INTRODUCCIÓN A XML (v.2)

©Luis Pilo Aceituno. Curso 2021-2022 *lpilo@educarex.es*

1.INSTALACIÓN DEL ENTORNO DE TRABAJO.

Proponemos emplear como entorno de trabajo ALTOVA XML Spy. La página oficial para hacer la descarga del software es: https://www.altova.com/es/xmlspy-xml-editor

También puedes emplear como entorno de trabajo NetBeans. La página oficial para hacer la descarga del software es: https://netbeans.apache.org//.

- Otras son posibilidad es XML Copy Editor y notepad++

Si te decides a usar NetBeans tienes, entre otras, las siguientes opciones.

- chequear y validar

- Ver el documento. Botón derecho > View

2.INTRODUCCIÓN A XML

- XML es una herramienta para almacenar y transportar datos. Sus siglas significan. Extensible Markup Language
- XML Sirve para enviar y recibir información.Se ocupara de almacenar la información que será interpretada por un programa
- Podemos emplearlo para
    - Intercambiar información entre sistemas.
    - Usarlo como bases de datos
    - Emplearlo para definir otros lenguajes
- XML y HTML presentan las siguientes diferencias.
    - XML fue diseñado para centrarse en los datos mientras que HTML se centra más en la presentación de estos (es decir en su visualización)
    - XML es extensible, es decir, el programador puede definir sus propias etiquetas,
    - XML no tiene etiquetas predefinidas
    - XML es sensible a mayúsculas.
- El aspecto de un documento XML es similar a este,

```
<?xml version="1.0" encoding="UTF-8"?>
<tutoriales>
        <tutorial>
                <nombre>XML Y SUS TECNOLOGÍAS</nombre>
                <url>https://www.udemy.com</url>
        </tutorial>
        <tutorial>
                <nombre>HTML Tutorial</nombre>
                 <url>https://www.udemy.com</url>
         </tutorial>
</tutoriales>
```

ACTIVIDAD 1. Busca información sobre los componentes de un documento XML describirlos brevemente y poner algunos ejemplos

3.ESTRUCTURA DE UN DOCUMENTO XML

1.Árboles XML

- Los documentos XML se componen de texto plano y etiquetas definidas por el desarrollador. estas etiquetas se anidan en una estructura de árbol.
- Está estructura de árbol comienza con un elemento raíz que debe ser único y del que nacen las ramas.
- Si nos colocamos en un elemento, el elemento por encima de él es el elemento padre y los elementos inferiores son los elementos hijos. Los elementos de su mismo nivel son los hermanos.
- La estructura de un fichero XML es similar a está.

1. Comienza con una declaración que indica que es un fichero XML

```
<?xml version="1.0" encoding="UTF-8"?>
```

2. Siguen unas instrucciones de procesamiento que indican información al programa para que procese el documento. Estas instrucciones son optativas,

```
<?xml-stylesheet type="text/css" href="estilo-libreria.css"?>
```

.

3. Siguen con el elemento raíz

<libreria>

4. A continuación las ramas con sus hijos

<libro>

Llegados a este punto debemos mencionar dos conceptos importantes.

- Documento XML bien formado es el que no tiene errores de sintaxis.
- Documento XML válido es el que además de no tener errores de sintaxis no incumple ninguna de las normas establecidas en su estructura (DTD, XML Schema,..)

4.ETIQUETAS

Distinguimos estas etiquetas.

- Etiqueta Raíz (Root). Se ubica por encima de cualquier elemento y sirve como punto de partida para recorrer el árbol y ubicar el resto de nodos. Es el elemento que contiene al resto de los elementos.
- Elementos. Es la unidad básica de un XML. Se identifica por la etiqueta de apertura y cierre. En su interior contiene información, es decir otros elementos, o puede estar vacía

```
<tag>ejemplo1234</tag>
<tag_vacia/>
```

- Atributos.Son similares a los atributos de HTML. Permiten almacenar datos adicionales acerca de un elemento.

```
<tag atributo="ejemplo"/>
```

- Los atributos se utilizan para distinguir entre elementos del mismo nombre. Cuando no se desea crear un nuevo elemento en cada situación. Los atributos pueden agregar un poco más de detalles a la hora de diferenciar dos o más elementos similares.

- Ejemplo: En archivo xml que te muestro se distingue una película de otra ,aunque todas tengan los mismos datos, por el género al que pertenece.En caso de realizar una búsqueda podemos evitar procesar toda la ficha de películas viendo si el atributo género corresponde al que buscamos

```
<cartelera>
        <pelicula genero="  ">
             <titulo >

             </titulo>
        </pelicula>
</cartelera>
```

- Texto. Puedes incluirlo en el interior de un elemento o de un atributo. No puedes incluirlo en otro lugar, y los espacios reciben un tratamiento especial.

Tabulador	*\t*	*&#9*	*TAB*
Nueva Línea	*\n*	*&#10*	*LF*
Espacio	*\s*	*&#32*	*Barra Espaciadora*

- Comentario. Empiezan por <!-- y se cierran con ..>. Pueden colocarse en cualquier parte del documento, excepto dentro de una etiqueta o antes de la declaración del documento. <?xml …?>
- Entidades predefinidas. Son caracteres reservados de XML.Si se desea que sean interpretados como texto deben emplearse sentencias de escape.

&	&
<	<
>	>
'	'
"	"

- Secciones CDATA. Es un conjunto de caracteres que el procesador no analiza. No puede aparecer antes del elemento raíz, ni después de su cierre y no pueden contener al propio delimitarse de final de sección CDATA . Este delimitador es (]])
    - CDATA se usa para escapar algunos caracteres y evitar que sean tratados como XML normal.Los datos contenidos no se analizaran.
    - Podemos incluir fragmentos de código, expresiones matemáticas etc

```
<codigo>
    <![CDATA[<script>alert("Esto es un fragmento CDATA");</script>]]>
</codigo>
```

5.DOCUMENTO VÁLIDO Y BIEN FORMADO.

- Los ficheros denominados DTD y XSD (XML Schema) nos permiten definir reglas que amplían las restricciones sobre la sintaxis de un XML.
    - Un documento XML que cumple con las restricciones allí declaradas se dice que es un documento XML válido
- Algunas reglas referidas a la forma correcta de denominar las etiquetas y atributos son las siguientes.
    - Pueden comenzarse con una letra que puede ser de un alfabeto no latino.
    - Los demás caracteres pueden ser letras, números, guiones bajos , comas y dos puntos
    - Los nombres que empiezan por XML (mayúsculas o minúsculas) quedan reservados para la estandarización.
    - No puedes incluir ningún espacio, ni ningún signo que sea diferente de los citados con anterioridad.

1.Documento XML bien formado

- Un documento XML estará bien formado si cumple con una serie de reglas establecidas por el W3C.

ACTIVIDAD 2. Busca las reglas que debe cumplir un documento bien formado y comenta algunas de ellas

- Algunas de las más importantes son.

