XML i Validació d'XML

lloc: CIFP Francesc de Borja Moll
Curs: Llenguatges de marques i sistemes de gestió d’informació
Llibre: XML i Validació d'XML
Imprès per: Albert Perelló Puertas
Data: dilluns, 19 abril 2021, 15:34

Taula de continguts

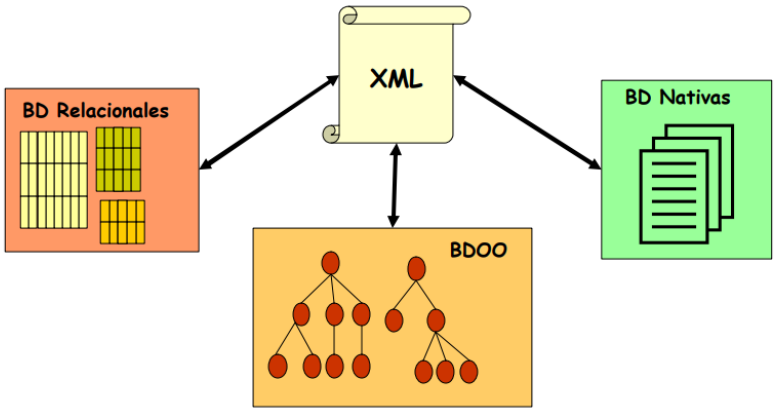
1. Llenguatge XML

XML (*eXtensible Markup Language*,) és un llenguatge desenvolupat per W3C què està basat SGML (S*tandard Generalized Markup Language*). En realitat XML no és un llenguatge de marcat si no més aviat un metallenguatge, que ens permet definir llenguatges de marcat adequats a usos determinats.

És indubtable que avui en dia, la necessitat de gestionar els continguts d'una manera ràpida i organitzada, es fa absolutament necessària. XML ofereix una àmplia gamma de solucions als diferents problemes que sorgeixen en aquest camp.

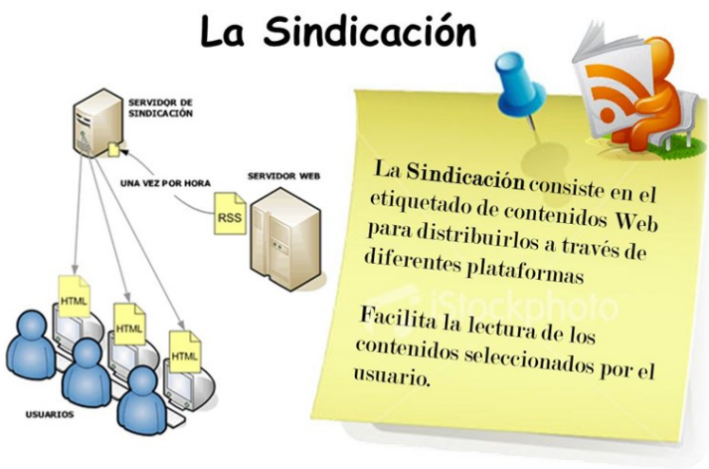
Alguns dels seus usos més estesos són:

- Intercanvi de dades entre sistemes: Possibilita l'intercanvi de dades de forma estructurada entre diferents sistemes. Al tractar-se d'un format de text pla i ser un llenguatge estandarditzat, fa que aquesta transferència sigui molt àgil i independent de la plataforma utilitzada.
- Publicar i intercanviar continguts de bases de dades: Permet guardar dades de forma estandarditzada per després poder ser tractats per multitud de llenguatges diferents. El seu maneig és molt més senzill que bases de dades com MySQL i molt més ric que utilitzar fitxers de text plans. Per exemple tenim una petita web on diàriament es posa algun article o nota, i no tenim doblers per pagar un hosting amb bases de dades MySQL i PHP, llavors XML pot reemplaçar a MySQL.



Font: Universidad de Oviedo (Ana Belén Martínez Prieto) http://di002.edv.uniovi.es/~labra/cursos/ver06/pres/XMLBD.pdf

- Sindicació de notícies o de continguts (RSS, Atom): Tecnologies que tenen com a base documents XML. Són serveis que permeten obtenir informació d'un document XML generat automàticament per un sistema de publicació.



Font: Blog: Tecnología , Información y Comunicación De Las Redes Educativas

- Definir formats de missatges per a la comunicació entre aplicacions B2B (*Bussiness-to-Bussiness,* B2B és un model de negoci normalment associat a majoristes que consisteix en la prestació de serveis, consum de continguts i/o transaccions entre dues empreses, https://madridnyc.es/que-es-el-b2b-ejemplos/).

Font: https://leticiadelcorral.com/

Resumint: XML és un llenguatge simple de descripció d'informació.

- És un estàndard que permet dissenyar i desenvolupar llenguatges de marques.
- Representa informació estructurada al web, de manera que aquesta informació pugui ser emmagatzemada, transmesa, processada, visualitzada i impresa, per molt diversos tipus d'aplicacions i dispositius.

Es diu que XML és un metallenguatge, això vol dir que pot ser emprat per definir altres llenguatges anomenats dialectes XML. Alguns dels llenguatges que es poden definir a partir d'XML són:

- GML (*Geography Markup Language*, Llenguatge de Marcat Geogràfic). És una gramàtica en XML Schema per al modelatge, transport i emmagatzematge d'informació geogràfica. La seva importància radica que a nivell informàtic es constitueix com una llengua franca per al maneig i transvasament d'informació entre els diferents programaris que fan ús d'aquest tipus de dades, com els Sistemes d'informació Geogràfica.
- MathML (*Mathematical Markup Language*, Llenguatge de Marcat Matemàtic). Serveix per descriure expressions matemàtiques capturant tant el seu contingut com la seva estructura.
- RSS (*Really Simple Syndication*, Sindicació Realment Simple). Es tracta d'una família de formats de canals web XML utilitzats per publicar continguts actualitzats de manera sovint, com per exemple llocs de notícies, blogs i podcasts, i per mitjà dels quals es pot compartir la informació i usar-la en altres llocs web o programes. És en essència una sindicació de continguts. (es veurà en propers capítols).
- SVG (*Scalable Vector Graphics*, Gràfics Vectorials Escalables). És una família d'especificacions d'un format de fitxer basat en XML per descriure gràfics vectorials bidimensionals, tant estàtics com dinàmics (p. ex. interactius o animats).
- XHTML (*eXtensible HyperText Markup Language*, Llenguatge de Marcat d'Hipertexte eXtensible). És, bàsicament HTML expressat com XML vàlid. És més estricte a nivell tècnic, però això permet que posteriorment sigui més fàcil a fer canvis o buscar errors entre d'altres.

Com a tots els llenguatges de marques, els documents XML es componen de dades-caràcter (que seria la informació pròpiament dita) i marcat (marques XML). El marcatge afegeix informació addicional que possibilita una nova manera de tractar la informació, ja que permet realitzar sobre els documents tasques informàtiques com són per exemple recerques precises, filtrats, generació automàtica d'informes, etc.

- A un document XML tota la informació es representa com a text.
- No hi ha tipus de dades numèriques, binaries, lògiques, etc.
- **XML és extensible**, això és, que en XML les marques no estan predefinides, sinó que podem definir les nostres pròpies marques, per tal que compleixin els requisits establerts en el llenguatge (que veurem més endavant). Gràcies a aquesta informació XML s'adapta a qualsevol classe de situació, necessitats de l'autor i programari de processament, ja que en funció dels requeriments es pot fer servir un programari més senzill o més complex.

Estructura d'un document XML

L'estructura general d'un document XML està formada per dues parts:

- **Pròleg (opcional)**:
  Conté una seqüència d'instruccions de processament i/o declaració del tipus de document. Es pot dividir en dues parts:
    1) Declaració XML: Estableix la versió d'XML, el tipus de codificació i si és un document autònom. És opcional.
    2) Declaració de tipus de document: Estableix el tipus de document que és.

- **Cos**:
  És el contingut informatiu de document, organitzat com un arbre únic d'elements marcats.

Pròleg

El pròleg afegeix informació sobre el document.Tot i que el pròleg és opcional, la seva inclusió és molt recomanable, ja que facilita un processament fiable i robust de la informació continguda en el document.

- La declaració de tipus de document:
  - Ens diu el tipus al qual s'ajusta el document.

Com hem vist al quadre anterior, el pròleg pot dividir-se en dues parts:

1) Declaració XML:

- Estableix que el document és un document XML.
- Inclou informació sobre la versió d'XML utilitzada per a escriure-ho.
- Pot incloure informació sobre el tipus de codificació de caràcters utilitzat en el document,
- Diu si és autònom, és a dir, que el document conté en si mateix tota la informació necessària per processar-ho o no.

Si està present, la declaració XML, ha de ser la primera línia del document.

Un exemple de declaració XML completa podria ser:

<? xml version = "1.0" encoding = "ISO-8859-1" standalone = "yes"?>

Els camps dins de la declaració XML han de seguir l'ordre estricte de la declaració:
- Atribut "version": Permet indicar la versió per a la qual es va elaborar el document (per exemple a l'exemple versió 1.0) i permetre que els documents s'adaptin a l'evolució de l'estàndard.

- Atribut "encoding": Permet indicar el joc de caràcters utilitzat en el document. El valor per defecte és UTF-8. La codificació i representació de caràcters.

- Atribut "standalone": És la declaració de document autònom i pot ser "yes" o "no". El valor "yes", indica que el document conté en el seu interior tota la informació rellevant per a la seva interpretació, per tant és independent o autònom.

* Direm que sí s'escriu la declaració XML, l'atribut "version" és obligatori, en canvi els altres dos no, i per defecte tendriem una codificació "UTF-8" i "no" (respectivament).

