EXTENSIBLE MARKUP LANGUAGE: XML

INTRODUCCIÓN

El lenguaje HTML, utilizado para crear las páginas que componen la Web y XML tienen muchos puntos en común.

Pondremos frente a frente dos documentos, uno escrito en HTML y otro en XML. De esa forma seremos capaces de compararlos y comprender mejor las carencias de HTML, un lenguaje muy útil para describir la presentación de los datos, pero no para los datos en sí mismos.

El siguiente es el código HTML de una página muy sencilla que muestra detalles sobre una película:

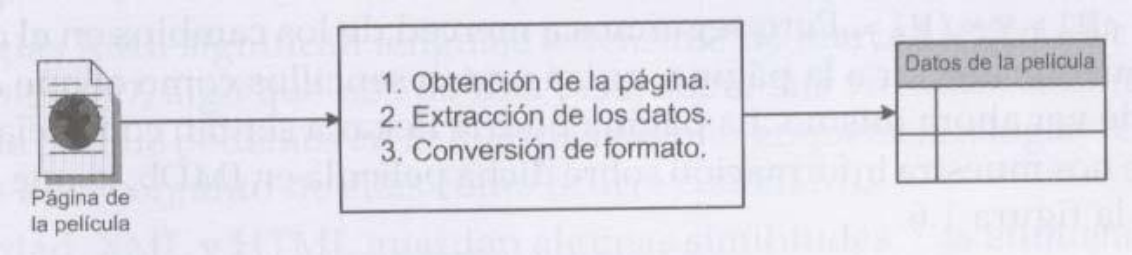
```
<HTML>
    <HEAD>
      <TITLE> Brazil </TITLE>
    </HEAD>
    <BODY>
      <H1>Brazil</H1>
      <B>Director:</B>Terry Gilliam<BR>
      <B>Protagonista:</b> Jonathan Pryce<BR>
    </BODY>
</HTML>
```

La página que acabamos de ver muestra el nombre del director y el del protagonista precedidos por una etiqueta que indica qué función cumplieron en la película.

Sin embargo, a pesar de que en ningún lugar se indica que Brazil sea el título de la película, todos hemos entendido que así es, sin necesidad de ninguna etiqueta. Ahora bien, ¿habría sido capaz de hacer lo mismo un proceso automático?

Supongamos que tenemos a nuestra disposición infinidad de páginas como la que hemos visto anteriormente, con detalles sobre multitud de películas, y que queremos crear un programa que se encargue de extraer detalles de cada una de ellas. El propósito de tal acción podría ser la creación de una compilación de toda esa información en otro formato, como un archivo de texto, o quizá la creación de una base de datos.

Puede que, en principio, nos parezca una tarea muy sencilla. En esta figura podemos ver un diagrama del funcionamiento de nuestro hipotético programa.



Tras obtener la página, hemos de extraer los datos que nos interesan. Por ejemplo, para localizar el título sólo hay que encontrar el texto que se encuentra entre las etiquetas <H1> y </H1>. Podemos extraer el resto de datos de forma igualmente trivial. De esta forma, cada vez que queramos pasar a nuestro formato particular los datos de una película, sólo tenemos que utilizar el programa que hemos creado.

Pero aquí viene el problema. Como ya sabrá, el aspecto de las páginas de la Web no es inmutable. ¿Qué ocurre si el diseñador de las hipotéticas páginas con información sobre películas decide que no quiere utilizar la etiqueta <H1> para resaltar el título y pasa a utilizar otra etiqueta, <H2>?

¿No sería fantástico que, de alguna manera, los datos de la película estuviesen siempre en el mismo sitio, aunque el aspecto de la página cambiase? De esa forma, el programa que extrae la información de las páginas podría seguir funcionando sin importar los cambios que los diseñadores decidiesen introducir en las mismas. Pues éste es uno de los objetivos con los que nació XML, aunque no se trata del único.

XML

Las siglas XML significan lenguaje extensible de marcas (eXtensible Mar-kup Language). XML es extensible porque podemos crear nuestras propias etiquetas, en lugar de estar atados a un conjunto de ellas como ocurre con HTML.

El siguiente código pertenece a un documento XML que intenta ofrecer los mismos datos que la página HTML que vimos anteriormente:

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

<pelicula>

    Brazil

   <director>Terry Gilliam</director>

   <actores>

      <actor>Jonathan Pryce</actor>

   </actores>

</pelicula>
```

Un documento XML se compone de dos partes. La primera es el prólogo. En ella se puede anunciar, entre otras cosas, que se trata de un documento XML y que se ajusta a una versión específica de la norma, en este caso la 1.0. La segunda parte es el cuerpo del documento. El inicio del cuerpo lo determina una etiqueta, en este caso <pelicula>, y su final la etiqueta de cierre correspondiente, es decir, </pelicula>.

Dentro del cuerpo del documento se pueden encontrar tantas etiquetas de apertura y cierre como se desee, siempre que estén correctamente anidadas. Esto quiere decir que si a continuación de una etiqueta de apertura se escribe otra etiqueta, la de cierre de la segunda debe aparecer antes que la de cierre de la primera.

Aplicando esto al documento XML que nos ocupa, como a continuación de la etiqueta <actores> aparece la etiqueta <actor>, hasta que esta última no se cierre no

se puede cerrar la primera. El siguiente fragmento sería, por tanto, incorrecto:

...

<actores>

<actor>Jonathan Pryce

</actores>

</actor>

...

Se trata de ser ordenado y seguir una lógica, algo beneficioso si se quiere utilizar XML de forma conjunta con un programa. Sin esta simple regla, no sería posible escribir un programa que de forma simple y eficiente pudiese leer los documentos XML y extraer información de ellos. Podríamos incluir elementos vacíos, es decir, que no contengan a otros elementos entre sus etiquetas de apertura y cierre. Por ejemplo, si no conociésemos ninguno de los intérpretes de la película podríamos incluir la etiqueta actores de la siguiente forma:

...

<actores></actores>

...

Aunque la siguiente forma, abreviada, también sería válida:

...

<actores/>

...

Ver una página XML en algunos clientes Web nos proporciona tres claras ventajas frente a verla en un editor de textos como el Bloc de notas de Windows:

1. Resalte de sintaxis: Mientras que en un editor de textos sencillo el texto que componga el documento XML aparecerá todo del mismo color, Internet Explorer, Firefox o Mozilla suelen mostrar lo que se conoce como resalte de sintaxis. El prólogo se muestra en un color, las etiquetas de apertura y cierre en otro y el texto envuelto por las etiquetas en otro. Esto ayuda a diferenciar unas partes del documento de otras de un vistazo.

2. Expansión/contracción de elementos: El puntero del ratón en lugar de ser una flecha, se ha convierte en una mano cuando se puede realizar una acción sobre el elemento en el que se encuentra. Si hace clic sobre cualquier elemento que presente un guión a su izquierda, ese elemento se contraerá, ocultando los elementos que aparezcan a continuación de él y antes de la etiqueta de cierre correspondiente.

Cuando se hace clic sobre uno de estos elementos: el elemento se contrae y el guión cambia al símbolo de suma (+ ▼). Si hace clic sobre él se volverá a expandir.

3. Validación del documento: Si el documento XML que intentamos ver está mal construido, el cliente Web nos avisará de ello.

FILOSOFÍA DE XML

La **filosofía** central de XML **reside en la división del documento en sus tres componentes principales:**

El contenido: la información del documento.

La estructura: el tipo y la organización de los elementos componentes del documento.

La presentación: la manera en que la información es presentada al lector.

Los objetivos planteados por el grupo de desarrollo del XML fueron diez puntos [Young M. 2000]:

1. **XML debe ser directamente utilizable sobre Internet.**
2. **XML debe soportar una amplia variedad de aplicaciones.**
3. **XML debe ser compatible con SGML.**
4. **Debe ser fácil la escritura de programas que procesen documentos XML.**
5. **El número de características opcionales en XML debe ser absolutamente mínimo, idealmente cero.**
6. **Los documentos XML deben ser legibles por los usuarios de este lenguaje y razonablemente claros.**
7. **El diseño de XML debe ser formal, conciso y preparado rápidamente.**
8. **XML debería ser simple pero perfectamente formalizado.**
9. **Los documentos XML deben ser fáciles de crear.**
10. **La brevedad en las marcas XML es de mínima importancia.**

VENTAJAS DE XML

- **Comunicación de datos**. Si la información se transfiere en XML, cualquier aplicación podría escribir un documento de "texto plano" (esto es, texto en el cual todos los caracteres se representan visualmente, sin existir caracteres no visibles, exceptuando los de salto de línea, tabulados o espacio) con los datos que estaba manejando en formato XML y otra aplicación recibir esta información y trabajar con ella.
- **Migración de datos**. Si tenemos que mover los datos de una base de datos a otra sería muy sencillo si las dos trabajasen en formato XML.
- **Aplicaciones web**. Hasta ahora cada navegador interpreta la información a su manera y los programadores del web tienen que modificar la página en función del navegador del usuario. Con XML tenemos una sola aplicación que maneja los datos y para cada navegador o soporte podremos tener una hoja de estilo o similar para aplicarle el estilo adecuado. Si mañana nuestra aplicación debe correr en WAP solo tenemos que crear una nueva hoja de estilo o similar.

DESARROLLO DE APLICACIONES WEB CON XML

Se pueden establece cuatro tipos de aplicaciones que impulsarán el desarrollo del XML:

- Aplicaciones que exijan que el cliente Web medie entre dos o más Bases de Datos. Se hará posible la integración de bases de datos distribuidas en los navegadores que admitan XML, pudiéndose modificar el contenido y la estructura de esta.

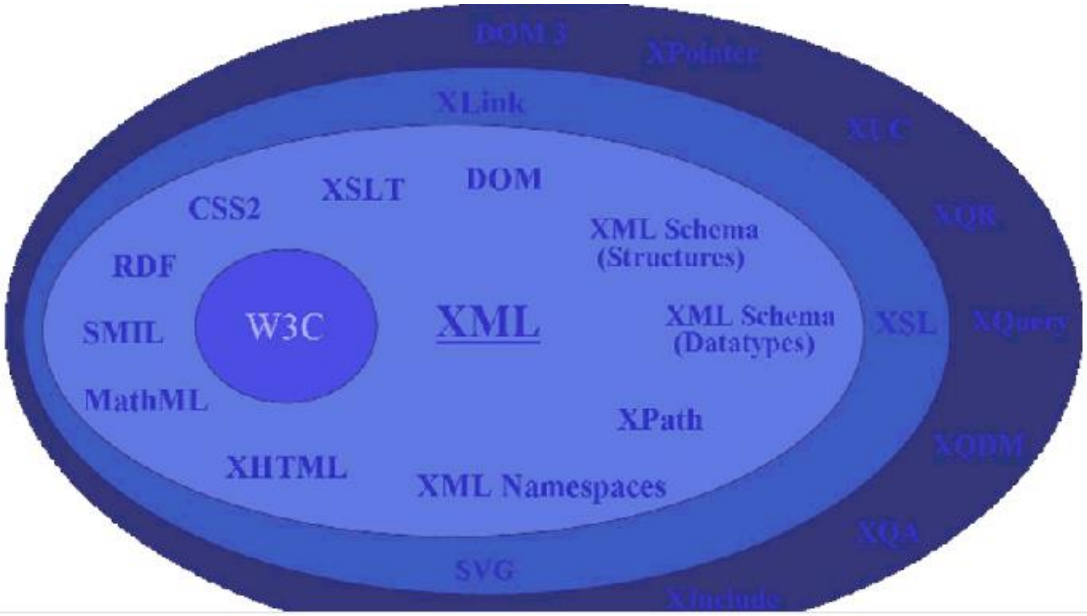
- Aplicaciones que intentan transferir una parte significativa de la carga del proceso del servidor al cliente Web. Esta carga hará que muchas de las funciones de modificación puedan desarrollarse desde el mismo navegador Web del cliente. El lado más negativo es que se necesitará mayor ancho de banda y mayor potencia del procesador del equipo para poder soportar esta arquitectura de tres capas.

- Aplicaciones que precisen que el cliente Web presente diferentes versiones de los mismos datos a diferentes usuarios.

- Aplicaciones en las que agentes Web inteligentes intentan adaptar la búsqueda de información a las necesidades de los usuarios individuales. Habrá una interacción entre la información requerida y las preferencias del usuario de la aplicación. Con el XML vendrá una segunda generación de aplicaciones con una mayor precisión de la búsqueda.

Su objetivo consiste en crear una serie de tecnologías que trabajen juntas, originando un conjunto extensible de controles de formato de documentos que permita la reutilización de definiciones de documento para cualquier combinación de documentos y necesidades.

La siguiente imagen nos muestra la estructura de componentes/recomendaciones que presenta XML:



El conjunto interior contiene las recomendaciones actuales, el intermedio las tecnologías candidatas a recomendación y el exterior son los documentos de trabajo actuales (*working draft*). Algunas de las tecnologías más importantes son:

- **XML (versión 1.0):** Identifica los requisitos de un documento XML bien formado, así como el origen y los objetivos del XML. http://www.w3.org/XML]
- **DTD:** Una definición de tipo de documento (*Document Type Definition*) contiene las reglas por las que es posible validar la información de un documento XML.
- **XLink** (*XML Linking Language*): anteriormente conocido como **XLL**, define la forma en que los documentos XML deben enlazarse entre sí. [http://www.w3.org/TR/xlink/]
