2. Documents XML

Vegeu la recomanació de l'W3C relativa a l'XML en la secció "Adreces d'interès" del web del mòdul.

XML són les sigles d'*extensible markup language* (llenguatge d'etiquetatge extensible). És un llenguatge estàndard, una recomanació del World Wide Web Consortium (W3C, www.w3.org/TR/REC-xml). Està basat en l'estàndard ISO SGML.

L'XML està tan basat en SGML que qualsevol document XML és un alhora un document SGML correcte (encara que a l'inrevés no passa).

L'XML va sorgir per intentar superar els problemes que tenia l'HTML a l'hora de processar automàticament la informació que contenen les pàgines web, però que a més pogués funcionar de la mateixa manera que es fa servir l'HTML. A més, s'hi van afegir altres requisits, com que:

- Fos fàcil crear documents XML i que els humans els poguessin llegir i entendre fàcilment.
- No fos complicat fer programes d'ordinador que treballessin amb XML.
- L'XML es pogués fer servir en el màxim possible de camps d'aplicació i mantingués la compatibilitat amb SGML.

El resultat ha estat que l'XML:

- Defineix la sintaxi genèrica per marcar les dades amb valors comprensibles per als humans.
- És una manera de donar format als documents que és prou flexible per ser personalitzada per a diferents destinacions: web, impressores, bases de dades, etc.
- Està pensat perquè tothom el pugui fer servir sigui quina sigui la seva àrea d'interès.

2.1 Metallenguatge

En realitat l'XML no és un llenguatge de marques, sinó que és un llenguatge que ens permetrà definir els nostres llenguatges de marques propis. Això vol dir que podrem definir llenguatges de marques específics per a cada un dels camps d'interès. Si treballem en el món dels gràfics podem definir el nostre llenguatge

específic per definir aquests gràfics, i si treballem en el món de la premsa podrem definir el nostre llenguatge per representar les notícies.

Per aquest motiu sovint es diu que l'XML és un **metallenguatge**, ja que ens permet definir l'estructura i el vocabulari d'altres llenguatges de marques.

> L'XML és un llenguatge per crear llenguatges de marques.

Aquesta llibertat a l'hora de definir les marques que ens interessin és un dels punts forts d'XML, que el fan molt més potent i adaptable a les diferents complexitats dels entorns en els quals pugui caldre.

2.2 Elements

La base de l'XML són els elements. Un element normalment estarà format per l'obertura d'una etiqueta –amb atributs o sense–, un contingut –que també pot ser un grup d'etiquetes–, i el tancament de l'etiqueta.

En aquest exemple podem veure el que acabem de comentar. Definim una etiqueta `nom`, després el contingut `Pere Martí` i posteriorment tanquem l'etiqueta:

```
1 <nom>Pere Martí</nom>
```

Aquest és un dels punts forts de l'XML: queda bastant clar que el contingut de dins de l'etiqueta és un nom.

Es podria fer l'exemple una mica més complex afegint-hi un atribut.

> Els atributs s'afegeixen sempre en l'obertura de l'etiqueta.

```
1 <nom càrrec="director">Pere Marti</nom>
```

Un altre dels aspectes bàsics de l'XML és que no es preocupa de la presentació sinó que parteix de la idea que l'important és el contingut de les dades i no la manera com es visualitzaran. Aquesta característica el fa ideal per presentar la informació i posteriorment per mitjà d'algun procés convertir la informació en un format de presentació específic com HTML, PDF, PostScript, etc. A més, aporta una característica molt interessant des del punt de vista de la informàtica: permet tenir les dades separades de la manera de representar-les.

> L'XML permet separar el contingut de la manera com serà presentat aquest contingut als usuaris.

2.2.1 Etiquetes

Les etiquetes es defineixen dins del document XML i es defineix un format per separar-les clarament del contingut de dades. Estrictament parlant hi ha dos tipus d'etiquetes:

- **Les etiquetes d'obertura**
- **Les etiquetes de tancament**

La informació es distribueix entre els dos tipus d'etiquetes. Així s'aconsegueix una manera senzilla de definir quines parts del document són dades i quines són estructura.

Les *etiquetes d'obertura* es defineixen amb els símbols de més petit "<" i més gran ">" amb un nom d'etiqueta al mig.

```
<etiqueta>
```

I les *etiquetes de tancament* es defineixen igual que les d'obertura però a l'hora de començar l'etiqueta s'hi especifiquen dos símbols: un símbol de més petit i una barra "</". D'aquesta manera es poden distingir fàcilment els dos tipus d'etiquetes i queda clar en quin lloc s'ha d'afegir el contingut.

```
<etiqueta>Contingut</etiqueta>
```

L'XML no defineix cap etiqueta sinó que permet que les defineixi cadascú en funció del que necessiti representar. Això fa més senzill crear documents XML, ja que no cal conèixer cap etiqueta per poder començar a treballar i perquè permet adaptar-se a qualsevol tasca.

A l'hora de definir les etiquetes es pot deixar un espai o un salt de línia darrere del nom de l'etiqueta però mai abans ni al mig. Per tant aquestes etiquetes serien correctes:

```
<etiqueta1>
<etiqueta >
</etiqueta>
</etiqueta1 >
</etiqueta2
>
```

I en canvi aquestes altres no serien correctes:

```
< etiqueta>
<etiqueta 1>
< /etiqueta2>
```

Com s'acaba de veure, l'especificació XML defineix clarament com s'han de crear les etiquetes en els documents XML, però en canvi no defineix cap tipus ni significat associat a cap de les etiquetes. Si, per exemple, fem servir en HTML

l'etiqueta <b>, aquesta ens està indicant que el contingut que contindrà s'haurà de representar en negreta. Això obliga que per fer documents XML s'hagin de conèixer les etiquetes.

Per a l'XML les etiquetes només són una manera de separar el contingut i definir l'estructura de les dades que conté. La interpretació de què signifiquen les dades i com s'han de representar es deixa a qui llegeixi el document (independentment que aquest sigui un humà o bé un programa).

El tractament d'excepcions que s'ha de fer per interpretar HTML és una de les coses que fa més complexa la tasca de programar els navegadors web actuals.

A pesar de la llibertat a l'hora de deixar que els usuaris triïn les seves pròpies etiquetes, l'XML és molt més rigorós que altres llenguatges a l'hora de definir com s'han d'escriure. L'objectiu de tenir regles d'escriptura està determinat per la necessitat que els documents XML en un moment o un altre seran processats per un programa, i els programes es fan més complexos a l'hora de tractar amb excepcions. Per tant, una de les coses que fa l'XML és **evitar les excepcions tant com sigui possible**.

> Les etiquetes que s'obrin sempre s'hauran de tancar.

Per evitar les excepcions una de les regles bàsiques d'XML és que qualsevol etiqueta que s'obri **sempre s'ha de tancar**. De manera que aquest no seria un document XML vàlid:

```
1 <nom>Pere Garcia
```

I en canvi aquest sí:

```
1 <nom>Pere Garcia</nom>
```

A pesar que no hi ha etiquetes definides, ja que l'XML **permet crear les etiquetes que ens facin falta**, per un motiu de llegibilitat i d'interpretació de les dades **es recomana que les etiquetes siguin autoexplicatives i pronunciables**. Això és per evitar coses com la de l'exemple següent:

```
1 <pccc>Figueres</pccc>
```

Segons aquest segon exemple, com determinem què és "Figueres"? Per a un programa d'ordinador segurament no seria un problema massa gran perquè simplement agafaria la dada i la interpretaria tal com l'hagin programat però en canvi per a un humà un document amb etiquetes com aquesta seria impossible d'interpretar.

> Es recomana triar els noms de les etiquetes de manera que puguin ser interpretades per qui les llegirà independentment de si les processa una persona o un programa.

L'exemple anterior serà més fàcil d'interpretar si canviem l'etiqueta:

```
1 <ciutat>Figueres</ciutat>
```

2.2.2 Contingut

El contingut d'un element serà tot allò que hi hagi entre les etiquetes d'obertura i de tancament. En el contingut podem tenir simplement text com aquí:

```
<persona>Pere Martí</persona>
```

O bé el contingut poden ser altres elements. En l'exemple següent l'element `<persona>` conté dos elements més: `<nom>` i `<cognom>`, que a més tenen contingut dins seu:

```
<persona>
    <nom>Pere</nom>
    <cognom>Martí</cognom>
</persona>
```

Una tercera possibilitat seria combinar els dos casos anteriors i que el contingut sigui una mescla de text i elements.

```
<persona>
    Pere Martí
    <carrec>Director</carrec>
</persona>
```

També és possible definir etiquetes sense contingut de manera que el seu significat està determinat pel nom de l'etiqueta.

```
<persona>
</persona>
```

Els elements buits també es poden definir fent servir el símbol `/>` en tancar l'etiqueta. De manera que no hi ha cap diferència entre definir `<director>` d'aquesta manera:

```
<director/>
```

o d'aquesta:

```
<director></director>
```

Els elements buits, encara que es defineixin amb el seu format especial, han de seguir les mateixes normes que els elements normals. Per tant, serien correctes les definicions següents:

```
<director />
<director
/>
```

I no ho serien aquestes:

```
< director/>
<director/ >
```

L'XML no defineix cap restricció a l'hora de definir el contingut dels elements. Per tant:

- Podem fer servir qualsevol caràcter representable amb el codi de caràcters.
- El contingut pot ser tan llarg com ens faci falta.
- Es pot escriure en qualsevol idioma del món.
- No importa que hi hagi espais en blanc o salts de línia dins del contingut.

Entitats

L'única cosa que s'ha de tenir en compte a l'hora de definir continguts és que hi ha una sèrie de caràcters que poden provocar que hi hagi confusions a l'hora d'interpretar el document XML, i per tant s'han declarat il·legals. No cal oblidar que un dels objectius de l'especificació és "que sigui fàcil escriure programes que processin els documents", i per tant s'han d'evitar les interpretacions.

Si passem el codi següent a un programa d'ordinador tindrem un problema

```
<algebra>
x<y i y>z
</algebra>
```

Per a un humà això no seria un problema perquè pot interpretar les dades per mitjà del seu coneixement, però en canvi un programa en llegir el contingut del document interpretaria que `<y i y>` és el començament d'una nova etiqueta.

Per evitar aquest problema s'han definit una sèrie de valors, anomenats **entitats**, que es fan servir per poder incloure els caràcters problemàtics dins del contingut de les etiquetes (taula 2.1).

Referències de caràcters

En realitat qualsevol caràcter es pot representar com una entitat fent servir el seu valor numèric en Unicode, emprant els símbols `&#` i els tres dígits del seu valor numèric. Per exemple: &#169

Taula 2.1. Entitats definides en XML

Símbol	Substitució
<	<
>	>
”	%quot;
’	'
&	&

Seccions CDATA

Es pot donar el cas que el contingut d'un element sigui HTML o bé codi en algun llenguatge de programació. Això obligaria a substituir per entitats una gran quantitat de símbols i a més restaria llegibilitat al document. Per evitar-ho l'XML permet fer servir les seccions CDATA dins del contingut d'un element. Tot el que estigui dins d'una secció CDATA no serà interpretat per cap programa.

`CDATA` és un terme heretat d'SGML i resumeix el text *character data*.

Les seccions CDATA funcionen com les etiquetes normals. Es defineixen començant per **<![CDATA[** abans d'especificar el contingut i s'acaben amb la combinació de caràcters **]]>**. Per tant, podem definir contingut amb símbols prohibits dins de les seccions CDATA sense problemes.

```
<valor>
    <![[CDATA[
        if (x <y and y>z)
        {
            printf("Correcte!");
        }
    ]]>
</valor>
```

A més es pot veure fàcilment que per a un humà és més difícil interpretar això:

```
<titol>Els documents HTML comencen per &lt;html&gt;</titol>
```

Que el seu equivalent amb CDATA:

```
<titol>
  <![[CDATA[
    Els documents HTML comencen per <HTML>
  ]]>
</titol>
```

Un altre problema que solucionen les seccions CDATA és que permeten afegir contingut en un llenguatge de marques que no cal que estigui ben format. Per exemple, si el contingut té etiquetes que no es tanquen només es poden definir en una secció CDATA o bé el programa les interpretarà com a etiquetes del document i donarà un error.

Les seccions CDATA normalment es fan servir per al següent:

- Definir grans fragments de text que requereixin moltes substitucions d'entitats.
- Si s'ha d'incloure contingut en HTML, Javascript o algun llenguatge similar. La substitució amb entitats li resta llegibilitat.
- Si el contingut està en un llenguatge de marques i no està ben format.

2.2.3 Atributs

Una manera alternativa d'afegir contingut als documents XML és per mitjà dels atributs. Els atributs són un parell de valors separats per un = que **només s'especifiquen en les etiquetes d'obertura**.

El primer valor del parell ens indica quin és el nom del contingut i es farà servir per interpretar què indiquen les dades que té associades, mentre que el segon ens definirà el contingut de l'atribut.

```
<nom carrec="professor">Frederic Garcia</nom>
```

Es pot veure com l'atribut `carrec` dóna un nivell més d'informació a `nom`. Ara se sap que fa referència al nom d'un professor.