2) La declaració del Tipus de Document: És opcional, ja que està inclosa en el pròleg i té un format especial, diferent de les marques i de les instruccions de processament.

- Proveeix una sèrie de mecanismes que aporten funcionalitat a XML.
- Gràcies a ella és possible definir una sèrie de restriccions addicionals que han de complir els documents.
- També incorpora la possibilitat d'utilitzar certes eines que facilitaran a l'usuari XML algunes tasques.

Un exemple d'una declaració de tipus de document és el següent:

<! DOCTYPE CIFP FRANCESC DE BORJA MOLL SYSTEM "http://www.cifpfbmoll.eu">

cos

És la part més important i conté la informació del document, és a dir, les dades a les quals s'ha afegit el marcat.

- **Elements XML**: Els documents XML estan formats per text pla (és a dir, sense format) i contenen marques (etiquetes) definides pel desenvolupador.
  - Les marques han de ser el més descriptives possibles.
  - La sintaxi general d'una marca és: <etiqueta>valor</etiqueta>, això seria un element.

  Per exemple, si en un document XML es vol guardar el nom Aina, es pot escriure: <nom>Aina</nom>, Per tant, seguint la sintaxi bàsica, l'etiqueta d'inici seria <nom> i l'etiqueta de fi </nom>. El valor en aquest cas seria Aina.
- **Elements buits**: En un document XML, un element pot no tenir cap contingut, en aquest cas hauríem d'escriure: <etiqueta></etiqueta> o bé simplement: <etiqueta/>.
- **Relacions pare-fill entre elements XML**: Un element (pare) pot contenir un altre(s) element(s) (els fills).

En el següent exemple podem veure que tenim un element "pare", que conté 4 elements "fills" (nom, dona, data_de_naixement i ciutat), i a la vegada un element fill (data_de_naixement) té tres elements "fills" (dia, mes, any).

```
<persona>
  <nom>Aina</nom>
  <dona/>
  <data_de_naixement>
      <dia>10</dia>
      <mes>3</mes>
      <any>2000</any>
   </data_de_naixement>
   <ciutat>Palma de Mallorca</ciutat>
</persona>
```

- **Element arrel d'un document XML**: Qualsevol document XML ha de tenir un únic element arrel (pare), d'aquesta manera, qualsevol document XML es pot representar com un arbre invertit d'elements. Es diu que els elements són els que donen estructura semàntica al document.
- **Elements amb contingut mixt**: Un element d'XML pot contenir contingut mixt, és a dir, text i altres elements.

Per exemple:

```
<persona>
   <nom>Aina</nom> fa <alçada>1,65</alçada>
</persona>
```

Normes de sintaxi d'XML

En un document XML, tots els noms dels elements es diu que són "case sensitive", és a dir, sensibles a lletres minúscules i majúscules. Hauran de complir les següents normes:

- Poden contenir lletres minúscules, lletres majúscules, números, punts ("."), guions ("-") i guions baixos ("_"), també poden contenir el caràcter dos punts (":") però només per definir espais de noms.
- El primer caràcter ha de ser una lletra o un guió baix.
- Darrere el nom d'una etiqueta es permet escriure un espai en blanc o un salt de línia, però mai abans del nom de l'etiqueta.
- Les lletres "á", "Á", "ñ", "Ñ" estan permeses però és recomanable no utilitzar-les per reduir possibles incompatibilitats amb programes que no les suportin.
- El guió ("-") i el punt ("."), encara que estan permesos per nomenar etiquetes, s'aconsella evitar el seu ús., ja que el guió es pot confondre amb el signe menys de la resta i el punt es podria interpretar com una propietat d'un objecte (veurem les propietats dels objectes més endavant).

Atributs a XML

Un atribut serveix per proporcionar informació extra sobre l'element que el conté. Els elements d'un document XML poden tenir atributs definits a l'etiqueta d'inici.

Exemple: Definim atributs

```
codi: F001
Nom: Falda
Color: blau
Preu: 24,90

La seva representació en un document XML podria ser:

<article codi="F001">
   <nom color="blau" preu="24,90">Falda</nom>
</article>
```

- **Normes de sintaxi dels atributs:**

  1. Els noms dels atributs han de complir les mateixes normes de sintaxi que els noms dels elements.
  2. A més els atributs d'un element han de ser únics.
  3. Els atributs continguts dins d'un element s'han de separar amb espais en blanc.

Com crear un document XML

Podem escriure un document XML amb qualsevol editor de text pla, com per exemple el bloc de notes, i amb programes específics, com per exemple Visual Studio Code.

Exemple:

```
<?xml version="1.0" encoding="UTF-8"?>
<biblioteca>
  <llibre>
    <titol>Cien años de soledad</titol>
    <autor>Gabriel García Márquez</autor>
    <any_de_publicacio="1967/>
  </llibre>
  <llibre>
    <titol>Ulisses</titol>
    <autor Datanaixement="02/02/1882">James Joyce</autor>
    <any_de_publicacio="1923"/>
  </llibre>
</biblioteca>
```

2. XML ben format, Validació, Llenguatges d'esquemes

XML vs HTML

XML, és un llenguatge que va ser creat per a descriure informació. La seva funció principal és ajudar-nos a organitzar continguts i això fa que els documents XML siguin portables cap a diferents tipus d'aplicacions.

Una dada important és que en parlar d'XML parlem de documents ben formats (*well formed)*. Això vol dir que totes les etiquetes han d'estar niuades (<EtiqA><EtiqB>Continguts</EtiqB></EtiqA>. Una cosa molt important, és tenir en compte que són "*casi sensitiv*e" de tal forma que <B> no és el mateix que <b>, sinó són dues etiquetes diferents. Una altra característica d'XML és que és un llenguatge que pot estendre's. En treballar amb documents XML podem determinar les nostres pròpies etiquetes i estructura de documents per a treballar.

Si el comparam amb **HTML**, **podem dir que és un llenguatge concebut per a mostrar informació, determinar com actua i que fa**. La seva funció radica a ajudar-nos a donar-li format als diversos continguts d'una pàgina. En HTML, a més totes les etiquetes ja han estat predefinides i són part de l'estàndard HTML definit pel W3C.

A partir d'aquesta diferència, es pot dir, que XML no és ni serà mai un reemplaçament de l'HTML sinó un complement que serveix per a manejar la informació separada del format.

- **Característiques d'XML**
  - Simple: L'especificació completa són menys de 30 pàgines.
  - Extensible: Les etiquetes les inventa el creador del document XML de forma lliure i pot compartir-les o no.
  - Estàndard obert: Això significa que no és necessari saber programar. Existeixen moltes eines eficients que ens permeten crear, manejar o implantar a un ordinador documents, així com distribuir-los.
  - Consensuat: El disseny d'XML inclou els punts de vista dels organismes coordinadors d'HTML i SGML, així com de persones que han desenvolupat importants aplicacions amb aquests estàndards.
  - Lliure: Ningú té la propietat ni la patent d'XML.
  - Llest per poder ser utilitzat: Els navegadors web, en les seves darreres versions, són capaços de llegir especificacions XML.
  - Validable: Té tècniques que permeten la validació dels documents involucrats.

Documents XML ben formats

Un document XML està ben format si s'apliquen les regles de sintaxi que ja hem comentat. En cas que no sigui així, l'analitzador sintàctic de torn generarà un missatge d'error i s'aturarà l'anàlisi de el document, com es pot comprovar amb aquest exemple en un simple navegador.

Direm que un document XML està ben format si:

- Els noms dels elements i els seus atributs estan escrits correctament.
- Els valors dels atributs estan entre cometes dobles o simples.
- Els atributs d'un element estan separats amb espais en blanc.
- Hi ha un únic element arrel.
- Tot element té un element pare, excepte l'element arrel.
- Tots els elements tenen una etiqueta d'obertura i una altra de tancament.
- Les etiquetes estan correctament niades.

Validació de documents XML

Existeix un procés per comprovar que uns fitxers XML determinats segueixen un determinat vocabulari, que s'anomena validació i els documents XML que segueixen les regles del vocabulari s'anomenen documents vàlids.

```
La validació és el procés de comprovar que un document compleix amb les regles del vocabulari en tots el
s seus aspectes.

- La validació permet assegurar que la informació està exactament en la manera com ha d'estar.
- Això és especialment important quan es comparteix informació entre sistemes, ja que validar el documen
t és assegurar-se que realment l'estructura de la informació que s'està passant és la correcta. Si es vo
l fer que dos programes situats en ordinadors diferents col·labori, és necessari que la informació que e
s passen entre un i l'altre segueixi l'estructura prefixada per enviar-se missatges.
```

Els documents vàlids són els que compleixen les especificacions dels llenguatges d'esquemes com són DTD (Document Type Definition) i XSD. No hi ha esquemes de documents idèntics, per això és difícil concretar les regles que han de complir.