- El documento puede empezar por una instrucción que indica la **versión** de XML y opcionalmente, el **encoding** y requerimiento de archivos externos (**standalone**). Es muy recomendable, aunque no obligatorio.
- Solo puede existir **un único elemento raíz**. Este elemento contendrá a todos los demás elementos.
- Los elementos que no estén vacíos deberán tener una etiqueta de **apertura y otra de cierre**, mientras que los elementos vacíos deberán cerrarse con />
- Los elementos tienen que aparecer correctamente **anidados y no solaparse**.
- Los nombres de etiquetas y atributos son **case sensitive** (sensibles a mayúsculas y minúsculas).
- Los valores de los atributos deben aparecer entre **comillas** simples o dobles, pero siempre del mismo tipo.
- No puedes incluir dos atributos con el mismo nombre que estén asociados a un mismo elemento.
- No puedes escribir nada antes de la instrucción **<?xml ... ?>**
- No pueden aparecer los signos '<' ni '**&**' en el contenido textual de elementos ni atributos, salvo algunas excepciones.

- Para verificar que un documento XML está bien formado podemos abrir el navegador (es preferible Firefox). Si se muestra el árbol de nodos significa que está bien formado

ACTIVIDAD 3. Realiza los siguientes ejercicios del listado nº 1 y nº 2

Ejercicios de Estructura

1. Estructura un documento XML que pueda almacenar la siguiente información:

Ciudades

Nombre	País	Continente
Calcuta	India	Asia
Oporto	Portugal	Europa
Alejandría	Egipto	África

2. Escribe un documento XML que almacene la siguiente información:

Hechos Históricos

Descripción de cada hecho	*Fecha*		
	Día	*Mes*	*Año*
IBM da a conocer el PC.	12	8	1981
Se funda Google.	4	9	1998
Se funda Facebook.	4	2	2004

SOLUCIONES

- Ejercicio nº 1
- Ejercicio nº 2

6.XML CON ESTILOS

- Podemos dar estilos a un archivo xml para mejorar su presentación. Para ello tenemos dos opciones.
    - Archivos css. Se vinculan al archivo xml mediante la instrucción
        - ***<?xml-stylesheet type="text/css" href="nombre_archivo.css"?>***
        - Está instrucción debe ser la segunda del archivo xml, justamente después de la instrucción de declaración
    - Archivos xsl. Las opciones son mayores que la de los archivos css. Lo veremos en profundidad durante el curso.

ACTIVIDAD Nº 4. Realiza el ejercicio nº 3.

3. Dado el archivo "articulosStyles.css" cuyo contenido es:

```
nombre{color:blue;font-size:20px}
precio{color:green;font-size:28px}
```

Escribe un documento XML asociado a un archivo de estilos CSS y que represente la siguiente información.

Artículos

Nombre	*Precio*
Sofá	642
Mesa	305
Lámpara	89

- La salida debe ser similar a está.

SOLUCIÓN

- Archivo xml
- Archivo.css

ACTIVIDAD Nº 5. Realiza el siguiente ejercicio.

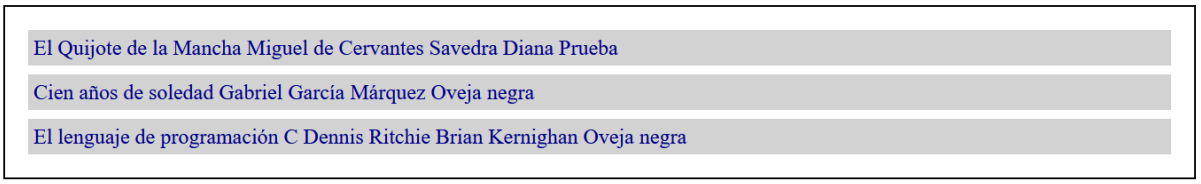
4. Queremos almacenar los datos de un formulario web. Para ello, se decide almacenar la información empleando un fichero XML. Del formulario se almacena el correo electrónico, el nombre de usuario, el país del que procede y si desea o no una suscripción a la revista online de la web. A parte, la web proporciona información sobre la fecha (día, mes y año) en el que se envía el formulario.

Estructura el documento XML para de modo que pueda almacenar esta información y almacena información sobre tres usuarios que inventes.

SOLUCIÓN.

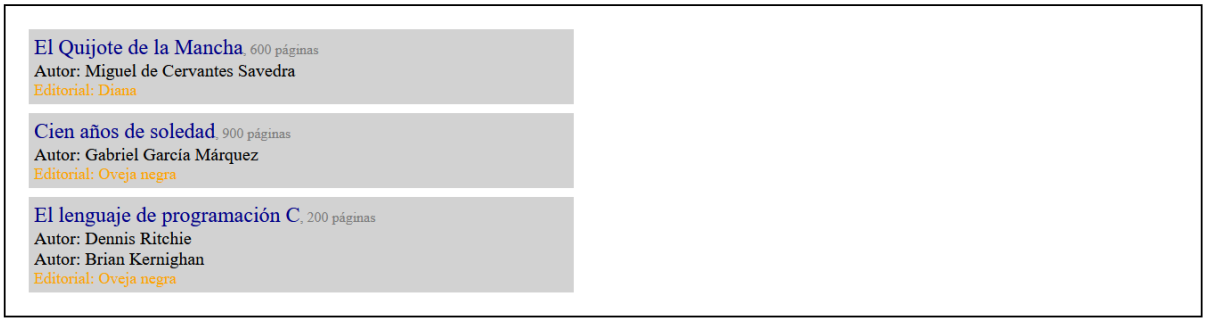
- Archivo xml

***ACTIVIDAD Nº 5.1*. Crea un archivo XML y el CSS correspondiente que permita dar estilos similares a los de la imagen**



Solución XML Ejercicio 5.1
Solución CSS Ejercicio 5.1

***ACTIVIDAD Nº 5.1 (ACTIVIDAD EVALUABLE)*. Crea un archivo XML y el CSS correspondiente que permita dar estilos similares a los de la imagen**



Solución XML Ejercicio 5.2.
Solución CSS Ejercicio 5.2

ACTIVIDAD Nº 6. Realiza los siguientes ejercicios

- ***Debes entregarlos a tu docente en la forma y plazos que él establezca.***

1.- Estructura un documento XML en el que pueda almacenarse la siguiente información acerca de los habitantes de un pueblo:

- ***Nombre***

- *Dirección (calle, número, piso)*
- *email*
- *mascota*

Una vez estructurado incluye información sobre dos habitantes

2.- Una empresa está almacenando información acerca de los equipos con los que trabajan sus informáticos para poder saber si alguno necesita actualizarse en un futuro. De cada equipo se guarda el nombre del operario que suele utilizarlo y el sistema operativo que tiene instalado. Además, se debe conocer su marca, modelo y si es portátil o de sobremesa. Del hardware, se quiere almacenar el procesado y la cantidad de memoria de la que dispone (RAM, HDD y SSD). Por último, se quiere almacenar si se conecta a la red de la empresa empleando una tarjeta WIFI o Ethernet. Incluye información de al menos dos equipos.