- **XPointer:** describe cómo debe apuntarse a un lugar específico dentro de un documento. [http://www.w3.org/TR/xptr/]
- **XPath:** proporciona un medio estándar de hacer referencia a direcciones y ubicaciones dentro de documentos XML así como de manipulación y comparación de cadenas, números y valores booleanos. [http://www.w3.org/TR/xpath/]
- **CSS** (*Cascading Style Sheets*): define un modelo de formato visual con controles avanzados para medios paginados e impresos. [http://www.w3.org/Style/CSS/]

- **XSL** (*eXtensible Stylesheet Language*): es un lenguaje diseñado específicamente para la creación de hojas de estilo con páginas XML. Se compone de: Un conjunto de propiedades y controles de formato para presentar la información de los documentos XML. Y **XSLT** (*XSL Transformations*)que define la sintaxis y la semántica que se utiliza para convertir documentos XML.
- **XML Schemas**: permiten incorporar más funcionalidad en los documentos XML de la que proporcionan actualmente las DTD.
- XQL (XML Query Language): proporciona un modelo de datos para su utilización con documentos XML, así como un lenguaje de consulta y un conjunto de operadores completos para el modelo de datos.
- **XHTML** (*eXtensible Hipertext Markup Language*): es una reformulación de la especificación 4.0 de HTML existente en forma de módulos que siguen las reglas impuestas por la especificación XML.

…… Y MAS

SOFTWARE XML

Los documentos XML no se diseñan ni se crean para permanecer aislados, su objetivo es el de facilitar información a una aplicación principal a través de un conjunto de herramientas software que permiten la manipulación, creación, edición, presentación y análisis sintáctico de los documentos. Básicamente podemos dividir el software asociado a XML en tres grandes grupos: editores, analizadores e interfaces de programación.

- Los **editores** nos facilitan la creación de los documentos XML. Se puede utilizar un editor tan sencillo como el bloc de notas de Windows o algo un poco más sofisticado, como el navegador y editor Amaya del W3C.
- Los **analizadores** nos proporcionan una herramienta capaz de entender la sintaxis de un documento XML y realizar comprobaciones sobre su estructura. Existen dos tipos básicos, los **no validadores** que solamente verifican que el documento XML está **"bien formado"** (tanto IE6 como NN7 integran estos analizadores como parte integral de sus componentes software) y **los validadores** que realizan las **tareas adicionales de verificación del contenido de un documento XML respecto a una DTD.**
- Las **interfaces de programación de aplicaciones** (API, *Application Programming Interface*) proporcionan métodos para poder manipular documentos XML desde lenguajes de programación o de script. Dentro de este grupo cabe destacar el **DOM** (*Document Object Model*).

  Existen validadores XML en línea, como XML Validation (http://www.xmlvalidation.com)

Algunas de las herramientas software:

- Editores: XMetal, XML Spy, XML Authority…
- Analizadores sintácticos no validadores: Expat, Lark, XP, TclXML, XML Testbed, RUWF…
- Analizadores sintácticos validadores: LT XML, XML4Java, MS XML, MXL Checker…

XML: ESTRUCTURA DE DOCUMENTOS

1. ¿QUÉ ES XML?

El lenguaje de marcado XML es un estándar de Internet susceptible de ser utilizado desde o por distintos tipos de programas o aplicaciones ejecutándose en distintos tipos de plataformas. Ha sido creado por la organización W3C en octubre de 1996

XML es, básicamente, un lenguaje derivado de uno anterior denominado SGML, que se orienta a la creación de especificaciones o modelos de documentos ("meta-lenguajes") o datos con el fin de poder estandarizar el intercambio de datos entre los componentes de una aplicación, o entre distintas aplicaciones o sistemas.

Un documento XML se basa en un lenguaje de marcas similar al que presenta una página HTML, en el sentido en que ambos se basan en documentos de texto plano (que pueden ser editados por cualquier editor de textos, desde el Bloc de Notas de Windows al vi de Unix), basados en un lenguaje de marcas (que define una serie de etiquetas como <FONT> o <TABLE>), y que son fácilmente entendibles a primera vista. El XML busca, ante todo, separar los datos del formato. A través de la creación de las hojas de estilo, HTML ha experimentado una notable evolución. Sin embargo, XML separa los datos de su presentación de manera más eficaz a que su principal objetivo es el de almacenar la información y no el de mostrarla.

XML es un lenguaje básico. Por tanto, muy probablemente nunca tengamos que estar preocupados de si estamos utilizando la versión correcta o "actualizada" del lenguaje

Comparativa

La siguiente tabla recoge las diferencias entre estos tres lenguajes.

Tabla 1.1. Diferencias entre los tres sistemas de lenguajes de estilo.

	HTML	*XML*	*SGML*
Especificación de marcas	Fija	Extensible	Extensible
Aplicación de formato	CSS	XSL / CSS / DSSSL	DSSSL
Búsquedas	Sin sistema propio de búsquedas	XPath y jerarquía de objetos	Motor propio
Complejidad	Baja		
Nivel de diseño y herramientas de autor	Mediana		
Herramientas de ayuda y SDK	Alta		
Herramientas no muy modernas			
Etiquetas de marcado	Fijadas por estándar HTML 4	Sin límite	Sin límite
Hipervínculos	Hipervínculos simples	Sistema propio (XPath, XPointer)	HyTime
Organización interna	Semi-estructurada	Jerárquica (árbol)	Jerárquica (árbol)
Validación de gramática	Sin validación	Validación DTD o XML Schema opcional	Obligatorio DTD

2. REGLAS BÁSICAS PARA LA CONSTRUCCIÓN DE DOCUMENTOS XML

Las reglas de validación que presenta el lenguaje XML y que precisamente le distinguen del resto de lenguajes de marcado sin validación:

- **Primera regla**: Un documento XML contiene uno o más elementos o etiquetas. Las etiquetas XML son marcas que se abren y cierran en el mismo orden de su creación. Esta regla se basa en la meta de poder construir un árbol de documento bien formado
- **Segunda regla**: Todas las etiquetas XML se constituyen por una etiqueta de apertura y una de finalización, que tendrá idéntico nombre que la de apertura, pero comenzando por el carácter barra "\". Por motivos de comodidad, es posible crear etiquetas XML únicas que actúen como apertura y finalización al mismo tiempo. Estas etiquetas deberán acabar con el carácter barra "/".
- **Tercera regla**: Los identificadores de etiqueta son sensibles a la utilización de caracteres en mayúsculas y minúsculas. Por esta razón, tanto la etiqueta de apertura y finalización de un mismo elemento, como todas las etiquetas iguales de un documento deberán guardar esta similitud en lo que a mayúsculas y minúsculas se refiere.

Todo documento XML se basa en un estructura jerárquica (en forma de árbol) en la que siempre se encuentra un primer elemento (raíz o root).

3. ESTRUCTURA DE UN DOCUMENTO

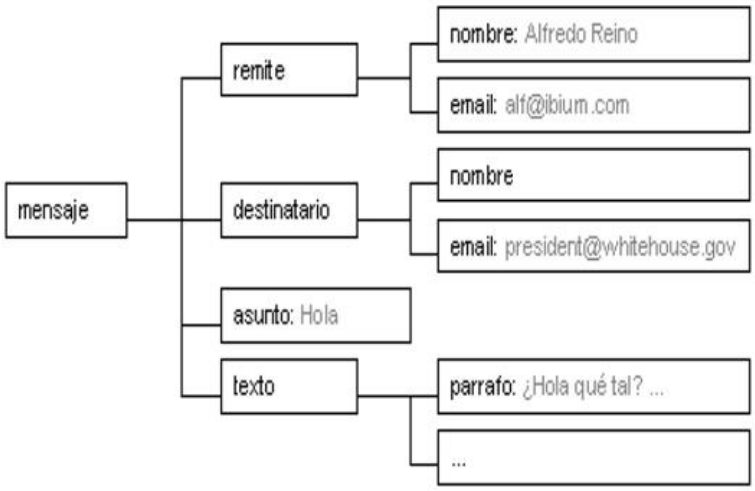
Un documento XML es un archivo de texto con extensión **".xml"** que puede editarse con cualquier editor de texto y que es interpretado por los navegadores Web. Se compone de etiquetas que pueden personalizarse, pero a diferencia del HTML es que puede ser reutilizado.

En el siguiente ejemplo podemos ver la estructura de un documento

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

<mensaje>

        <remite>

                <nombre>Alfredo Reino</nombre>

                <email>alf@ppp.com</email>

        </remite>

        <destinatario>

                <nombre>Obama</nombre>

                <email>president@whitehouse.gov</email>

        </destinatario>

        <asunto>Hola presidente</asunto>

        <texto>

                <parrafo>Hola que tal? desearía visitarte </parrafo>
```

```
        </texto>
</mensaje>
```