Per especificar els atributs es poden fer servir tant les comentes dobles com les cometes simples. Això sí, sempre s'ha de tancar les cometes tal com s'han obert. `<corredor posicio="1">` o `<corredor posicio='1'>` però mai `<corredor posicio="1'>`

A diferència del que passa en altres llenguatges de marques com l'HTML, l'XML és molt més estricte i obliga que **tots els valors dels atributs han d'estar envoltats per cometes**. En altres llenguatges de marques es permet que els valors numèrics no vagin entre comentes però en XML no es pot fer. Per tant l'exemple següent seria incorrecte perquè l'atribut no té cometes.

```
<corredor posicio=1>Manel Roure</corredor>
```

Si es vol especificar s'ha de posar entre cometes simples:

```
<corredor posicio='1'>Manel Roure</corredor>
```

o dobles:

```
<corredor posicio="1">Manel Roure</corredor>
```

Fins i tot quan els atributs no cal que tinguin cap valor, perquè es considera que ja se'n pot interpretar el significat pel nom de l'atribut, s'hi ha d'especificar valor entre cometes. Per tant, l'exemple següent no seria correcte perquè l'atribut no té valor:

```
<alumne delegat>Jaume Ravent</alumne>
```

Ho hauríem d'especificar amb la cadena buida:

```
<alumne delegat="">Jaume Ravent</alumne>
```

O posant-hi algun valor:

```
<alumne delegat="si">Jaume Ravent</alumne>
```

L'XML **permet definir tants atributs com faci falta** sense cap tipus de restricció. Les úniques condicions que hem de seguir és que cada un dels atributs ha d'estar separat dels altres almenys per un espai.

```
<persona nom="Xavier" cognom="Sala" cognom="Pujolar"/>
```

L'ordre en què apareixen els atributs no té cap importància, de manera que els dos codis següents serien equivalents:

```
<persona nom="Xavier" cognom="Sala" />
```

```
<persona cognom="Sala" nom="Xavier" />
```

Quan s'han de fer servir atributs i quan elements?

Hi ha hagut molts debats sobre quan s'han de fer servir atributs i quan s'han de fer servir elements sense que s'hagi trobat cap argument totalment acceptat per tothom.

Al final el resultat pràctic és que és virtualment el mateix fer això:

```
<persona nom="Pere" />
```

Que fer:

```
<persona>
   <nom>Pere</nom>
</persona>
```

I per tant sempre es pot fer servir el mètode que ens agradi més.

Una altra característica important dels atributs és que **no es poden repetir els noms dels atributs dins d'un mateix element**. De manera que no és correcte especificar dos atributs `carrec`:

```
<polític carrec="alcalde" carrec="diputat">Jordi Rufi</polític>
```

Per tant, s'ha de buscar una altra manera d'especificar-ho. Per exemple, amb una llista de valors:

```
<polític carrec="alcalde diputat">Jordi Rufi</polític>
```

o bé posant noms d'atributs diferents:

```
<politic alcalde="si" diputat="si">Jordi Rufi</polític>
```

Atribut xml:lang

L'atribut `xml:lang` és un atribut predefinit i serveix per especificar l'idioma que s'ha fet servir per escriure el contingut tant de l'element com dels altres atributs.

Vegeu les formes definides per l'IETF en el BCP 47 en la secció "Adreces d'interès" del web del mòdul.

El valor de l'atribut `xml:lang` pot ser buit o bé ha d'estar en una de les formes definides per l'IETF en el BCP 47 (www.rfc-editor.org/rfc/bcp/bcp47.txt), que fan servir els codis ISO que defineixen les regions i els idiomes. Sense entrar-hi en profunditat això implica que hi podem definir el codi ISO d'un idioma o bé una combinació de codis ISO que defineixin la regió i l'idioma.

> L'atribut `xml:lang` ens permet definir en quin idioma està el contingut d'un element o d'un document XML.

Per tant, el valor de l'atribut `xml:lang` d'aquest exemple seria vàlid perquè el codi ISO 639 de dues lletres per a la llengua catalana és `ca`.

```
<missatge xml:lang="ca">Hola</missatge>
```

Però també seria vàlid fer servir qualsevol de les combinacions de regió i idioma:

```
<missatge xml:lang="ca-ES">Hola</missatge>
<missatge xml:lang="ca-AD">Salut</missatge>
```

L'atribut només fa referència a l'element en què s'ha especificat i al seu contingut, de manera que aquí:

```
<saludar xml:lang="ca">
  <arribar>Hola</arribar>
  <marxar>Adéu</marxar>
</saludar>
```

es pot considerar que els elements `arribar` i `marxar` tenen també l'atribut `xml:lang` perquè són contingut de `<saludar>`, i per tant el seu contingut i el valor dels seus atributs està en català.

Sempre es pot evitar aquesta forma d'herència de l'atribut redefinint l'atribut `xml:lang` com en aquest exemple:

```
<saludar xml:lang="ca">
  <arribar>Hola</arribar>
  <arribar xml:lang="en">Hello</arribar>
</saludar>
```

S'hi defineixen dos elements `arribar`, un que té el contingut en català, ja que l'ha heretat de l'element `saludar`, que els conté, i l'altre que té el contingut en anglès perquè se l'hi ha especificat l'idioma explícitament.

L'objectiu de l'atribut `xml:lang` és permetre que els programes puguin interpretar l'idioma en el qual hi ha el contingut dels documents o dels elements XML. Hi ha molts programes que fan servir aquesta característica per poder fer que els programes adaptin el contingut que han de mostrar en l'idioma del qui els fa servir.

Si es té un programa que vol mostrar missatges per pantalla en l'idioma que faci servir l'usuari se li podria preparar un fitxer de dades com el següent, que li permetria fer-ho un cop hagués determinat l'idioma del sistema operatiu:

```
<programa>
   <missatge xml:lang="ca">Benvingut</missatge>
   <missatge xml:lang="es">Bienvenido</missatge>
   <missatge xml:lang="fr">Soyez le bienvenu</missatge>
   <missatge xml:lang="en">Welcome</missatge>
</programa>
```

L'atribut "xml:space"

L'atribut `xml:space` fa referència a com s'han de tractar els espais que hi ha en el contingut d'un element determinat.

Quan algú edita un document XML és corrent que faci servir espais i salts de línia perquè el document sigui més "bonic" però realment aquests espais i salts de línia no formen part del contingut. **Amb l'ús de l'atribut `xml:space` es defineix si aquests espais són part del contingut o no**.

Només hi ha dos valors acceptats per a l'atribut `xml:space`: **default** i **preserve** (taula 2.2).

Taula 2.2. Valors acceptats per a l'atribut "xml:space"

Valor	Significat
default	Implica que el document ha de mostrar només un sol espai de separació entre les paraules. Qualsevol altra cosa ha de ser suprimida.
preserve	Fa que els espais es deixin tal com estan en el document. Per tant, si hi ha 5 espais després d'una paraula en el contingut s'han de preservar aquests espais.

Qualsevol altre valor es considerarà un error lleu i no es tindrà en compte.

Seguint el que hem definit, en el document següent:

```
<missatge xml:space="default">
Hola:
    Com va tot?
</missatge>
```

Com que l'element `<missatge>` té de paràmetre `xml:space=''default''`, li estem especificant que se suprimeixin els espais, o sigui, que seria com tenir:

```
<missatge>Hola: Com va tot?</missatge>
```

En canvi, si l'atribut hagués estat `preserve` el valor del contingut de `<missatge>` seria interpretat respectant els espais i els salts de línia.

```
<missatge>
Hola:
    Com va tot?
</missatge>
```

Els processadors XML han de passar tots els espais que no siguin etiquetes al programa que llegeixi el document. Per tant, a pesar del nom per defecte es fa servir `preserve`.

2.2.4 Noms vàlids XML

L'XML permet definir les nostres etiquetes i atributs lliurement però no totes les combinacions possibles són acceptables. Els noms de les etiquetes i dels atributs han de complir unes regles per ser considerats "correctes".

Les regles per definir noms correctes són senzilles:

1. Els noms han de començar per una lletra de l'alfabet, el caràcter de subratllat (_) o un guió (-). També s'accepta el caràcter de dos punts (:), però està reservat.

Tot i que els dos punts estan acceptats no es recomana que es facin servir en la creació d'etiquetes perquè es reserven per ser usats en *espais de noms*.

2. Els caràcters en majúscules són diferents dels caràcters en minúscules.
3. No hi pot haver espais enmig del nom.
4. No poden començar per la paraula *XML* tant si qualsevol de les lletres està en majúscules o en minúscules. Aquestes paraules es reserven per a estandarditzacions futures.

Seguint aquestes regles tindrem que l'exemple següent:

```
<Ciutat comarca="Alt Empordà">Figueres</Ciutat>
```

L'etiqueta `<Ciutat>` és correcta perquè compleix totes les regles. La primera lletra del nom comença per un caràcter de l'alfabet, `C`, no hi ha cap espai enmig del nom, i no comença per les lletres `xml`.

Alhora, l'atribut `comarca` també és correcte perquè comença amb un caràcter de l'alfabet, `c`, no té espais i no comença per `xml`.

En canvi no és correcta l'etiqueta `<Alt Empordà>` perquè té espais enmig del nom:

```
<Alt Empordà></Alt Empordà>
```

Ni ho serien les etiquetes `<1aPosicio>` i `<2aPosicio>`, ja que comencen per un dígit:

```
<carrera>
  <1aPosicio>Manel Garcia</1aPosicio>
  <2aPosicio>Pere Pi</2aPosicio>
</carrera>
```

En la taula 2.3 hi ha exemples d'etiquetes correctes i incorrectes.

Taula 2.3. Exemples d'etiquetes correctes i incorrectes

Etiquetes correctes	Etiquetes incorrectes
<correu1/>	<1Correu/>
<correu-electronic/>	<correu electronic/>
<element />	< element/>
<_Carai/>	<%descompte/>
<svg:rectangle />	<xml-rectangle/>

La versió 1.1 d'XML permet que siguin vàlids tota una sèrie de caràcters extra per adaptar-se a les noves versions de Unicode, però en general no aporten res si no fem servir algun idioma asiàtic.

A pesar d'aquestes restriccions la llibertat que permet l'estàndard és prou important.

Una de les coses que no s'ha d'oblidar mai és que un dels objectius d'XML és fer que els documents amb XML puguin ser llegits i entesos fàcilment per les persones i, per tant, es recomana que no es facin servir noms que no tinguin sentit o que siguin difícilment entesos per una persona.

2.2.5 Comentaris

Sovint en els documents XML s'especifiquen dades extra que realment no formen part del document. Aquestes dades s'anomenen *comentaris* i es fan servir per a moltes coses diverses, com ara:

- Generar documentació sobre el document.
- Indicar a la gent que pugui rebre el document què es volia fer en crear-lo.
- Altres.

Els analitzadors XML són programes que, a partir de l'estructura dels documents XML, ens permeten l'accés al contingut a partir de les seves etiquetes.
El més habitual sol ser processar el document per generar-ne una sortida que pugui ser visualitzada més fàcilment, etc..

Generalment els analitzadors XML simplement solen descartar els comentaris, ja que l'especificació XML diu que no cal que siguin processats.

Els comentaris s'especifiquen encloent-los entre els símbols "<!–" i "–>" i es poden posar comentaris en qualsevol lloc del document XML excepte dins de les etiquetes. Com a exemple, en el document següent s'ha inclòs el comentari "Llista d'alumnes" per a indicar en quin lloc comença la llista d'alumnes:

```
<professors>
   <nom>Marcel Alaba</nom>
</professors>
<!–– Llista d'alumnes ––>
<alumnes>
   <nom>Frederic Garcia</nom>
   <nom>Federicu Pi</nom>
</alumnes>
```

2.2.6 Instruccions de procés

Les instruccions de procés són una manera de donar instruccions als programes que llegiran el document. Es defineixen dins dels símbols "<?" i posteriorment **sempre s'hi ha d'especificar a quin programa van dirigides** amb l'única restricció que no poden ser les lletres "XML" en qualsevol combinació de majúscules o minúscules.

Per tant, per afegir informació que vagi destinada a un programa anomenat `php` s'han de definir les instruccions de procés amb una instrucció de procés:

```
<?php ... ?>
```

Aquest sistema permet que en un document XML es puguin afegir instruccions de procés diferents per a programes diferents:

```
<?programa1 ... ?>
<?programa2 ... ?>
```

El contingut de les dades pot ser qualsevol cosa que tingui sentit per al programa que les llegirà, i per tant no necessàriament ha de tenir sentit per als processadors XML. Com que no formen part del document XML en si poden aparèixer en qualsevol lloc del document excepte en mig d'una etiqueta.

2.3 Declaració XML

L'especificació és una línia que defineix una manera d'identificar que un document és XML per mitjà d'una etiqueta especial anomenada *declaració XML*, que serveix per indicar que un document és XML.

```
1 <?xml version="1.0"?>
```

La declaració XML és opcional, tot i que es considera recomanable, ja que té alguns atributs que poden ajudar als programes a entendre característiques del document com el codi de caràcters que s'hi fa servir, la versió d'XML que s'ha usat, etc.

Si la declaració és present s'ha de tenir en compte que:

- La declaració ha d'estar en la primera línia del document i ha de començar en el primer caràcter del document.
- Ha de tenir obligatòriament l'atribut `version`, que actualment només pot ser `1.0` o `1.1`.