Perquè un document XML sigui vàlid, durant el procés de validació es comprova:

- Quins elements o atributs es permeten en un document del tipus definit en l'esquema.
- L'estructura dels elements i atributs (elements niats, atributs obligatoris o opcionals, etc.).
- L'ordre dels elements.
- Els valors de les dades d'atributs i elements (segons enumeracions, rangs de valors delimitats, tipus de dada correcta (per exemple, format correcte d'una data, utilitzar un sencer per expressar un nombre), etc.
- La unicitat de valors dins d'un document (per exemple, referències de productes que no poden repetir-se).

```
IMPORTANT: S'ha de tenir clara la diferència entre el que és un document ben format i un document vàlid

- Un document és ben format quan segueix les regles d'XML.
- Un document és vàlid si segueix les normes del vocabulari que té associat.

Un document pot complir perfectament amb les regles de creació de documents XML i, per tant, estar ben f
ormat, però en canvi no seguir les normes del vocabulari i, per tant, no ser vàlid.
I un document pot ser ben format però no vàlid, i en canvi si és vàlid segur que és ben format.

La majoria dels navegadors ofereixen un depurador que pot identificar documents XML mal formats.
```

Llenguatge d'esquemes

Hi ha molts llenguatges de definició d'esquemes, però els més utilitzats en XML són:

- DTD: Definició de Tipus de Document (en anglès *Document type definitions*).
- XSD: Definició d'esquema XML (en anglès, *XML definition language*).
- Relax NG (RNG): Llenguatge regular per XML Següent Generació. És una notació senzilla i fàcil d'aprendre que s'està fent molt popular. No té tantes possibilitats com XML Schema, però té una sintaxi més senzilla. A més admet afegir instruccions de tipus XML Schema per el que s'està convertint en una de les formes de validació més completes.
- Schematron: És part integrant del projecte ISO DSDL (*Document Schema Definitiion Languages*). Permet establir regles que faciliten establir les relacions que han de complir les dades d'un document XML. Però, per contra, no és tan bo per establir la resta de regles de validació (ordre d'elements, tipus de dades, etc.).

Processadors d'XML (*parsers*)

El procés de validació es fa per mitjà de programes especials anomenats processadors o validadors (en anglès *parsers*).

Els processadors XML són els encarregats de validar els documents i, per tant, és molt important triar un processador que sigui capaç de validar el llenguatge de validació que fem servir:

- El fet que la DTD sigui el sistema més antic i que se'n parli en l'especificació d'XML ha fet que la majoria dels processadors XML puguin validar llenguatges definits amb DTD.
- Amb el pas del temps, XML Schema (XSD), s'ha convertit en la manera estàndard de validar vocabularis XML, i altres tecnologies XML en fan servir alguns aspectes. Per tant, la majoria dels processadors també suporten aquest format.
- Molts processadors a part dels dos més emprats, també poden fer servir altres llenguatges (en especial Relax NG).

2.1. Validació per DTD

DTD o Definició de Tipus de Document: És la tècnica més veterana i en realitat ve del SGML (el llenguatge base d'XML). És la més utilitzada, però també la menys coherent amb les regles XML. Però, la seva enorme compatibilitat amb molts de productes de software l'ha convertida en una eina de molt d'èxit.

```
IMPORTANT: DTD (en anglès Document Type Definition), ens serveix per definir l'estructura que ha de segu
ir la informació en un arxiu XML i poder concloure si aquest és vàlid segons una sèrie de regles que nos
altres mateix definim en funció de les nostres necessitats, en altres paraules, es podria dir que és una
definició gramatical que ens dirà si un XML és vàlid o no. Una DTD està escrita en el seu propi metallen
guatge i no en XML.

- Defineix tots els elements.
- Defineix les relacions entre els diferents elements.
- Proporciona informació addicional que pot ser inclosa en el document (atributs, entitats, notacions).
- Aporta comentaris i instruccions per al seu processament i representació dels formats de dades.
```

Com ja s'ha comentat, és un mètode bastant senzill, i per aquesta raó presenta diverses limitacions, ja que no suporta noves ampliacions d'XML i no és capaç de descriure certs aspectes formals d'un document en l'àmbit expressiu. Les DTD poden ser internes o externes a un document, o totes dues coses a la vegada.

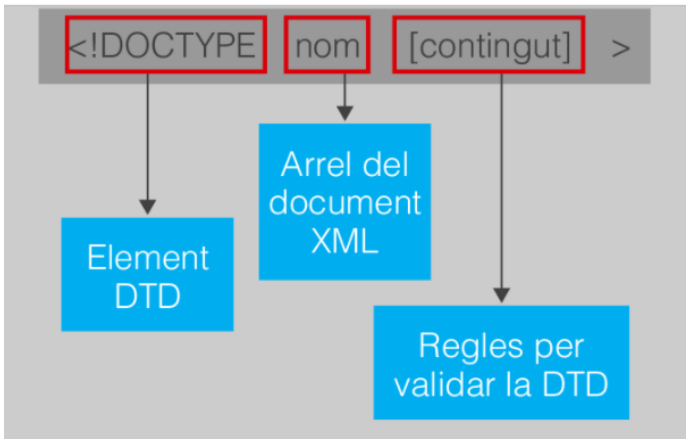
Sintaxi d'un document DTD

```
<!DOCTYPE element
[
   declaració 1
   declaració 2
   ........
]>

On:
- DOCTYPE: Defineix que és un document DTD
- element: Nom del node arrel del document XML
- declaració n: Definició d'una característica que ha de complir el document XML
```

- **Referència a una DTD en un document propi**

  Es pot definir l'estructura que ha de complir un document XML mitjançant el codi DTD, inserit en el mateix document. El desavantatge és que les regles definides només es podran aplicar en el mateix document.



font: IOC

Un document XML que defineixi internament la seva DTD, simplement escriu instruccions DTD dins del mateix document emprant l'etiqueta anomenada DOCTYPE.

- La sintaxi és:

```
<!DOCTYPE nom [....codi DTD...]>

El que s'escriu dins dels claudàtors seran les instruccions/declaracions DTD:

<! DOCTYPE nom [
 ... declaracions ...
]>
```

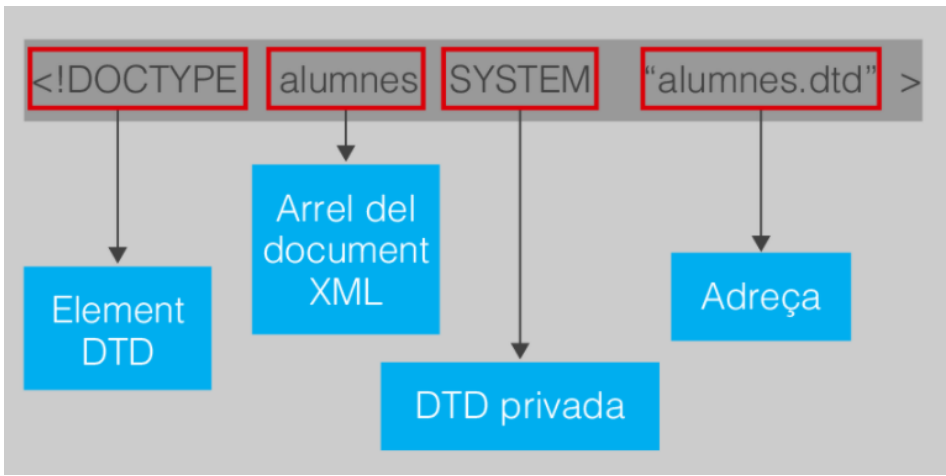
```
EXEMPLE DTD dins un document XML


<?xml version=”1.0” encoding=”UTF-8”?>
   <!DOCTYPE persona [
         <!ELEMENT persona (nom)>
         <!ELEMENT nom (#PCDATA)>
       ]>
   <persona>
      <nom>Antonio</nom>
   </persona>

Aquest document és vàlid ja que compleix les regles establertes directament a l'etiqueta DOCTYP
E.
```

- **Referència a una DTD en document extern i privat**

  En aquest cas la validació es realitza mitjançant un document DTD extern i privat. D'aquesta manera, si volem que un document compleixi les regles del document DTD, s'haurà d'indicar la ruta (relativa o absoluta) dins del document XML.



Font: IOC

- La sintaxi és:

```
<! DOCTYPE nom SYSTEM "ruta o camí al DTD">

On la ruta o camí potser, com s'ha indicat abans potser absoluta o relativa:

- En el cas de ruta absoluta s'haurà d'indicar el seu URL. Per exemple: <!DOCTYPE nom SYSTEM "ht
tp://.../document.dtd">
- En el cas de ruta relativa s'haurà d'escriure el nom de l'arxiu. Per exemple: <!DOCTYPE arrel
 SYSTEM "document.dtd">
```

EXEMPLE DTD extern al document XML

Fitxer: llibre.xml

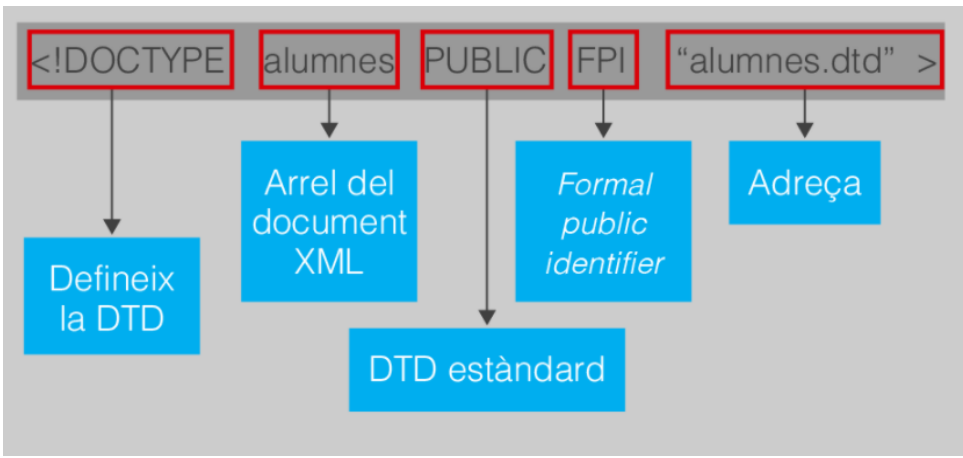
```
<?XML version="1.0" ?>
<!DOCTYPE llibre SYSTEM "llibre.dtd">
<llibre>
  <titol>Luces de Bohemia</titol>
  <autor>Ramon del Valle Inclán</autor>
</llibre>
```

Fitxer: llibre.dtd

```
<!ELEMENT llibre (titol,autor)>
<!ELEMENT titol (#PCDATA)>
<!ELEMENT autor (#PCDATA)>

*Nota: Les declaracions per "llibre", "titol" i "autor", les veurem un poc més endavant
```

- **Referència a una DTD en document extern i públic:**



Font: IOC

  - La sintaxi és:

```
<! DOCTYPE nom PUBLIC "Identificador_DTD" "ruta o camí">
```

Declaracions d'una DTD

1- Declaració d'elements

Les declaracions d'elements permeten definir quins elements estan permesos per a un document i quin és el seu contingut.

