DAM

Linguaxes de Marcas e Sistemas de Información

UD3 XML

PROFESORA | CRISTINA PUGA FDEZ	UNIDADE | UD3	TRIMESTRE | 2
MÓDULO | LMSXI		

Tabla de contenido

3. XML

3.1 Introducción

La característica principal de un lenguaje de marcas es que añade al contenido del texto marcas o etiquetas que permiten completar este con información sobre su estructura o presentación.

En el lenguaje HTML existe una serie de etiquetas predefinidas (HTML, HEAD, BODY, etc.) las cuales en un principio eran utilizadas para definir tanto la estructura como la apariencia del documento. Pero con la aparición de las hojas de estilo CSS se pasó a utilizar este lenguaje para la definición de la apariencia.

El lenguaje XML (eXtensible Markup Language, Lenguaje de Macado Extensible) fue creado para estructurar, almacenar y transportar información. Un documento en XML es meramente información textual limitada por etiquetas.

Por lo tanto, la principal diferencia del lenguaje XML con otros lenguajes de marcas es que en XML no existen etiquetas predefinidas. El conjunto de etiquetas a utilizar se definen para cada documento, de forma que permitan identificar su contenido.

Un ejemplo de un documento XML con información sobre pedidos podría ser:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE pedidos SYSTEM "pedidosEmpresa.dtd">
<!-- comentario en el documento -->
<pedidos>
 <pedido num="1">
  <idCliente>345</idCliente>
  <fecha>2009/1/15</fecha>
  <linea>
       <idProducto>325</idProducto>
       <cantidad>2</cantidad>
       <precioUnidad>1,35</precioUnidad>
  </linea>
  <linea>
       <idProducto>27</idProducto>
       <cantidad>1</cantidad>
       <precioUnidad>32,50</precioUnidad>
  </linea>
 </pedido>
 <pedido num="2">
  <idCliente>24</idCliente>
```

