Tema 1 Fundamentos de XML

1 ¿qué es XML?

1.1 objetivos de XML

XML es un lenguaje de marcas que se ha estandarizado y se ha convertido en uno de los formatos más populares para intercambiar información.

Se trata de un formato de archivos de texto con marcado que deriva del lenguaje **SGML**, y que ha establecido una serie de reglas más estrictas, pero más fáciles y coherentes.

La realidad es que XML siempre ha estado muy ligado al éxito de HTML. Su aparición se justifica por los problemas crecientes que se fueron observando en las páginas web.

El lenguaje **HTML**, propio de las páginas web, a finales de los años 90 tenía estos problemas como formato de intercambio de información:

- **La mayoría de etiquetas HTML no eran semánticas**. Es decir, no sirven para decir el tipo de contenido que tenemos sino para indicar el formato. Por ejemplo, la etiqueta *h1* en HTML marca un título de primer nivel. Esta etiqueta sí es semántica.

Pero existían otras etiquetas que no son semánticas.

Así *font,* sirve para colorear o cambiar el tipo de letra y no para indicar qué tipo de texto tenemos.

- **HTML es un lenguaje rígido, no es extensible**. Es decir, no podemos añadir etiquetas nuevas acordes con nuestras necesidades, ya que ningún navegador las reconocerá. Las nuevas etiquetas deben de ser aprobadas por los estándares y luego aceptadas por los navegadores. Proceso extremadamente lento.

- **Requiere añadir lenguajes distintos de HTML para añadir potencia y funcionalidad**, por lo que los diseñadores tienden a incrustar dentro del código HTML código de otros lenguajes como **PHP** o **Javascript**, lo que dificultan su legibilidad y comprensión.

Por ello al crear XML se plantearon estos objetivos:

[1] Debía de ser **similar a HTML** (de hecho, se basa en el lenguaje SGML base para el formato HTML), dada su enorme aceptación.

[2] Debía de ser **extensible**, es decir que sea posible añadir nuevas etiquetas sin problemas. Esa es la base del lenguaje XML.

[3] Debía de tener unas **reglas concisas y fáciles**, además de **estrictas**.

[4] Debía de ser **fácil de implantar en todo tipo de sistemas**. XML nace con una vocación multiplataforma, como base de intercambio de información entre sistemas de toda índole.

[5] Debía ser **fácil de leer para todas las personas**.

[6] Debía ser **fácil crear herramientas capaces de procesar XML**

1.2 XML y SGML

Básicamente XML es SGML. Es decir, una persona que conozca SGML no tendrá ningún problema en aprender XML. Las bases son las mismas, pero XML elimina gran parte de su complejidad. De hecho, se dice que XML es un subconjunto de SGML. Es decir, es SGML, pero restringiendo o eliminando ciertas normas.

Por otro lado, XML sirve para lo mismo que SGML: para diseñar lenguajes de marcas. Es decir, XML define tipos de documentos, de forma que, junto a la información del documento, hay una serie de etiquetas que sirven para clarificar lo que significa la información.

De esa manera XML (como ya hizo SGML) es un lenguaje que permite especificar otros lenguajes. Entendiendo como lenguajes, en este caso, al conjunto de etiquetas que se pueden utilizar en un documento. No solo qué etiquetas, sino en qué orden y de qué forma se pueden utilizar.

1.3 lenguajes basados en XML

Algunos de los lenguajes estándares de marcado basados en XML son:

- **RSS**. (*Really Simple Syndication*, aunque hay otras interpretaciones de este acrónimo). Para producir contenidos sindicables, los cuales se utilizan, fundamentalmente, para producir contenidos simples a los que nos podemos suscribir. El ejemplo más claro son los contenidos que muestran noticias, cuyo flujo es continuo.
- **Atom**. Formato semejante al anterior, pensado también para distribuir contenidos a los que nos podemos suscribir. Muchas veces complementa a RSS
- **ePUB**. Formato de libro digital que se ha convertido en un estándar de facto por la gran implantación que está teniendo en todos los dispositivos de lectura digitales. En realidad, es un archivo comprimido (ZIP) que contiene tres documentos XML que son los que especifican la estructura y contenido del documento.
- **SVG**. *Scalable Vector Graphics*, gráficos de vectores escalables. Permite definir imágenes vectoriales pensadas para ser publicadas en una página web.
- **DITA**. *Darwin Information Typing Architecture*, se utiliza para producir documentos técnicos y está siendo cada vez más popular. Tiene capacidad para incluir numerosos metadatos y puede adaptarse a las necesidades de todo tipo de entidades. Permite dividir en temas los documentos.