Respecte als elements, perquè un document XML sigui vàlid ha de complir que:

- Tots els elements estiguin reconeguts mitjançant declaració de tipus (és a dir, tots els elements han de pertànyer a un tipus declarat).
- El contingut de cada element s'ajusta al que s'ha declarat en el seu tipus.

```
Les declaracions dels elements segueixen la següent sintaxi:

<!ELEMENT nom_element (contingut)>

On:
- "nom_element" és el nom de l'element i "contingut" una expressió que descriu el contingut de l'ele
ment.
- I s'empra l'etiqueta <!ELEMENT>
```

El contingut podrà ser:

- **EMPTY:** Significa que l'element és buit, és a dir, que no pot tenir contingut.
  - Els elements buits es poden escriure amb etiquetes d'obertura i de tancament sense res enmig, ni espais.
  - EMPTY s'ha d'escriure sense parèntesi.
  - L'element pot tenir atributs.

```
Exemple amb EMPTY:

<!DOCTYPE element_buits [
  <!ELEMENT element_buit EMPTY>
]>

<element_buit></element_buit>   <!-- Un element buit amb etiqueta oberta i tancada-->
<element_buit />               <!-- Una altra manera d'escriure elements buits-->

Nota: Els comentaris van entre <!-- y -->
```

- **(#PCDATA)**: Significa que l'element pot contenir text.
  - S'ha d'escriure entre parèntesis.
  - No contenen etiquetes ni altres elements.
  - Hi ha algunes restriccions de tipus de text:
    - El caràcter "<", perquè es pot confondre amb el començament d'un nou element.
    - El caràcter &, perquè aquest símbol indica el començament d'una entitat.
    - La cadena "]]>", perquè marca el final d'una secció CDATA (que veurem més endavant).

```
Exemple amb #PCDATA:

<!DOCTYPE client [
  <!ELEMENT client (nom)>     <!-- L'element "client" tendrà un fill "nom" -->
  <!ELEMENT nom (#PCDATA)>    <!-- L'element "nom" serà de tipus text-->
]>

<client>
  <nom> Joana </nom>
</client>
```

- **ANY:** Significa que l'element pot contenir qualsevol cosa (text i altres elements), és a dir, amb contingut mixt.
  - Pot contenir atributs.
  - No s'empra massa, ja que és preferible estructurar els continguts.

Exemple amb ANY:

```
<!DOCTYPE element_any[
  <!ELEMENT element_any ANY>
  <!ELEMENT element_x ANY>
]>

<element_any />
<!--Validació correcta-->

<element_any>Això és un exemple</element_any>
<!--Validació correcta-->

<element_any>Això és <element_x> un exemple</element_x></element_any>
<!--Validació correcta-->
```

- **, (coma)**: Significa que l'element conté una sèrie de fills en l'ordre indicat.

Exemple:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE alumne [
   <!ELEMENT alumne (nom, data_matricula, modul)>
   <!ELEMENT nom (#PCDATA)>
   <!ELEMENT data_matricula (#PCDATA)>
   <!ELEMENT modul (#PCDATA)>
]>

<alumne>
   <nom>Carlos</nom>
   <data_matricula>26-09-2020</data_matricula>
   <modul>Llenguatges</modul>
</alumne>
<!--Validació correcta-->

<alumne>
   <nom>Carlos</nom>
   <data_matricula>26-09-2020</data_matricula>
</alumne>
<!--Validació incorrecta-->
```

- **| (o lògic)**: Significa que l'element conté només un dels dos fills indicats.

Exemple:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE alumne [
   <!ELEMENT alumne (nom, (data_matricula | curs), modul)>
   <!ELEMENT nom (#PCDATA)>
   <!ELEMENT data_matricula (#PCDATA)>
   <!ELEMENT curs (#PCDATA)>
   <!ELEMENT modul (#PCDATA)>
]>

<alumne>
   <nom>Carlos</nom>
   <data_matricula>26-09-2020</data_matricula>
   <modul>Llenguatges</modul>
</alumne>
<!--Validació correcta-->

<alumne>
   <nom>Cecília</nom>
   <curs>segon</curs>
   <modul>Llenguatges</modul>
</alumne>
<!Validació correcta-->
```

És possible declarar elements mixtes,és a dir, que poden contenir dades caràcter i elements secundaris, per exemple: <!ELEMENT nom_element (#PCDATA | element_fill)*>

- **? (interrogant)**: Significa que l'element pot aparèixer o no, però només una vegada.

Exemple:

```
<!DOCTYPE pizza [
  <!ELEMENT pizza (nom, forma?)>
  <!ELEMENT nom (#PCDATA)>
  <!ELEMENT forma (#PCDATA)>
]>

<pizza>
   <nom>Margarita</nom>
   <forma>redona</forma>
</pizza>
<!--Validació correcta-->

<pizza>
   <nom>Napolitana</nom>
</pizza>
<!--Validació correcta-->
```

- *** (asterisc)**: Significa que l'element pot no aparèixer o aparèixer una o més vegades.

Exemple:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE arcoiris [
  <!ELEMENT arcoiris (color*)>
  <!ELEMENT color (#PCDATA)>
]>

<arcoiris>
  <color>Blau</color>
  <color>Negre</color>
  <color>Groc</color>
</arcoiris>
<!--Validació correcta-->

<arcoiris> </arcoiris>
<!--Validació correcta-->
```

- **+ (signe de la suma)**: Significa que l'element ha d'aparèixer una o més vegades (no pot no parèixer).

Exemple:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE arcoiris [
  <!ELEMENT arcoiris (color+)>
  <!ELEMENT color (#PCDATA)>
]>

<arcoiris>
  <color>Blau</color>
  <color>Negre</color>
  <color>Groc</color>
</arcoiris>
<!--Validació correcta-->

<arcoiris>
  <color>Blau</color>
</arcoiris>
<!--Validació correcta-->
```

- **() (parèntesis)**: Permet agrupar expressions.

Exemple:

```
<!DOCTYPE element_agrupacio [
  <!ELEMENT element_agrupacio (a, (a|b))>
  <!ELEMENT a EMPTY>
  <!ELEMENT b EMPTY>
]>
```

2- Declaració d'entitats

Fina ara hem associat un document XML amb un únic fitxer, però això no sempre és així. L'XML ens permet organitzar el contingut d'un document XML de manera més flexible. Per exemple, si tenim un llibre en format XML, aquest no té perquè incloure-se en un únic fitxer, podriem voler que cada capítol estigués en un fitxer diferent. El que permet aquesta flexibilitat s'anomena "entitat".

Una entitat és una unitat d'emmagatzemament que pot contenir qualsevol cosa, des de una cadena de caràcters fins un fitxer, un gràfic, etc. i per tant ens permet dividir el document XML en diferents objectes d'emmagatzemament. Després l'aplicació XML que hagi de treballar amb aquest document XML tractarà aquest conjunt d'objectes com un únic objecte XML.

- Una entitat consisteix en un nom i el seu valor (són similars a les constants en els llenguatges de programació).
- Amb algunes excepcions, el processador XML substitueix les referències a entitats pels seus valors abans de processar el document.
- Un vegada definida l'entitat, es pot utilitzar en el document escrivint una referència a l'entitat, que comença amb el caràcter "&", segueix amb el nom de l'entitat i acaba amb ";". (És a dir, & nom_entitat;).

- **Declaracions d'entitats generals**

  Les entitats generals poden utilitzar-se en el cos d'un document XML i en el seu DTD. Poden representar caràcters, paràgrafs i també documents sencers.

```
Sintaxi:   <!ENTITY nom_entitat "text">
```

```
Exemple: Entitat general

<?xml version="1.0"?>
<!DOCTYPE cadenes_text [
   <!ELEMENT cadenes_texts (text+)>
   <!ELEMENT text (#PCDATA)>
   <!ENTITY escriptor "Miguel de Cervantes">
   <!ENTITY obra "El Quijote">
]>
<cadenes_texts>
   <text>&obra; va ser escrita per &escriptor; </text>
   <text>&escriptor; era espanyol</text>
</cadenes_text>

Quan un analitzador XML troba una referència "&nom_entitat;", reportarà un text de recanvi a l'aplic
ació en el lloc de la referència.
Pots fer la prova: Guarda el codi en un document XML i visualitza el resultat en una navegador
```

Les entitats poden ser internes o externes:

- **Declaracions d'entitats internes**

  Si una entitat es declara en una DTD s'anomenarà entitat interna.

```
Exemple: Entitat interna

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE direccio [
   <!ELEMENT direccio (#PCDATA)>
   <!ENTITY nom "Tonieta Sampol">
   <!ENTITY empresa "Info, S.A.">
   <!ENTITY telefon "670666888">
]>

<direccio>
   &nom
   &empresa
   &telefon
</direccio>
```