Este mismo documento puede ser visto de forma gráfica, para comprender mejor la estructura de un documento XML.



Ejemplo:

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<tarjetavisita>
 <apellido>DEL VALLE</apellido>
 <nombre>Cristobal</nombre>
 <sociedad/>
 <profesion>Doctor</profesion>
 <direccion>
  <numero>19</numero>
  <calle>C/ Florida</calle>
  <codigopostal>18080</codigopostal>
  <poblacion>MADRID</poblacion>
 </direccion>
 <numerotelefono>123456789</numerotelefono>
 <numeromovil/>
```

```
 <numerofax/>
<email/>
</tarjetavisita>
```

Para verificar los documentos XML, simplemente se arrastra el documento.xml a un navegador de Internet.

Un documento XML puede estar formado por:

- **Un prólogo** (es una parte opcional) que añade información sobre el documento.
- **Un ejemplar** que es la parte más importante y que contiene la información del documento, es decir, los datos a los que se les ha añadido el marcado.

A. EL PRÓLOGO

Si *se incluye, el prólogo debe preceder al ejemplar del documento.*

Aunque el prólogo es opcional, su inclusión es muy recomendable ya que facilita un procesado **fiable y robusto** de la información contenida en el ejemplar.
El prólogo puede a su vez dividirse en dos partes:

- **La Declaración XML**, que cumple las siguientes funciones:
  - **Marca el documento como texto XML. (xml)**
  - **Declara cuál es la versión de XML utilizada para elaborar el documento a través de lo que se denomina una declaración de versión. (atributo versión)**
  - **Aporta información sobre la codificación empleada para representar los caracteres mediante una declaración de codificación. (atributo encoding)**
  - **Incluye también una declaración de documento autónomo. (standalone)**

Si está presente, la declaración XML debe ser la primera línea del documento, incluirla en otro lugar supone un error de buena formación que impedirá que el documento sea procesado.
El hecho de que sea opcional tiene como objetivo permitir el procesamiento de documentos HTML y SGML como si fueran XML, si fuera obligatoria éstos (los documentos HTML y SGML) deberían incluir una declaración de versión XML, que obviamente, no tienen.

Un ejemplo:

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

EJEMPLO:

```
<?xml version="1.0"  encoding="ISO-8859-1" standalone="yes" ?>
<directorio>
<entrada>
 <apellido>Guevara</apellido>
 <nombre>Ernesto</nombre>
```