- **MathML**. Pensado para representar expresiones matemáticas.
- **ODF** u **OD**. *Open Document for Office Applications*. Es un formato abierto que pretende ser un estándar como formato de intercambio entre documentos de oficina (hojas de cálculo, procesadores de texto, presentaciones, diagramas…). Su popularidad aumentó notablemente cuando fue adoptado por el software **Open Office**, actualmente perteneciente a la fundación **Apache**.
- **OOXML**. *Office Open XML*. Formato rival del anterior, propiedad de Microsoft y utilizado como formato XML para el software Microsoft Office. Pretende también ser un estándar.
- **OSDF**. *Open Software Description Format*, basado en especificaciones de las empresas **Marimba** y **Microsoft**, sirve para describir la tecnología y los componentes con los que se ha desarrollado un determinado software, a fin de facilitar el proceso de instalación.
- **RDF**, *Resource Description Format*, Sirve para desarrollar documentos que describan recursos. Se trata de un proyecto, ya antiguo, para definir modelos de metadatos. Se basa en los modelos conceptuales como el modelo entidad/relación o los diagramas de clases, aunque actualmente se utiliza fundamentalmente para describir recursos web.
- **SMIL**. *Synchronized Multimedia Integration Language*, Lenguaje sincronizado de integración multimedia. Utilizado para producir presentaciones de TV en la web, fundamentalmente.
- **SOAP**. *Simple Object Access Protocol*. Protocolo estándar de comunicación entre objetos utilizado para comunicar con Servicios Web.
- **WSDL**. *Web Services Description Language*. Lenguaje para definir Servicios Web.
- **VoiceXML**. Se utiliza para representar diálogos vocales.
- **XHTML**. Versión del lenguaje de creación de páginas web, HTML, que es compatible con las normas XML.

1.4 usos de XML

*contenido web

XML fue defendido por **Tim Bernes-Lee** el creador de la web. La idea era que acabara retirando a HTML como formato principal de los documentos de la web.

De hecho, XHTML, durante un tiempo, fue el HTML aceptado por la comunidad de desarrolladores web. XHTML es un lenguaje creado bajo las normas de XML.

Sin embargo, el triunfo de HTML 5 ha limitado completamente esta idea. Ahora XHTML no tiene, prácticamente, ninguna influencia. forma se pueden utilizar en el documento.

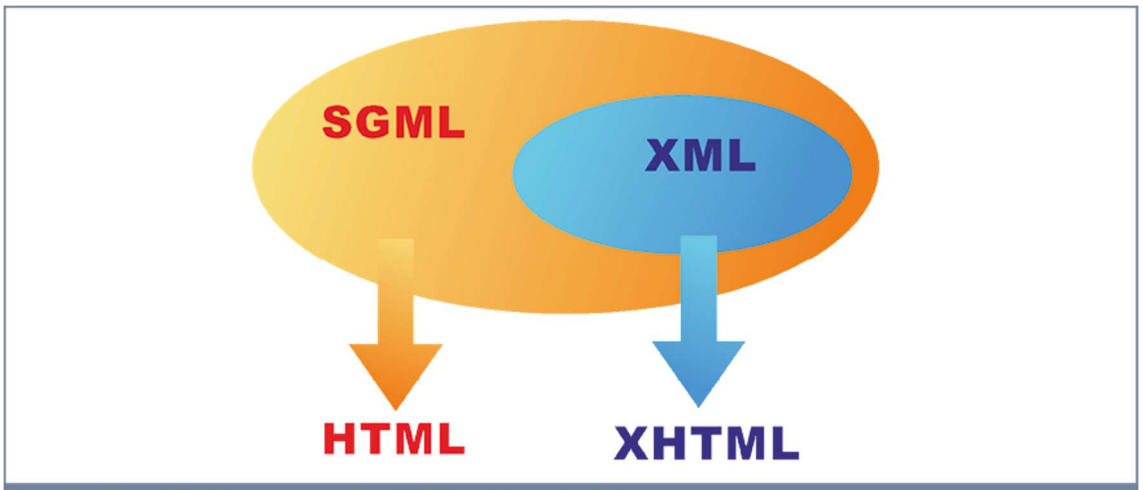

Ilustración 20. Relación entre SGML, XML y HTML

*intercambio de información entre aplicaciones

El hecho de que XML almacene información mediante documentos de texto plano, facilita que se utilice como estándar, ya que no se requiere software especial para leer su contenido: es texto y es entendible por cualquier software.

Numerosos servicios en Internet ofrecen resultados de consultas en este formato. Entre ellos el tiempo, resultados de elecciones, datos estadísticos, consultas a bases de datos, etc.

*documentación

Es un formato muy defendido para almacenar documentos. Su simplicidad y gran capacidad semántica son la clave del éxito en este tipo de usos. El formato documental basado en XML más famoso es el formato ePUB, el más utilizado como formato de **eBooks**.

Como formato documental para las empresas es muy interesante por su versatilidad y facilidad de manipulación.

***bases de datos**

Todas las grandes bases de datos empresariales son capaces de utilizar XML para extraer datos o incluso como formato fundamental de algunos sistemas gestores de bases de datos. Lenguajes como **XQuery** o **XPath** permiten encontrar o navegar entre los datos de un documento XML.

Las principales bases de datos incluso tienen a XML como un posible formato para sus atributos o columnas, de modo que se pueden almacenar directamente en ellas y luego utilizar técnicas de búsqueda de información propias de XML para extraer la información que nos interese.