- **Declaracions d'entitats externes**

  És quan l'entitat es declara fora de la DTD. En podem trobar de dos tipus:

- **L'entitat fa referència a un fitxer de text:** En aquest cas l'entitat se substitueix pel contingut de l'arxiu. Són entitats analitzables.
  - L'entitat pot ser una entitat de sistema, amb la sintaxi: <! ENTITY nom_entitat SYSTEM "camí">
  - Pot ser una entitat pública, amb la sintaxi: <! ENTITY nom_entitat PUBLIC "Identificador_DTD" "camí">
- **L'entitat fa referència a un fitxer que no és de text (per exemple, una imatge).** Són entitats no analitzables, ja que fan referència a dades que un processador XML no té perquè analitzar**.** En aquest cas l'entitat no es substitueix pel contingut de l'arxiu:
  - L'entitat pot ser una entitat de sistema, amb la sintaxi: <! ENTITY nom_entitat SYSTEM "camí" NDATA tipus>
  - Pots ser una entitat pública, amb la sintaxi: <! ENTITY nom_entitat PUBLIC "Identificador_DTD" "camí" NDATA tipus>
- *En tots aquests casos:*
  - *"nom_entitat" és el nom de l'entitat.*
  - *"camí" és el camí (absolut o relatiu) fins a un arxiu.*
  - *"tipus" és el tipus de fitxer (gif, jpg, etc).*
  - *"Identificador_DTD" és un identificador públic formal (Formal Public Identifier).*

Exemple: Entitat externa analitzable privada

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE direccio
[
  <!ELEMENT direccio (empresa, telefon)>
  <!ENTITY nom SYSTEM "clients.txt">
  <!ELEMENT empresa (#PCDATA)>
  <!ELEMENT telefon (#PCDATA)>
]>

<direccio>
   &nom;
   <empresa>Info, S.A.</empresa>
   <telefon>670666888</telefon>
</direccio>

Suposam que "clients.txt" conté "Tonieta Sampol".
```

- **Declaracions d'entitats de caràcter**

  Les entitats de caràcter s'utilitzen per nombrar qualcuna de les entitats que són representacions simbòliques de información, és a dir, caràcters que són difícils o impossibles d'escriure.

Exemple: Entitats de caràcters

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE autor[
   <!ELEMENT autor (#PCDATA)>
   <!ENTITY escriptora "Tonina Sampol">
   <!ENTITY copyright "&#169;"> Aquest valor representa ©.
]>

*Nota: Consulta de les entitats de caràcter a: https://es.wikipedia.org/wiki/Anexo:Referencias_a_ent
idades_de_caracteres_XML_y_HTML#Est%C3%A1ndares_p%C3%BAblicos_de_conjuntos_entidades_de_caracteres
```

- **Entitats ja existents**

  A XML ja estan definides les següents entitats:

  - ampersand (seria l'And o i)): &
  - Single quote (cometes simples): '

- Major que: >
- Menor que: <
- Double quote (cometes dobles) : "

Aquestes entitats no hi ha que declarar-les dins cap DTD.

```
Exemple:

<?xml version="1.0"?>
<nota>
   <descripcio>És una empresa de serveis &amp; assessoria </descripcio>
</nota>

Es veuria així:
<nota>
<descripcio>És una empresa de serveis & assessoria </descripcio>
</nota>
```

3- Declaració d'atributs

Els atributs donen més informació sobre un element, és a dir, defineixen una propietat d'un element.

A un atribut d'XML, sempre el trobem en forma de "name-value pair'". Un element pot tenir molts o pocs atributs únics.

La declaració d'atributs és molt similar a la declaració d'elements en molts de sentits, excepte en un: en lloc de declarar contingut permès per elements, es declara una llista d'atributs permesos per a cada element. Aquestes llistes s'anomenen: declaració 'ATTLIST' (de llista d'atributs).

```
Una declaració d'atributs segueix la següent sintaxi:

<!ATTLIST nom_element nom_atribut tipus_atribut valor_inicial_atribut >

On:

- "nom_element" és el nom de l'element per el qual es defineix un atribut.
- "nom_atribut" és el nom de l'atribut.
- "tipus_atribut" és el tipus de dades.
- "valor_inicial_atribut" és el valor predeterminat de l'atribut.
```

```
Per definir diversos atributs d'un mateix element, es poden utilitzar una o diverses declaracions d'atri
buts:

<!ATTLIST nom_element nom_atribut1 tipus_atribut1 valor_inicial_atribut1>
<!ATTLIST nom_element nom_atribut2 tipus_atribut2 valor_inicial_atribut2>

<!ATTLIST nom_element
  nom_atribut1 tipus_atribut1 valor_inicial_atribut1
  nom_atribut2 tipus_atribut2 valor_inicial_atribut2
 >

On: Les dues declaracions són equivalents.
```

- **Tipus d'atributs:**
  - **CDATA**: L'atribut conté caràcters. No pot contenir etiquetes.

```
<!DOCTYPE element_x [
  <!ELEMENT element_x EMPTY>
  <!ATTLIST element_x color CDATA>
]>


<element_x color="vermell" />
<!--Validació correcte-->
```

- **NMTOKEN:** L'atribut només conté pot contenir lletres, dígits i els caràcters punt, guió, subratllat i el dos punts.

```
<!DOCTYPE element_x [
  <!ELEMENT element_x EMPTY>
  <!ATTLIST element_ color NMTOKEN #REQUIRED>
]>


<element_x color="blau_turquesa" />
<!--Validació correcta-->

<element_x color="blau turquesa" />        <!-- ERROR: hi ha un espai en blanc -->
<!--Validació correcta-->
```

- **NMTOKENS**: L'atribut és com NMTOKEN acceptant també espais en blanc.

```
<!DOCTYPE element_x [
  <!ELEMENT element_x EMPTY>
  <!ATTLIST element_x color NMTOKENS #REQUIRED>
]>

<element_x color="1" />
<!--Validació correcta-->

<element_x color="blau turquesa" />
<!Validació correcta-->

<element_ color="2*2" />        <!-- ERROR: hi ha un asterisc i NMTOKEN no admet asteriscs-->
<!--Validació incorrecta-->
```

- **valors**: L'atribut només pot contenir uns dels elements d'una llista. La llista ha d'anar entre parèntesis, separats els elements per una barra vertical "|".

  Si volem establir un element per defecte, es posa entre cometes.

```
<!DOCTYPE arcoiris [
  <!ELEMENT arcoiris EMPTY>
  <!ATTLIST arcoiris color (blau|blanc|vermell) #REQUIRED>
]>


<arcoiris color="blau" />
<!--Validació correcta-->

<arcoiris color="verd" />       <!-- ERROR: "verd" no està a la llista de valors -->
<!Validació correcta-->
```

- **ID**: El valor de l'atribut (no el nom) ha de ser únic i no es pot repetir a altres elements o atributs. No pot contenir espais en blanc.

```
<!DOCTYPE biblioteca [
  <!ELEMENT biblioteca (llibre*)>
  <!ELEMENT llibre (#PCDATA) >
  <!ATTLIST llibre codi ID #REQUIRED>
]>

<biblioteca>
  <llibre codi="Lli1">Poema de Mio Cid</llibre>
  <llibre codi="Lli2">Cien años de Soledad</llibre>
</biblioteca>
<!--Validació correcta-->

<biblioteca>
  <llibre codi="Lli1">Poema de Mio Cid</llibre>
  <llibre codi="Lli1">Cien años de Soledad</llibre>       <!-- ERROR: no es pot repetir un atribut d
e tipus ID -->
</biblioteca>
```

- **IDREF**: El valor de l'atribut ha de coincidir amb el valor de l'atribut ID d'un altre element.

```
<!DOCTYPE biblioteca [
  <!ELEMENT biblioteca ((llibre|prestec)*)>
  <!ELEMENT llibre (#PCDATA) >
  <!ATTLIST llibre codi ID #REQUIRED>
  <!ELEMENT prestec (#PCDATA) >
  <!ATTLIST prestec codi_prestec IDREF #REQUIRED>
]>

<biblioteca>
  <llibre codi="Lli1">Poema de Mio Cid</llibre>
  <prestec codi_prestec="Lli1">Pepe Mir</prestec>
</biblioteca>
<!--Validació correcta-->

<biblioteca>
  <llibre codi="Lli1">Poema de Mio Cid</llibre>
  <prestec codi_prestec="Lli2">Pepe Mir</prestec>          <!-- ERROR: el valor "Lli2" no és ID de c
ap element -->
</biblioteca>
<!--Validació incorrecta-->
```

- **IDREFS**: El valor de l'atribut és una sèrie de valors separats per espais que coincideixen amb el valor de l'atribut ID d'altres elements.

```
<!DOCTYPE biblioteca [
  <!ELEMENT biblioteca ((llibre|prestec)*)>
  <!ELEMENT llibre (#PCDATA) >
  <!ATTLIST llibre codi ID #REQUIRED>
  <!ELEMENT prestec (#PCDATA) >
  <!ATTLIST prestec codi_prestec IDREFS #REQUIRED>
]>

biblioteca>
  <llibre codi="Lli1">Poema de Mio Cid</llibre>
  <llibre codi="Lli2">Cien años de Soledad</llibre>
  <prestec codi_prestec="Lli1 Lli2">Pepe Mir</prestec>
</biblioteca>
<!Validació correcta-->

<biblioteca>
  <llibre codi="Lli1">Poema de Mio Cid</llibre>
  <llibre codi="Lli2">Cien años de Soledad</llibre>
  <prestec codi_prestec="Lli3"> Pep Matas</prestec>          <!-- ERROR: el valor "Lli3" no és ID de
cap element -->
<biblioteca>
<!--Validació incorrecta-->
```