```
</entrada>

</directorio>
```

Hay que tener en cuenta que **estas declaraciones siguen un orden estricto** (el **mostrado en el ejemplo) y no se pueden especificar en otro orden.**
Además, si se especifica la codificación ("encoding") o la declaración de documento autónomo ("standalone") es necesario incluir también la declaración sobre la versión ("version").
Las comillas que delimitan los valores de las distintas declaraciones son obligatorias, aunque pueden ser dobles o simples.

La **versión** permite **indicar la versión** para la que se elaboró el documento (actualmente la 1.0) y permitir que los documentos se adapten a la evolución del estándar.

La **codificación** nos permite indicar **el juego de caracteres utilizado en el documento**, el valor por defecto es UTF-8 (ASCII de 7bits). Esta codificación no admite acentos ni otros caracteres comunes en español. Es por eso que la declaración de codificación nos será de gran utilidad ya que de otra manera habría que incluir estos caracteres a través de referencias a carácter, lo que puede resultar cuando menos laborioso (salvo que sean pocos). Para incluir los acentos o caracteres especiales del castellano, hay que utilizar la codificación de 8 bits, ISO-8859-1 asociada a los lenguajes de Europa Occidental.

Algunos **otros códigos** de caracteres ampliamente utilizados son:

ISO-8859-2: asociada a los lenguajes de Europa del Este.
ISO-8859-3: asociada a los lenguajes del sur de Europa.
ISO-8859-4: asociada a los lenguajes del norte de Europa.
ISO-8859-5: contiene los caracteres del Cirílico.
ISO-8859-15: como el ISO-8859-1 pero con el símbolo del Euro.
UTF-8: Unicode comprimido.
UTF-16: UCS (*Universal Charcter System*) comprimido.

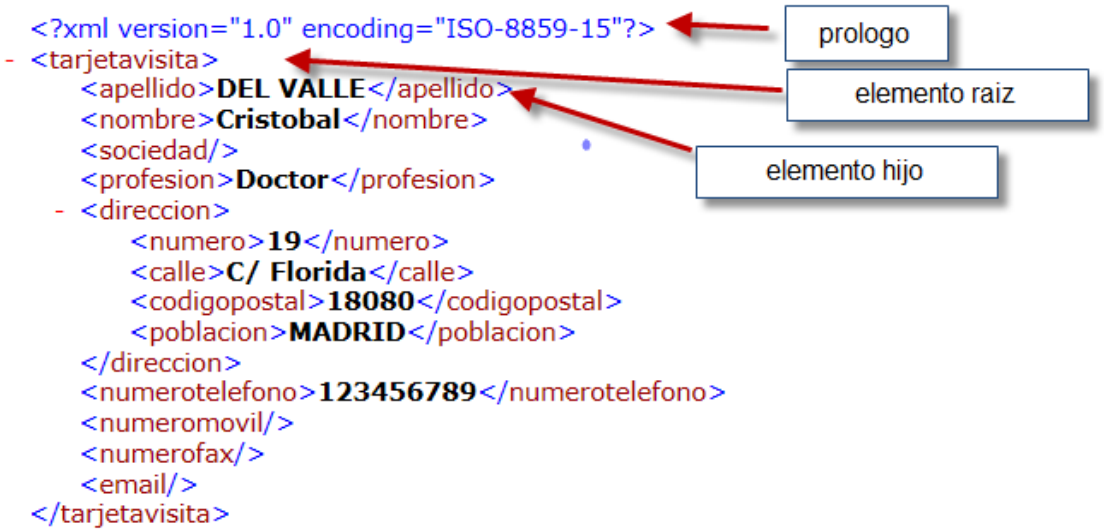
Por último, la declaración de **documento autónomo**, indica que **el documento contiene en su interior toda la información relevante para su interpretación**. Baste decir por ahora que pueden existir ciertos contenidos, fuera del documento actual, que modifiquen la forma en la que se procesará el documento, esta característica implica que el documento no es autónomo.

- **La Declaración del Tipo del Documento**,

Todos los documentos XML deben cumplir con la sintaxis de XML. La **Declaración de Tipo del Documento** provee una serie de mecanismos que aportan funcionalidad a XML, gracias a ella es posible definir una serie de restricciones adicionales que deben cumplir los documentos, también incorpora la posibilidad de utilizar ciertas herramientas que facilitarán al usuario XML algunas tareas. Todas estas propiedades adicionales se engloban bajo lo que se denomina un **tipo**. Los documentos que tienen un tipo asociado, y que cumplen con él, podrán distinguirse del resto y formarán lo que se denomina un tipo de documentos o una clase de documentos.
Existen situaciones en las que la declaración de tipo del documento no es necesaria, es perfectamente posible trabajar con XML sin emplearlas, sobre todo en entornos en los que los documentos XML se generan automáticamente por programas y no es necesario comprobar ciertas condiciones.

B. EL EJEMPLAR

Es la parte más importante y que **contiene la información del documento, es decir, los datos a los que se les ha añadido el marcado**. Se le llama ejemplar porque se supone que es una ocurrencia (un ejemplar) de un tipo de documento son sus restricciones y sus características.



ESTRUCTURA LÓGICA DE UN EJEMPLAR

Un documento está formado por:

- **Elementos y atributos**
- **Comentarios**
- **Instrucciones de procesamiento,**
- **Secciones CDATA.**

Todo lo cual está indicado en el documento por el marcado. **En un documento XML podemos distinguir entre marcado y datos carácter. El marcado es todo aquello situado entre los caracteres [<] y [>] y [&] y [;].**

Todos los caracteres de un documento XML proceden del conjunto de caracteres Unicode - ISO/IEC 10646, según regula la recomendación de XML.

Unicode: es un estándar internacional elaborado por el consorcio Unicode que pretende dar cobertura a la mayoría de los sistemas de escritura actualmente en uso en el mundo facilitando una representación digital para una gran cantidad de caracteres. Cada carácter tiene una codificación binaria de 16 bits.

ISO/IEC 10646: es otro estándar internacional de 32 bits, su objetivo es más amplio, ya que abarca un mayor número de caracteres, procedentes de sistemas de escritura antiguos pero también de sistemas actuales.

En la práctica, ambos estándares se pueden considerar equivalentes, ya que la versión actual de Unicode (2.0) está incluida en el estándar ISO/IEC 10646 y su codificación binaria no varía. Para más información sobre Unicode es posible visitar su sitio Web en http://www.unicode.org/.
Aunque todos los caracteres de un documento XML deben estar recogidos en estos estándares, no todos los caracteres de estos estándares se permiten en un documento XML. Así, los caracteres legales son: tabulador, retorno de carro, fin de línea y los caracteres gráficos de Unicode y ISO/IEC 10646. El uso de la mayoría de los caracteres no gráficos está prohibido, es decir, no se pueden incluir en un documento XML, ya sea directamente o mediante referencias a carácter.

Podemos incluir en nuestro documento los caracteres permitidos de estos conjuntos mediante una **referencia a carácter** que se compone de la cadena [&#], seguida del número del carácter que se desee y finalizando con un carácter [;]. El número del carácter puede estar expresado en decimal o en hexadecimal (precedidos de una "x" obligatoriamente minúscula).

Mediante **&#nº:** se puede incluir un carácter Unicode Ejemplo: ∀
&nombre: permite referenciar macros (se definen en una DTD)
Algunas macros predefinidas:

<	<
>	>
"	"
'	'
&	&

EJEMPLO:

```
<?xml version="1.0" ?>
<texto_con_acentos>
<!-- El elemento nl se incluye para facilitar la lectura en el
navegador Web -->

  <nl>Esta letra es una A acentuada: &#193; &#xc1; &#xC1;</nl>
  <nl>Esta letra es una E acentuada: &#201; &#xc9; &#xC9;</nl>
  <nl>Esta letra es una I acentuada: &#205; &#xcd; &#xCD;</nl>
  <nl>Esta letra es una O acentuada: &#211; &#xd3; &#xD3;</nl>
  <nl>Esta letra es una U acentuada: &#218; &#xda; &#xDA;</nl>
  <nl>Esta letra es una a acentuada: &#225; &#xe1; &#xE1;</nl>
  <nl>Esta letra es una e acentuada: &#233; &#xe9; &#xE9;</nl>
  <nl>Esta letra es una i acentuada: &#237; &#xed; &#xED;</nl>
  <nl>Esta letra es una o acentuada: &#243; &#xf3; &#xF3;</nl>
  <nl>Esta letra es una u acentuada: &#250; &#xfa; &#xFA;</nl>
  <nl>¿Hay otras letras que el espa&#241;ol necesite?</nl>
  <nl>¿Hay otras letras que el espa&#xF1;ol necesite?</nl>
  <nl>¡Hay otras letras que el ESPA&#209;OL necesite!</nl>
  <nl>¡Hay otras letras que el ESPA&#xD1;OL necesite!</nl>
  <nl>Algunos caracteres no parecen dar problemas &#191; es decir
"¿"</nl>
  <nl>Algunos caracteres no parecen dar problemas &#xbf; es decir
"¿"</nl>

  <nl>Algunos caracteres no parecen dar problemas &#161; es decir
"¡"</nl>
  <nl>Algunos caracteres no parecen dar problemas &#xa1; es decir
"¡"</nl>

</texto_con_acentos>
```

BUSCAR TABLA DE CARACTERES UNICODE

XML: ELEMENTOS Y ATRIBUTOS

Los dos componentes clave del ejemplar del documento XML son los **elementos** y los **atributos**.

ELEMENTOS

XML nos permite crear nuestras propias marcas.

Las distintas piezas de información en las que podemos dividir un documento, reciben, en XML, el nombre de elementos. Son la estructura básica que nos permite en un documento XML representar su estructura lógica y semántica. Los elementos constituyen la arquitectura en árbol de un documento XML.

Los elementos sirven para:

- Especificar contextos
- Delimitar contenidos
- Estructurar contenidos
- Jerarquizar elementos

La verdad es que como mejor se puede entender qué son los elementos es a través de ejemplos, veamos el siguiente, un poema XML:

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

<Poema>

      El Reino Perdido

      Las huestes de don Rodrigo

      desmayan y huian

      cuando en la octava batalla

      sus enemigos vencian

</Poema>
```

Éste es un documento XML muy sencillo, está reducido al mínimo y tiene **un único elemento**, Poema, gracias al cual podríamos saber que el documento contiene un poema, y podríamos diferenciarle de otros documentos como cartas, pedidos, recetas, etc. Lo cierto es que la

información que añade este elemento no es demasiada y seguramente no queramos conformarnos con eso. El documento de partida se puede refinar para añadirle otros elementos (en negrita).

```
<?xml version="1.0">
<Poema>
  <Titulo>El Reino Perdido</Titulo>
  <Cuerpo>
Las huestes de don Rodrigo
  desmayan y huian
  cuando en la octava batalla
  sus enemigos vencian
  </Cuerpo>
  </Poema>
```

En este segundo ejemplo existen elementos que identifican el título (Titulo) y el cuerpo (Cuerpo) del poema, con su ayuda, seríamos capaces, por ejemplo, de listar todos los títulos de los poemas de los que disponemos, o buscar un título determinado y mostrar el cuerpo del poema. Las piezas de texto que forman el título y el cuerpo están perfectamente identificadas y delimitadas. Este proceso de refinamiento y adición de elementos podría continuarse mientras fuera necesario.