```
  <fecha>2009/01/17</fecha>
  <linea>
       <idProducto>73</idProducto>
       <cantidad>1</cantidad>
       <precioUnidad>7,10</precioUnidad>
  </linea>
  </pedido>
</pedidos>
```

Las diferentes aplicaciones que utilicen estos archivos pueden acceder a esta información y administrarla según le interese: enviarla, transformarla, visualizarla, etc.

3.2 Ventajas del uso de XML

Las principales ventajas derivadas del uso de documentos XML son:

- La estructura de un documento XML se puede entender sin dificultad
- Aporta estructura e información a los datos de los documentos
- Facilita la comunicación de información entre aplicaciones, de forma independiente de la plataforma en la que estén desarrolladas y de los orígenes de datos que utilicen.
- Se pueden añadir nuevas etiquetas en el futuro en el documento XML, manteniendo la compatibilidad con las ya existentes (es extensible).
- Existen analizadores estándar de documentos XML que podemos utilizar para procesar cualquier lenguaje basado en XML, lo que acelera el desarrollo de aplicaciones que utilicen documentos XML.

3.3 Herramientas de edición de XML

La edición de documentos XML se puede llevar a cabo por medio de editores de texto genéricos, pero existen herramientas que incorporan funciones específicas como:

- Reconocimiento de la sintaxis y resaltado automático del texto.
- Comprobación de si un documento está o no bien formado.
- Comprobación de la gramática del documento.
- Evaluación de expresiones XPath.
- Utilización de hojas de estilo XSL para realizar transformaciones.

Sin embargo, lo habitual no es que los usuarios interaccionen directamente con ficheros XML, sino que éstos se emplean normalmente para intercambiar información entre aplicaciones.

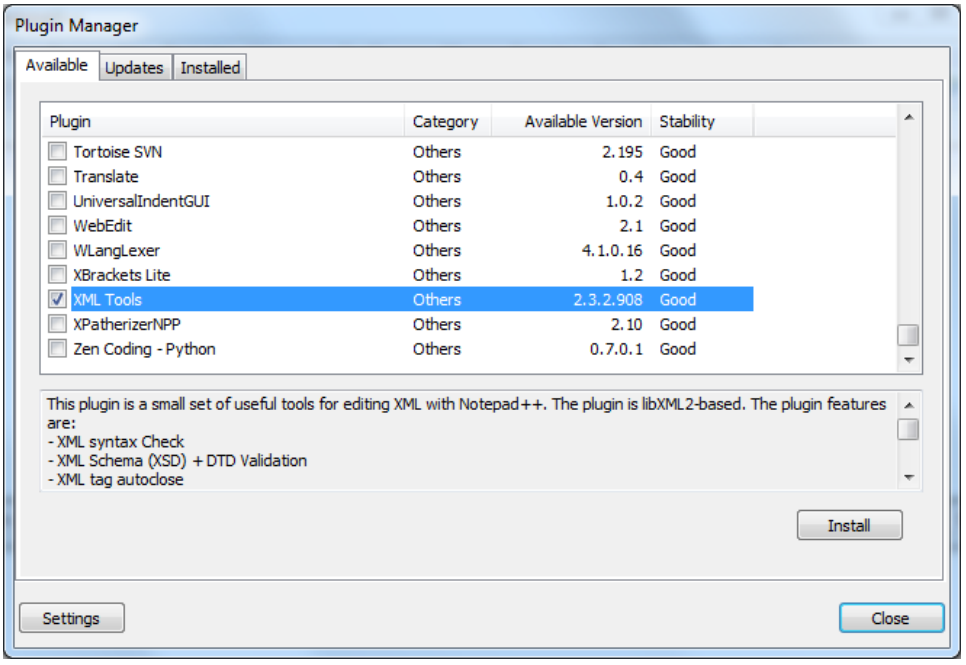
Las aplicaciones analizan e interpretan los archivos XML por medio de una herramienta llamada **parser** (analizador de sintaxis), cuyo objetivo es leer y estructurar los elementos del documento. Cada lenguaje de programación dispone de sus propias librerías que implementan parsers para analizar documentos XML.

3.3.1 Notepad++

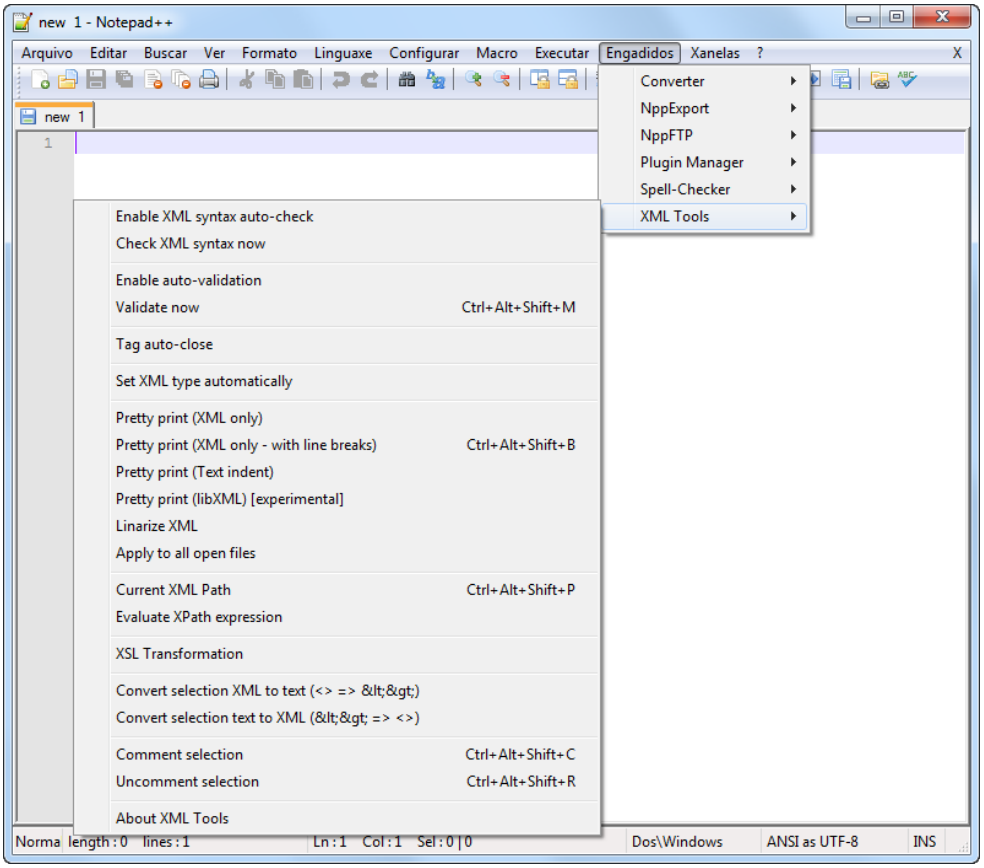
Notepad++ ofrece reconocimiento de código para muchos lenguajes de programación, entre ellos XML, y soporta una amplia gama de codificaciones de documentos.