ACTIVIDAD Nº 7. Realiza los siguientes ejercicios

- *Debes entregarlos a tu docente en la forma y plazos que él establezca.*

EL Objetivo de estos ejercicios es que practiques la sintaxis correcta del XML para ello te propongo que compruebes su validación. Debes realizarlo en dos fases

1. *Intenta corregirlo sin el uso de ninguna herramienta software, observando tan solo su estructura*
2. *Comprueba su correcto una vez que has corregido los errores usando alguna herramienta, como NetBeans*

1. Corrige los errores que hay en el siguiente extracto de “frutas.xml” para que esté bien formado.

```
<?xml version="1.0" encoding="UTF-8">
< frutas >
        < fruta >
                < nombre >cereza< nombre \>
        < fruta \>
        < fruta >
                < nombre >naranja< nombre \>
        < fruta \>
< frutas \>
```

2. Corrige los errores que encuentres en el siguiente código (“vehiculos.xml”) para que esté bien formado. Puede ser necesario crear nuevas etiquetas o atributos.

```
<!-- Documento XML con errores de sintaxis. --!>
<? xml versión="1.0" encodin = "UTF-8" >
<terrestres>
        <vehiculo>bicicleta<vehiculo>
        <vehiculo>coche<vehiculo>
        <vehiculo>tractor<vehiculo>
<acuaticos>
        <vehiculo>canoa<vehiculo>
<aereos>
        <vehiculo>avioneta<vehiculo>
        <vehiculo>helicóptero<vehiculo>
```

3. Tienes que corregir los errores que hay en los siguientes ejemplos XML. Recuerda que puede ser necesario crear nuevas etiquetas o atributos.

```
<?xml version="1.0" encoding="UTF-8"?>
<figuras>
        <figura plana>
                <nombre>cuadrado</nombre>
                </lados 4>
        </figura>
        <figura plana>
                <nombre>triángulo</nombre>
                </lados 3>
        </figura>
        <figura tridimensional>
                <nombre>cubo</nombre>
                </aristas 12>
                </caras 6>
        </figura>
</figuras>
```

-

```
<?Xml version="1,0" encoding="UTF8"?>
<triangulo base="7"altura="5">
<triangulo base="2"altura="6">
<triangulo base="3"altura="3">
```

-

```
<?xml version="1.0" encoding="UTF-8"?>
<numeros>
        <1 letra="u" letra="n" letra="o">1</>
        <2 letra="d" letra="o" letra="s">22</>
        <6 letra="s" letra="e" letra="i" letra="s">666666</>
</numeros>
```

Solución de los ejercicios propuestos.

Ejercicio 1
Ejercicio 2
Ejercicio 3.1
Ejercicio 3.2
Ejercicio 3.3 (Posible ejercicio de examen)

ACTIVIDAD DE AULA nº 1

- Realiza los siguientes ejercicios

1. Construir XML de una cartelera de cine. Para ello, primero definimos la información que se almacena de una película tipo. La siguiente table muestra la información básica:

TRON
Título original :"Tron". EEUU, 1982. 96min.
Director: Steven Lisberger
Actores: Jeff Bridges, Bruce Boxleitner, David Warner, Cindy Morgan, Barnard Hughes ...
Kevin Flynn es un programador joven y presumido que trabajaba en la megacorporación ENCOM. Uno de los ejecutivos de esta corporación es Dillinger . Flynn fue engañado por Dillinger respecto a las ganancias y autoría de los juegos que ha creado. De hecho, Dillinger vendió los videojuegos de Flynn y pasaron a su propiedad. Ante la incapacidad de probar que él es el autor, y renunciando a la compañía, Flynn se ve obligado a trabajar en arcades. Muchos de los juegos que él mismo ha creado se encuentran en su local de *arcade*.
Www.tron.com
Ciencia Ficción. Autorizada para mayores de 7 años.

Una cartelera está compuesta de más de una película. Debes tener en cuenta que el título original solo aparecerá cuando la película no sea hispana, y que no siempre existe una web con la información de la película. Además, se requiere guardar información sobre el fichero gráfico que contiene el cartel de la película, aunque este fichero no siempre está disponible.

Debes generar un documento XML que contenga al menos 5 películas y que contemple las diferentes posibilidades que se puedan dar.

Solución .

ACTIVIDAD DE AULA nº 2

- ***Crea un documento xml que guarde información sobre la página web de un supermercado***
- ***Debe incluirse información sobre al menos 5 productos diferentes***
- ***Proporciona la mayor información que puedas, es decir la mayor complejidad posible de nodos y atributos.***
    - ***Más tarde este fichero se emplea con ejercicios de XPath y XQuery***

Ejemplo de solución

7.VALIDACIÓN DE XML CON DTD

Se considera un documento XML bien formado cuando cumple las reglas de XML y que no hay ninguna incoherencia al usar el lenguaje. Sin embargo no es válido por que en este caso se podrán definir documentos sin ninguna restricción. Para que esto no sea así debemos definir unas reglas que permitan que un documento sea válido

Para ello se crea un documento que contendrá las reglas que deben cumplir los documentos XML que se basen en él.

- El documento XML deberá indicar que plantilla de reglas utilizará y deberá completarlas para que se considere válido.
- Con la validación tenemos la seguridad de que los documentos cumplen unas reglas concretas y es fácil para, por ejemplo, una empresa establecer un protocolo que deben cumplir sus documentos..

IMPORTANTE: Debes distinguir claramente entre estos dos conceptos.

- ***Cuando un documento XML cuempl estrictamente laas reglas generales de creación de XML se dice que está bien formado***
- ***Cuando el documento además sigue las reglas de un documento de validación se dice que es válido***

- Son varias las técnicas empleadas para validar documentos entre ellas tenemos,
    - ***DTD. Document Type Definition***. Validación por documentos de definición de tipos. Se utilizaba en el lenguaje SGML y de ahí debe su popularidad No obstante recibe críticas porque su sintaxis no es XML
    - ***XML Schema o esquemas XML.***Es la que se aconseja actualmente pero no tiene una implantación total
    - ***Relax NG.*** Es una notación sencilla y fácil de aprender pero no ofrece tantas posibilidades como el XML Schema.

