de la programación para acceder, añadir y cambiar dinámicamente contenido estructurado en documentos con lenguajes de programación.





# DOM

```
● ● ●  
1  <?xml version="1.0"?>  
2  <libros>  
3  <libro>  
4  <autor>Carson</autor>  
5  <precio format="dollar">31.95</precio>  
6  <pubdate>05/01/2001</pubdate>  
7  </libro>  
8  <pubinfo>  
9  <publisher>MSPress</publisher>  
10 <estado>WA</estado>  
11 </pubinfo>  
12 </libros>
```



# Librerías

Trabajando en JAVA



# Librerías

- **DOM Parser:** Carga el fichero en memoria, creando la representación del mismo. Trabajamos nodo a nodo. Recomendable para ficheros pequeños
  - [https://www.tutorialspoint.com/java\\_xml/java\\_dom\\_parser.htm](https://www.tutorialspoint.com/java_xml/java_dom_parser.htm)
- **SAX Parser:** No carga el fichero en memoria ni crea representación del mismo. Los analizadores SAX están basados en un modelo de eventos: a medida que el parser recorre el documento éste informa de la ocurrencia de eventos, tales como el comienzo de un elemento XML o el final del documento, a un manejador de eventos (event handler).
  - [https://www.tutorialspoint.com/java\\_xml/java\\_sax\\_parser.htm](https://www.tutorialspoint.com/java_xml/java_sax_parser.htm)
- **JDOM Parser:** Nos permite leer, editar y escribir XML muy fácilmente y nos permite poder elegir entre SAX Parser, DOM Parser, STAX Event Parser, y STAX Stream Parser.
  - Recomendable usar Maven: <https://mvnrepository.com/search?q=jdom>
  - [https://www.tutorialspoint.com/java\\_xml/java\\_jdom\\_parser.htm](https://www.tutorialspoint.com/java_xml/java_jdom_parser.htm)
- **JAXB: Java Architecture for XML Binding (JAXB) define como los objetos son convertidos en JAVA desde/a XML.**
  - <https://www.baeldung.com/jaxb>



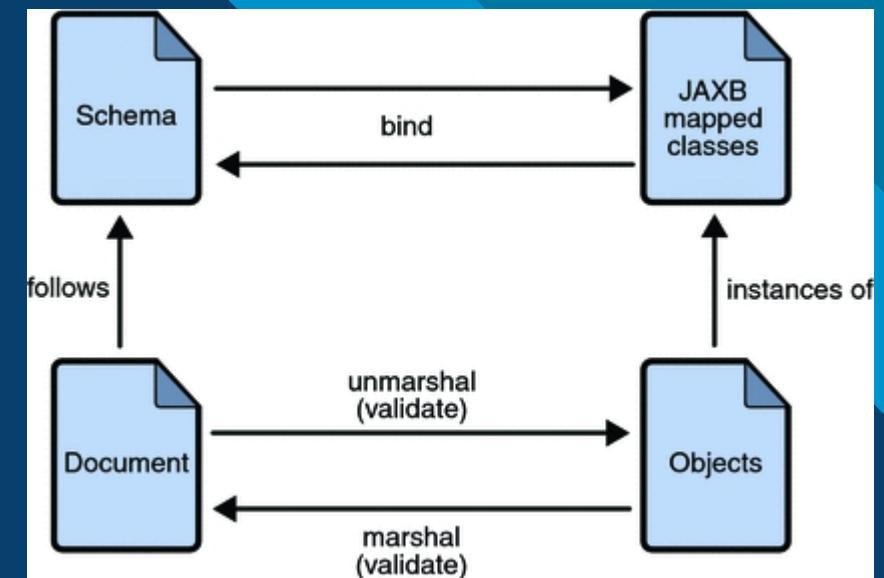
# Librerías. JAXB

- JAXB permite mapear clases Java a representaciones en XML y viceversa.
- JAXB proporciona dos principales características:
  - La capacidad de serializar (marshalling) objetos Java a XML.
  - Lo inverso, es decir, deserializar (unmarshalling) XML a objetos Java.
- O sea que JAXB permite **almacenar y recuperar datos en memoria en cualquier formato XML, sin la necesidad de implementar un conjunto específico de rutinas XML de carga y salvaguarda para la estructura de clases del programa.**
- El compilador de JAXB (schema compiler) permite generar una serie de clases Java que podrán ser llamadas desde nuestras aplicaciones a través de métodos sets y gets para obtener o establecer los datos de un documento XML.
- En Java 9 y Java 10 debemos usar --add-modules=java.xml.bind.
- In Java 11, JAXB está fuera deñ JDK, debemos usarlo via Maven o Gradle.
  - <https://mvnrepository.com/search?q=jaxb>



# Librerías. JAXB

- El funcionamiento esquemático al usar JAXB sería:
  - Crear un esquema (fichero .xsd) que contendrá las estructura de las clases que deseamos utilizar.
  - Compilar con el JAXB compiler ese fichero .xsd, de modo que nos producirá los POJOs, o sea, una clase por cada uno de los tipos que hayamos especificado en el fichero .xsd. Esto nos producirá los ficheros .java.
  - Compilar esas clases java.
  - Crear un documento XML: validado por su correspondiente esquema XML, o sea el fichero .xsd, se crea un árbol de objetos.
  - Ahora se puede parsear el documento XML, accediendo a los métodos gets y sets del árbol de objetos generados por el proceso anterior. Así se podrá modificar o añadir datos.
  - Después de realizar los cambios que se estimen, se realiza un proceso para sobrescribir el documento XML o crear un nuevo documento XML.



# CRUD

Create, Read, Update & Delete

# CRUD



“

"XML no es más lenguaje de  
programación que unas notas sobre una  
servilleta de papel"

- Charles Simonyi

”



# Recursos

- Twitter: <https://twitter.com/joseluisgonsan>
- GitHub: <https://github.com/joseluisgs>
- Web: <https://joseluisgs.github.io>
- Discord: <https://discord.gg/eFMKxfPY>
- Aula Virtual: <https://aulavirtual33.educa.madrid.org/ies.luisvives.leganes/course/view.php?id=247>



# Gracias

José Luis González Sánchez