Para trabajar con Notepad++ en la edición de documentos XML necesitamos instalar el plug-in "XML Tools".

Desde "Plugins->Plugin Manager->Show plugin manager", seleccionamos "XML Tools" y pulsamos el botón "Install".



Una vez instalado el plugin, aparecerá un nuevo submenú “XML Tools” en el menú de Plugins.



3.3.2 BaseX

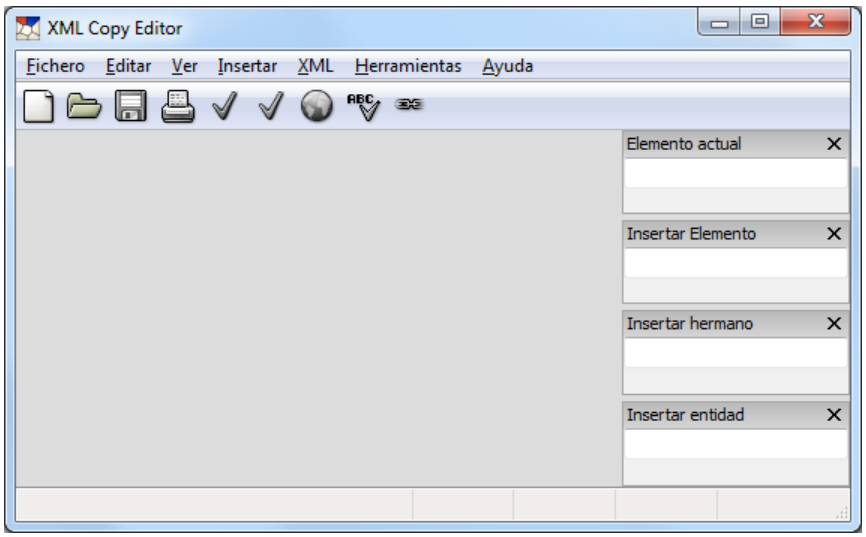
BaseX es un sistema de gestión de bases de datos XML nativo y ligero y un procesador XQuery, desarrollado como un proyecto comunitario en GitHub. Cumple con las especificaciones del Consorcio World Wide Web (W3C) y está especializado en almacenar, consultar y visualizar grandes documentos y colecciones XML. Es independiente de la plataforma y se distribuye como software libre.

A diferencia de otras bases de datos orientadas a documentos, las bases de datos XML brindan soporte para lenguajes de consulta estandarizados como XPath y XQuery. La GUI incluida permite a los usuarios buscar interactivamente, explorar y analizar sus datos, y evaluar expresiones XPath / XQuery en tiempo real.

Ofrece también APIs para trabajar con lenguajes como Python, PHP, Perl, Java, C##, RESTful API, etc.

3.3.3 XML Copy Editor

XML Copy Editor es una herramienta de código libre, con versiones para sistemas operativos Windows e Linux, orientada únicamente al trabajo con documentos XML.



Se puede descargar de: https://xml-copy-editor.sourceforge.io/, y una vez instalada ya está lista para trabajar. Los sistemas de codificación que soporta son más reducidos que los que maneja Notepad++, pero suficientes para la mayoría de casos.

3.4 Contenido de un documento XML

3.4.1 Elementos

La base de un documento XML es el *elemento*. Cada elemento está formado por una etiqueta de apertura y otra de cierre, utilizando en ambos casos los caracteres “<” y “ >”. El contenido de un elemento es todo lo que aparece entre ambas etiquetas.

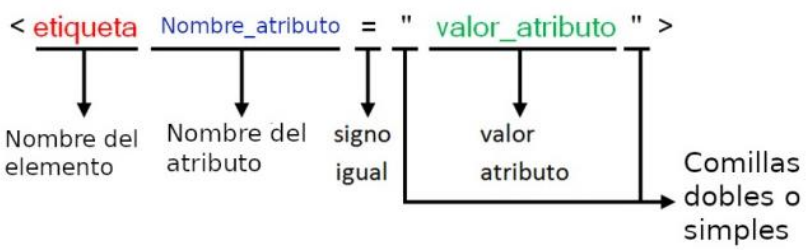


Ilustración 1- Elementos XML

Los elementos pueden contener:

- Atributos: aparecen siempre en la etiqueta de apertura.
- Texto: entre la etiqueta de apertura y la de cierre.
- Otros elementos.

Por ejemplo, el elemento “modulo” contiene atributos “sesiones” y “horas”, un texto y un subelemento “profesorado”.