***formato de imagen vectorial**

En estos últimos años, el formato SVG es reconocido por todos los navegadores. SVG es un lenguaje XML que sirve para representar imágenes vectoriales. AL ser imágenes que no pierden calidad al ampliar o reducirse, se han convertido en las ideales para los logotipos, líneas, formas e iconos de las páginas.

***archivos de configuración**

Para almacenar información de configuración o de archivos tipo log (archivos que graban los eventos ocurridos en un determinado dispositivo) de diversos aparatos hardware, es también un formato ideal.

Numerosos switches, routers, impresoras o servidores utilizan el lenguaje XML para este tipo de archivos. De esa forma la información se estructura de forma muy semántica y es muy sencillo modificar la configuración de estos aparatos.

1.5 tecnologías relacionadas con XML

El ecosistema XML aporta un gran número de tecnologías y lenguajes (la mayoría de lenguajes están también basados en XML) para conseguir obtenener más funciones de los documentos XML.

Las tecnologías más importantes son:

- **DTD**. *Document Type Definition*, definición de tipo de documento. Es un lenguaje que permite especificar reglas que han de cumplir los documentos XML a los que se asocien. Es decir, permite crear documentos de validación para archivos XML.
- **XML Schema**. La función que realiza esta tecnología es la misma que la anterior. La diferencia está en que los documentos XML Schema poseen una sintaxis 100% XML (DTD se basa en SGML), por lo que es un formato orientado a reemplazar al anterior.
- **Relax NG**. Otro formato de definición de validaciones para documentos XML. Es una alternativa a las dos anteriores. Tiene una sintaxis más sencilla.
- **Namespacing**, espacios de nombres. Es una norma que permite conseguir nombres de elementos que carecen de ambigüedad. Es decir, nombres únicos para los distintos elementos que están dentro de los documentos XML.
- **XPath**. Lenguaje de consulta que permite seleccionar o acceder a partes de un documento XML.
- **CSS**. *Cascade StyleSheet*. Hojas de estilo en cascada. Permiten dar formato a los documentos XML o HTML.
- **XSLT**. Sirve para lo mismo que CSS: dar formato a un documento XML. Tene muchas más posibilidades que CSS. Tiene la capacidad de convertir un documento XML en un documento HTML o incluso a tipos comerciales como PDF.
- **XQuery**. Permite consultar datos de los documentos XML, manejándolos como si fueran parte de una base de datos.
- **DOM**. *Document Object Model*, tecnología que permite acceder a la estructura jerárquica del documento. Normalmente para poder ser utilizada por un lenguaje de programación (como **JavaScript**) y poder dar dinamismo a su contenido.

- **SAX**. *Simple API for XML*, permite el uso de herramientas para acceder a la estructura jerárquica del documento XML a través de otro lenguaje. Se usa mucho en la programación de aplicaciones en lenguaje Java.
- **XForms**. Formato de documentos pensados para ser usados como formularios de introducción de datos.
- **XLink**. Permite crear hipervínculos en un documento XML.
- **XPointer**. Semejante al anterior, especifica enlaces a elementos externos al documento XML.

1.6 software para producir XML

En principio XML se puede escribir desde cualquier editor de texto plano (como el **bloc de notas** de Windows o el editor **vi** de Linux). Pero es más interesante hacerlo con un editor que reconozca el lenguaje y que además marque los errores en el mismo.

Además, las distintas funciones habituales que se realizan con los documentos XML, requieren de otro software capaz de manipular el contenido de los documentos.

El software necesario, habitual, es el siguiente:

[1] Un **editor de texto** plano para escribir el código XML. Bastaría un editor como el bloc de notas de Windows o el clásico vi de Linux.

Pero hay editores mucho más apropiados. Son los editores de código de múltiple propósito. Son editores que colorean el código especial porque conocen casi todos los lenguajes (C, Java, Python, HTML). Ejemplos de ellos son **emacs**, **Notepad++** o **SublimeText**.

[2] Un **analizador sintáctico o parser,** programa capaz de entender y validar el lenguaje XML. **Apache Xerces** es quizá el más popular validador de documentos XML.

[3] Un **procesador XML.** Software capaz de producir una presentación visual sobre el documento XML. Un simple navegador puede hacer esta función, pero cuando se aplican lenguajes de presentación como CSS.

Pero hay software que transforma los documentos XML mediante el lenguaje XSL, para producir resultados, por ejemplo, en HTML. **Apache Xalan** y **Saxon** son los dos procesadores de XSL más conocidos