7.2.Introducción a los DTD

+ DTD significa “Document Type Definition”

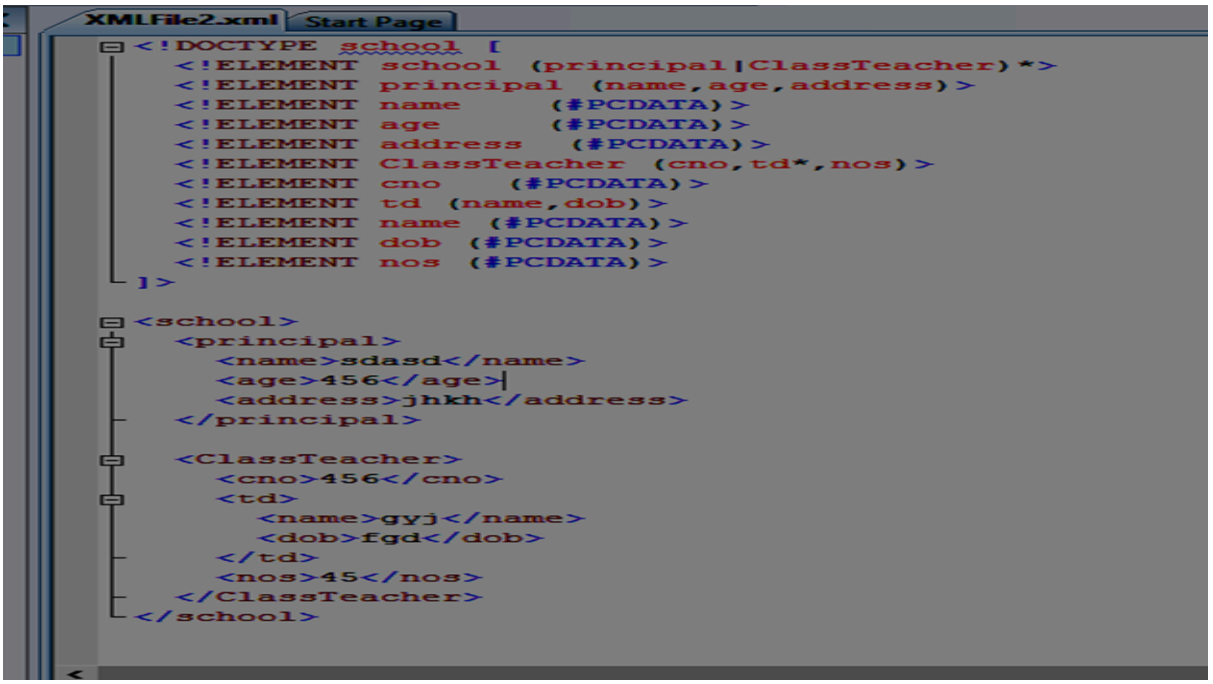
+ Describe la estructura y sintaxis de un documento XML o SGML

+ Mantiene coherencia entre documentos que usen el mismo DTD

Dicho con sencillez, **un DTD es un fichero de definición de tipo de documento. Permite definir la estructura, los elementos y los atributos permitidos para uno o varios documentos XML. Esto permite mantener la coherencia entre documentos que usen el mismo DTD.**

Como se puede deducir de lo anterior, **un DTD contiene las reglas de estructura y sintaxis de un fichero XML.** Así pues, ***sirve para comprobar si un fichero XML es válido o no***. Según esto, podemos afirmar que con un DTD, grupos independientes de personas pueden acordar un DTD estándar para intercambiar datos. También podemos decir que una aplicación podría usar un DTD para verificar que sus datos almacenados en XML son válidos.

Ejemplo de DTD



7.3.-¿Cuál es el aspecto de un documento DTD?

Ejemplos de DTD

Ejemplo 1:

<!ELEMENT lista_de_personas (persona*)>

<!ELEMENT persona (nombre, fechanacimiento?, sexo?, numeroseguridadsocial?)>

<!ELEMENT nombre (#PCDATA) >

<!ELEMENT fechanacimiento (#PCDATA) >

<!ELEMENT sexo (#PCDATA) >

<!ELEMENT numeroseguridadsocial (#PCDATA

Ejemplo 2

```
<?xml version="1.0"?><!DOCTYPE books [
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT authors (author)+>
<!ELEMENT subject (#PCDATA)>
<!ATTLIST subject class CDATA "">
<!ELEMENT book (title,authors,subject)>
<!ATTLIST book
    bookid CDATA #REQUIRED
    pubdate CDATA #REQUIRED>
]>
<books name="My books">
    <book bookid="1" pubdate="03/01/2002">
        <title>Java Web Services</title>
        <authors>
```

Ejemplo 3

DTD - Example

```
1   <?xml version="1.0"?>
2   <!DOCTYPE name [
3   <!ELEMENT name (first, middle, last)>
4   <!ELEMENT first (#PCDATA)>
5   <!ELEMENT middle (#PCDATA)>
6   <!ELEMENT last (#PCDATA)>
7   ]>
8   <name>
9       <first>Joseph</first>
10      <middle>John</middle>
11      <last>Fawcett</last>
12  </name>
```

7.4.-Declaración de bloques DTD

Los bloques para construir una DTD son:

- ***Elemento***. Es el bloque principal con el que se construyen los documentos XML.
    - Los elementos se declaran de una de estas dos maneras
        - <!ELEMENT nombre_del_elemento categoria>
        - <!ELEMENT nombre_del_elemento (nodos_hijos)
    - En el segundo caso se define el nombre_del_elemento como el nodo padre del que cuelgan un conjunto de hijos.
    - Si existe algún elemento que debe estar vacío se emplea la palabra reservada EMPTY que puesta en la zona de categoría permite indicar este hecho.
    - Ejemplo. La etiqueta de salto de línea del HTML se declara como
 </br> Si es declarada como vacía (EMPTY) ya no es necesario emplear etiqueta de cierre y podría emplearse como
. Una posible declaración sería
        - <!ELEMENT saltoLinea EMPTY>
- ***Atributo***. Es una manera de añadir más información a un elemento.
- ***Entidad***.En XML existen algunos caracteres con un significado especial. Como esos caracteres cuando se almacenan puede confundirse con entidades propias del documento XML usamos la siguiente notación
    -   =>' ' //Espacio en blanco
    - &gt => >
    - &lt => <
    - &quot =>"
    - &apost =>'
    - &amp => &
        - Se pueden usar dentro del documento XML
    - El usuario puede definir entidades propias para que después de analizar el documento XML se sustituyan por el valor indicado.
    - Ejemplos:
        - <!ENTITY pi "3.1141516"> => π
        - <!ENTITY textFile SYSTEM "fichero.txt"> =>&textFile
- ***PCDATA***. Indica que entre las etiquetas de apertura y cierre de ese elemento se almacenarán caracteres como texto y serán analizados por el procesador (parser) de XML
- ***CDATA***. En la práctica es como el PCDATA pero el contenido de ese elemento no se analiza

<u>Ejemplos de declaraciones de elementos</u>

1. Elementos vacíos <!ELEMENT nombre-elemento EMPTY>
2. Elementos que contienen texto.<!ELEMENT nombre-elemento (#PCDATA)
3. Elementos que contiene cualquier combinación de elementos conocidos <!ELEMENT nombre-elemento(#ANY) .=>Podría ser teléfono, fecha, mensaje,..
4. Elementos que suponen una secuencia de elementos (hay un elemento principal y otros secundarios). <!ELEMENT nombre-element(hijo1,hijo2,..). **<u>Deben aparecer en el mismo orden en que aparecen en el archivo XML</u>**

5. Las ocurrencias mínimas indican que un elemento ha de aparecer al menos una vez dentro de la secuencia.Se representan añadiendo un + al elemento que deba aparecer al menos una vez <!ELEMENT nombre-element(hijo1,+hijo2,..).
6. Para indicar que un elemento aparezca 0 o más veces usamos un asterisco <!ELEMENT nombre-element(hijo1* ,hijo2,..).
7. Para indicar que puede aparecer 0 o una vez usamos el signo de interrogación. <!ELEMENT nombre-element(hijo1,hijo2?,..).
8. Para indicar que aparece un elemento u otro los separamos por una barra.