Per tant, la declaració següent és errònia perquè deixa una línia en blanc, deixa espais davant de la declaració i no defineix l'atribut `version`:

```
1 .
2   <?xml ?>
```

Com que la declaració és opcional podem trobar documents XML sense aquesta declaració. Si això passa els programes que llegeixin el document han de suposar que és tracta d'un document XML versió 1.0.

2.3.1 Atributs de la declaració XML

La declaració XML permet definir tres atributs (taula 2.4).

Taula 2.4. Atributs de la declaració XML

Atribut	Objectiu	Obligatori ?
version	Defineix quina versió d'XML s'ha fet servir per crear el document.	Sí

TAULA 2.4 (continuació)

Atribut	Objectiu	Obligatori ?
encoding	Defineix el codi de caràcters que fa servir el document.	No
standalone	Serveix per indicar si el document no depèn d'altres documents.	No

Atribut "version"

L'atribut `version` serveix per especificar mitjançant quina versió d'XML s'ha d'interpretar el document. Tot i que la definició XML és opcional, si hi és aquest atribut esdevé obligatòria.

Per tant, si es carrega el document següent en un programa li estem donant instruccions que ha de tractar el document segons les especificacions de la versió 1.1.

```
1 <?xml version="1.1" ?>
2 <nom>Pere Garcia</nom>
```

Si no s'especifica la definició o bé el valor és incorrecte els programes poden ignorar el que hi hagi i suposar que la versió és la 1.0. Per tant, els dos documents següents han de ser interpretats com a 1.0:

```
1 <?xml version="1.9" ?>
2 <nom>Pere Garcia</nom>
```

```
1 <nom>Pere Garcia</nom>
```

Atribut encoding

L'atribut `encoding` permet definir amb quin codi de caràcters s'han codificat les dades.

Per defecte tots els programes que llegeixin XML han de poder interpretar els codis de caràcters que estiguin en UTF-8 i UTF-16 i fins i tot detectar en quina de les dues codificacions està fet el document. Això implica que si el document està creat amb alguna d'aquestes codificacions no caldria especificar l'atribut `encoding`. Tot i això, per motius de llegibilitat del document sovint s'hi especifica igualment.

Com que l'estàndard ASCII és un subconjunt d'UTF-8 els documents que estiguin en ASCII tampoc no requereixen cap declaració.

En canvi, si el document segueix alguna altra codificació l'atribut s'ha d'especificar explícitament. Per un document creat fent servir el codi de caràcters ISO-8859-15 és obligatori tenir definit l'atribut que indiqui el codi de caràcters que s'ha fet servir:

```
1 <?xml version="1.0" encoding="ISO-8859-15"?>
```

Es poden fer servir les abreviacions de codis de caràcters de la taula 2.5.

Taula 2.5. abreviacions de codis de caràcters acceptades per l'XML

Tipus	Codis de caràcters
Unicode	UTF-8, UTF-16, ISO-10646-UCS-2, i ISO-10646-UCS-4
ASCII i variants	ISO-8859-1, ISO-8859-2, ..., ISO-8859-n
JIS X-0208-1997	ISO-2022-JP, Shitf_JIS, EUC-JP

Vegeu la definició de codificacions de l'IANA en la secció "Adreces d'interès" del web del mòdul.

Per a altres codis de caràcters es recomana fer servir els noms que defineix l'IANA (www.iana.org/assignments/character-sets).

Si a pesar d'això el codi de caràcters que es fa servir no està en la llista s'ha de fer començar el nom amb `x-`.

Atribut "standalone"

Els documents d'esquemes poden fer que algunes de les etiquetes les hagin de tractar de manera diferent els programes. Per exemple, definint atributs per defecte en algunes etiquetes i que, per tant, no cal que siguin específicament declarades en el document XML.

En la definició del vocabulari es pot definir que l'element `<persona>` té un atribut `versió` que per defecte és `home`. En crear el document es podria definir l'etiqueta sense especificar-ne l'atribut perquè el seu valor està implícit.

```
1 <persona>Joan</persona>
```

Si un programa tracta aquest document ha de tenir en compte aquest valor, i per tant haurà de llegir el document que declara els atributs per defecte.

Amb l'atribut `standalone` s'està definint si el document XML es pot entendre per si sol o bé necessita que sigui interpretat amb l'ajuda d'algun altre document, i per aquest motiu només té dos valors possibles (taula 2.6).

Taula 2.6. Valors possibles de l'atribut standalone

Valor	Significat
yes	El document és complet, i per tant no li calen altres documents per ser interpretat.
no	El document no es pot entendre per si sol i n'hem de llegir d'altres per poder-lo entendre.

Si l'atribut no s'especifica es considera que el valor de l'atribut `standalone` és **no**.

2.4 Correctesa

Els programes d'ordinador no tenen la flexibilitat que tenen els documents XML i necessiten poder treballar amb dades molt pautades, i precisament això és el que intenten fer les regles, evitar que hi hagi parts del document que puguin ser malinterpretades. Com que la majoria de les vegades els documents XML hauran de ser llegits per programes, és molt important tenir alguna manera de comprovar que aquest document XML estigui ben format, que compleixi les regles de definició de l'XML.

> Comprovar la correctesa d'un document XML és comprovar que no s'incompleix cap de les regles de definició d'XML.

Es considera que un document és correcte o ben format si compleix amb les regles bàsiques de creació d'un document XML. En general podem definir aquestes regles com:

- Només hi pot haver un element arrel.
- Totes les etiquetes que s'obren s'han de tancar.
- Les etiquetes han d'estar imbricades correctament.
- Els noms de les etiquetes han de ser correctes.
- Els valors dels atributs han d'estar entre cometes.

2.4.1 Només hi pot haver un element arrel

En l'exemple següent:

```
<persona>
   <nom>Pere</nom>
   <cognom>Garcia</cognom>
</persona>
```

L'etiqueta que surt en primer lloc, `<persona>`, i que acaba al final, `</persona>`, defineix el que s'anomena **element arrel**. Que un element sigui arrel vol dir que no té cap element que el contingui.

En aquest altre exemple, en canvi, hi ha dos elements arrel, `<persona>` i `<gos>`:

```
<persona>
   <nom>Pere Garcia</nom>
</persona>
<gos>
   <nom>Rufy</nom>
</gos>
```

Es pot veure que tant `<persona>` com `<gos>` no tenen cap element que els contingui i, per tant, tots dos són elements arrel. **Això fa que no sigui un document XML ben format**.

També hi ha dos elements arrel en el cas en què les etiquetes siguin les mateixes. Per exemple, en el document següent tenim dos elements arrel `<persona>`. Per aquest motiu, aquest document tampoc no està ben format.

```
<persona>
   <nom>Pere</nom>
   <cognom> Pérez </cognom>
</persona>
<persona>
   <nom>Marià</nom>
   <cognom>Garcia</cognom>
</persona>
```

Si es volgués convertir el document anterior en un document ben format es podria posar un element nou que englobés els dos elements arrel. Per exemple, s'hi s'afegeix un element `<persones>` i així es crea un document XML ben format:

```
<persones>
    <persona>
       <nom>Pere</nom>
       <cognom> Pérez </cognom>
    </persona>
    <persona>
        <nom>Marià</nom>
        <cognom>Garcia</cognom>
    </persona>
</persones>
```

2.4.2 Totes les etiquetes que s'obren s'han de tancar

A pesar del que passa en altres llenguatges de marques, per exemple HTML, en l'XML totes les etiquetes que s'obrin han de ser tancades obligatòriament. Per tant, aquest seria un exemple correcte:

```
<nom>Pere</nom>
```

Mentre que aquest no seria correcte:

```
<nom>Pere
```

En alguns moments pot caldre reflectir alguna dada que no requereixi cap valor. Per exemple si tenim una llista d'alumnes ens podria interessar marcar quin d'ells és el delegat. Per definir-ho no ens caldria cap dada, ja que simplement l'alumne que tingui l'etiqueta `<delegat>` és el delegat. Per tant, per definir que en Frederic Pi és el delegat podríem estar temptats de fer això:

```
<alumnes>
    <alumne>
        <nom>Pere Garcia</nom>
```

```
    </alumne>
    <alumne>
        <nom>Frederic Pi</nom>
        <delegat>
    </alumne>
    <alumne>
        <nom>Maria Puigdevall</nom>
    </alumne>
</alumnes>
```

Això és incorrecte. Si obrim una etiqueta l'hem de tancar; per tant, hauríem de fer:

```
...
<alumne>
   <nom>Frederic Pi</nom>
   <delegat></delegat>
</alumne>
...
```

O bé fer servir la versió de definició d'etiquetes buides acabant l'etiqueta amb "/>":

```
...
<alumne>
    <nom>Frederic Pi</nom>
    <delegat/>
</alumne>
...
```

2.4.3 Les etiquetes han d'estar imbricades correctament

Per mantenir la jerarquia, l'XML defineix que les etiquetes no es poden tancar si encara hi ha alguna etiqueta que forma part del contingut que no ha estat tancada. Això és perquè no es permet que un element tingui una part del seu contingut en un element i una altra part en un altre, ja que això trencaria la jerarquia de dades.

Tancament de les etiquetes desordenat

Alguns llenguatges de marques sí que permeten obrir i tancar les etiquetes en qualsevol ordre. Per exemple, les versions anteriors a la 4.0 d'HTML permetien obrir i tancar etiquetes en l'ordre que volguéssim sempre que s'acabessin tancant totes. Però a partir de la versió 4.0 ja no s'accepta, a pesar que la majoria dels navegadors sí que ho permeten.

Per tant, sempre que s'obri una etiqueta dins d'un element aquesta ha de ser tancada abans d'acabar l'element. Aquest exemple seria correcte perquè l'element `<persona>` en el seu contingut obre l'etiqueta `<nom>` però abans d'acabar tanca `</nom>`:

```
<persona><nom>Pere</nom></persona>
```

En canvi, si les etiquetes no es tanquen en l'ordre correcte, encara que es compleixi la regla que s'han de tancar totes les etiquetes, no tenim un document XML ben format.

```
<persona><nom>Pere</persona></nom>
```

En l'exemple que acabem de veure s'està creant un bucle, ja que `<persona>` conté una part de `<nom>` i alhora `<nom>` conté una part de `<persona>`. Això trenca amb la idea de jerarquia de les dades que defineix l'XML i, per tant, no és correcte.

Per facilitar l'escriptura de documents XML i minimitzar els possibles errors per culpa de tancaments desordenats, els documents XML **sovint es representen sagnats**.

Sagnia

La sagnia consisteix a escriure les etiquetes de manera sagnada en funció del seu nivell dins de la jerarquia d'elements. A mesura que es van creant nous elements dins d'altres es van afegint sagnies.

En l'XML la sagnia es fa en funció del nivell en què es troben les dades. En general cada un dels fills fa que s'incrementi la sagnia. Per exemple:

```
<alumne>
    <persona>
        <nom>
        </nom>
    </persona>
</alumne>
```

Fent servir la sagnia és més fàcil determinar en quin lloc s'ha produït un error de tancament d'etiquetes i arreglar-ho. Com podem veure en l'exemple següent, si les etiquetes de tancament no estan en la mateixa columna que les d'obertura és que hi ha alguna cosa malament.

```
<institut>
    <classe>
        <alumne>Pere</alumne>
        <alumne>Joan</alumne>
    </institut>
</classe>
```

En general la sagnia aporta dos avantatges a l'XML:

1. Evita que es cometin errors a l'hora de tancar les etiquetes.
2. Fa que sigui més fàcil veure d'un cop d'ull l'estructura del document.

2.4.4 Els noms de les etiquetes i dels atributs han de ser correctes

Vegeu l'apartat "Noms vàlids XML" en aquest apartat.

L'XML dóna molta llibertat a l'hora de definir els noms dels elements i dels atributs però no tots els noms són acceptats.

En general tindrem que:

- No es poden posar noms que no comencin per un caràcter ni que continguin espais dins seu.
- Les majúscules i les minúscules són caràcters diferents per a XML.
- Els noms que comencin per `xml` estan reservats.

Per tant, seria incorrecte escriure una etiqueta en minúscules i després tancar-la en majúscules:

```
<persona>Pere</Persona>
```

Els noms han d'estar escrits de la mateixa manera:

```
<persona>Pere</persona>
```

No es poden posar etiquetes ni atributs que comencin per `xml` perquè estan reservats per l'estàndard.

En l'exemple següent hi ha un error en el nom de l'atribut `xmlpersona`:

```
<persona xmlpersona="si">Pere</persona>
```

De fet, ja hi ha alguns atributs començats per `xml` que s'accepten perquè estan definits en l'estàndard i tenen un objectiu clar: `xml:lang`, `xml:space`, `xmlns`...

2.4.5 Els valors dels atributs han d'estar entre cometes

En l'apartat de definició dels atributs ja es comenta que els valors dels atributs sempre han d'anar entre cometes però podem resumir el més important.

Vegeu l'apartat "Atributs" en aquest apartat.