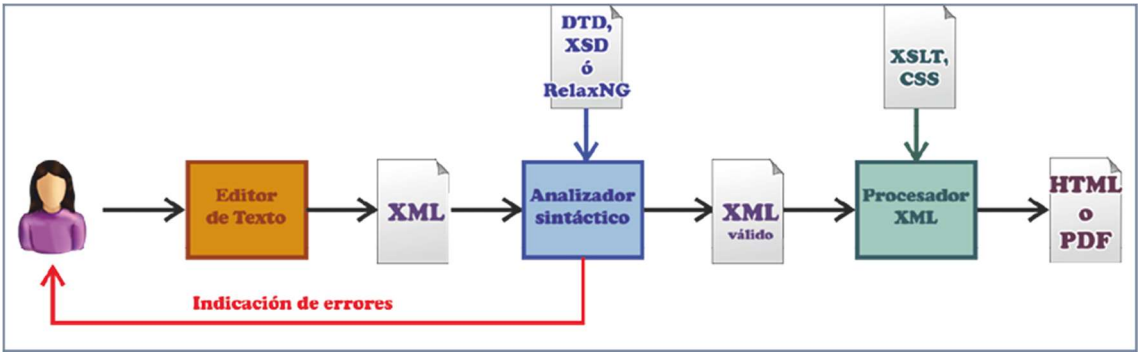

Ilustración 21. Proceso completo de funcionamiento productivo de un XML

2 funcionamiento de XML

2.1 cuestiones básicas

Los documentos XML, en definitiva, son documentos de texto cuyos metadatos se indican mediante etiquetas (marcas) que permiten clasificar el texto a través de su significado.

Las etiquetas en XML se deciden a voluntad, no hay una lista concreta de etiquetas que se pueden utilizar. La idea es que con XML estamos definiendo un tipo de documento cuya forma, significado y estructura dependerán solo de nuestras necesidades.

Ejemplo de XML:

```
<persona>
    <nombre>Jorge</nombre>
```

<apellido>Sánchez**</apellido>**

</persona>

En este código, se define un elemento llamado **persona**, que contiene a dos elementos más: *nombre* y *apellido*, cuyos valores son *Jorge* y *Sánchez* respectivamente.

Algunos detalles sobre el funcionamiento son:

- Un elemento se define con un nombre entre los símbolos **<** y **>**. Eso forma una etiqueta de apertura.
- El elemento termina con la etiqueta de cierre, la cual se indica con el símbolo **/** antes del nombre de la etiqueta y todo ello nuevamente encerrado entre los símbolos **<** y **>**.
- El contenido entre la apertura y el cierre queda afectado por esas etiquetas. De hecho, se tratará del contenido del elemento.
- XML **distingue entre mayúsculas y minúsculas**, siendo buena práctica escribir las etiquetas en minúsculas.
- Se pueden espaciar y tabular las etiquetas a voluntad. Es buena práctica que un elemento interno a otro aparezca en el código con una sangría mayor. Por eso en el código anterior, *nombre*, que está dentro de *persona*, aparece con una tabulación en su lado izquierdo.
- Los comentarios en el código se inician con los símbolos **<!--** y terminan con **-->**. El texto dentro de esos símbolos no se tienen en cuenta como parte del documento XML y se usa para hacer anotaciones.
- Según la **W3C** (organismo de estandarización de XML), el texto en un documento XML debe de estar **codificado en Unicode** (normalmente UTF-8)

2.2 estructura de los documentos XML

Los documentos XML se dividen en:

- **Prólogo**. Se trata de la primera zona del documento y sirve para indicar qué tipo de documento es. Los apartados del prólogo suelen ser:

    - Declaración del documento, que permite indicar el tipo de documento XML, es decir su versión y la forma en la que está codificado.
    - Instrucciones para el procesado del documento
    - Comentarios
    - Ruta hacia el documento **DTD**, **XSD** o **Relax NG** que sirve para validar el documento XML actual.
    - Indicación de otros documentos que afectan al actual. Como por ejemplo los documentos XSLT que sirven para dar formato al documento XML.
- **Elemento raíz**. Tras el prólogo tenemos el contenido en sí del documento. Todo ese contenido debe de estar incluido dentro de un mismo elemento llamado elemento raíz. Se trata de un **elemento obligatorio que se abre tras el prólogo y se debe cerrar justo al final del documento**. Es decir, no puede haber otro elemento después del elemento raíz.
- Dentro de un elemento (sea raíz o no), puede haber:
    - Más elementos
    - Atributos
    - Texto normal
    - Entidades
    - Comentarios

2.3 reglas para los nombres

En XML los elementos y atributos tienen un nombre (más correctamente llamado **identificador**), el cual debe cumplir estas reglas

- En XML se distingue entre mayúsculas y minúsculas, por lo que si un elemento se abrió usando minúsculas, se debe cerrar de la misma forma. Usar minúsculas, en todo caso, es lo recomendable.
- El **nombre debe comenzar por una letra**. Después puede tener más letras, números o el signo de subrayado o guion bajo.

- No pueden empezar con la palabra XML ni en mayúsculas ni en minúsculas ni en ninguna combinación de mayúsculas ni minúsculas. La razón es que los nombres que empiezan ´con el texto “XML” se entiende que son parte de un lenguaje oficial relacionado con XML.