- Realiza un documento XML que resuelva el siguiente supuesto: *"...necesito 20 rotuladores RX2 de código R23, que escriban y 2 grapadoras Linde código G56 envueltas para regalo"*
- Realiza un documento XML que resuelva el siguiente supuesto: "Con fecha 03.05.13, remito el paciente E.J.C. HC 334455 a Neumología por presentar bronquitis aguda con broncoespasmo"

Si sabemos de antemano que: HC 334455 es el número de historia clínica de un paciente, que Neumología es un punto de asistencia ambulatorio para las consultas externas de un hospital y que bronquitis aguda es un diagnóstico que dispone del código 466.0 dentro del sistema de clasificación dentro del hospital, obviamente este trozo de texto aunque no disponga de ninguna estructura subyacente, para nosotros es significativo. Sin embargo, un ordenador por muy potente que sea poca cosa podrá hacer más que trocear las palabras, indexarlas y procesarlas por separado.

UN ELEMENTO CONSTA DE TRES PARTES

Se puede observar en los ejemplos que los elementos están formados por tres componentes:

- **Una etiqueta de inicio o apertura**, por ejemplo **<titulo>,**
- **Una etiqueta de finalización o cierre**, por ejemplo **</titulo>,**
- **Un contenido**, situado entre ambas etiquetas, y que en este caso es Contenido.

<titulo>Contenidoe</titulo>

Como veremos más adelante también existen elementos compuestos por una sólo etiqueta, aquellos que no tienen contenido.

Hay que notar que, al contrario de lo que sucede en HTML, todos los elementos de XML, están formados por estas tres partes, las etiquetas no son el elemento en sí, son una parte de él que lo delimita y que lo marca. **El hecho de olvidar una etiqueta de finalización o de inicio implica un error de buena formación.** Una buena costumbre para no olvidar una etiqueta de fin es escribir la de inicio y la de fin al mismo tiempo.

EL ELEMENTO RAÍZ

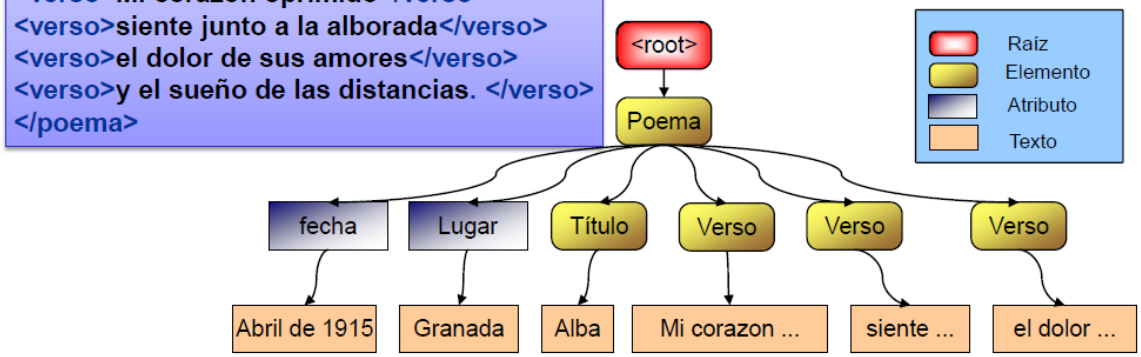
El **elemento raíz del documento**, recoge todo el contenido del documento. Por encima de cualquier elemento se ubica el nodo raíz, que se designa como "/". No es un componente que tenga representación dentro del documento XML, pero se utilizará más adelante como punto de partida para recorrer el árbol XML, y ubicar el resto de nodos.

Todos los documentos XML tienen obligatoriamente un elemento raíz (uno y sólo uno), un documento XML sin el elemento raíz o con dos o más elementos raíz no está bien formado.

Sólo puede haber un único elemento raíz

Cualquier documento puede representarse como un árbol

```
<poema fecha="Abril de 1915"
       lugar="Granada">

<titulo>Alba</titulo>

<verso>Mi corazón oprimido</verso>
<verso>siente junto a la alborada</verso>
<verso>el dolor de sus amores</verso>
<verso>y el sueño de las distancias. </verso>
</poema>
```



Realiza el documento XML de la figura completando el prólogo.

ELEMENTOS: REGLAS DE FORMACIÓN

Los elementos al igual que todos los componentes de un documento XML deben seguir ciertas reglas de buena formación, aplicables tanto a las etiquetas de inicio y fin como al contenido que admiten.

Un procesador XML conforme con la Especificación XML, comprobará que el documento cumple estas restricciones de buena formación. Cualquier restricción no cumplida será detectada y se tratará como un error fatal: el procesador informará a la aplicación y dejará de trabajar de una manera normal. Fundamentalmente, el objetivo de estas restricciones es asegurar que los documentos XML puedan ser interpretados por los procesadores XML sin ninguna ambigüedad, de manera que todos los elementos (y atributos) quedan perfectamente definidos.

NOMBRES

- Un nombre (Name) se compone como mínimo de **una letra**, (desde la "a" a la "z" o desde la "A" a la "Z"), de **un guión bajo** o de **un carácter de dos puntos**.
- El carácter inicial (letra, guión bajo o carácter de dos puntos) puede estar seguido de otros caracteres.
- La codificación de los nombres es sensible a mayúsculas y minúsculas.
- los nombres **no** pueden empezar por la **cadena xml**, incluidas todas las versiones cambiando el minúsculas por mayúsculas, (XML, Xml, xMl, ...), estos nombres se reservan para posteriores estandarizaciones.

El resto de los caracteres, destacando los espacios en blanco y los siguientes caracteres [,], [!] o [;], no pueden formar parte de un nombre. **Fíjese además que un nombre no puede comenzar ni por un dígito, ni por un punto, ni por un guión**.

Por lo tanto, la siguiente líneas es incorrecta: <2Marzo2015></2Marzo2015>
Esta sería una versión correcta: <Marzo-2-2015></Marzo-2-2015>

Dice la Recomendación XML que el carácter de dos puntos está reservado para la experimentación con espacios de nombres. Esto implica que no debe utilizarse en nombres, salvo con espacios de nombres. También especifica la Recomendación que los procesadores de XML deben aceptar este carácter como un carácter de nombre, a pesar de ello, algunos no lo hacen con lo que el carácter de dos puntos debe retirarse del nombre si se desea utilizar ese procesador en concreto.