```
<módulo sesiones="5" horas="133">Linguaxes de marcas e
 sistemas de información
   <profesorado>Maria Fernández</profesorado>
</módulo>
```

También pueden existir elementos vacíos, que pueden ser representados por dos o por una única etiqueta:

```
<activo></activo>
```

```
<activo />
```

3.4.2 Nombres de los elementos

Se considera como nombre de un elemento lo que aparece en la etiqueta de apertura y en la de cierre, los cuales deben coincidir. Puede contener cualquier carácter alfabético (con o sin tildes), números, y algunos signos de puntuación, pero no pueden contener espacios. Además, el primer carácter no puede ser un número ni signo de puntuación. Por ejemplo, **no** serían elementos **válidos**:

```
<1Enero2020></1Enero2020>
<+elemento></+elemento>
<.elemento></.elemento>
<-elemento></-elemento>
```

También están reservados los nombres de elementos que empiezan por los caracteres “xml”, en cualquier combinación posible de mayúsculas y minúsculas (Xml, XmL,xmL).

3.4.3 Atributos

Los elementos pueden tener atributos, los cuales son una forma de incorporarles características y tienen las mismas reglas que los nombres de elementos. Por ejemplo:

```
<módulo sesiones="5" horas="133">
```

No existe límite para el número de atributos ni es significativo su orden, pero un elemento concreto no puede tener atributos repetidos.

3.4.3.1 *Subelementos*

En ocasiones debemos elegir entre utilizar un subelemento o un atributo para almacenar cierta información. Para ello, debemos pensar que los atributos están pensados para guardar información (descripciones) sobre los datos y sólo pueden tomar un único valor, mientras que los elementos pueden almacenar varios datos e incluso extenderse con estructuras anidadas. Por ejemplo, si ponemos profesor como atributo de módulo, sólo podremos definir 1 único profesor, pero si lo ponemos como subelemento podremos definir 1 o varios profesores para ese módulo.

3.4.3.2 *Valores de los atributos*

El valor del atributo debe ir encerrado entre comillas, simples o dobles (pero sin mezclar ambos tipos en el mismo atributo) y se separa del nombre del atributo por un signo "=". Además, todos los atributos deben tener un valor definido. Por ejemplo, no sería válido:

```
<elemento atrib1></elemento>
```

Y por el contrario, sería válido:

```
<elemento atrib1=""></elemento>
```

3.4.4 Texto

Como parte del contenido de un elemento, entre las etiquetas de apertura y cierre puede aparecer texto, otros elementos, o ambos al mismo tiempo.

El texto de los documentos XML puede contener cualquier carácter salvo "z" y "&". Esta misma restricción se aplica para los valores de los atributos, los cuales además no pueden contener el mismo tipo de comilla que se utiliza para delimitarlos.

Ejemplos de valores no válidos serían:

```
<autor>Stevie Ray Vaughan & Double Trouble</autor>
<canción nombre='Don't bang the drum'></canción>
```

Para utilizar uno de estos caracteres no permitidos, podemos:

- Utilizar una referencia a un carácter Unicode: las referencias empiezan por “&#” (“&#x” para hexadecimal) y finalizan con el carácter “;”(Ejemplos: “&”→ "&", “<” → "<").
- Utilizar entidades: como veremos más adelante se podrían reemplazar los caracteres & y <, respectivamente, por las cadenas "&" y "<".

Ejemplos:

```
<autor>Stevie Ray Vaughan &#38; Double Trouble</autor>
```

```
<autor>Stevie Ray Vaughan &amp; Double Trouble</autor>
```

3.4.5 Comentarios

A veces es aconsejable insertar comentarios en un documento XML que serán ignorados al procesar la información del documento. Tienen el mismo formato que en HTML, es decir, comienzan con la cadena "<! --" y terminan con "-->".

Los comentarios pueden aparecer en cualquier lugar del documento y pueden contener cualquier carácter excepto la combinación de doble guión “--”.