2.4 elementos del prólogo

declaración XML

Se trata de la primera línea de un documento XML y sirve para clarificar el tipo de documento XML. En realidad, es opcional, pero es muy recomendable. Es:

<?xml version=”1.0” encoding=”UTF-8”?>

Indica la versión XML del documento (se suele usar 1.0) y la codificación (utf-8 es la habitual y recomendable por la W3C).

instrucciones de procesamiento

Un documento XML puede incluir instrucciones de este tipo para indicar un documento para validar el XML, darle formato, … u otras funciones. Por ejemplo:

<?xml-stylesheet type=”text/xsl” href=”stylesheet.xsl”?>

Esta instrucción **asocia un documento xsl** al documento XML para poder darle un formato de salida (para especificar la forma en la que los datos se muestran por pantalla, por ejemplo).

2.5 comentarios

Como se ha indicado antes comienzan con el símbolo *<!--* y terminan con -->. Dentro puede haber cualquier texto que se utiliza con fines explicativos o de documentación del código.

Los comentarios **no pueden meterse dentro de la etiqueta** de un elemento, ni tampoco puede contener etiquetas ni de apertura ni de cierre.

2.6 elementos

Son la base del documento XML. Sirven para dar significado al texto o a otros elementos o también para definir relaciones entre distintos elementos y datos.

Hay una confusión entre lo que es un elemento y lo que es una etiqueta. En este caso, por ejemplo:

<nombre>Jorge**</nombre>**

- *<nombre>* Es una etiqueta de apertura
- *</nombre>* Es una etiqueta de cierre
- *<nombre>***Jorge***</nombre>* Es un elemento (el elemento nombre)
- *Jorge* es el contenido del elemento

El contenido de un elemento puede **contener simplemente texto**:

<descripción>

Producto con precio rebajado debido a su escasa demanda

</descripción>

O puede contener otros elementos (o ambas cosas). En este el elemento persona consta de un elemento nombre y otro apellido.

<persona>

<nombre>Jorge**</nombre>**

<apellido>Sánchez**</apellido>**

</persona>

Los elementos se deben abrir y cerrar con la etiqueta que sirve para definir el elemento. **Siempre se debe cerrar el último elemento que se abrió.** Es decir, es un error:

<persona><nombre>Jorge**</nombre>**

<apellido>Sánchez**</persona></apellido>**

elementos vacíos

Los elementos vacíos **no tienen contenido.** Ejemplo:

<casado></casado>

Los elementos vacíos **pueden indicar el cierre en la propia etiqueta de apertura**:

<casado />

2.7 atributos

Se definen **dentro de las etiquetas de apertura de los elementos.** Sirven para indicar propiedades de los elementos a los que se les asigna un determinado valor.

Para ello **se indica el nombre del atributo seguido del signo = y del valor (entre comillas) que se le da al atributo.** Ejemplo:

<persona complejidad=”alta**”>**

<nombre>Jorge**</nombre>**

<apellido>Sánchez**</apellido>**

</persona>

Un elemento puede contener varios atributos:

<persona privacidad=”alta**” tipo=”**autor**”>**

<nombre>Jorge**</nombre>**

<apellido>Sánchez**</apellido>**

</persona>

2.8 texto

El texto como se comentó antes está siempre entre una etiqueta de apertura y una de cierre. Eso significa que todo texto es parte de un elemento XML.

Se puede escribir cualquier carácter Unicode en el texto, pero no es válido utilizar caracteres que podrían dar lugar a confusión como los signos separadores **<** y **>** por ejemplo.

2.9 CDATA

Existe la posibilidad de marcar texto para que sea **procesado literalmente como texto** y no como sintaxis de XML. Eso permite indicar texto que contiene caracteres que serían problemáticos.

Para ello, el texto se coloca dentro de un elemento CDATA. Este elemento funciona así

<! [CDATA [texto no procesable… **]]>**

Esto permite utilizar los caracteres < y >, por ejemplo, dentro de un texto literal y no serán considerados como separadores de etiquetas.

Ejemplo:

```
<?xml version=”1.0”?>
<documento>
    <título>Prueba</título>
    <ejemplo>
        <![CDATA[
            En HTML la negrita se escribe: <strong>
        ]]>
    </ejemplo>
```

</documento>

En el ejemplo, los símbolos < y > no se toman como una etiqueta XML, sino como texto normal.

Otro uso de CDATA es **colocar dentro de este elemento código de lenguajes de scripts** como Javascript para que no sean interpretados como parte de XML.

2.10 entidades

Las entidades **representan caracteres individuales.** Se utilizan para poder representar caracteres especiales o bien caracteres inexistentes en el teclado habitual.

Se trata de códigos que empiezan con el signo & al que sigue el nombre de la entidad o el número Unicode del carácter que deseamos representar.