AMPLIACIÓN DE CONTENIDOS: Declaración de elementos

- Para declarar un elemento en una DTD se utiliza la siguiente sintaxis

```
<!ELEMENT nombre-del-elemento tipo-de-contenido>
```

- En el tipo-de-contenido se especifica el contenido permitido en el elemento. Este puede ser:
    - Texto, (#PCDATA)
    - Otros elementos (hijo)
    - Estar vacío,EMPTY
    - Mixo (texto y otros elementos), ANY

EJEMPLOS

1.- El contenido de un elemento puede ser texto (#PCDATA)

En el siguiente documento XML el elemento ciudad puede contener cualquier texto

```
<?xml version=”1.0” encoding=”UTF-8”?>
<!DOCTYPE ciudad [
   <!ELEMENT ciudad (#PCDATA)>
]>
<ciudad>Roma</ciudad>
```

- Escribiendo #PCDATA entre paréntesis se ha indicado que el elemento ciudad pude contener una cadena de caracteres analizables

2.-El contenido de un elemento puede contener a otros elementos

En el siguiente ejemplo el elemento ciudad contiene los elementos “nombre” y “país”

```
<?xml version=”1.0” encoding=”UTF-8”?
<!DOCTYPE ciudad [
   <!ELEMENT ciudad (nombre, país)>
  <!ELEMENT nombre (#PCDATA)>
```

```
  <!ELEMENT país (#PCDATA))>
]>

<ciudad>
        <nombre>Roma</nombre>
        <país>Italia</país>
</ciudad>
```

3.Un elemento puede estar vacío.

En el siguiente ejemplo se ha declarado mayor de edad como vacío

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE persona [
   <!ELEMENT persona (nombre, mayor_de_edad, ciudad)>
   <!ELEMENT nombre (#PCDATA)>
   <!ELEMENT mayor_de_edad EMPTY>
   <!ELEMENT ciudad (#PCDATA)>
]>

<persona>
   <nombre>Elsa</nombre>
   <mayor_de_edad/>
   <ciudad>Pamplona</ciudad>
</persona>
```

Un elemento puede definirse para contener contenido mixto - **ANY**

EJEMPLO En la DTD interna del siguiente documento XML, se ha indicado que el elemento "persona" puede contener texto y otros elementos, es decir, contenido mixto, **ANY**:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE persona [
   <!ELEMENT persona ANY>
   <!ELEMENT nombre (#PCDATA)>
   <!ELEMENT ciudad (#PCDATA)>
]>
```

```
<persona>
   <nombre>Elsa</nombre> vive en <ciudad>Pamplona</ciudad>.
</persona>
```

Obsérvese que, por ejemplo, también sería válido el siguiente documento XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE persona [
   <!ELEMENT persona ANY>
   <!ELEMENT nombre (#PCDATA)>
   <!ELEMENT ciudad (#PCDATA)>
]>

<persona>
   <nombre>Elsa</nombre> vive en Pamplona.
</persona>
```

3. Secuencia de elementos.Estructura con hijos

- Supongamos el archivo xml correspondiente a un mensaje sms. Una posible DTD sería

……
<|ELEMENT BDsms(sms)>
<!ELEMENT sms (teléfono, fecha, hora, mensaje)>
……

- Es razonable pensar que puede que algunos de los sms recibidos puedan contener más de un mensaje. ¿Cómo recogemos está circunstancia en nuestra declaración?
- Las ocurrencias que pueden aparecer se indican con los siguientes operadores.
    - **:** indica que aparece obligatoriamente una vez.
    - **+** indica que puede haber una o más ocurrencias del elemento indicado
    - ***** indica que puede haber cero o más ocurrencias del elemento indicado
    - **?** indica que puede haber cero o una ocurrencia de elemento indicado
- Si suponemos que todos los sms tiene como mínimo un mensaje, pero puede haber varios la declaración correcta sería

<!ELEMENT sms (teléfono, fecha, hora mensaje+)>

7.5.Declaración interna

- Podemos compararla con la inclusión de un script dentro de un programa HTML
- Queremos realizar la declaración de este archivo xml

```xml
<?xml version='1.0' encoding='UTF-8' ?>
<receta>
<!--Primera receta-->
<titulo>...</titulo>
<ingredientes>...</ingredientes>
<procedimiento>...</procedimiento>
</receta>
```

1. ¡Comenzamos con !DOCTYPE seguido del nombre del nodo raíz o principal. Entre los corchetes declaramos cada uno de los elementos

   <?xml version=”1.0” encoding=”ISO-8859-1”?>
   <!DOCTYPE nombre Nodo Raíz
   [

   //Declaración de los elementos.
   ]

2. ***Cada elemento*** se declara con la palabra reservada !ELEMENT y el primero que debe declarase es el propio elemento raíz y entre paréntesis figuran los elementos que lo componen

<!DOCTYPE receta [
<! ELEMENT receta (título, ingredientes, procedimiento)
// Declaración del resto de elementos
]

3. Declaramos a continuación los restantes elementos.
   - En nuestro ejemplo son todos de tipo PCDATA.

- El resultado final sería.

```xml
<!DOCTYPE receta [
    <!ELEMENT receta (titulo,ingredientes,procedimiento)>
        <!ELEMENT titulo (#PCDATA)>
        <!ELEMENT ingredientes (#PCDATA)>
        <!ELEMENT procedimiento (#PCDATA)>

]>
```

7.6.-Declaración externa

1. Debemos crear un nuevo archivo y hacer en él la declaración.
2. A continuación en el XML se vincula está DTD. La sintaxis es .

<!DOCTYPE nombreNodoRaiz SYSTEM "nombreArchivoDTD.dtd">

EJERCICIO PARA REALIZAR EN CLASE. (Demostrado por el profesor)

- ***Enunciado***
- ***Solución***

ACTIVIDAD Nº 8. Crea el DTD correspondiente a los siguientes archivos XML. Puedes hacer una declaración interna y/o externa (lo dejo a tu criterio)

Ejercicio 1.

```
<aviso>
 <titulo>...</titulo>
 <parrafo>...</parrafo>
 <grafico>...</grafico>
 <parrafo>...</parrafo>
 <grafico>...</grafico>
 </aviso>
```

Ejercicio 2.