```
<!-- Isto é un comentario -->
```

3.4.6 Instrucciones de procesamiento

Se utilizan para dar información a las aplicaciones que procesan los documentos XML. Aunque pueden aparecer en cualquier parte del documento, suelen ponerse al principio y se delimitan por “<?” y “?>”.

```
<?xml-stylesheet type="text/xsl" href="plantilla.xsl"?>
```

3.5 Documentos XML bien formados

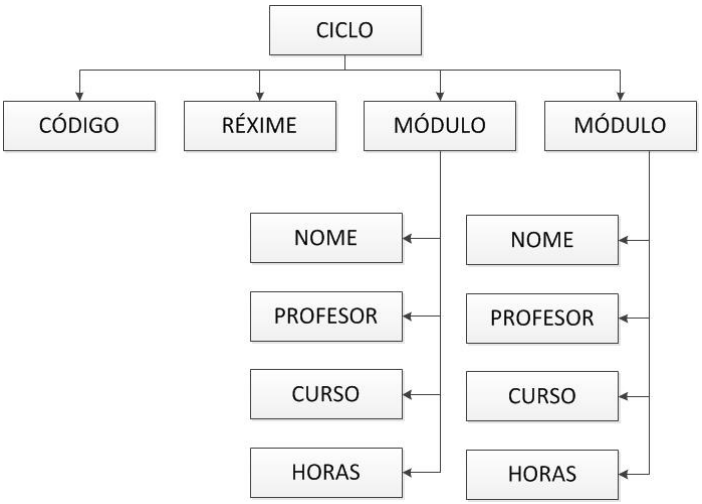
Un documento se considera “bien formado” si respeta las normas que se han indicado hasta ahora y además las siguientes descritas en la especificación oficial de XML:

- Todos los elementos del documento se anidan en forma de árbol, existiendo un único elemento raíz.

Ejemplo:

```
<ciclo>
  <código>ASIR</código>
  <réxime>Ordinario</réxime>
  <módulo>
    <nome>Linguaxes de marcas e sistemas de
 información</nome>
    <profesor>Xaime Louzán</profesor>
    <curso>1º</curso>
    <horas>133</horas>
  </módulo>
  <módulo>
    <nome>Servizos de Rede e Internet</nome>
    <profesor>Ana Quintela</profesor>
    <curso>2º</curso>
    <horas>140</horas>
  </módulo>
</ciclo>
```

Cuya vista en forma de árbol sería:



- Todos los elementos deben ser cerrados y de forma estructurada es decir, el último elemento en abrirse será el primero en cerrarse.

```
<elementoA>
  <elementoB>
  </elementoB>
</elementoA>
```

- La etiqueta de cierre de un elemento debe coincidir exactamente con el nombre de la etiqueta de apertura (mayúsculas y minúsculas incluidas).
- En la especificación 1.1 del lenguaje, los documentos XML deben empezar por un prólogo.
- Los navegadores web distinguen los documentos bien formados de los mal formados. Si abrimos un documento XML bien formado con un navegador, se mostrará la información que contiene en forma de árbol con nodos dinámicos. En caso de tener errores, el navegador señala el primer elemento que tiene error.

  Ejemplo documento XML bien formado:

```
<?xml version="1.1" encoding="ISO-8859-1"?>
<directorio>
 <entrada>
   <apelidos>Louzán Ferreiro</apelidos>
   <nome>Xaime</nome>
 </entrada>
</directorio>
```

3.5.1 Especificaciones de XML

Existen dos especificaciones del lenguaje XML: 1.0 y 1.1. Las diferencias entre ellas son escasas siendo la más utilizada la 1.1, que añade sobre la anterior:

- Soporte Unicode completo (la versión 1.0 sólo soporta los caracteres recogidos en Unicode 2.0)
- Obligatoriedad de incluir un prólogo en el cual se debe indicar la versión del lenguaje utilizada y opcionalmente puede incluir atributos como encoding (tipo de codificación utilizada) o standalone (para indicar dependencias de documentos de comprobación de la validez de la gramática).