No importa quines siguin les cometes que fem servir, dobles o simples, sempre que es facin servir les mateixes en obrir que en tancar.

```
<classe nom="Llenguatges de marques">
    <alumne delegat='si'>Pere Garcia</alumne>
    <alumne>Frederic Pi</alumne>
</classe>
```

Encara que el contingut dels atributs sigui un valor numèric s'ha d'especificar entre cometes:

```
<alumne nota="10">Maria Puigdevall</nom>
```

Podem especificar tants atributs com calgui en una etiqueta i l'ordre en què s'especifiquen no té importància:

```
<alumne nom="Pere" cognom="Garcia" nota="6" />
```

2.4.6 Processadors XML

L'objectiu principal de totes les regles que defineixen com s'ha d'escriure un document XML és definir una manera d'escriure documents XML que puguin ser llegits i interpretats per un programa d'ordinador. Els programes encarregats de detectar si un document XML està ben format s'anomenen genèricament **processadors d'XML**, tot i que sovint es fa servir el nom en anglès, ***parsers***.

Un processador d'XML és un programa que permet llegir un document XML i accedir-ne a l'estructura i a les dades que conté. Això ho fa llegint tot el document

i determinant quins dels caràcters que hi troba són part de l'estructura i quins són realment dades.

Vocabularis

XML permet definir etiquetes sense cap restricció però en canvi els programes només poden funcionar amb les etiquetes per les que han estat programats.

Per aquest motiu els programes defineixen quins conjunts d'etiquetes accepten. La llista d'etiquetes que s'accepten en un determinat llenguatge s'anomenen **vocabularis**.

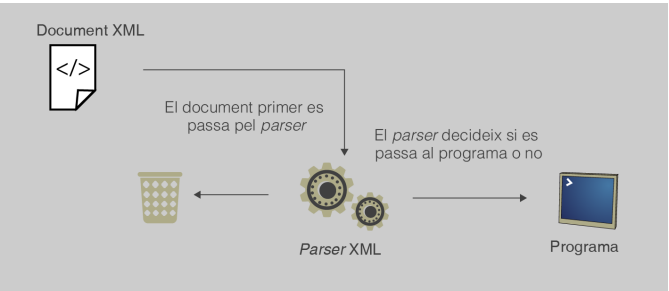
Generalment es classifiquen els processadors en dos grans grups:

- **Processadors validadors**: comproven que els documents compleixen les regles i restriccions definides en l'esquema del vocabulari que se li ha associat i, per tant, a part de la lectura del document XML també han de llegir l'esquema contra el qual es valida.
- **Processadors no validadors**: revisen que el document no incompleix cap de les normes i regles de definició d'XML.

Sempre que no es tingui cap tipus de definició de vocabulari associada a un document XML es pot comprovar que el document és correcte amb un processador que no validi. En canvi, si tenim el vocabulari ens caldrà un processador validador que comprovarà que realment estem posant les etiquetes allà on poden anar.

Les tasques dels processadors XML es fan sempre **abans** que el programa que ha de treballar realment amb les dades faci res. Per tant, el procediment sempre serà: primer passar el document pel processador, i si el processador el dóna per correcte llavors es passa al programa (figura 2.1).

Figura 2.1. Pas d'un document XML a un programa



Errors

A part de definir com s'han de crear els fitxers XML l'especificació també defineix com han de reaccionar els processadors davant dels diferents errors que puguin detectar en tractar un fitxer XML. Els errors es defineixen en dos grups:

- **Errors lleus**: els errors lleus són els que incompleixen alguna de les regles definides com a *recomanacions*. Si algun document XML incompleix alguna recomanació l'analitzador pot continuar analitzant el document intentant recuperar-se de l'error.

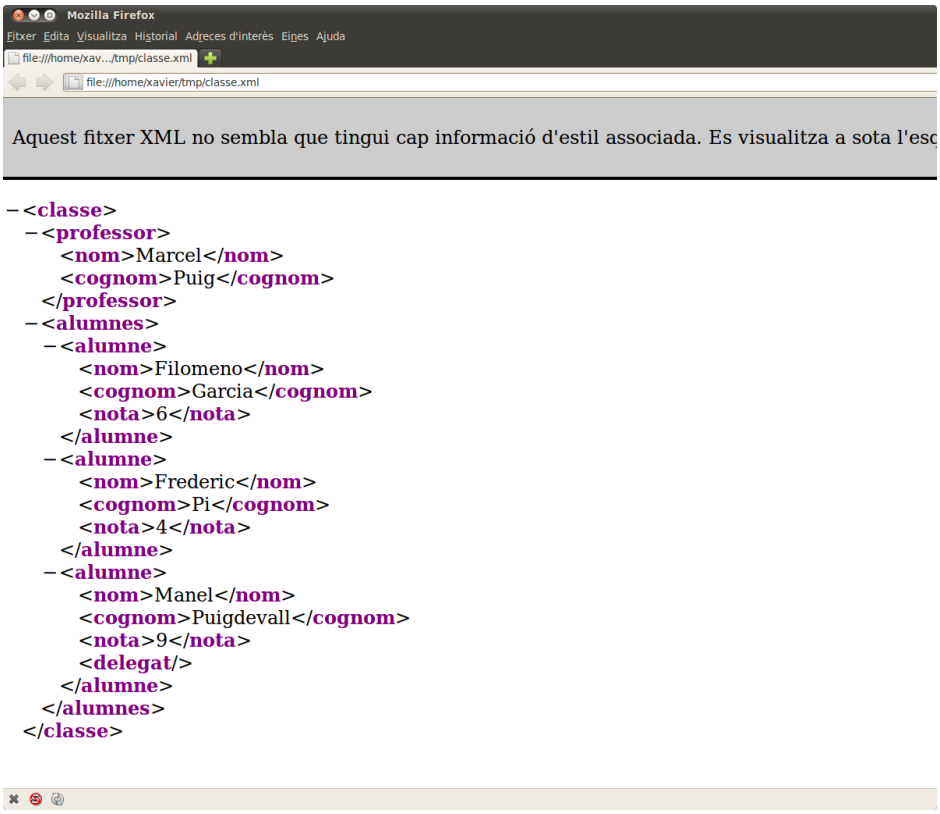
- **Errors greus**: es generen quan s'incompleix alguna de les regles *obligatòries* de l'especificació. Es defineix que en cas de trobar algun error greu l'analitzador s'ha d'aturar immediatament i no continuar amb el procés d'anàlisi. En general qualsevol error que faci que el document no sigui "ben format" serà un error d'aquest tipus.

Navegadors web

Un cop s'ha creat un document XML normalment caldrà comprovar que aquest document està **ben format** segons les regles d'XML fent servir un processador XML.

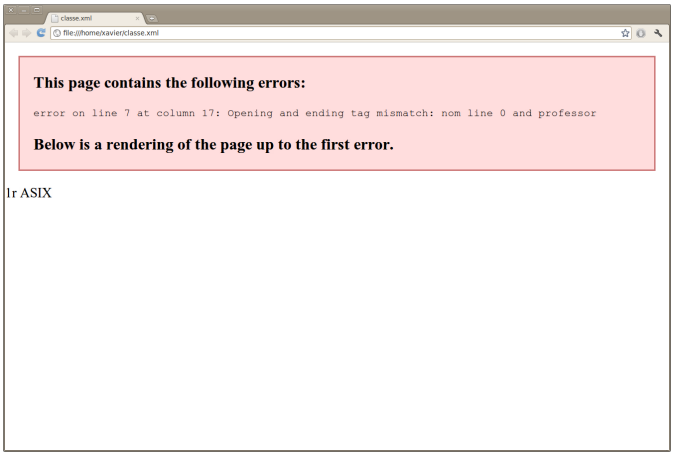
La manera més senzilla de comprovar que un document està ben format és carregar-lo des d'un navegador. La majoria dels navegadors porten un processador XML incorporat i detecten automàticament que un document és XML i el comproven (figura 2.2).

Figura 2.2. Document correcte des d'un navegador



En cas de produir-se un error els navegadors mostren quin error s'hi ha produït i en quina línia es troba l'error (figura 2.3).

Figura 2.3. En carregar un document XML incorrecte en el navegador Google Chrome ens mostra els errors.



Processadors XML

En comptes de fer servir un navegador web per comprovar els documents XML sovint es fa servir un processador. La majoria dels processadors formen part de biblioteques que els programadors poden usar des dels seus programes per carregar els documents XML i comprovar-los.

A vegades els processadors porten utilitats que permetran comprovar ràpidament si un document està ben format o no sense haver de desenvolupar un programa per fer-ho.

Volem destacar els processadors següents:

- libxml2
- Apache Xerces
- Expat
- MSXML

libxml2

libxml2 forma part del projecte Gnome i és un processador de codi obert que està escrit en llenguatge C, però que es pot fer servir des de diferents llenguatges de programació com Perl, Ruby, Python, C#, PHP... i des de diferents sistemes operatius com Windows, Linux o Mac OS X. Està instal·lat per defecte en moltes de les distribucions de Linux.

Entre les utilitats que inclou n'hi ha una que funciona des de l'entorn d'ordres, anomenada **xmllint**, que és un analitzador i validador de documents XML.

Es pot comprovar si un document està ben format simplement especificant-lo com a paràmetre. Per validar el document següent:

```
1 <?xml version="1.0"?>
2 <persona>
3   <nom>Pere</nom>
4 </persona>
```

Executarem l'ordre:

```
$ xmllint persona.xml
```

Si el document és correcte mostrarà el document XML per pantalla (es pot evitar aquest comportament amb el paràmetre `-noout`):

```
$ xmllint persona.xml
<?xml version="1.0"?>
<persona>
  <nom>Pere</nom>
</persona>
```

En el cas que hi hagi algun error es mostrarà per pantalla i donarà informació sobre en quin lloc es troba i quin és l'error. Si modifiquem el document XML perquè tingui un error:

```
<persona>
    <nom>Pere
</persona>
```

Quan el passem a `xmllint`, aquest ens informarà que hi ha un error a la línia 3 perquè ha detectat que s'ha tancat `<persona>` però `<nom>` no està tancat.

```
$ xmllint persona.xml
persona.xml:3: parser error : Opening and ending tag mismatch: nom line 2 and
    persona
</persona>
          ^
persona.xml:4: parser error : Premature end of data in tag persona line 1
^
```

També es pot fer servir `xmllint` per veure que els processadors intenten arreglar els errors lleus que hi pugui haver en un document. Si eliminem la declaració XML del document (no estem provocant cap error, ja que la declaració només és una recomanació):

```
<persona>
    <nom>Pere</nom>
</persona>
```

En passar-lo per xmllint es pot veure que ha detectat que la declaració no hi era i l'ha afegida automàticament:

```
$ xmllint  persona.xml
<?xml version="1.0"?>
<persona>
    <nom>Pere</nom>
</persona>
```

Apache Xerces

Sota el nom de Xerces la fundació Apache manté un projecte de desenvolupament de processadors XML per a Java, C++ i Perl. Es tracta d'una de les biblioteques més usades per processar XML.

Expat

XML Parser Toolkit són unes biblioteques per fer servir XML des del llenguatge de programació C i Perl.

El fan servir alguns dels programes d'ús comú, com per exemple el servidor web Apache HTTP Server, el navegador Mozilla i les biblioteques dels llenguatges de programació Perl, Python i PHP.

Per exemple, en l'Ubuntu, en instal·lar la biblioteca, també s'instal·la un programa anomenat *xmlwf* que permet comprovar si un document està ben format des de la línia d'ordres. Si no diu res el fitxer està ben format.

```
$ xmlwf persona.xml
```

En cas d'error mostrarà en quina línia i columna es troba:

```
$ xmlwf persona.xml
persona.xml:3:2: mismatched tag
```

MSXML

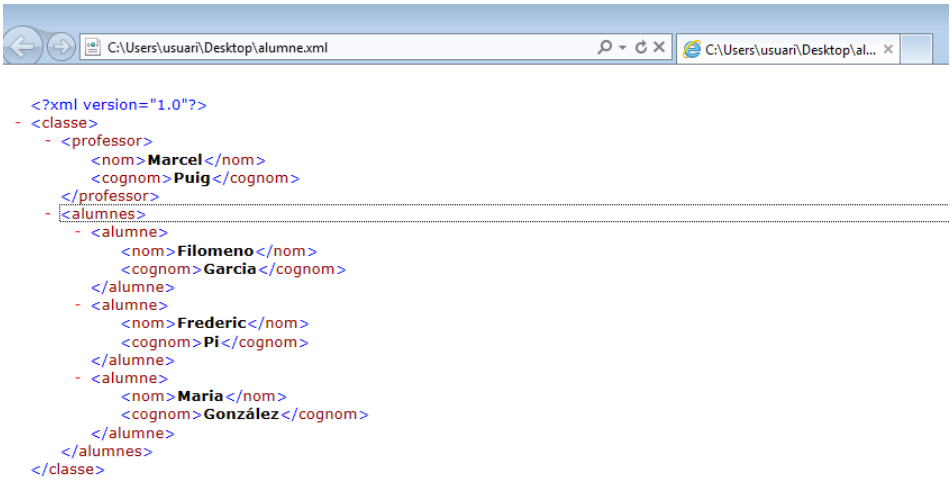
Microsoft XML Parser és la biblioteca oficial per processar XML en molts dels productes de Microsoft. Es tracta d'una biblioteca inclosa en el sistema operatiu, i per tant s'actualitza via Windows Update. N'han anat sortint versions en què s'han anat afegint funcions. Es pot fer servir des de diferents llenguatges de programació, llenguatges de *scripts* o des de llenguatges basats en .NET

Fan servir MSXML productes com Microsoft Windows, Microsoft Office, Microsoft SQL Server, Microsoft Internet Explorer...