```
<aviso>
</aviso>
```

Ejercicio 3.

```
<aviso>
<parrafo>...</parrafo>
<grafico>...</grafico>
<parrafo>...</parrafo>
<parrafo>...</parrafo>
<grafico>...</grafico>
</aviso>
```

Ejercicio 4

```
<aviso>
 <titulo>...</titulo>
 <parrafo>...</parrafo>
 <parrafo>...</parrafo>
 <grafico>...</grafico>
 </aviso>
```

Ejercicio 5

```
<receta>
<parrafo>Esto es un párrafo</parrafo>
<titulo>...</titulo>
<ingredientes>...</ingredientes>
<procedimiento>...</procedimiento>
</receta>
```

SOLUCIONES:

7.7.-Declaración de atributos

- Se declaran mediante la etiqueta ***<!ATTLIST ..>***. La sintaxis es .

<!ATTLIST nombre_elemento nombre_atributo tipo presencia/valorPorDefecto**>**

- En la declaración distinguimos los siguientes elementos.
    - nombre_elemento. Es el nombre del elemento que podrá utilizar el atributo
    - nombre_atributo. Es el identificador del atributo que estamos declarando.
    - tipo.Es el tipo de valores que podemos asignar al atributo
    - presencia/valorPorDefecto. Indica las características de los valores que puede tomar el atributo.Si es obigatorio, si hay valor por defecto,..También permite dar un valor que el atributo tomará en el documento XML en el caso de que no se le de ningún valor
- Podemos declarar varios atributos para el mismo elemento.

```
<!ATTLIST nombre_elemento
        nombre_atributo1  tipo  presencia/ valorPorDefecto
        nombre_atributo2  tipo  presencia/ valorPorDefecto ... >
```

Ejemplos de declaración de atributos

1.-Declaramos el atributo nacionalidad correspondiente al elemento persona de tipo “texto normal”

```
<!ATTLIST persona nacionalidad CDATA>
```

7.1.1.-Presencia/Valor por defecto y otras opciones

- En la declaración de un atributo lo último que se indica es la propiedad relativa al valor por defecto. Tenemos las siguientes posibilidades

A) Valor por defecto

- Si al final de la declaración del atributo aparece un valor concreto se entiende que ese será el valor por defecto. Es decir que se podría no utilizar el atributo en un elemento y entonces dicho atributo tomaria ese valor

Ejemplo:En el archivo ***directorio.dtd*** que se muestra a continuación se define el atributo nacionalidad para el elemento persona como un atributo que contendrá texto de todo tipo (#PCDATA) pero que por defecto toma el valor Española

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT  directorio (persona)+>
        <!ELEMENT  persona (#PCDATA)>
                <!ATTLIST persona nacionalidad CDATA "Española">
```

B) Valores fijos.

- Se puede utilizar el término #FIXED antes de indicar el valor por defecto de un atributo. En este caso en ningún documento XML se podrá modificar dicho atributo

Ejemplo. En el siguiente ejemplo el atributo nacionalidad no podrá tomar ningún otro valor que no sea Española

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT  directorio (persona)+>
        <!ELEMENT  persona (#PCDATA)>
                <!ATTLIST persona nacionalidad CDATA #FIXED "Española">
```

C) Valores requeridos.

- En este caso se emplea la palabra #REQUIRED indicando con ello que es obligatorio asignar algún valor en el documento al atributo

Ejemplo. En este caso se obliga a asignar valor al atributo nacionalidad

```
<!ATTLIST persona nacionalidad CDATA #REQUIRED>
```

D) Valor opcional

- La palabra #IMPLIED especificado en el atributo indica que dicho atributo puede quedarse sin valor. Es decir no posee valor por defecto pero puede quedarse sin especificar. (quedaría nulo)

```
<!ATTLIST persona nacionalidad CDATA #IMPLIED>
```

7.1.1.-Tipos de atributos.

- *Entre* los distintos tipos de atributos destacan.
  - ***CDATA***: es un texto. Podrán tener cualquier carácter
  - ***ID***:es un identificador que permite identificar al elemento de manera única en todo el documento XML
  - ***IDREF***.es un identificador de otro elemento del propio documento XML
  - **IDREFS**.es una lista de identificadores a otros elementos.
  - (tipo1 | tipo2 | …):es el valor de uno de los indicados en está lista número (definida por el usuario)
  - ***NMTOKEN***:es un texto que solo podrá tener letras, dígitos, gon, subrayado, punto y dos puntos
  - ***ENTITY***:el tipo de atributo es una entidad que se ha declarado con anterioridad
  - ***ENTITIES*** Es una lista de entidades.

7.1.2.-Miscelánea de ejemplos

1.-Atributo de pago:

- ***Atributo: tipo_de_pago***
- ***Tipo de datos:Lista de caracteres***
- ***Valor por defecto:pagado***

Seguimos la plantilla <!ATTLIST elemento atributo tipo valor> y una posible declaración sería : <!ATTLIST payment payment type CDATA "checked">

2.- Validación, interna, del siguiente archivo XML

```
<menu_almuerzo>
    <comida precio="$2.00" calorias="650">
        <nombre>Waffles</nombre>
        <descripcion>Waffles baratos de McDonalds</descripcion>
    </comida>
    <comida precio="$5.00" calorias="1500">
        <nombre>Hamburguesa</nombre>
        <descripcion>La hamburguesa mas comun de McDonalds</descripcion>
    </comida>
</menu_almuerzo>
```

Solución.

- *Es tan solo una de las posibles soluciones. Discute con tus compañeros y docentes otras posibles soluciones válidas, justificando tu respuesta*