- **Valors inicials dels atributs:**
  - **#REQUIRED**: L'atribut és obligatori, encara que no s'especifica cap valor predeterminat.

```
<!DOCTYPE obligatori [
  <!ELEMENT obligatori EMPTY>
  <!ATTLIST obligatori color CDATA #REQUIRED>
]>

<obligatori color="" />
<!--Validació correcta-->

<obligatori color="grog" />
<!Validació correcta-->

<obligatori />          <!-- ERROR: falta l'atribut "color" -->
<!--Validació incorrecta-->
```

  - **#IMPLIED**: L'atribut no és obligatori i no s'especifica cap valor predeterminat.

```
<!DOCTYPE no_obligatori [
  <!ELEMENT no_obligatori EMPTY>
  <!ATTLIST no_obligatori color CDATA #IMPLIED>
]>

<no_obligatori />
<!--Validació correcta-->

<no_obligatori color="" />
<!--Validació correcta-->

<no_obligatori color="grog" />
<!Validació correcta-->
```

- **#FIXED valor**: L'atribut té un valor fix.

```
<!DOCTYPE fixat [
  <!ELEMENT fixat EMPTY>
  <!ATTLIST fixat color CDATA #FIXED "verd">
]>

<fixat color="" />         <!-- ERROR: l'atribut "color" no té el valor "verd" -->
<!--Validació incorrecta-->

<exemple color="grog" />     <!-- ERROR: l'atribut "color" no té el valor "verd" -->
<!--Validació incorrecta-->
```

2.2. XML Schema: XSD

XSD: Significa definició d'esquema XML ((*XML Schema Definition*). S"utilitza al igual que la DTD, per definir l'estructura i el contingut d'un arxiu XML. Els documents d'esquema XML (XSD), en realitat són els successors de les DTD. Els dos poden ser utilitzats, però s'ha de tenir en compte que XSD és una millora general de la DTD.

Té més avantatges que DTD:

- XSD és extensible (més que la DTD).
- Suporta tipus de dades i espais de noms (la DTD no). Veurem al pròxim capítol que és un espai de noms.
- XSD proporciona més control sobre l'estructura que la DTD.
- És més potent que la DTD.
- Es basa en la gramàtica per proporcionar una potència expressiva molt gran.
- Utilitza sintaxi XML, cosa que li permet especificar de forma més detallada un extens sistema de tipus de dades.

Té un inconvenient:

A l'hora de la validació del document, la utilització de XSD suposa un gran consum en recursos i temps que causa del seva gran especificació i complexitat en la sintaxi (els esquemes són més difícils de llegir i d'escriure).

Comparativa DTD i XSD

Concepte	DTD	XSD
Llenguatge	• No és XML	• XML
Definició estructura	• Limitada. • Insuficient per a continguts mixts. • Repeticions de les etiquetes limitades a 0, 1 o moltes (poc concret).	• Permet definició acurada d'elements mixts. • Especificació del nombre de repeticions màxim i mínim que es pot repetir una etiqueta.
Tipus de dades	• Molt limitat	• Disposa dels tipus disponibles en els llenguatges de programació. • Possibilitat de definir nous tipus.
Múltiples espais de noms	• Només una DTD per document	• Possibilitat de disposar de múltiples diccionaris (espais de noms diferents).

2.3. Validació per XSD

- ### Arrel

  A l'arrel és on escriurem l'element arrel del esquema XML:

  ```
  <xs:schema>
  ...
  </xs:schema>
  ```

  A més pot contenir:

  ```
  Arrel completa:

  <?xml version="1.0"?>
  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="EspaiNomsPropi"
  xmlns="EspaiNomsPropi"
  elementFormDefault="qualified">
  ...
  ...
  </xs:schema>
  ```

  On:

  - **xmlns:xs="http://www.w3.org/2001/XMLSchema"**, és la definició de l'espai de noms (*namespace* W3C). És obligatori. Al crear un XML schema feim ús dels elements i atributs especificats a l'estàndard d'XML Schema. Per fer això possible s'ha d'incloure dins l'element arrel de l'esquema (element "schema") una referència a l'espai de noms "http://www.w3.org/2001/XMLSchema".

    Espai de noms:

    La premissa principal de XML és la creació d'un llenguatge de marques format per elements totalment personalitzats. Això és, el programador és el que donarà nom a totes les marques del seu arxiu XML, i aquestes marques poden tenir qualsevol nom (sempre que sigui vàlid). Quan es creen elements personalitzats en un entorn tancat no hi ha cap problema, ja que podem triar nosaltres els noms i assignar marques amb diferent nom de les ja utilitzades. El problema apareix quan es mesclen molts de vocabularis XML. És aquí on apareixen els espais de nom.

    Un espai de noms XML (XML *Namespaces)* és una recomanació de W3C (a partir de 1999) per a proporcionar elements i atributs amb nom únic a un arxiu XML. No formava part de l'especificació XML 1.0 original.

    Un arxiu XML pot contenir noms d'elements o atributs procedents de més d'un vocabulari XML. Si a cadascun d'aquests vocabularis se'ls assigna un espai de noms, un àmbit semàntic propi, referenciat a un URI*, es resol l'ambigüitat existent entre elements o atributs que s'anomenin igual (l'homonímia). Els noms d'elements dins de cada espai de noms han de ser únics. L'espai de noms ens permet diferenciar les etiquetes XML de l'esquema, respecte a les del document XML.

    ```
    Els espais de noms es van crear perquè no existissin col·lisions entre els diferents mòduls de p
    rogramari que eren capaços de reconèixer les marcacions (etiquetes i atributs) del llenguatge XM
    L, ja que els diferents documents que contenen marcacions de diferents fonts independents entre
     si, solien tenir problemes de reconeixement quan havien estat creats per altres programes de pr
    ogramari que, no obstant això, utilitzaven el mateix tipus d'element o nom d'atribut. Una de les
    motivacions per a aquesta modularitat és que, si hi ha un conjunt de marcacions disponibles, que
    són comprensibles i per a les que hi ha programari útil disponible, és millor la reutilització
     d'aquestes marques que el fet reinventar unes noves. Així, es va considerar que les construccio
    ns de documents havien de tenir noms universals, l'àmbit s'estengués més enllà del document que
     les contingui. L'especificació Namespaces XML descriu un mecanisme, els espais de noms XML, que
    du a terme aquesta missió.
    ```

***URI**: S'utilitzen perquè el nom sigui únic, no són enllaços, ni han de contenir informació. Per exemple "http://www.w3.org/2001/XMLSchema" no conté ningun codi, simplement descriu l'espai de noms XMLSchema a lectors humans. El fet d'emprar un URL per identificar un espai de noms, en lloc d'una simple cadena (com "XML Schema"), redueix la possibilitat que diferents espais de noms emprin identificadors iguals. Els identificadors dels espais de noms no necessiten seguir les convencions de les adreces d'Internet, encara que sovint ho facin. Un URI és una cadena curta de caràcters que identifica inequívocament un recurs, normalment accessible en una xarxa o sistema.*

Com se referència un espai de noms amb prefix? Quan ja s'ha declarat un espai de noms i s'ha assignat el prefix per indicar que un element pertany a aquest espai de noms, s'anteposa aquest prefix seguit de dos punts (:) al nom de l'element. Aquest prefix ha d'aparèixer tant a l'etiqueta d'inici com a la de tancament de l'element.

```
La sintaxi genèrica per descriure o definir l'espai de noms és:

xmlns:prefix ="uri_espai_noms"

On:
- "xmlns:xs ...", indica que tots els elements o atributs que duen el prefix "xs:" pertanyen a
 l'espai de noms especificat a la URI (http:...).
- Els prefixos s'utilitzen per distingir entre diferents espais de noms. Es pot utilitzar qualse
vol prefix, sempre que s'especifiqui l'espai de noms XML associat. Si l'esquema només fa referèn
cia a un únic espai de noms, no és obligatori emprar el prefix, en aquest cas la declaració ens
 pot quedar: "<schema xmlns="http://...">".
- Recalcar que els prefixos més utilitzats són: "xs", "xsd".

*Nota: Un espai de nom té en realitat dos noms: un "sobrenom" o el que seria el nom local (el pr
efix) i un nom real (l'URI).
```

- **targetNamespace="EspaiNomsPropi" i xmlns="EspaiNomsPropi",** ens permet definir l'espai de noms propi. Com a conseqüència, podem tenir diferents prefixos que ens permetran distingir entre elements propis i elements de l'espai de noms XML Schema:

```
Ens quedaria:

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace:="http://www.nostrelloc.es/doc">
xmlns:doc="http://www.nostrelloc.es/doc"

*Nota: Prefix espai de noms propi seria "doc"
```

- **elementFormDefault="qualified"**, ens permet definir si els elements i atributs de l'esquema necessiten anar precedits del prefix de l'espai de noms (per defecte és “unqualified”, que significa que no necessiten anar precedits per els prefix que indica l'espai de noms).

```
Ens quedaria:

elementFormDefault = "qualified"
attributeFormDefault = "unqualified"
```

- **xsi: schemaLocation**: Aquest atribut situa esquemes per a elements i atributs que es troben en un espai de noms especificat. El seu valor és una URI d'espai de noms seguit d'una URL relativa o absoluta on es pot trobar l'esquema per a aquest espai de noms. En general, s'adjunta a l'element arrel, però pot aparèixer més baix en l'arbre.
- **xsi: noNamespaceSchemaLocation**: Aquest atribut ens serveix per especificar la ubicació d'un esquema XML que no té un espai de noms de destinació. Aquest atribut no pren una llista d'URL, només pot especificar una ubicació d'esquema.