En XML hay definidas cinco entidades:

- **>** Símbolo **>** (mayor que)
- **&lt**; Símbolo **<** (menor que)
- **&** Símbolo **&**
- **"** Símbolo **“** (comillas)
- **'** Símbolo **‘** (apóstrofe)

Es **obligatoria usar esas cinco entidades en lugar de los símbolos que representan.**

También podemos representar caracteres mediante entidades con número. De modo que el *ñ* representa a la letra *ñ* (suponiendo que codificamos en Unicode, que es lo habitual). El número puede ser hexadecimal por ejemplo para la eñe de nuevo, sería *ñ*

3 jerarquía XML

Los elementos de un documento XML establecen una jerarquía que **estructura el contenido del mismo.** Esa jerarquía se puede **representar en forma de árbol.**

Así por ejemplo el archivo XML:

```
<?xml version=”1.0” ?>
<documento>
      <título>Apuntes de XML</título>
      <autor>Jorge Sánchez</autor>
      <fecha_pub>
            <día>18</día>
            <mes>Enero</mes>
            <año>2019</año>
      </fecha_pub>
</documento>
```

Este documento se puede representar de esta forma gráfica:

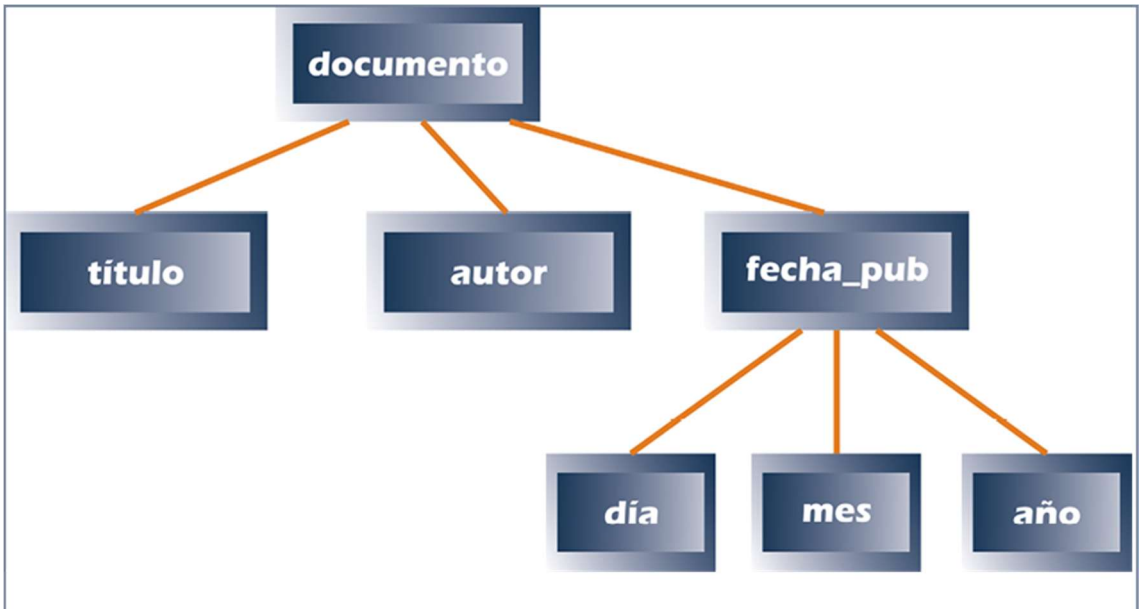

Ilustración 22. Representación gráfica que expresa la jerarquía del documento XML

4 XML bien formado

Se habla de XML *bien formado* (**well formed)** cuando el documento cumple reglas estrictas, las cuales están pensadas para facilitar su legibilidad y mantenimiento. Además, permiten cumplir los objetivos que se plantearon los creadores del lenguaje XML.

Debemos tomar las reglas para producir XML bien formado como reglas obligadas. Los documentos XML que no están bien formados, se les considera **incorrectos** y así serán marcados por los **analizadores (*parsers*) de XML** que se encargan de comprobar la sintaxis de los documentos XML.

Un documento XML bien formado es un **documento analizable.** Sin embargo, eso no significa que sea **válido**. Los documentos válidos **cumplen la sintaxis de un documento validador**, idea que se explicará en la siguiente unidad.

En los siguientes apartados se explica las reglas que deben de cumplir los documentos bien formados.

4.1 reglas generales

- Dentro del texto común no se pueden utilizar los símbolos de mayor (**>**), menor (**<**), ampersand (**&**) ni las comillas simples o dobles. Se deben de utilizar entidades o deben estar incluidos en una sección CDATA.
- En principio, en el texto normal, los símbolos de separación de caracteres como **espacios en blanco, tabuladores y saltos de línea, no se tienen en cuenta, se ignoran**. Pero sí es posible que sean significativos en algunos elementos (si es así se indica en su documento de validación). No tener en cuenta los espacios, significa que, en el texto contenido en un elemento, si dejamos 25 espacios entre dos palabras, se entenderá **que hemos dejado uno solo**.
- En cualquier caso, todos los caracteres escritos en el documento XML forman parte del mismo. Será una cuestión

posterior si se tienen en cuenta o no para presentar los datos del documento XML en pantalla o impresión.

4.2 reglas para los elementos

- Siempre se debe de **cerrar antes el último elemento abierto**:

Es decir, este código **no** sería XML bien formado:

<fecha><dia>13**</dia><mes>**Mayo**</fecha></mes>**

Lo correcto sería:

<fecha><dia>13**</dia><mes>**Mayo**</mes></fecha>**

Todos los elementos poseen etiquetas de apertura y de cierre. Es decir, toda etiqueta que se abra se debe de cerrar. Si la etiqueta no tiene contenido, **se debe cerrar en la propia etiqueta**. Por ejemplo:

**
**

- Todos los documentos XML deben de tener **un único elemento raíz que contendrá al resto de elementos.**
- Los nombres de los elementos comienzan con letras y pueden ir seguidos de letras, números, guiones o de puntos (aunque realmente **los puntos se deberían emplear cuando hay varios espacios de nombres**).