Para más información sobre los caracteres incluidos dentro del grupo "CombiningChar" y "Extender", consultar la *Recomendación*

ETIQUETAS DE INICIO, DE FIN Y DE ELEMENTO VACÍO

Anteriormente se dijo que los elementos se describían como un conjunto de tres piezas, una **etiqueta de inicio, un contenido y una etiqueta de fin**. Revisaremos estos conceptos para formalizar algunos detalles, y veremos cómo existen también elementos sin contenido, que pueden reducirse a una única etiqueta de inicio y fin.

- **Etiqueta de inicio**: las etiquetas de inicio comienzan con un carácter [<], y a continuación, sin ninguna separación, un identificador genérico (un nombre XML). A lo anterior, le pueden seguir espacios en blanco completamente opcionales. Las etiquetas finalizan con un carácter [>]. Todas las etiquetas de inicio deben tener su correspondiente etiqueta de fin con el mismo identificador genérico.
- **Etiqueta de fin**: las etiquetas de fin comienzan con la cadena [</], seguida, sin separación, de un identificador genérico (un nombre XML) seguido a su vez de un carácter [>]. Entre el nombre del elemento y el carácter [>] puede, opcionalmente, haber espacios en blanco. Para que una etiqueta de fin tenga sentido debe haber una etiqueta de inicio anterior, con el mismo identificador genérico (incluidas las mayúsculas y las minúsculas).

Los elementos sin contenido reciben el nombre de elementos vacíos, **estos elementos pueden aparecer expresados de dos maneras, mediante dos etiquetas, una de inicio y otra de fin, seguidas, o mediante una única etiqueta especial, la etiqueta de elemento vacío.**

- **Etiqueta de elemento vacío**: está formada por el carácter **[<]**, seguido de un identificador genérico (un nombre XML). Para finalizar, espacios en blanco opcionales, y la cadena **[/>].**

  Ejemplo: <elemento_vacio></elemento_vacio> o <elemento-vacio />

El siguiente ejemplo nos muestra el uso de este tipo de etiquetas:

```
<?xml version="1.0">
<Poema>
        <Titulo>El Reino Perdido</Titulo>
        <Datos/>
        <Cuerpo>
                Las huestes de don Rodrigo
                desmayan y huian
```

```
                cuando en la octava batalla
                sus enemigos vencian
                ...
        </Cuerpo>
</Poema>
```

Tal y como está ahora este nuevo elemento no aporta información, entonces, ¿cuál es el sentido de los elementos vacíos?. Tal y como veremos, los elementos vacíos normalmente se complementan con atributos. Los atributos se explicarán más adelante.

CARACTERES PROHIBIDOS DENTRO DEL CONTENIDO DE LOS ELEMENTOS

Dentro del contenido de los elementos no pueden aparecer los siguientes caracteres y cadenas ya que tienen funciones especiales. Si aparecen deben escaparse para que no se interpreten erróneamente:

- **El caracter [<]** porque podría confundirse con el comienzo de una etiqueta.
- **El caracter [&]** porque podría confundirse como comienzo de una referencia a entidad (todavía no se ha hablado de ellas).
- **La cadena []]>]** no debe aparecer dentro del contenido de un elemento. Esto es así para asegurar cierta compatibilidad hacia atrás con SGML.

En XML hay definidas cinco entidades o caracteres especiales que comienzan con el carácter &:

- > símbolo >
- < símbolo <
- & símbolo &
- " símbolo “
- ' símbolo ‘

RESTRICCIONES EN CUANTO AL ANIDAMIENTO DE LOS ELEMENTOS

Los elementos deben anidarse correctamente. Es decir, si la etiqueta de inicio de un elemento está dentro de otro elemento, la etiqueta de finalización del primer elemento debe estar dentro de la etiqueta de finalización del segundo elemento.

Siempre que estas reglas de anidamiento se cumplan, y a falta de una declaración que especifique el contenido del elemento (se estudiarán más adelante), los elementos admiten cualquier combinación de elementos y datos carácter.

• ATRIBUTOS

Existe otro componente de XML, **los atributos, que asociados a los elementos actúan como modificadores, como adjetivos que incluyen información adicional, aplicable a un elemento**. Los atributos en XML funcionan de un modo similar a como funcionan en HTML.

<elemento atributo=”valor del atributo”> valor del elemento</elemento>

<persona sexo=”masculino”>DEL VALLE</persona>

Si recordamos el ejemplo anterior vemos como ahora el elemento vacío Datos tiene otro aspecto.

```
<?xml version=“1.0”>
<Poema>
        <Titulo>El Reino Perdido</Titulo>
        <Datos Autor="desconocido" Fecha="1700"/>
        <Cuerpo>
                Las huestes de don Rodrigo
                desmayan y huian
                cuando en la octava batalla
                sus enemigos vencian
                ...
        </Cuerpo>
</Poema>
```

En él aparecen dos nuevos componentes Autor y Fecha, que son atributos aplicados al elemento Datos y que aportan información sobre el autor del poema y la posible fecha de creación del mismo. También podrían haberse incluido como atributos del elemento raíz Poema.

ATRIBUTOS: REGLAS DE FORMACIÓN

Al igual que los elementos los **atributos deben seguir ciertas reglas** de buena formación. Si no se cumplen estos requisitos el documento XML que contiene los atributos no puede llamarse "bien formado". La mayoría son fáciles de aprender y recordar.

```
<alumno sexo="mujer" f_nacimiento="01/01/2001">
        <nombre>Pepa</nombre>
```

```
        <apellido>Perez</apellido>

        <tfno tipo = "movil">1111111111</tfno>

</alumno>
```

NOMBRES

Los nombres de los atributos deben cumplir las mismas restricciones que los nombres de los elementos.

DÓNDE ESPECIFICAR LOS ATRIBUTOS

Los atributos y sus valores se especifican, **dentro de una etiqueta de inicio o de** elemento vacío, a continuación del nombre del elemento, entre ambos debe haber **al menos un espacio en blanco**. **Si un elemento tiene varios atributos éstos se deben separar entre sí, como mínimo, por un espacio en blanco.** Como ya sabemos, las etiquetas finalizan con un carácter [>], entre este carácter y los posibles atributos puede haber opcionalmente espacios en blanco.

CÓMO SE ESPECIFICA EL VALOR DE UN ATRIBUTO

La asignación del valor al atributo se realiza a través de un carácter [=]. A ambos lados del igual pueden aparecer tantos espacios en blanco como se desee (o ninguno).

En XML, a diferencia de HTML, para especificar el valor de un atributo éste **se tiene que escribir entre comillas simples o dobles**, el tipo que no se utilice se puede incluir dentro del valor para marcar literales. El mismo tipo que abre el valor debe cerrarlo, lo contrario representa un error de buena formación.