Molts dels programes que processen XML en sistemes Windows solen fer servir aquesta biblioteca (especialment si treballen en .NET).

Com que Internet Explorer fa servir aquesta biblioteca, simplement carregant un document XML amb Internet Explorer en realitat s'està comprovant amb l'MSXML (figura 2.4).

Figura 2.4. L'Internet Explorer ens permet comprovar documents amb l'MSXML



2.5 Estructura dels documents XML

Una de les coses que s'aconsegueixen en forçar que els documents XML segueixin les seves normes és fer que la informació que conté un document s'organitzi de manera jeràrquica.

Per exemple, tenim el document XML següent:

```
<?xml versión="1.0" encoding="UTF-8" ?>
<classe>
   <professor>
       <nom>Marcel</nom>
       <cognom>Puig</cognom>
   </professor>
   <alumnes>
       <alumne>
           <nom>Filomeno</nom>
           <cognom>Garcia</cognom>
       </alumne>
       <alumne>
           <nom>Frederic</nom>
           <cognom>Pi</cognom>
       </alumne>
       <alumne>
           <nom>Manel</nom>
           <cognom>Puigdevall</cognom>
           <delegat/>
       </alumne>
   </alumnes>
</classe>
```

Gràcies al fet que les etiquetes tenen noms clars es pot deduir que en aquest document XML s'estan definint estructuradament una classe amb un professor i tres alumnes. Hi ha un element arrel anomenat `<classe>` que engloba tot el document, i com a contingut té dos elements més, `<professor>` i `<alumnes>`.

> Els elements que formen part del contingut d'un node s'anomenen fills.

Per tant, `<classe>` té dos elements **fills**: `<professor>` i `<alumnes>`. Alhora l'element `<professor>`, que és un dels fills de `<classe>`, també té dos fills: `<nom>` i `<cognom>`.

```
...
    <professor>
        <nom>
            Marcel
        </nom>
        <cognom>
            Puig
        </cognom>
    </professor>
...
```

Per equivalència amb les relacions humanes els fills dels fills d'un element s'anomenen **néts**. Per tant, `<nom>` i `<cognom>` són fills de `<professor>` i **néts** de `classe`:

classe -> professor -> nom Els elements <nom> i <cognom> no tenen nodes fills sinó que contenen les dades del document. En aquest cas dues cadenes de text `Marcel` i `Puig`. Els nodes finals s'anomenen **fulles** i generalment seran sempre nodes de dades.

Es pot fer amb l'altre element fill de <classe>, però amb la diferència que té un nivell més. Ara es té una estructura més llarga:

classe -> alumnes -> alumne -> nom Tots els elements que pengen d'un element s'anomenen genèricament **descendents**. Per tant, <alumnes>, <alumne> i <nom> són descendents de <classe>.

Com podeu veure, és relativament senzill interpretar quines són les dades que conté un document XML, gràcies al fet que es fan servir noms d'etiqueta que tenen sentit per a un possible lector, però també es pot deduir quina és l'estructura de les dades i les relacions que tindran aquestes dades entre si.

El fet de seguir aquest sistema, que és flexible en determinats moments, però estricte en d'altres, ens permet fer que un programa d'ordinador també pugui de manera senzilla detectar ràpidament quina part del document són dades i quina són metadades, però que a més pugui establir quina és la relació entre aquestes dades.

En cap moment no s'ha especificat res sobre de quina manera s'hauran de representar les dades, ja que aquesta és una de les bases fonamentals del llenguatge XML:

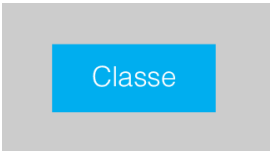
> L'objectiu de l'XML és representar l'estructura de les dades oblidant-se de com es farà per presentar-les.

2.5.1 Representació en forma d'arbre

Una de les característiques interessants dels documents XML és que, en tenir les dades estructurades de manera jeràrquica, aquesta jerarquia fa que els documents XML puguin ser representats gràficament en forma d'arbre. La representació en arbre és útil per comprendre millor quina és l'estructura del document.

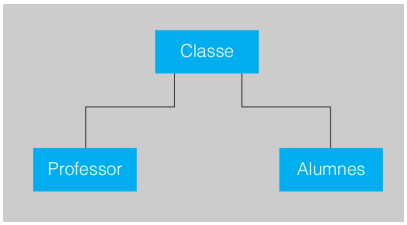
Per fer la representació en forma d'arbre sempre es parteix del node arrel, que en aquest cas és `classe` (figura 2.5).

Figura 2.5. Es comença definint l'element arrel com un node



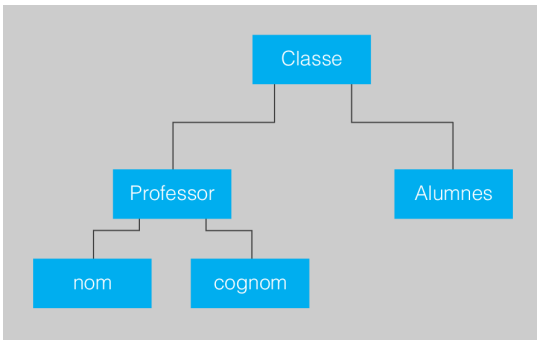
A partir del node arrel s'agafen els fills directes que tingui i s'enllacen amb un arc, que serà el que indicarà la relació que tenen els elements entre ells (figura 2.6).

Figura 2.6. S'hi afegeixen els fills



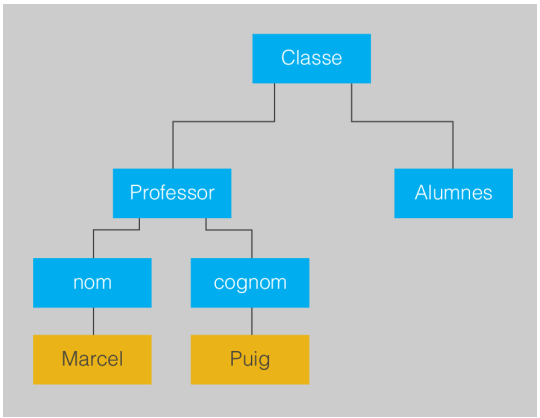
El mateix es pot fer per als fills de cada un dels nodes. Per exemple, `professor` té dos fills més, que són `nom` i `cognom`, que s'uniran a `professor` amb un arc per indicar que tenen relació pare/fill (figura 2.7). De fet, es fa és més o menys el mateix que es faria per crear un arbre genealògic.

Figura 2.7. I es va fent el mateix amb els fills de cada node



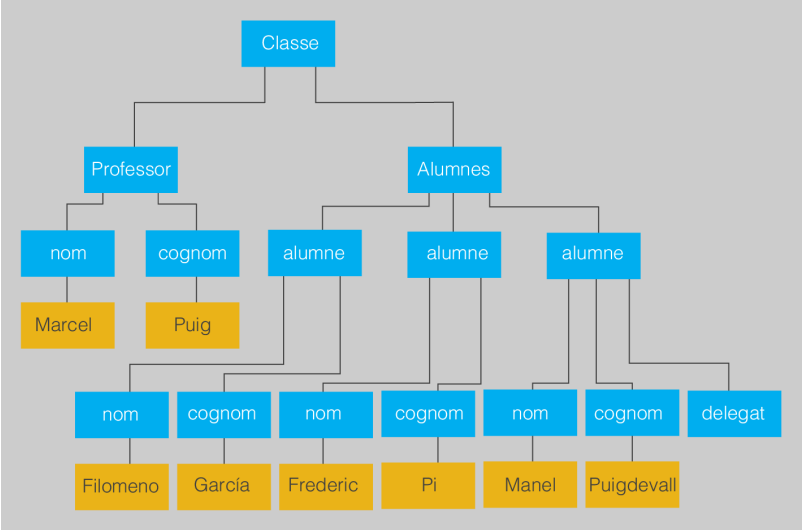
Com que ja no hi ha més elements per representar ja es poden pintar els nodes de dades, que generalment s'acostumen a pintar de color diferent (figura 2.8).

Figura 2.8. Els nodes de dades se solen representar d'una manera especial



Si se segueix amb el document el resultat serà un arbre que ens representarà gràficament la relació que tenen els elements entre si (figura 2.9).

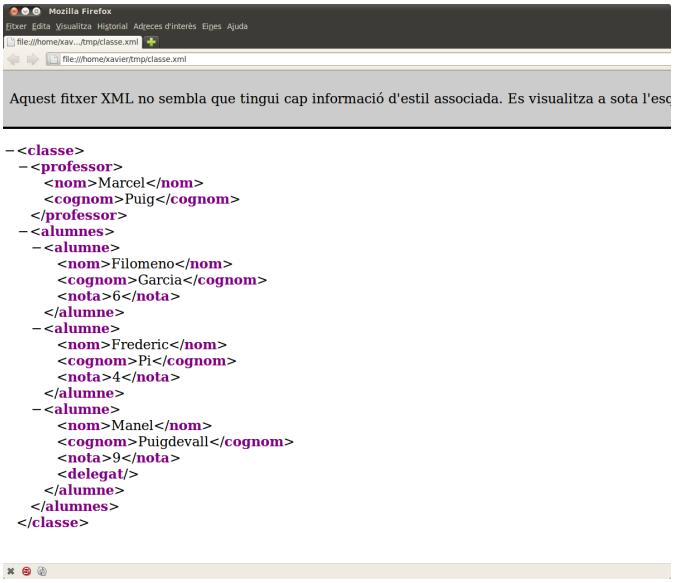
Figura 2.9. El resultat és la representació del document en forma d'arbre



2.5.2 Representació en els navegadors

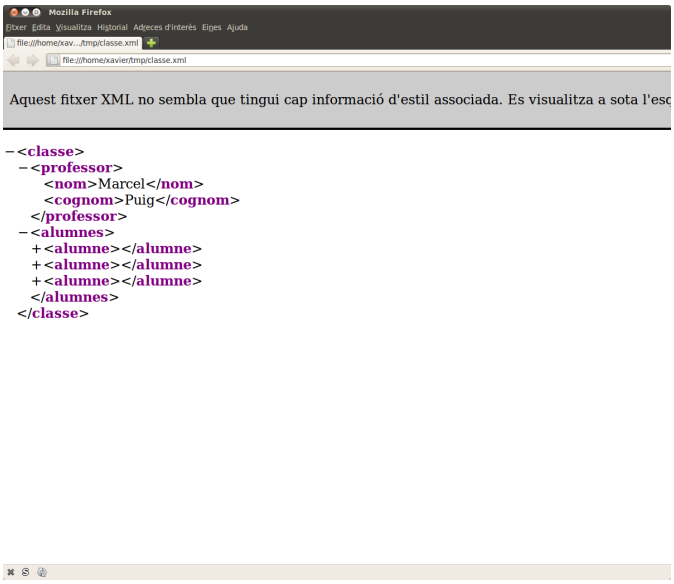
Una representació semblant a la de l'arbre és la que han triat els navegadors per mostrar els documents XML si no els diem el contrari afegint-hi algun full d'estil. Quan s'obre un document XML en un navegador automàticament és representa sagnat i s'hi afegeixen nodes davant dels elements pares que permetran mostrar o ocultar els fills que conté (figura 2.10).

Figura 2.10. Representació d'un document XML correcte en el navegador Mozilla Firefox



En la vista d'arbre que ofereixen els navegadors normalment es poden plegar i desplegar els fills dels nodes per poder personalitzar la visió del document (figura 2.11).

FIGURA 2.11. El Firefox permet plegar i desplegar els fills dels nodes



2.6 Creació de documents XML

Generalment els documents XML seran creats i llegits des de programes d'ordinador, però algunes vegades també es pot donar el cas que s'hagin de crear manualment.

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.

2.6.1 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.

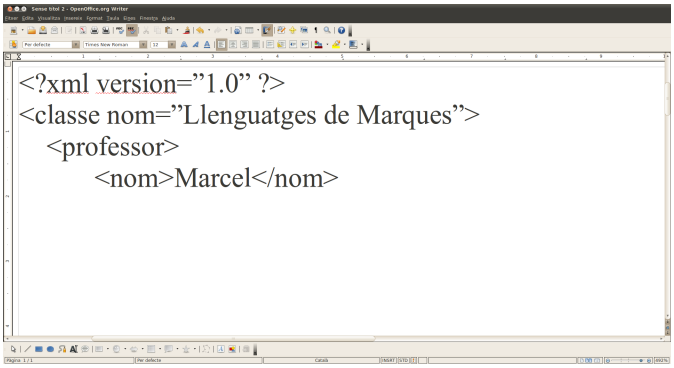
A pesar d'això també han aparegut tota una sèrie d'editors pensats per fer l'edició de documents XML més senzilla. Aquests editors són molt diversos i normalment ofereixen diferents tipus d'assistència per evitar que es cometin errors en crear el document: comprovar interactivament que el document sigui correcte, aconsellar etiquetes, acolorir...