4.3 reglas para los atributos

- Los nombres de atributos siguen las mismas normas que los nombres de los elementos.
- Los atributos **sólo se pueden colocar en etiquetas de apertura** y nunca en las de cierre.
- Los valores de los atributos deben ir **entrecomillados** (sin importar si se usan comillas simples o dobles). Ejemplos

<nombre sexo=”Hombre”>Antonio</nombre>

<nombre sexo=’Mujer’>Sara</nombre>

<nombre sexo=’Mujer”>Eva</nombre> <!-- error -->

- La última línea es incorrecta porque no se pueden combinar ambas comillas.
- Todos los atributos **deben de tener valor asociado**. Es decir esta etiqueta, válida en HTML, no sería válida en XML:

  <hr noshade>

- Debería ser, por ejemplo, así:

  <hr noshade=”noshade”>

4.4 comprobadores de sintaxis de documentos XML

Para comprobar si un documento XML está bien formado, necesitamos un software de validación. Posibilidades:

- **Validadores en línea.** Se trata de páginas web a las que las indicamos la dirección de Internet (es decir, la URL) en la que se encuentra el documento XML a comprobar, o bien copiamos el código XML a comprobar. Algunas son:
  - **http://www.w3schools.com/xml/xml_validator.asp** El validador de la organización w3schools que tiene documentación extensa sobre diversas tecnologías relacionadas con la web.
  - **www.xmlvalidation.com** Otro validador muy popular.
- **Validadores offline**. Es decir, validadores que son programas a descargar. Sin duda el más popular hoy día es **xmllint**, basado en la librería **libxml2** creada para el lenguaje C. Para utilizarlo:

[1]En el caso de Windows, hay que descargar la librería **libxml2** de la dirección ftp://ftp.zlatkovic.com/libxml/ y desde ahí descargar los zip necesarios, elegir el archivo ZIP de la librería con la última versión (en el momento de escribir estas líneas es el archivo *libxml2-2.7.8.zip*)

[2]Descomprimir el archivo en el sitio deseado y asignar en la que viene dicho validador (lo normal es descargar un archivo ZIP en el caso de Windows y descomprimirlo).

- **Validadores integrados.** En la mayoría de editores completos existen validadores integrados en el sistema que incluso nos marcan los errores a medida que escribamos el código.

5 espacios de nombres

5.1 el problema de los nombres de elementos

Puede ocurrir que cuando se manejan diversos documentos XML que varios de ellos utilicen las mismas etiquetas, aunque el contexto sería distinto. De ser así, tendríamos un problema si manejamos ambos documentos con el mismo software, ya que el analizador, no sabría cómo manejar ambas etiquetas iguales.

Los espacios de nombres (*namespacing* en inglés) evitan el problema indicando en cada elemento una indicación que sirve para indicar el contexto de cada etiqueta y así diferenciar las que son iguales.

Ejemplo:

```
<?xml version=”1.0” encoding=”UTF-8”?>
<document>
<title>TITULO DEL DOCUMENTO</title>
    <content>
      <html>
          <head>
              <title>Titulo HTML</title>
          </head>
          <body>
              Texto del documento
          </body>
      </html>
```

```
    </content>
    <author>Jorge</author>
</document>
```

En el ejemplo anterior se usan etiquetas en inglés para el documento (algo muy habitual en el mundo empresarial) y eso hace que **la etiqueta *title* se repita en contextos distintos**; la primera aparición **es para poner un título genérico al documento** (y es una etiqueta de la empresa en cuestión) y la segunda **se corresponde a la etiqueta *title* del lenguaje HTML.**

5.2 diferenciar elementos

La solución para diferenciar es **anteponer al nombre de la etiqueta un nombre distintivo separado por un punto.**

Por ejemplo:

```
<?xml version=”1.0” encoding=”UTF-8”?>
<document>
<jorge.title>TITULO DEL DOCUMENTO</jorge.title>
        <content>
        <html>
            <head>
                <html.title>Titulo HTML</html.title>
            </head>
            <body>
                Texto del documento
            </body>
        </html>
    </content>
    <author>Jorge</author>
</document>
```

Ese prefijo diferenciador e**s el espacio de nombres al que pertenece la etiqueta**, pero usado así tendríamos el problema de que con un sufijo tan corto, se podría repetir.