```
<!DOCTYPE  menu_almuerzo
 [
<!ELEMENT menu_almuerzo (comida*)>
<!ELEMENT comida (nombre,descripcion)>
    <!ATTLIST comida precio CDATA #REQUIRED>
    <!ATTLIST comida calorias CDATA #REQUIRED>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT descripcion (#PCDATA)>
]>
```

ACTIVIDAD Nº 9. Crea el DTD correspondiente a los siguientes archivos XML. Puedes hacer una declaración interna y/o externa (lo dejo a tu criterio)

Ejercicio nº 1. Haz clic en este enlace para obtener el archivo XML.

- ***Solución***

Ejercicio nº 2. Haz clic en este enlace para obtener el archivo XML. (posible ejercicio de examen)

- ***Solución***

Ejercicio nº 3. Haz clic en este enlace para obtener el archivo XML (posible ejercicio de examen)

- ***Solución***

Ejercicio nº 4. Haz clic en este enlace para obtener el archivo XML posible ejercicio de examen)

- ***Solución***

7.1.3.-Atributos o elementos.

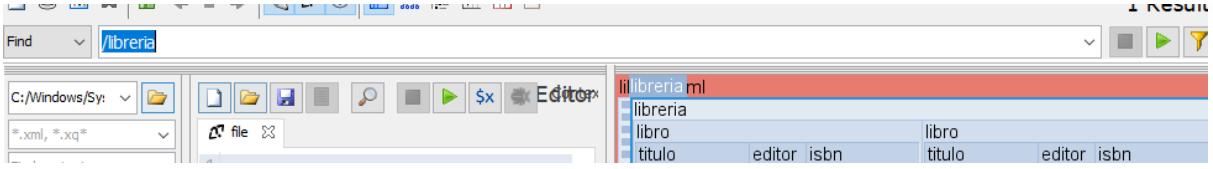
- ***En XML no hay reglas fijas acerca de cuándo usar elementos y cuando atributos pero puede serte de ayuda las siguientes observaciones,***
    - ***Los atributos no pueden contener valores múltiples pero los elementos o etiquetas si***
    - ***Los atributos no son fáciles de ampliar para hacer frente a cambios futuros***
    - ***Los atributos no describir estructuras pero los elementos si***
- ***No obstante posiblemente sea aconsejable usar atributos cuando se necesite alguna especie de identificador. (recuerda el ejercicio del archivo xml sobre la cartelera de cine.)***

8.CONSULTAS XPATH

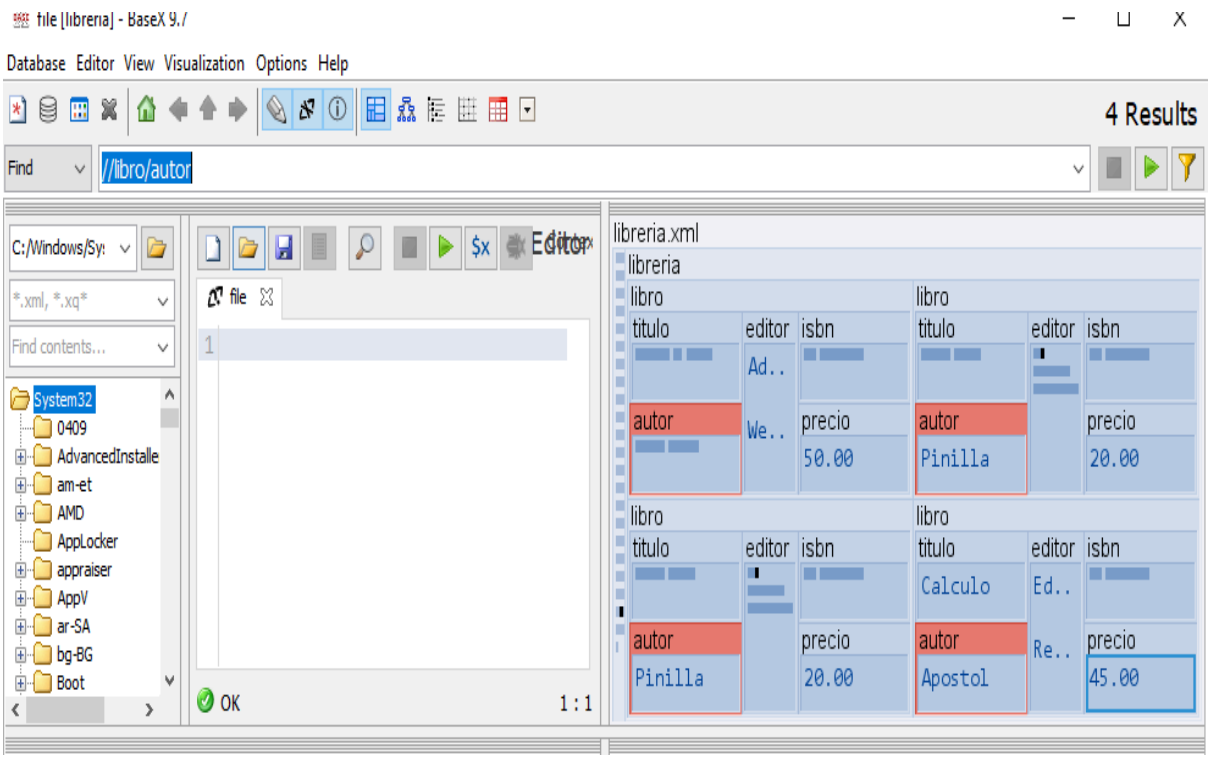
- XPath es un lenguaje que permite construir expresiones que recorren y procesan un documento XML. XPath permite buscar y seleccionar teniendo en cuenta la estructura jerárquica del XML
- XPath se puede usar para navegar a través de los elementos y atributos que conforman un documento XML.Bien a ser como un lenguaje de consulta
    - XPath es el núcleo de XSLT