Molts dels editors especialitzats en XML normalment a més ofereixen moltes altres funcions com generació d'expressions XPath, creació de fulls d'estil, depurament de transformacions, peticions XQuery...

Editors de text

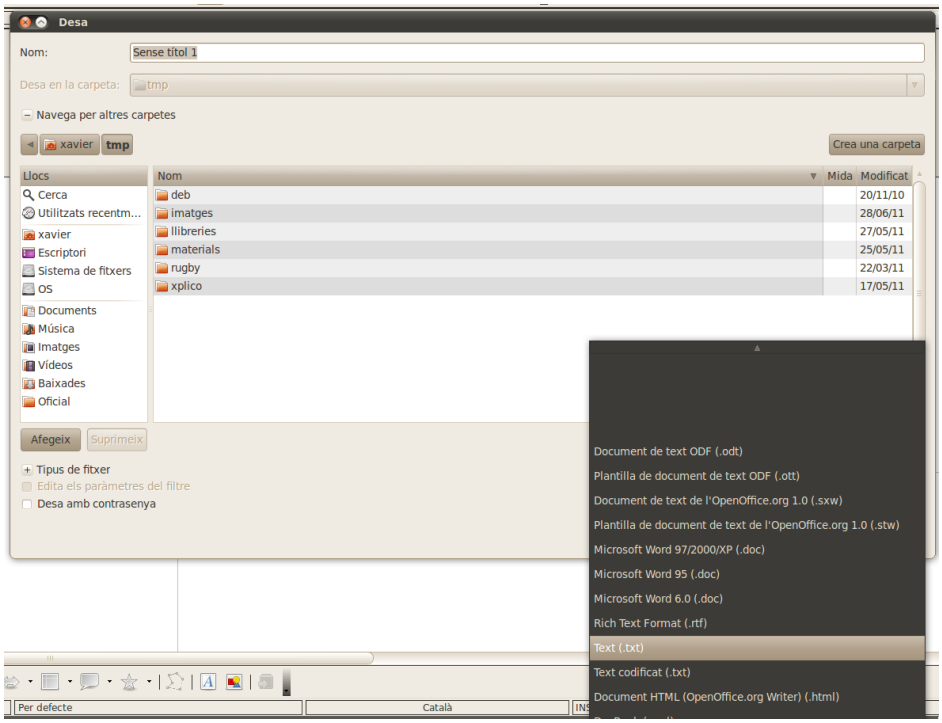
L'única cosa que cal per crear un document XML és un editor de text normal i corrent que no enriqueixi el text. Els documents de text que permeten afegir format, com ara text en negreta, canviar el tipus de lletra, etc., com per exemple el Microsoft Word, l'OpenOffice.org Writer, el LibreOffice Writer, etc., solen generar documents de text enriquit que es concentren més en com s'ha de mostrar el document que no pas en les dades. A més, aquests programes sovint canvien automàticament alguns caràcters del text que anem escrivint (les cometes solen ser canviades sistemàticament) per fer-lo més "agradable" a l'hora d'imprimir-lo, com es pot veure a la figura 2.12.

Figura 2.12. L'OpenOffice.org modifica les cometes per fer-les més "agradables" a la vista



Per tant, aquests programes no són els més adequats per crear documents XML, tot i que és possible fer-ho si a l'hora de desar els documents es va amb compte d'especificar que es volen desar com a "text" (figura 2.13).

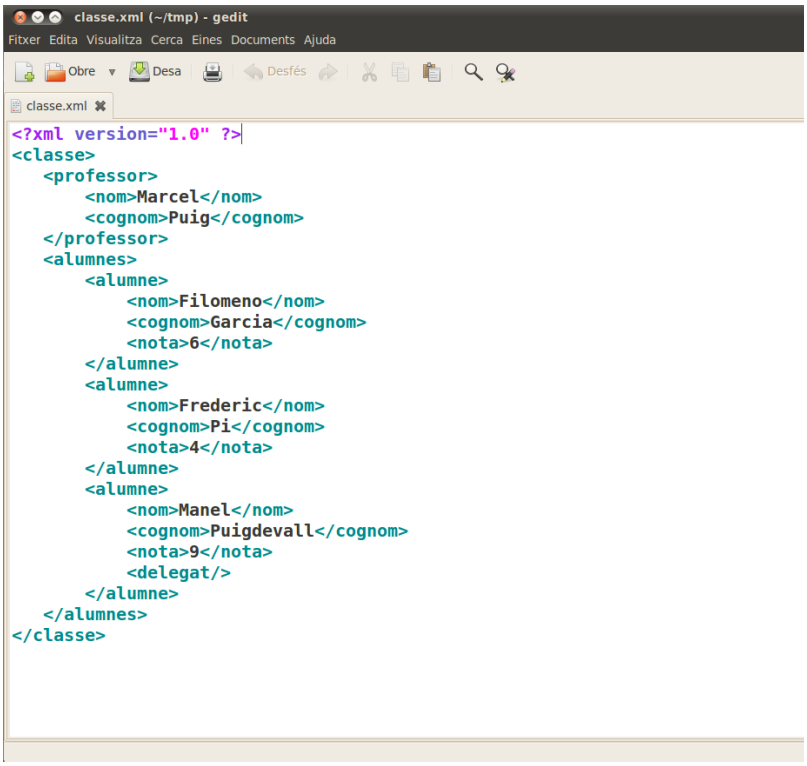
Figura 2.13. Es pot fer servir l'OpenOffice.org però anant en compte a l'hora de desar el document



La realitat és que per crear documents XML van molt millor els editors més senzills (Gedit, Bloc de notes, *vi*, etc.) que no pas els editors de text enriquit.

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 (figura 2.14).

Figura 2.14. L'editor Gedit de Gnome fins i tot ofereix acoloriment de text en els fitxers XML



Editors amb suport d'XML

També hi ha un segon grup d'editors que, tot i no estar especialitzats en XML, hi poden tenir suport. 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 autocompletament d'etiquetes.

Aquest comportament és típic dels editors pensats per ser usats pels programadors, com per exemple el Komodo IDE (figura 2.15), l'Eclipse o el Visual Studio.

L'èxit d'XML ha fet que alguns editors hagin incrementat el seu suport per a XML. Per exemple, les noves versions del Microsoft Visual Studio a més de l'autocompletament d'etiquetes permeten definir esquemes i depurar transformacions des de l'entorn integrat (figura 2.16).

Figura 2.15. Molts dels editors dels entorns de programació ofereixen suport als documents XML. Per exemple, l'editor de codi obert Komodo

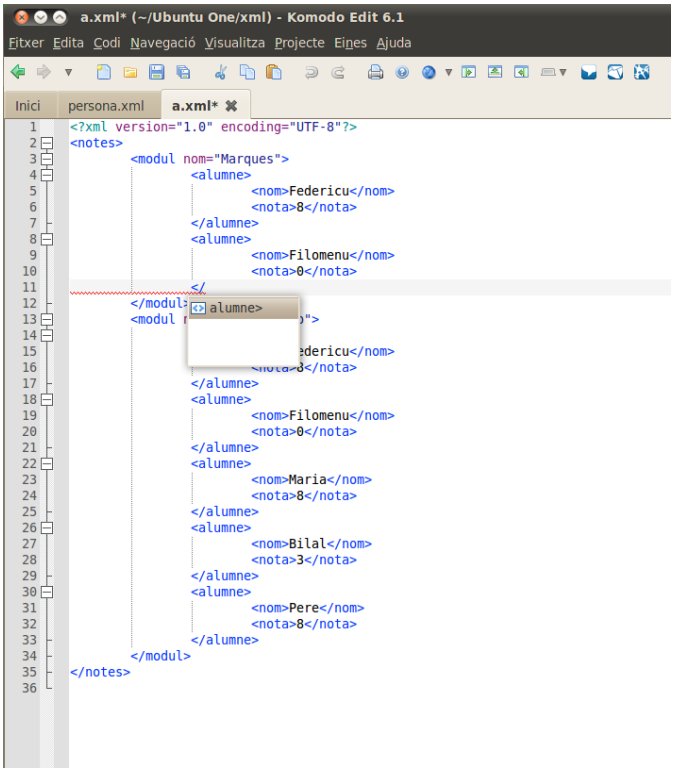
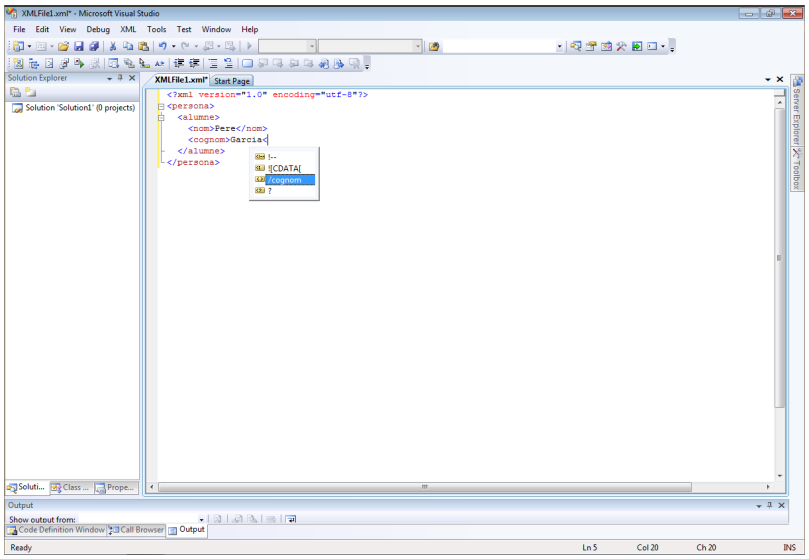


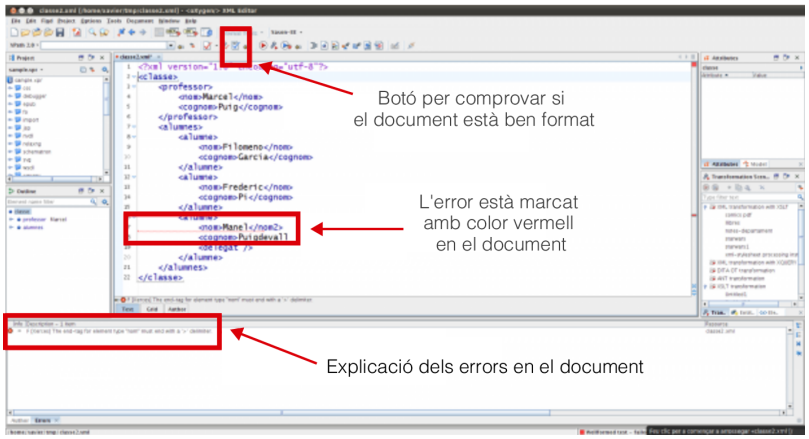
Figura 2.16. Microsoft ha afegit al Visual Studio suport per a diverses tecnologies XML



Editors especialitzats en XML

Tot i que amb els altres editors es poden crear documents XML, quan es vol fer un treball professional normalment s'ha d'acabar recorrent a un editor d'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 (figura 2.17).

Figura 2.17. L'editor EditiX ens va creant l'arbre XML alhora que anem escrivint el document XML

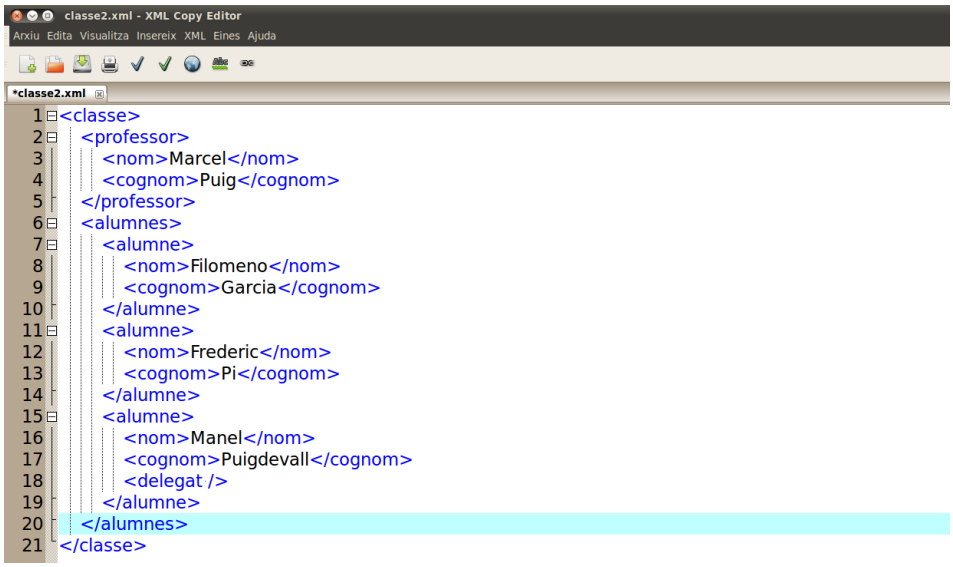


A part de la simple edició de documents XML, aquests editors permeten tot un ampli ventall de tasques amb les tecnologies relacionades amb l'XML com editar documents XML restringint les etiquetes que s'hi fan servir; definir un esquema; crear, convertir i depurar esquemes XML, XSLT, XPath, XQuery, WSDL,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

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.