Por ello una solución más completa, **es indicar la URL de la entidad responsable de la etiqueta**:

```
<?xml version=”1.0” encoding=”UTF-8”?>
<www.jorgesanchez.net.document>
    <www.jorgesanchez.net.title>
        TITULO DEL DOCUMENTO
    </www.jorgesanchez.net.title>
    <www.jorgesanchez.net.content>
        <www.w3c.org.html>
            <www.w3c.org.head>
                <www.w3c.org.title>
                    Titulo HTML
                </www.w3c.org.title>
            </www.w3c.org.head>
            <www.w3c.org.body>
                Texto del documento
            </www.w3c.org.body>
        </www.w3c.org.html>
    </www.jorgesanchez.net.content>
    <www.jorgesanchez.net.author>
        Jorge
    </www.jorgesanchez.net.author>
</document>
```

Como se puede observar, **el documento ahora es muy poco legible**.

5.3 uso de espacios de nombres

La complejidad del código anterior nos traslada el por qué necesitamos de otra solución más simple pero igual de definitiva.

Los espacios de nombres permiten **indicar un prefijo a las etiquetas.** A ese prefijo **se le asocia una URL.** De este modo el nombre es único, pero, a la vez, el código no es tan complejo. El **prefijo se separa con dos puntos del nombre.**

La URL que se asocia es en realidad una **URI** (*Universal Resource Identifier*) un identificador único de recurso, de modo que la raíz de la **URI** es el dominio universal (Internet) de la empresa y a él se añade la ruta al recurso.

Así si hemos definido documentos XML cuyo elemento raíz es *document*, la URL relacionada con el espacio de nombres del documento podría ser:

www.jorgesanchez.net/document.

5.4 atributo xmlns

Todas las etiquetas en XML pueden hacer uso del atributo **xmlns (*xml namespacing*)** que **permite asignar un espacio de nombres a un prefijo en el documento** dentro del elemento en el que se usa el espacio de nombres. Ejemplo:

```
<?xml version=”1.0” encoding=”UTF-8”?>
<document
   xmlns:jorge=”http.//www.jorgesanchez.net/document”
   xmlns:html=”htp://www.w3c.org/html”>
    <jorge:title>
        TITULO DEL DOCUMENTO
    </jorge:title>
    <jorge:content>
        <html:html>
            <html:head>
               <html:title>Titulo HTML</html:title>
```

```
            </html:head>
            <html:body>
                Texto del documento

            </html:body>
        </html:html>

    </jorge:content>
    <jorge:author>
        Jorge
    </jorge:author>
</document>
```

En el ejemplo se usa el **prefijo *jorge* para indicar etiquetas del espacio de nombres *www.jorgesanchez.net/document* y *html* para el espacio de nombres oficial de HTML.**

5.5 espacios de nombre por defecto

En el caso de que las etiquetas, mayoritariamente, en un documento pertenezcan a un mismo espacio de nombres, lo lógico es **indicar el espacio de nombres por defecto.** Eso se hace sin indicar prefijo en el atributo **xmlns**. Ejemplo:

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

   xmlns="http.//www.jorgesanchez.net/document"

   xmlns:html="htp://www.w3c.org/html">

    <title>
        TITULO DEL DOCUMENTO

    </title>
    <content>
        <html:html>
            <html:head>
               <html:title>Titulo HTML</html:title>

            </html:head>
            <html:body>
                Texto del documento
```

```
            </html:body>
        </html:html>

    </content>
    <author>
        Jorge
    </author>
</document>
```

Las etiquetas sin prefijo se entienden que pertenecen al espacio de nombres *www.jorgesanchez.net/document*, para las del otro espacio se usa el prefijo.

5.6 uso de espacios de nombres en etiquetas interiores

El atributo **xmlns** no tiene por qué utilizarse en el elemento raíz, se puede **posponer su declaración en el primer elemento que pertenezca al espacio de nombres deseado.** Por ejemplo:

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

<document

   xmlns=”http.//www.jorgesanchez.net/document”>
    <title>
           TITULO DEL DOCUMENTO

    </title>

    <content>
        <html:html xmlns:html=”htp://www.w3c.org/html”>

            <html:head>
                <html:title>
                    Titulo HTML

                 </html:title>
            </html:head>
            <html:body>
               Texto del documento

            </html:body>
        </html:html>
```

```
    </content>
    <author>
        Jorge
    </author>
</document>
```

5.7 uso de varios espacios por defecto en etiquetas interiores

Un documento **puede declarar espacios por defecto en etiquetas interiores** lo que permite aún más versatilidad en los documentos.

Ejemplo:

```
<?xml version=”1.0” encoding=”UTF-8”?>
<document xmlns=”http.//www.jorgesanchez.net/document”>
<!-- comienza el espacio de nombres de jorgesanchez.net -->
    <title>
        TITULO DEL DOCUMENTO
    </title>
    <content>
        <html xmlns=”htp://www.w3c.org/html”>
     <!-- desde aquí el espacio ahora es el de html -->
            <head>
                <title>
                    Titulo HTML
                </title>
            </head>
            <body>
                Texto del documento
            </body>
        </html>
<!-- fin del espacio html, regresa el espacio jorgesanchez.net -->
    </content>
    <author>
        Jorge
    </author>
</document>
```

5.8 cancelar espacios por defecto

Si se usa el atributo *xmlns=””*, entonces se está indicando (en el interior del elemento en el que se use) **que ese elemento y sus hijos no usan ningún espacio de nombres.**