    - Es necesario para que funciones XSLT
- Podemos descargarlo de https://basex.org/. Te aconsejo la opción de descarga .zip para evitar problemas con los antivirus.
    - Una vez descomprimida la carpeta puede usarse sin necesidad de instalación

8.1.-Base & XPath

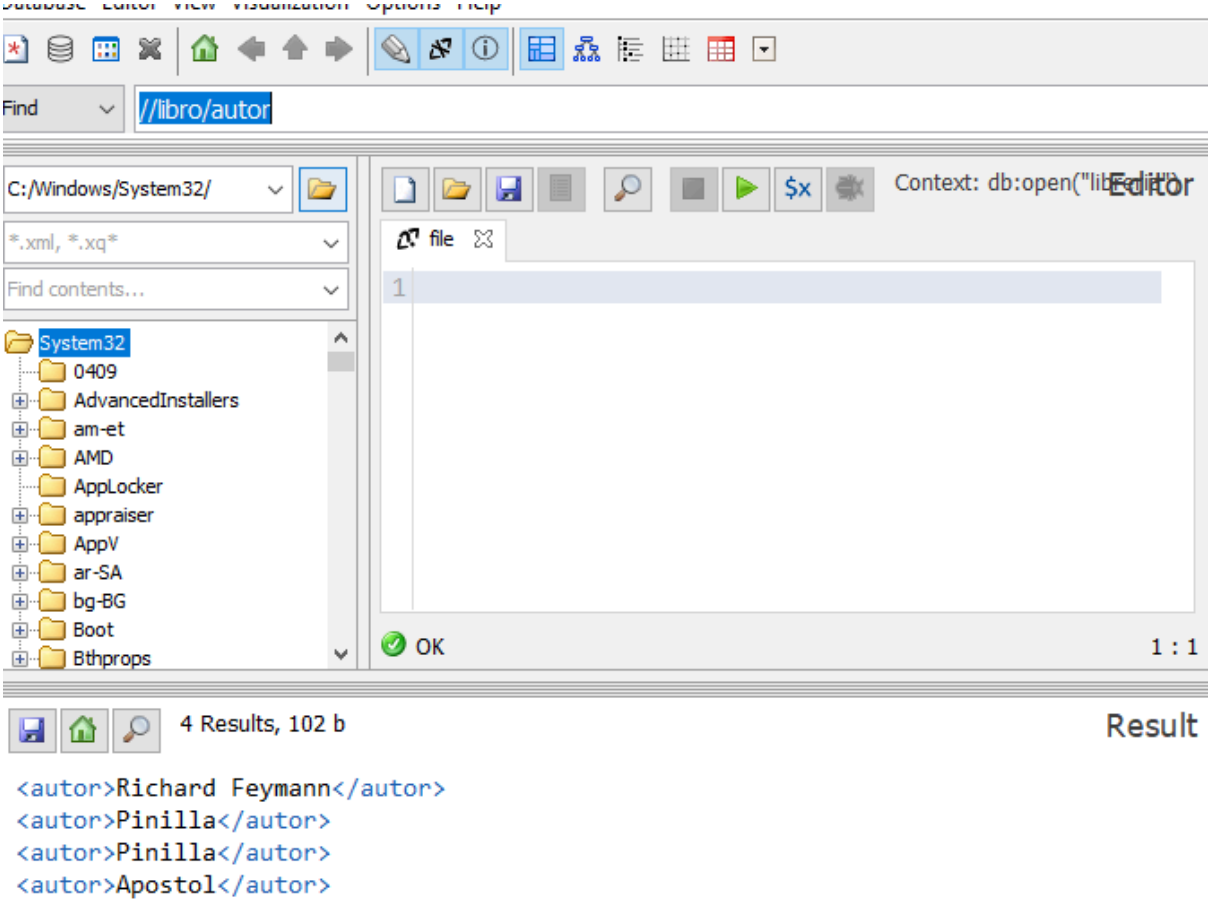
- Para seleccionar un archivo xml hacemos clic en el botón nuevo con lo cual lo cargamos
- En la barra de direccione si ponemos * hacemos referencia a todo el xml y si ponemos // solo al nodo raíz



- Si quiero seleccionar un nodo en concreto añado /nodo y me selecciona ese nodo.
    - Puede observarse en la subpantalla de la derecha como selecciona solo los autores



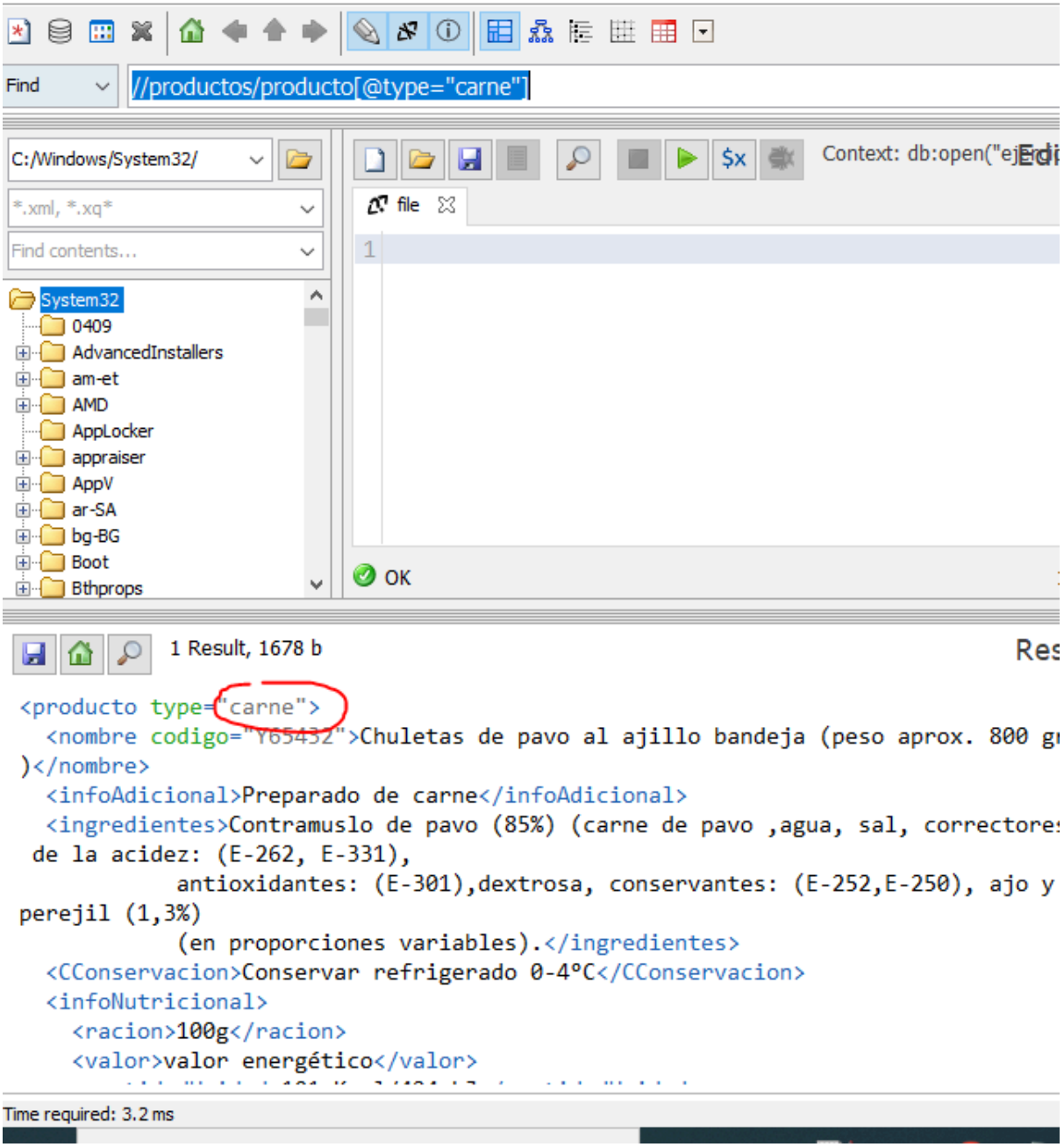
- También puedo observar en la pantalla inferior izquierda que solo se han seleccionado los autores



- Para filtrar por un atributo en la barra de tareas ponemos @nombre del atributo =”valor del atributo” esto entre corchetes
  //….[@nombreAtributo=”valor”]

    - la @ hace mención a un atributo. Lo anterior me permite

Observa el ejemplo en que se selecciona a los productos de atributo type=”carne”



8.2.-Tipos de nodos

- XPath tiene los siguientes tipos de nodos
    - Element
    - Attribute
    - Text
    - Namespace
    - Processing-Instruction
    - Comment
    - Document Node
- Estos elementos tienen una estructura de árbol.
    - En el ejemplo que te muestro compruebas que <bookstore> contiene a <author> y este a lang

```
<?xml version="1.0" encoding="UTF-8"?>

<bookstore>
  <book>
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
</bookstore>
```

```
<bookstore> (root element node)

<author>J K. Rowling</author> (element node)

lang="en" (attribute node)
```

- Las relaciones entre nodos son: Parent, Children y Sibling
    - Los nodos por debajo de otro son Children
    - El nodo que un nodo tiene por encima es su Parent
    - Sibling son los que están al mismo nivel.