```
<?xml version="1.1" encoding="utf-8"?>
```

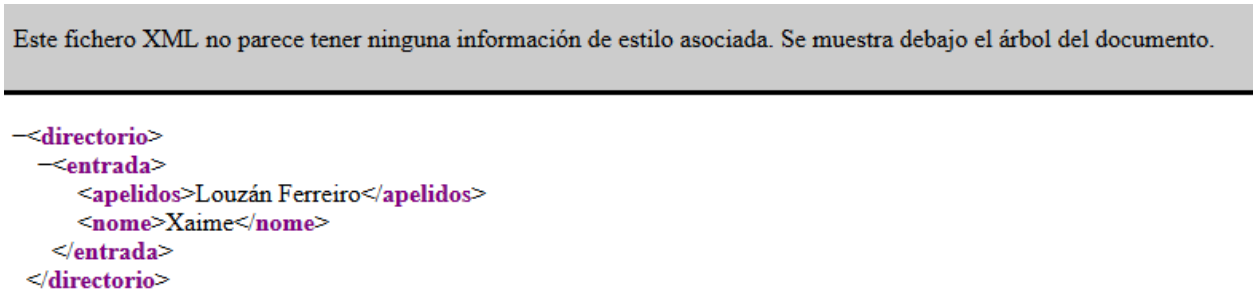
3.5.2 Documentos bien formados y navegadores web

Los navegadores web pueden distinguir entre documentos bien formados y mal formados. Si abrimos con el navegador un documento bien formado nos mostrará la información que contiene en forma de árbol de nodos dinámicos.

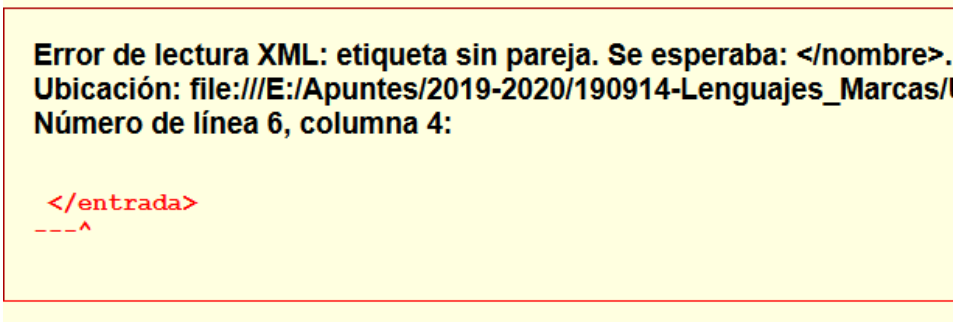
Por ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<directorio>
 <entrada>
   <apellidos>Louzán Ferreiro</apellidos>
   <nombre>María
 </entrada>
</directorio>
```

Se mostraría en el navegador como:



En caso de que el documento esté mal formado, el navegador señalará el primer elemento que contenga un error. Por ejemplo, si eliminamos en el código anterior </nome>, nos mostraría:



3.5.3 Espacios en blanco

XML define un conjunto de caracteres como “espacios en blanco” para hacer más legible el código. Estos caracteres son:

- Espacios (32).
- Tabuladores (9).
- Saltos de línea (10)
- Retorno de carro (13).

Normalmente las aplicaciones que procesan los documentos XML eliminan de su contenido los espacios sobrantes, pero si queremos que los mantenga debemos utilizar un atributo especial de

XML “xml:space”, que puede tener dos valores:

- default: se procesa el documento de forma normal.
- preserve: se preservan los espacios tal y como están en el contenido del elemento.

```
<autor>Rosalía de Castro</autor>
<poesía xml:space="preserve">
Adiós, ríos; adios, fontes;
adios, regatos pequenos;
adios, vista dos meus ollos:
non sei cando nos veremos.
</poesía>
```

3.5.4 Lenguaje del contenido

Para identificar el idioma en el que está escrito el contenido de un elemento se utiliza: “xml:lang”, cuyo valor será un código de dos letras de los que recoge la norma ISO 639-1(ejemplo, gl para gallego, es para castellano).

```
<canción título="Automatic imperfection" xml:space="preserve"
 xml:lang="en">
There are children playing on the manpost
'tending they are kids
They can climb up and fall down hard on their knees
…
</canción>
<canción título="Lo que sueñas vuela" xml:space="preserve"
 xml:lang="es">
Caminando sin pensar,
despacito, sin tiempo que ganar
…
</canción>
```

3.5.5 Las entidades

Las entidades son estructuras XML con nombre asociado. Para referenciar una entidad se utiliza la siguiente sintaxis:

```
&nombreDeEntidad;
```

Al referenciar una entidad por su nombre, se incluye automáticamente su contenido en lugar de su referencia. Algunas entidades predefinidas en la especificación de XML son:

Carácter referenciado	Referencia
&	&amp;
<	&lt;
>	&gt;
“	&quot;
‘	&apos;
Sustitución de un carácter utilizando su código hexadecimal. Ejemplo: ©	&#codChar;

3.5.6 La sección CDATA

A veces, el uso de entidades puede ser muy incómodo, y la lectura del documento muy complicada para los *humanos.* Por esto, puede ser útil el uso de secciones, ya que el parser XML ignorará su contenido.