Exemple:

Document instància XML:

```
<persona  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="humans.xsd">
  <nom>Joan</nom>
  <direccio>C/Sol</direccio>
</persona>
```

**Nota: Si ens fixam, posam "xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="humans.xsd" dins l'àmbit de l'etiqueta arrel, on també hi podran anar atributs d'informació*

El document XSD que conté l'esquema XSD és "humans.xsd":

```
<?xml version="1.0" encoding="UTF-8"?>
 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" attributeFormDefault="qualified">
 <xs:element name="persona">
    <xs:complexType>
       <xs:sequence>
          <xs:element name="nom" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="direccio" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
       </xs:sequence>
    </xs:complexType>
 </xs:element>
 </xs:schema>
```

**Nota: Aquest XSD defineix o declara un element anomenat "persona" que és de tipus compost que veurem més endavant*

Declarar elements

- **Sintaxi:**

```
<xs:element name="nomElement"  type="xs:integer"  minOccurs="num_min"  maxOccurs="num_max"  fixed="v
alorElement"   default="valorXdefecte">
</xs:element>

També ho podem escriure:
<xs:element name="nomElement"  type="xs:integer"  minOccurs="num_min"  maxOccurs="num_max"  fixed="v
alorElement"   default="valorXdefecte"/>

On:
- name: És el nom de l'element.
- type: Defineix el tipus d'element. Poden ser Tipus simples i tipus complexos. El tipus de'element,
que pot ser:
    xs:boolean
    xs:date
    xs:decimal
    xs:integer
    xs:string
- minOccurs i maxOccurs (opcionals): Indiquen el mínim i màxim (respectivament) de números d'ocurrèn
cies de l'element.
    El valor per defecte dels és 1.
    Si es vol indicar que l'element pot aparèixer un número il·limitat de vegades, l'atribut "maxOcc
urs" tendrà el valor "unbounded".
- fixed (opcional): Especifica un valor fix per l'element.
- default (opcional): Especifica un valor per defecte per l'element.

* Nota:
   -Els valors "fixed" i "default" dels elements només s'afegeixen al document XML si l'element està
present.
   -Els elements amb un valor "fixed", poden estar buits. Si no són buits el seu contingut ha de coi
ncidir amb el definit a l'atribut "fixed".
```

```
Exemple: Declarar un valor per defecte a un element

<xs:element name="edat" type="xs:integer" default="18">
</xs:element>
```

```
Exemple: Establir un valor obligatori (automàticament assignat a l'element)

<xs:element name="altura" type="xs:string" fixed="300px">
</xs:element>
```

- **Elements simples**

  Són elements que només poden contenir dades de tipus caràcter, no poden contenir altres elements ni atributs.
  Alguns dels tipus de dades primitius són : string, boolean, decimal, float, double, de tipus data-hora (duration, dateTime, time, date, gYearMonth, etc.), hexBinary, anyURI, etc.
  Alguns del tipus de dades no primitius són : normalizedString, token, language, IDREF, Name, integer, long, etc.

```
Sintaxi per declarar un element simple:

<xs:element name="nom_element" type="tipus_element">
</xs_element>
```

Exemple: Declaram un element anomenat "fecha", de tipus "date".

XSD:
```
<xs:element name="fecha" type="xs:date"/>
```

XML:
```
<fecha>2002-09-24</fecha>  <!--Validació correcta-->
```

Exemple: Declaram dos elements simples anomenats "titol" de tipus "string", i "capitols" de tipus "integer".

XSD:
```
<xs:element name="titol" type="xs:string"/>
<xs:element name="capitols" type "xs:integer"/>
```

Si iniciam el procés de Validació podem dir que els següents elements són vàlids:
XML:
```
<titol>La dama de las camelias</titol>
<capitols>10</capitols>
```

- Tipus personalitzats

Són creats per l'usuari a partir d'un tipus de dada predefinida. Se defineix un nom que després es pot emprar a qualsevol element del document mitjançant l'atribut "type". També poden emprar-se directament a la definició d'un element en lloc d'emprar-se amb l'atribut type.

Exemple: Tipus personalitzat directament a la definició de l'element.

```
<xs:element name="edat" minOccurs="1" maxOccurs="1">
  <xs:simpleType>   <!--Aquí declaram el tipus de l'element-->
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="0"/>
      <xs:maxInclusive value="100"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Nota: xs:restriction, xs:minInclusive, xs:maxInclusive es veuen a la part de restriccions.

Exemple: Tipus personalitzat mitjançant l'atribut type.

```
<xs:simpleType name="maxim-llarg">
  <xs:restriction base="xs:string">
    <xs:minLength value="0"/>
    <xs:maxLength value="20"/>
  </xs:restriction>
</xs:simpleType>

<xs:element name="nombre" type="maxim_llarg"/>
```

Nota: xs:restriction, xs:minLength, xs:maxLength es veuen a la part de restriccions.

- **Restriccions**

  Les restriccions s'utilitzen per definir els valors que són acceptables pels elements i atributs XML.

Sintaxi:

```
<xs:simpleType name=”nom”>
   <xs:restriction base="tipus">
      ...definicions de les restriccions...
   <xs:restriction>
</xs:simpleType>
```

- Rang de números: S'utilitza en els tipus de dades numèriques i data/hora. Defineix el mínim i màxim dels números permesos (minInclusive i maxInclusive, minExclusive i maxExclusive, depenent si es volen incloure els límits)

Exemple 1:

```
<xs:element name="edat" minOccurs="1" maxOccurs="1">
  <xs:simpleType>
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="0"/>
      <xs:maxInclusive value="100"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Exemple 2:

```
<xs:element name="edat" minOccurs="1" maxOccurs="1">
  <xs:simpleType>
    <xs:restriction base="xs:integer">
      <xs:minExclusive value="0"/>
      <xs:maxExclusive value="100"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

- Dígits: S'utilitza en els tipus de dades numèriques. Defineix el número màxim de dígits permesos (totalDigits).

Exemple:

```
<xs:element name="telefono" minOccurs="1" maxOccurs="1">
  <xs:simpleType>
    <xs:restriction base="xs:integer">
      <xs:totalDigits value="9"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

- Llista de valors: S'utilitza en tots els tipus de dades. Defineix una llista de valors permesos a l'element (enumeration).

Exemple:

```
<xs:element name="color" minOccurs="1" maxOccurs="1">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="Vermell"/>
      <xs:enumeration value="Verd"/>
      <xs:enumeration value="Grog"/>
      <xs:enumeration value="Blau"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

- Longitud: S'utilitza en tipus de dates de text. Defineix el número exacte de caràcters permesos, o el mínim i màxim d'ells (length, minLength, maxLength).

Exemple 1:

```
<xs:element name="text" minOccurs="1" maxOccurs="1">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:length value="25"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Exemple 2:

```
<xs:element name="text" minOccurs="1" maxOccurs="1">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:minLength value="1"/>
      <xs:maxLength value="25"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

- Plantilla de caràcters: S'utilitza en tipus de dades de text. Especifica un patró o expresió regular que ha de complir el contingut de l'element (pattern).

Exemple 1:

```
<xs:element name="text" minOccurs="1" maxOccurs="1">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[A-Z][A-Za-z][A-Za-z]"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Exemple 2:

```
<xs:element name="genere">
 <xs:simpleType>
   <xs:restriction base="xs:string">
     <xs:pattern value="Home|Dona"/>
   </xs:restriction>
 </xs:simpleType>
</xs:element>
```

- Restriccions als espais en blanc: Especifica com s'ha de tractar els possibles espais en blanc, les tabulacions, els salts de línia i els retorns de carro que puguin aparèixer (whiteSpace).

Exemple:

```
<xs:element name="direccio">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:whiteSpace value="preserve"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>

On:
"preserve": Les dades no se modifiquen, queden tal i com estan escrits.
"replace": Els tabuladors, salts de línia i retorn de carro són substituïts per espais.
"collapse": Fa el mateix que "replace", però a més substitueix espais múltiples per un només.
```

- **Elements complexos**

  Un element complex és el que pot contenir altres elements i/o atributs. Per declarar elements complexos utilitzam "complexType".

  - Consideracions sobre elements complexos
    - Un element complex pot estar buit, és a dir, no contenir ni elements ni text, però si algun atribut. Si no contenen atributs, es poden declarar com a elements simples.