L'edició en la **vista de text** (figura 2.18) no sol diferir gaire de l'edició en un editor normal i corrent però sol oferir alguns avantatges afegits com autocompletament d'etiquetes, acoloriment del contingut, finestres d'ajuda que mostren l'estructura del document, etc.

Figura 2.18. La vista de text és la ideal per editar els arxius veient clarament el format XML



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 tant

sols sàpiga que està creant un document XML. Una d'aquestes vistes alternatives és la **vista d'arbre** (figura 2.17). 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 (figura 2.19 i figura 2.20).

Figura 2.19. L'usuari va creant les dades de manera estructurada i l'editor crea el document XML per ell

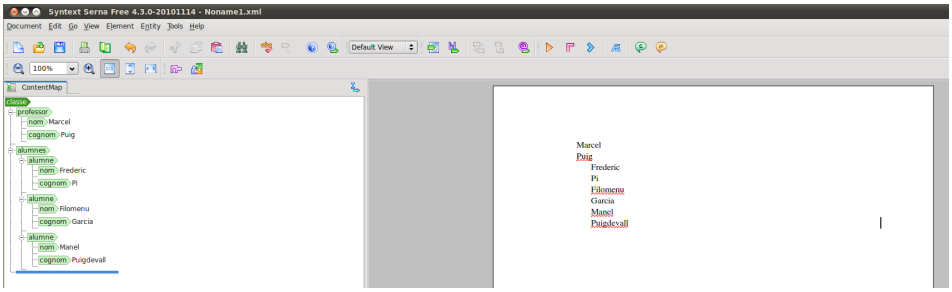
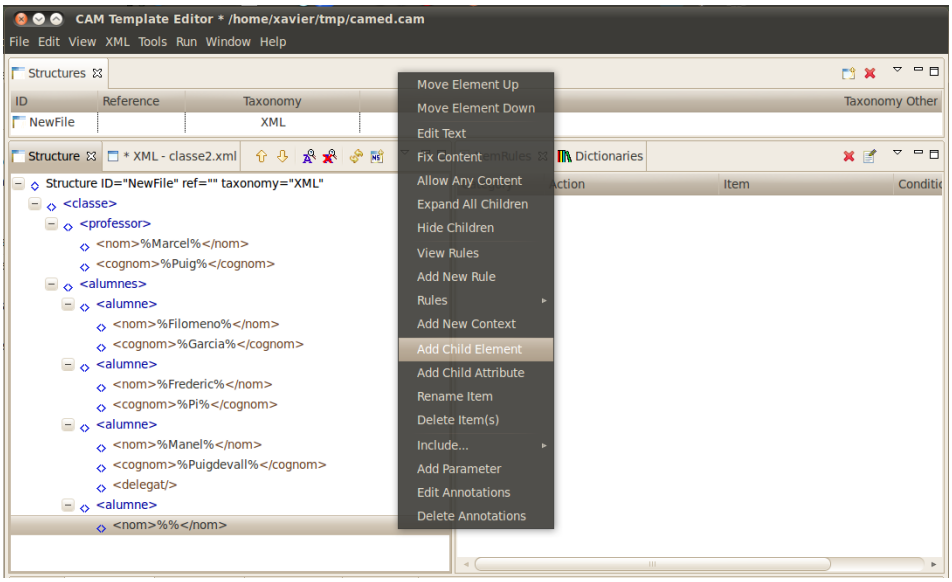
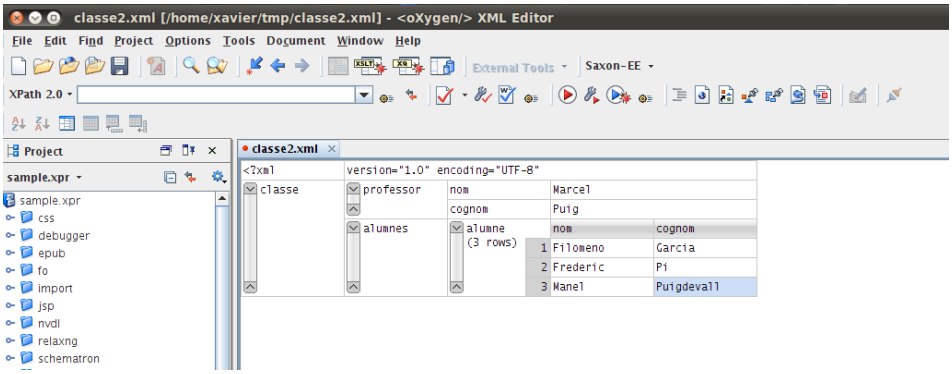


Figura 2.20. La vista d'arbre està pensada per als usuaris que no coneixen el llenguatge XML



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** (figura 2.21).

Figura 2.21. Vista de graella



La vista de graella està pensada per a usuaris que només es volen preocupar de l'estructura del document i no de com crear-lo.

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. Afortunadament la majoria dels editors comercials ofereixen un temps de prova abans d'obligar a comprar el programa, i per tant es poden provar diferents editors abans de decidir quin és el que s'adapta millor a les necessitats que tenim. A més, alguns editors tenen versions limitades gratuïtes o de codi obert que en molts casos poden ser suficients per satisfer les necessitats que es puguin tenir.

Entre els editors comercials normalment es destaquen aquests:

- **oXygen XML Editor** (Windows, Linux, Mac OS X)
- **Editix XML Editor** (Windows, Linux, Mac OS X)
- **Altova XMLSpy XML editor** (Windows)
- **Stylus Studio** (Windows)
- **XMLmind** (Windows, Linux, Mac OS X)
- **XMLwriter** (Windows)
- **Liquid XML Studio** (Windows)
- **Serna Enterprise XML Document Editor** (Windows, Linux, Mac OS X, Solaris)

També n'hi ha algun de codi obert però sovint les seves prestacions solen ser molt inferiors:

- **Serna Free Open Source XML Editor** (Windows, Linux, Mac OS X, Solaris)
- **XML Copy Editor** (Linux)

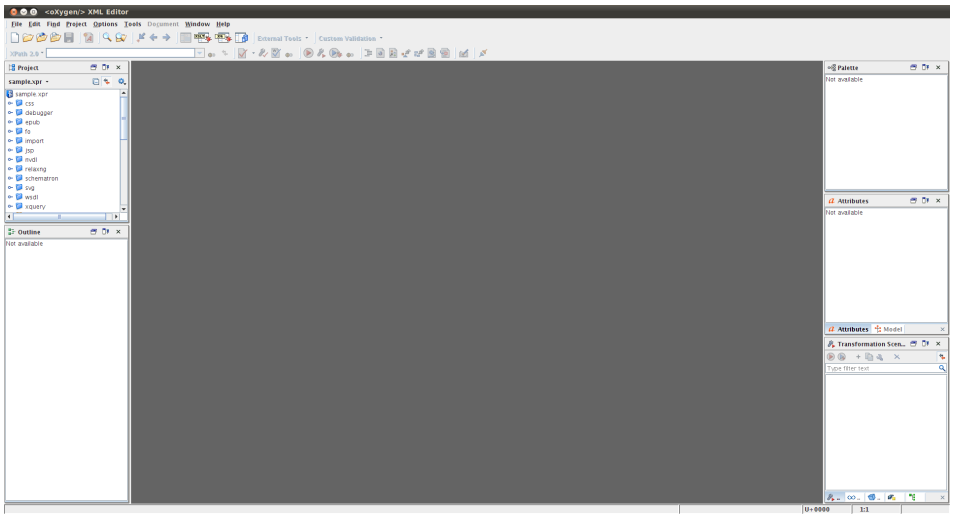
A pesar que en l'exemple es fa servir l'oXygen, la majoria dels editors funcionen d'una manera similar.

Edició d'un document amb un editor d'XML

Si bé és cert que podem crear un document XML amb un editor de text senzill, en aquest exemple es farà servir un editor especialitzat per veure'n les possibilitats. L'editor que farem servir per fer l'exemple és l'oXygen XML Editor.

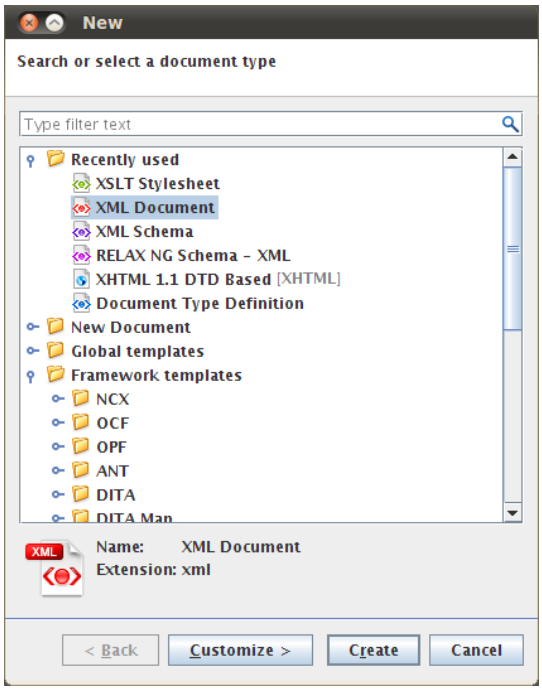
Generalment els editors d'XML divideixen l'espai de treball en finestres en les quals cada una té una funció específica (figura 2.22).

Figura 2.22. Pantalla inicial de l'oXygen XML Editor, on es veuen els diferents espais



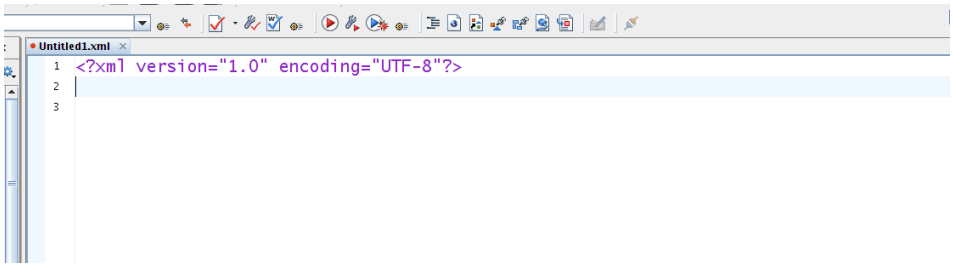
A l'hora de crear un fitxer nou els editors d'XML normalment obren un assistent (figura 2.23) que permet crear fitxers de diversos tipus, de manera que un cop se n'ha triat un se li afegiran automàticament les opcions predeterminades del tipus de document triat.

Figura 2.23. Assistent de creació d'arxius de l'oXygen



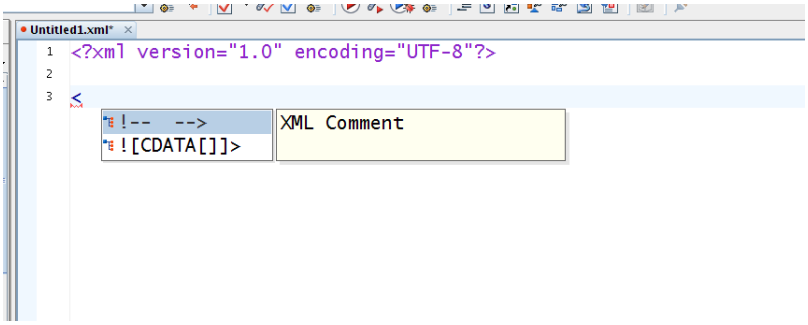
En crear un document XML s'afegeix la capçalera XML automàticament (figura 2.24).

Figura 2.24. Document XML creat per l'assistent d'oXygen



En editar, un assistent anirà oferint les possibilitats que consideri adients en cada punt concret (figura 2.25).

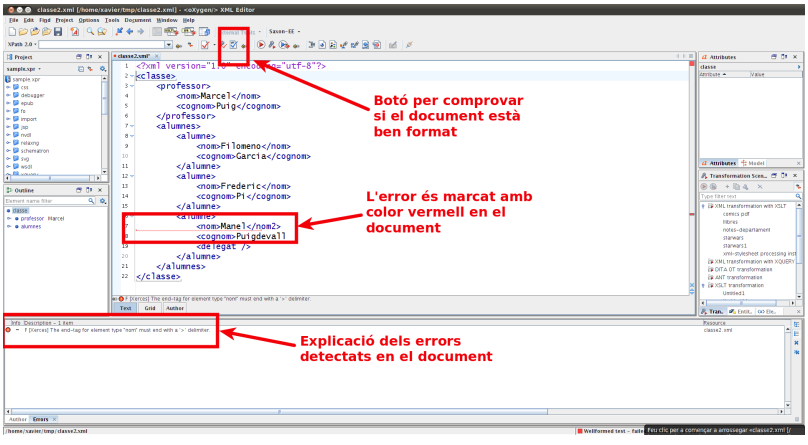
Figura 2.25. L'assistent d'edició ens indicarà què podem escriure en cada punt del document



L'edició no té secrets, ja que el que es fa és simplement editar un arxiu de text amb un assistent que ofereix ajudes diverses:

- En crear una etiqueta d'obertura es crearà l'etiqueta de tancament.
- Mentre s'edita es van comprovant automàticament els errors que sorgeixin.
- Hi sol haver alguna manera de comprovar si un document està ben format, i un panell en el qual es poden veure els errors (figura 2.26).