Para la definición de una sección se utilizar:

<![CDATA[...]]>

Por ejemplo, el siguiente código XML:

```
<texto>Tendremos este animal en distintas categorías de &lt; a
 &gt; serían: caniche, mastín</texto>
```

Puede sustituirse por:

```
<texto><![CDATA[Tendremos este animal en distintas categorías
 de < a > serían: caniche, mastín ]]></texto>
```

3.6 Necesidad de tener espacios de nombres

Un espacio de nombres es una forma de agrupar elementos y atributos con origen común, para diferenciarlos de otros elementos con el mismo nombre.

Por ejemplo, supongamos una tienda de alimentación con la siguiente gramática XML para sus

productos:

```
<?xml version="1.1" encoding="utf-8"?>
<productos>
  <producto>
    <cod>LACT02330993</cod>
    <descripción>Leche entera envase 1L</descripción>
  </producto>
  <producto>
    <cod>LACT00493112</cod>
    <descripción>Margarina vegetal tarrina 250g</descripción>
  </producto>
  …
</productos>
```

Si más adelante la misma empresa crea una gramática XML para definir los clientes habituales:

```
<?xml version="1.1" encoding="utf-8"?>
<clientes>
  <cliente>
    <cod>CL09384</cod>
    <nombre>Uxío Fuentes Neira</nombre>
    <dirección>Rúa Europa 24, 3ºA</ dirección >
    <teléfono>555098433</teléfono>
  </cliente>
  <cliente>
    …
  </cliente>
  …
</clientes>
```

En ambos casos existe un elemento de nombre “cod” y en caso de que se mezclen los datos de clientes con los de los productos, tendremos que diferenciar los elementos según su significado.

Se podría cambiar los nombres de los elementos para que no coincidan, pero en muchas ocasiones esto no es posible.

Los espacios de nombres son una recomendación de W3C (*World Wide Web Consortium*) para que los nombres comunes de elementos y etiquetas no colisionen.

Debemos utilizar espacios de nombres en los documentos XML si existe la posibilidad de que sean compartidos y mezclados con otros documentos XML.

3.7 Declaración de un espacio de nombres

Para declarar un espacio de nombres se utiliza la palabra reservada “xmlns” (*XML NameSpace*). Por ejemplo:

```
<?xml version="1.1" encoding="utf-8"?>
<productos
xmlns="http://www.atendadepaco.com/espaciodenombres/productos/">
    <producto>
    <cod>LACT02330993</cod>
    <descripción>Leche entera envase 1L</descripción>
  </producto>
  <producto>
    <cod>LACT00493112</cod>
    <descripción>Margarina vegetal tarrina 250g</descripción>
  </producto>
  …
</productos>
```

Si declaramos el espacio de nombres en el elemento raíz, todos los elementos del documento pasan a formar parte del espacio de nombres. Se denomina *espacio de nombres por defecto*.

3.7.1 URIs, URLs y URNs

Para identificar un espacio de nombres, y diferenciarlo de otro, se asocia con una cadena de texto, la cual debe ser única.

- Una solución es utilizar una URL URL (*Uniform Resource Locator*, Localizador Uniforme de Recurso) para la cadena de texto del espacio de nombres.

```
http://www.atendadepaco.com/espaciosdenommbres/productos/
```

- Una URI (*Uniform Resource Identifier*, Identificador Uniforme de Recurso), que es un identificador, no tiene que ser la dirección de un recurso existente.
- Una URN (*Uniform Resource Name*, Nome Uniforme de Recurso) es un nombre que identifica algo. Cada URN está formada por un espacio de nombres y una cadena que identifica un ítem de ese espacio de nombres.

```
urn:[espazo de nomes]:[cadea de texto]
```

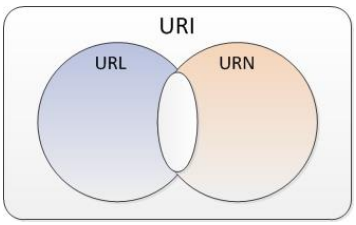


Ilustración 2- URI

3.7.2 Espacios de nombres con prefijo

Cuando necesitamos utilizar más de un espacio de nombres en un documento XML, debemos elegir un prefijo para ellos, el cual se indica a continuación de la palabra “xmlns:”. Este prefijo será el que debemos utilizar para reflejar a que espacio de nombres pertenece el elemento.