      Exemple: Element buit

      ```
      <xs:element name="autor" type="buit_tipus_cadena">
         <xs:complexType name="buit_tipus_cadena">
            <xs:atribute name="nom" type="xs:string"/>
         </xs:complexType>
      </xs:element>
      ```

      L'exemple següent és vàlid, validat amb l'element de tipus complex anterior:

      ```
      <autor nom="Almudena Grandes"/>
      ```

    - Un element complex pot contenir altres elements, ho indicam amb "mixed".

```
<xs:element name="confirmacioComanda">
  <xs:complexType mixed="true">
    <xs:sequence>
      <xs:element name="intro" type="xs:string"/>
      <xs:element name="nom" type="xs:string"/>
      <xs:element name="data" type="xs:string"/>
      <xs:element name="comanda" type="xs:string"/>
   <xs:sequence>
  </xs:complexType>
</xs:element>
```

Aquest element, es podria emprar dins d'un document XML de la següent manera:

```
<confimacioComanda>
   <intro>Per:</intro>
   <nom>Joan Pérez</nom>
   Confirmació amb data de <data>24-01-2020</data> que s'ha rebut la comanda <comanda>aspira
dor</comanda>.
</confimacioComanda>
```

Existeixen tres formes d'especificar els subelements dins de l'element, mitjançant tres tipus d'elements predefinits:

- "sequence": Serveix per especificar l'ordre en el que obligatòriament han d'aparèixer els elements fill d'un element.

  ```
  <xs:element name="persona">
     <xs:complexType>
      <xs:sequence>
        <xs:element name="Primer_llinatge" type="xs:string"/>
        <xs:element name="Segon_llinatge" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  XML:
  <persona>
     <primer_llinatge> López </primer_llinatge>
     <segon_llinatge> Sureda </segon_llinatge>
  </persona>

  <!--Validació correcta-->
  ```

- "choice": Serveix per especificar que només es permet escriure un dels element fills.

```
<xs:element name="persona">
  <xs:complexType>
    <xs:choice>
      <xs:element name="nom" type="xs:string"/>
      <xs:element name="llinatge" type="xs:string"/>
    </xs:choice>
  </xs:complexType>
</xs:element>

XML's:
<persona>
   <nom> Aina </nom>
</persona>
<!--Validació correcta-->

<persona>
   <llinatge> Gómez </llinatge>
</persona>
<!--Validació correcta-->
```

- “all”: Serveix per indicar que els elements fills d'un element poden aparèixer en qualsevol ordre.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="lloc">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="ciutat">
          <xs:complexType>
            <xs:all>
              <xs:element name="nom" type="xs:string"/>
              <xs:element name="pais" type="xs:string"/>
            </xs:all>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>

XML:
<lloc>
   <ciutat>
     <pais>Espanya</pais>
     <nom>Palma de Mallorca</nom>
   </ciutat>
</lloc>
<!--Validació correcta-->
```

Declarar d'atributs

Els elements simples no poden tenir atributs, quan tenim atributs a un element, aquest és un element complex.

Un atribut ha de ser de un tipus simple, és a dir, no poden contenir altres elements. S'han de declarar just abans de tancar l'etiqueta xs:complexType. Poden ser opcionals i poden tenir un valor assignat per defecte.

Sintaxi per definir un atribut

```
<xs:attribute name="nom atribut" type="tipus atribut" use="valor" default="valor" fixed="valor"/>
```

On:
  -name: Indica el nom de l'atribut.
  -type: Indica el tipus de dada que emmagatzemarà l'atribut.
  -use: Indica si l'existència de l'atribut és opcional (optional), obligatòria (required) o prohibida (prohibited). Por defecte opcional.
  -default: És el valor que prendrà l'element quan es processa si en el document XML no ha rebut cap valor. Només es pot emprar amb tipus de dada de text.
  -fixed: Indica l'únic valor que pot contenir l'element en el document XML. Només es pot emprar amb tipus de dades de text.

Exemple: declaració atribut

```
<xs:attribute name="nacionalitat" type="xs:string" use="required"/>
```

XML:

```
<llinatge nacionalitat="Espanyola">Servera</llinatge>
<!--Validació correcta-->
```

Exemple: Element complex amb atribut

```
<xs:element name="client">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="nom" type="xs:string"/>
         <xs:element name="llinatge" type="xs:string"/>
      </xs:sequence>
      <xs:attribute name="codi" type="xs:integer"/>
   </xs:complexType>
</xs:element>
```

2.4. Eines d'edició per XML

Creació de documents XML

Crear un document XML manualment és molt més senzill que crear un document binari, ja que no difereix gaire de crear un document de text. Simplement necessitem un editor de text que no enriqueixi el text.

- **Editors**

  Els documents XML són simples documents de text en què hem afegit algun tipus de metadades. Això permet que la creació de documents XML sigui realment senzilla, ja que es pot fer servir l'editor més senzill que trobem en qualsevol sistema operatiu per poder crear els nostres documents. A pesar d'això també han aparegut tota una sèrie d'editors pensats per fer l'edició de documents XML més senzilla. Aquests editors són molt diversos i normalment ofereixen diferents tipus d'assistència per evitar que es cometin errors en crear el document, comproven interactivament que el document sigui correcte, aconsellen etiquetes, etc. Molts dels editors especialitzats en XML normalment a més ofereixen moltes altres funcions com generació d'expressions XPath, creació de fulls d'estil, depuració de transformacions, peticions XQuery, etc.

  L'única cosa que s'ha de fer, és crear un document XML és un editor de text normal i corrent que no enriqueixi el text (els programes editors de text com són Microsoft Word, l'OpenOffice.org Writer, el LibreOffice Writer, etc., no són adequats). Els editors que si es poden emprar, són per exemple: Gedit de Linux, Bloc de notes, etc., i en alguns casos aquests editors més senzills fins i tot detecten que s'està editant un document XML i marquen amb colors diferents les etiquetes i les dades (per exemple Gedit).

  - **Editors amb suport d'XML**

  Són editors que donen suport a XML. Aquests editors normalment ofereixen una assistència mínima en editar XML, com acoloriment de les diferents seccions, comprovació automàtica del tancament de les etiquetes, o fins i tot se'n poden trobar amb auto completament d'etiquetes.

  - **Editors especialitzats en XML**

  Aquests editors estan dissenyats específicament per crear i editar documents XML de manera eficient i senzilla minimitzant les possibilitats que es cometin errors en l'edició. Generalment tots ofereixen un entorn amb un grup de finestres amb diferents vistes de l'edició per intentar que no es perdi la visió de conjunt, del que s'està creant.

  A part de la simple edició de documents XML, aquests editors permeten tot un ampli ventall de tasques amb les tecnologies relacionades amb l'XML com: Editar documents XML restringint les etiquetes que s'hi fan servir; Definir un esquema; Crear, convertir i depurar esquemes XML, XSLT, XPath, XQuery, WSDL, SOAP; Ajudes per crear documents en vocabularis basats en XML.

  Una característica interessant que ofereixen és la possibilitat d'editar documents XML des de diferents punts de vista. El més corrent sol ser fer-ho per mitjà de vistes de text o diferents vistes gràfiques destinades a amagar la complexitat dels documents XML als usuaris que fan servir l'editor.

  A part de l'edició de text molts editors també ofereixen vistes que permeten que un usuari pugui crear dades estructurades de manera gràfica sense que l'usuari ni tan sols sàpiga que està creant un document XML. Una d'aquestes vistes alternatives és la vista d'arbre. La vista d'arbre permet editar el document visualment a partir de l'estructura jeràrquica, de manera que no cal que el que està creant l'arbre conegui la sintaxi XML. Mentre l'usuari va creant l'arbre, l'editor en segon pla va creant el document XML corresponent. Amb la mateixa idea de fer que l'edició dels documents XML sigui més fàcil per als usuaris no especialitzats, també hi ha la vista de graella.

  La popularitat del format XML està fent que el nombre d'editors especialitzats no pari de créixer i que per tant es faci difícil triar l'editor que s'adapta més bé a les necessitats que un usuari pugui tenir.

  Alguns exemples d'editors per XML són: Visual Studio Code, File Viewer Plus 3, XML Explorer, XML Notepad 2007, EditiX XML Editor, Essential XML Editor, XML Tree Editor, ... i n'hi ha d'altres en línia, com per exemple: xmlGrid.net, XML Viewer,etc.

  - Extensions interessants per a Visual Studio Code:

Soporte de lenguaje XML de Red Hat. Esta extensión de VS Code brinda soporte para la creación y edición de documentos XML, basados en LemMinX XML Language Server.

Caracteristicas basicas: Informe de errores de sintaxis, Finalización de código general, Etiquetas de cierre automático, Sangría automática de nodo, Resaltado de símbolos, Enlaces de documentos, Símbolos y esquema del documento, Soporte de cambio de nombre, Formateo de documentos, Validación DTD, Finalización DTD, Formateo DTD, Validación XSD, Hover basado en XSD, Finalización de código basado en XSD, Soporte XSL, Catálogos XML, Asociaciones de archivo, Acciones de código, Almacenamiento en caché de esquemas.

Xml CompletePreview (Ayudante de edición XML (usando XSD schemaLocation). Esta extensión ayuda a editar archivos XML proporcionando sugerencias. Archivos de esquema de muestra proporcionados para tipos de archivo XAML (WPF, Avalonia) y para archivos csproj. No requiere ningún tiempo de ejecución como java, python o xmllint, mientras que realiza un análisis XSD parcial.

Características: Linter básico (XML + validación XSD parcial), Autocompletado rápido basado en XSD (utiliza comentarios de XSD), Formateo de XML (rango seleccionado o documento completo), Cierre automático y cambio de nombre automático para la etiqueta editada actualmente (funciona solo para una sola etiqueta en una línea determinada), Documentación de desplazamiento del mouse para nodos / atributos xml (utiliza comentarios de XSD)

Software per validar XML

- Per validar documents XML per DTD o per XSD podem emprar XML Copy Editor, que és gratuït, llançat sota la Llicència Pública General GNU. Està disponible en català, xinès (simplificat i tradicional), holandès, anglès, francès, alemany, italià, eslovac, suec i ucraïnès. Hi ha paquets Linux per a Arch, Fedora, Gentoo, openSUSE, Slackware i Ubuntu. Una versió portàtil de WinPenPack també està disponible. En el següent enllaç teniu una breu manual per descarregar el software i com emprar-lo: XML Copy Editor. Per Mac podem descarregar BaseX
- Per a validar documents XML amb DTD intern podem utilitzar el servei de validació d'XML online: https://www.xmlvalidation.com/. Simplement hem de carregar el fitxer ".xml" que conté l'especificació DTD o bé enganxar tot el document en l'àrea de text de la plana web. Posteriorment clicam damunt el botó "Validate". En el cas que hi hagi errors, la plana web ens informarà de la ubicació d'aquests.