Figura 2.26. Els errors en l'edició



Els errors en l'edició són detectats per l'editor en escriure o bé quan li demanem que comprovi si el document està ben format.

2.6.2 Creació d'un document XML

A l'hora de definir un grup de dades dins d'un document XML caldrà fer tota una sèrie de passes prèvies que permetin determinar quines són les dades que cal emmagatzemar i posteriorment definir quina és l'estructura que s'ha de donar a aquestes dades; així doncs:

1. Determinació de les dades
2. Determinació de l'estructura

Determinació de les dades

És bàsic abans de crear un document XML saber clarament quines són les dades que s'hi han de posar. Sovint això estarà determinat pel programa que les ha de processar posteriorment, però també pot ser que el programa encara no existeixi i per tant la creació estigui determinada per les preferències personals, etc.

Si no s'està restringit per un programa que ja determini l'estructura del document XML, el que cal és determinar quines dades s'han d'emmagatzemar. Hi ha molts sistemes per fer-ho però el més senzill és fer una llista amb totes les dades rellevants.

Document XML per desar les dades d'una biblioteca

Volem crear una biblioteca en què es puguin emmagatzemar les dades dels llibres que hi ha. Per tant, fem una llista amb les dades que considerem que cal que hi hagi en el document:

- Títol
- Subtítol
- Autor
- Any de publicació
- Editorial
- Nom de la col·lecció
- Idioma

Determinació de l'estructura

Una altra de les coses bàsiques a l'hora de crear un document XML és definir quina és l'estructura que hauran de tenir les dades. Aquesta estructura estarà determinada per les necessitats del programa o de la persona que farà servir el document XML. De manera que les possibilitats a l'hora de crear una estructura per a les dades que volem fer servir són moltes: agrupar per autor, agrupar per llibre, agrupar per editorial, agrupar per tema...

Tria de l'estructura

En una biblioteca podem fer l'estructura des de molts punts de vista. Per exemple, la podem fer a partir dels autors, com aquesta:

- autor 1
  - llibre 1
  - llibre 2
- autor 2
  - llibre 1
- etc.

O bé podem la fer a partir dels llibres d'aquesta manera:

- llibre 1
  - autor 1
- llibre 2
  - autor 1
- etc.

La primera opció és la que s'ha triat per desenvolupar en aquest exemple.

Creació del document

Com que s'ha triat una forma d'organització per mitjà dels autors el que queda clar és que el primer nivell serà una llista d'elements `autor` més o menys d'aquesta manera:

```
<biblioteca>
    <autor></autor>
    <autor></autor>
    <autor></autor>
    ...
</biblioteca>
```

Dins de cada un dels autors les dades per emmagatzemar seran les dades personals de l'autor (el nom, en el nostre exemple) i la llista dels llibres de l'autor que hi hagi a la biblioteca:

```
<autor>
   <nom>Nom de l'autor</nom>
   <llibres>
       Llista de llibres
   </llibres>
</autor>
```

En el darrer nivell es poden posar totes les dades del llibre obviant les que ja estan implícites perquè estan en etiquetes superiors. En l'exemple, l'autor ja queda implícit i, per tant, no cal tornar-lo a posar:

```
<llibre>
    <titol>Titol del llibre</titol>
    <subtitol>Subtítol</subtitol>
    <any>Any en què s'ha publicat el llibre</any>
    <idioma>Idioma en què està escrit</idioma>
    <editorial>
        <nom>Nom de l'editorial</nom>
        <col·lecció>Adreça de l'editorial</col·lecció>
    </editorial>
</llibre>
```

En qualsevol moment es poden crear nivells nous per agrupar les dades segons algun sentit que interessi marcar. Per exemple, això és el que s'ha fet amb l'editorial.

Cal tenir en compte que no cal preocupar-se gaire que en alguns casos les dades no tinguin sentit o no existeixin, ja que en crear el document simplement es poden eliminar les etiquetes que no tinguin sentit.

Per tant, el document final podria quedar d'aquesta manera:

```
<bibloteca>
    <autor>
        <nom>John Ronald Reuel Tolkien </nom>
        <llibres>
          <llibre>
              <titol>El Hòbbit</titol>
              <any>2010</any>
              <idioma>català</idioma>
              <editorial>
                  <nom>Edicions de la magrana</nom>
                  <col·lecció>L'Esparver</col·lecció>
              </editorial>
          </llibre>
          <llibre>
              <titol>El senyor dels anells</titol>
              <subtitol>La germandat de l'anell</subtitol>
              <any>2002</any>
              <idioma>català</idioma>
              <editorial>
                  <nom>Editorial Vicens Vives</nom>
              </editorial>
          </llibre>
        </llibres>
    </autor>
    <autor>
        <nom>Isaac Asimov</nom>
        <llibres>
              <titol>Jo, robot</titol>
              <any>2001</any>
              <idioma>català</idioma>
              <editorial>
                  <nom>Edicions Proa</nom>
                  <col·lecció>Proa Butxaca</nom>
              </editorial>
        </llibre>
    </autor>
</biblioteca>
```

2.7 Espais de noms

Amb l'XML es poden crear les etiquetes amb el nom que es vulgui en el moment en què calguin. Per tant, si tenim un document XML en el qual s'especifiquen habitacions de lloguer podríem tenir un document com aquest:

```
<lloguer>
    <adreca>
        <carrer>Escudillers, 23</carrer>
        <ciutat>Figueres</ciutat>
        <codi_postal>17600</codi_postal>
    </adreca>
    <habitacio>
        <rect>
            <finestres>2</finestres>
            <portes>1</portes>
        </rect>
    </habitacio>
</lloguer>
```

O sigui, en el darrer document XML estem especificant que hi ha una habitació de lloguer al carrer Escudillers, 23, de Figueres, que té forma rectangular amb dues finestres i una porta.

Tot i que teòricament el sistema sembla perfecte, en la pràctica té el problema que **el llenguatge humà és limitat** i a més moltes vegades hi ha paraules amb diversos sentits. Això fa que molta gent pugui triar les mateixes etiquetes per fer coses que siguin totalment diferents. Això d'entrada no és un problema fins que cal mesclar documents que vénen de fonts diferents.

En XML hi ha un llenguatge estàndard que serveix per representar gràfics 2D anomenat *SVG* (*scalable vector gràfics*). SVG serveix per definir gràfics vectorials, o sigui, que en comptes de desar els punts que formen les imatges es desa com s'han de dibuixar els gràfics fent servir figures geomètriques. Es poden veure alguns dels elements de SVG a la taula 2.7.

Taula 2.7. Etiquetes bàsiques del format SVG per representar gràfics vectorials

Etiqueta	Ús
svg	L'arrel dels documents SVG.
line	Serveix per definir línies d'un punt a un altre.
rect	Es fa servir per definir rectangles a partir de quatre punts.
circle	Amb aquesta etiqueta es defineixen cercles a partir del punt central i el radi.
ellipse	Ens permet definir el·lipses a partir del punt central i els dos radis.
polygon	Permet dibuixar polígons a partir d'un grup de punts.

Per tant, algú podria decidir que el document milloraria si s'hi afegeix una representació gràfica de la planta dels pisos, i que es pot aprofitar el llenguatge SVG per fer-la.

Com que l'XML permet mesclar diferents vocabularis, simplement s'hi afegeix. Per fer-ho més estructurat, a més, s'ha definit l'etiqueta `<imatge>`:

```
<lloguer>
    <adreca>
        <carrer>Escudillers, 23</carrer>
        <ciutat>Figueres</ciutat>
        <codi_postal>17600</codi_postal>
    </adreca>
    <habitacio>
        <rect>
            <finestres>2</finestres>
            <portes>1</portes>
        </rect>
    </habitacio>
    <imatge>
        <svg>
            <rect x="0" y="0" width="30" height="60" />
        </svg>
    </imatge>
</lloguer>
```

Per un a lector humà en llegir-ho no hi hauria cap problema perquè ràpidament pot detectar que la imatge està dins de l'element `<imatge>`, però per a un programa, determinar si l'etiqueta és del vocabulari original o bé d'SVG, és pràcticament impossible. Això vol dir que cal algun sistema de poder definir a quin vocabulari pertanyen les etiquetes. Això és el que fan els **espais de noms**.

> Els espais de noms permeten mesclar llenguatges XML en el mateix document i a més definir clarament a quin vocabulari pertany cada etiqueta.

El que fan els espais de noms és canviar els noms de les etiquetes perquè siguin únics. En teoria es podria fer servir qualsevol combinació de caràcters, però com que hi hauria el mateix problema que amb les etiquetes (algú les podria usar) generalment es fa per mitjà d'una URL (*uniform resource locator*), que en principi són úniques. Així es podria definir una URL única al nostre espai de noms (http://www.ioc.cat/lloguer) i la URL d'SVG (`http://www.w3.org/2000/svg`) davant dels elements i el programa ja no tindrà problemes per determinar a quin vocabulari pertany cada etiqueta:

```
<http://www.ioc.cat/lloguer:lloguer>
    <http://www.ioc.cat/lloguer:adreça>
        <http://www.ioc.cat/lloguer:carrer>Escudillers, 23</http://www.ioc.cat/
            lloguer:carrer>
        <http://www.ioc.cat/lloguer:ciutat>Figueres</http://www.ioc.cat/
            lloguer:ciutat>
        <http://www.ioc.cat/lloguer:codi_postal>17600</http://www.ioc.cat/
            lloguer:codi_postal>
    </http://www.ioc.cat/lloguer:adreca>
    <http://www.ioc.cat/lloguer:habitacio>
        <http://www.ioc.cat/lloguer:rect>
            <http://www.ioc.cat/lloguer:finestres>2</http://www.ioc.cat/
                lloguer:finestres>
            <http://www.ioc.cat/lloguer:portes>1</http://www.ioc.cat/
                lloguer:portes>
        </http://www.ioc.cat/lloguer:rect>
    </http://www.ioc.cat/lloguer:habitacio>
    <http://www.ioc.cat/lloguer:imatge>
        <http://www.w3.org/2000/svg:svg>
```

```
            <http://www.w3.org/2000/svg:rect x="0" y="0" width="30" height="60"
                 />
        </http://www.w3.org/2000/svg:svg>
    </http://www.ioc.cat/lloguer:imatge>
</http://www.ioc.cat/lloguer:lloguer>
```

Com que escriure totes les adreces cada vegada fa que es perdi la llegibilitat del document, l'XML permet definir àlies per a cada una de les URL. Els àlies es defineixen per mitjà de l'atribut `xmlns` dels elements i s'hereten a tot el contingut de l'element:

```
<element xmlns:alies="http://adreca"/>
```

Per tant, si es defineix l'atribut en l'arrel del document s'heretaran els àlies a tots els elements del document.

```
<lloguer xmlns:lloguer="http://www.ioc.cat/lloguer"
         xmlns:svg="http://www.w3.org/2000/svg" >
    <lloguer:adreca>
        <lloguer:carrer>Escudillers, 23</lloguer:carrer>
        <lloguer:ciutat>Figueres</lloguer:ciutat>
        <lloguer:codi_postal>17600</lloguer:codi_postal>
    </lloguer:adreca>
    <lloguer:habitacio>
        <lloguer:rect>
            <lloguer:finestres>2</lloguer:finestres>
            <lloguer:portes>1</lloguer:portes>
        </lloguer:rect>
    </lloguer:habitacio>
    <lloguer:imatge>
        <svg:svg>
            <svg:rect x="0" y="0" width="30" height="60" />
        </svg>
    </lloguer:imatge>
</lloguer:lloguer>
```

`xmlns` també permet un espai de noms per defecte i que, per tant, podrà fer servir les seves etiquetes sense àlies. **Si algun atribut `xmlns` no defineix àlies es converteix en l'espai de noms per defecte**.

```
<lloguer xmlns="http://www.ioc.cat/lloguer"
         xmlns:svg="http://www.w3.org/2000/svg" >
    <adreca>
        <carrer>Escudillers, 23</carrer>
        <ciutat>Figueres</ciutat>
        <codi_postal>17600</codi_postal>
    </adreca>
    <habitacio>
        <rect>
            <finestres>2</finestres>
            <portes>1</portes>
        </rect>
    </habitacio>
    <imatge>
        <svg:svg>
            <svg:rect x="0" y="0" width="30" height="60" />
        </svg:svg>
    </imatge>
</lloguer>
```

Com que els espais de noms es poden definir en qualsevol element, una possibilitat alternativa seria **definir els espais de noms en les etiquetes adequades**:

```
<lloguer xmlns="http://www.ioc.cat/lloguer">
    <adreca>
        <carrer>Escudillers, 23</carrer>
        <ciutat>Figueres</ciutat>
        <codi_postal>17600</codi_postal>
    </adreca>
    <habitacio>
        <rect>
            <finestres>2</finestres>
            <portes>1</portes>
        </rect>
    </habitacio>
    <imatge>
        <svg xmlns="http://www.w3c.org/2000/svg">
            <rect x="0" y="0" width="30" height="60" />
        </svg>
    </imatge>
</lloguer>
```

D'aquesta manera tots els descendents de l'element <lloguer> segueixen el seu espai de noms excepte quan s'arriba a l'element <svg>, el qual canvia l'espai de noms per defecte per a ell i per als seus descendents.