Por ejemplo, si creamos un documento de ventas, podríamos tener:

```
<?xml version="1.1" encoding="utf-8"?>
<venta
 xmlns="http://www.atendadepaco.com/espaciosdenombres/ventas/"
 xmlns:cl="http://www.atendadepaco.com/espaciosdenombres
 /clientes/" xmlns:pr="http://www.atendadepaco.com/
 espaciosdenombres/productos/">
  <cl:cliente>
    <cl:cod>CL09384</cl:cod>
    <cl:nombre>Uxío Fuentes Neira</cl:nombre>
    <cl:direccion>Rúa Europa 24, 3ºA</cl: direccion >
    <cl:teléfono>555098433</cl:teléfono>
  </cl:cliente>
  <pr:productos>
    <pr:producto>
      <pr:cod>LACT02330993</pr:cod>
      <pr:descripción>Leche entera envase 1L</pr:descripción>
    </pr:producto>
    <pr:producto>
      <pr:cod>LACT00493112</pr:cod>
      <pr:descrición>Margarina vegetal tarrina
 250g</pr:descrición>
    </pr:producto>
    …
  </pr:productos>
  …
  <importe_total>16,34€</importe_total>
</venta>
```

3.7.3 Ámbito de un espacio de nombres

El ámbito de un espacio de nombres es el conjunto de elementos del documento donde podemos utilizarlo. El ámbito comprende el elemento en el que se declara, y todos los elementos que este contiene.

Se puede redefinir un espacio de nombres por defecto, o un espacio de nombres asociado a un prefijo, dentro de su ámbito. Para esto, simplemente debemos volver a definirlo.

También es posible eliminar la definición anterior de un espacio de nombres por defecto. Sólo tenemos que redefinirlo utilizando como identificador una cadena vacía. Por ejemplo:

```
<?xml version="1.1" encoding="utf-8"?>
<elemento_raiz xmlns="http://www.ejemplo.com/">
  <elemento1 xmlns="">
    …
  </elemento1>
  …
</elemento_raiz>
```

En este caso, el espacio de nombres por defecto declarado en el elemento raíz, no se aplica al elemento “elemento1” ni a todos sus subelementos.

3.7.4 Atributos y espacios de nombres

Los espacios de nombres se aplican sólo a los elementos, no se aplican a los atributos ni a otros elementos como comentarios o instrucciones de procesamiento.

Por ejemplo:

```
<?xml version="1.1" encoding="utf-8"?>
<elemento_raiz xmlns:ns="http://www.ejemplo.com/">
  <ns:elemento1 cod="A334">
    …
  </ns:elemento1>
  …
</elemento_raiz>
```

El espacio de nombres “http://www.ejemplo.com/” se declara en el elemento raíz, y se aplica al elemento “elemento1”, lo cual no incluye el atributo “cod”. En los documentos XML, los atributos pertenecen a un elemento concreto, lo cual evita que puedan colisionar con atributos pertenecientes a otros espacios de nombres.

En caso de querer aplicar un espacio de nombres a un atributo, podríamos hacerlo utilizando la misma sintaxis que con los elementos:

```
<?xml version="1.1" encoding="utf-8"?>
<elemento_raiz xmlns:ns="http://www.ejemplo.com/">
  <ns:elemento1 ns:cod="A334">
    …
  </ns:elemento1>
  …
</elemento_raiz>
```

ANEXO I: Materiales

I. Textos de apoyo o de referencia

- Wikipedia. http://www.wikipedia.org
- W3schools CSS: https://www.w3schools.com/xml/default.asp

II. Recursos didácticos

- Apuntes en el aula virtual.
- Ordenador personal, con navegador web y conexión a internet.
- Software para elaboración de documentos de texto.