MÁS REQUISITOS

A falta de una declaración que especifique qué valores puede tomar un atributo (se estudiarán más adelante), **los atributos contienen cualquier combinación de datos carácter que no contenga los caracteres [<] y [&] ya que se interpretarían de forma errónea**

Dentro de la etiqueta de inicio de un elemento no pueden aparecer dos referencias al mismo atributo. Esta restricción es lógica, ya que el procesador XML no podría conocer cuál es el valor correcto.

Cada elemento puede contener atributos en la etiqueta inicial

```
<poema fecha="Abril de 1915" lugar="Granada">
  . . .
</poema>
```

El orden de los atributos no es significativo

No puede haber 2 atributos con el mismo nombre

Las comillas dobles y simples pueden mezclarse

```
<autor frase="Juan dijo 'Hola' ">
 . . .
</autor>
```

```
<autor frase='Juan dijo "Hola" '>
 . . .
</autor>
```

USO DE LOS ATRIBUTOS

A veces a la hora de diseñar, puede ser complicado elegir entre un atributo o un elemento, si bien muchas veces el resultado es el mismo. **Hay que tener en cuenta que los atributos no pueden contener subelementos, ni subatributos, ni se organizan en ninguna jerarquía, por lo que tienen una capacidad de representación mucho más reducida que los elementos y no pueden reflejar una estructura lógica**. La información que añaden suele ser de poca entidad, sencilla, sin estructura y no subdivisible.

La regla general a utilizar es que se deben emplear elementos a menos que se necesite por alguna razón especial utilizar atributos. Esto se debe a las siguientes ventajas:

- No podemos tener varios atributos con el mismo nombre en el mismo elemento pero podemos tener varios elementos anidados con el mismo nombre.

- Los atributos son menos flexibles si queremos cambiar la estructura del documento.

- Los atributos no se pueden utilizar para describir estructuras jerárquicas mientras que los elementos si.

- Los atributos son más difíciles de manipular por programas

- **No podemos especificar el orden apropiado de los atributos.**

Si se utilizan atributos para colocar datos en un documento XML, el documento termina siendo más difícil de leer y de mantener.

Los atributos son útiles en las siguientes circunstancias:

- **Permiten la representación de metadatos. (Información sobre los datos). (Por ejemplo la versión de un fichero.)**

- **Proporcionan identificadores únicos para elementos que pueden utilizarse como referencias cruzadas.**

- **Especifican un conjunto de valores posibles para algunos bloques de datos (tipo enumerado).(semáforo: verde, rojo o amarillo)**

- **Especifican un valor constante.**

- **Especifican valores predeterminados**

- **Referencian entidades externas (como pueden ser otros archivos)**

Los atributos a menudo se procesan mucho más rápido que las entidades.

```
<alumno id="1111">
        <nombre>Pepa</nombre>
        <apellido>Perez</apellido>
        <tfno tipo ="movil">1111111111</tfno>
        <F_N>
                <dia>1</dia>
                <mes>1</mes>
                <anno>1991</anno>
        </F_N>
</alumno>
```

OTROS ELEMENTOS

En esta sección vamos a ver los otros elementos de un documento XML:

- **Comentarios,**
- **Instrucciones de procesamiento,**
- **Secciones CDATA**

1. COMENTARIOS

Los comentarios pueden aparecer en cualquier lugar de un documento fuera de otras marcas. Los comentarios son importantes y aportan información que puede ser útil en algún momento de la vida del documento XML, ya sea para nosotros mismos o para otras personas que deban trabajar con él en el futuro.

Un procesador de XML puede, pero no tiene que, hacer posible que la aplicación recupere el texto de comentarios.

- Comienzan por la cadena [<!--] y finalizan con la cadena [-->], sin separación dentro de estas cadenas.
- Los comentarios no se consideran datos carácter, y el procesador XML no tiene la obligación de pasarlos a la aplicación.
- Admiten cualquier carácter en su interior, salvo, por compatibilidad con SGML, la cadena [--].
- Los comentarios pueden aparecer antes o después de un elemento, pero no pueden aparecer dentro de una etiqueta.
- Tampoco pueden aparecer antes de la declaración XML.

Ejemplo:

```
<?xml version="1.0">
<!-- Este documento contiene un libro -->
<Libro>
        <Titulo>El Reino del Dragón de Oro</Titulo>
        <!-- título del libro -->
        <Datos Autor="Isabel Allende" Fecha="2003"/>
        <Descripcion>
        <!-- sinopsis del libro -->
                El reino del dragón de oro es la segunda parte de una trilogía, Memorias de águila y jaguar, iniciada con La ciudad de las bestias. Esta vez, Nadia Santos y Alexander Cold acompañarán a la periodista y escritora Kate Cold al Reino Prohibido, un pequeño país situado en el corazón del Himalaya.                ...
        </Descripcion>
</Libro>
```

2. SECCIONES CDATA

Estas secciones de datos son complementarias al marcado. Se denominan Datos de **carácter o Character DATA**. Permiten que determinados datos **no sean procesados por los analizadores** (no sean considerados datos de marcado).

Pueden contener:

- Texto,
- Caracteres reservados (como <) y
- Caracteres blancos.

Las secciones CDATA, empiezan con la cadena "<![CDATA ["y finalizan con la cadena "]]>" .

Ejemplo: Para poner la siguiente expresión 6 es <7 y 7 > 6 tendríamos que incorporarlo como:

<![CDATA[6 es <7 y 7 > 6]]>

Dentro de una sección CDATA sólo la cadena final se reconoce como marcado y gracias a ello, caracteres como, < y & pueden darse en su forma literal con lo que no se necesita utilizar los respectivos "&alt;"y "&".

Las secciones CDATA no pueden anidarse, y por ello en:

<![CDATA[<saludo>Hola amigo</saludo>]]>

Tanto <saludo> como </saludo> se reconocen como caracteres de datos y no de marcado.

La razón de ser de las secciones CDATA es la posibilidad de incluir un código de secuencia de comandos, que muy a menudo incluye los caracteres &,<,> y " . Para ver su utilidad, el listado siguiente muestra un documento XML que compara un texto en una sección CDATA con caracteres de datos especiales:

```
<?xml version = "1.0" ?>
<formula>
        <usasecuencia>
                Si (velocidad &gt; 120 )
                Multa por exceso de velocidad
        </usasecuencia>
        <sinusarsecuencia>
                <![CDATA[
                        Si ( velocidad > 120 )
                        Multa por exceso de velocidad
                ]]>
</sinusarsecuencia>
</formula>
```

Las secciones CDATA permiten escribir texto literal que no será procesado

```
<código>
  if x < 3 && x > 4 then
     print “Hola”
</código>
```

```
<código>
  if x &lt; 3
   &amp;&amp; x &gt; 4 then
     print &quot;Hola&quot;
</código>
```

```
<código>
<![CDATA[
if x < 3 && x > 4 then
      print “Hola”
]]>
</código>
```

3. INSTRUCCIONES DE PROCESO

La Instrucción de Proceso (IP) es un mecanismo que permite a los documentos XML **contengan instrucciones específicas para las aplicaciones que los van a usar, sin que éstas formen parte de los datos del propio documento.** En consecuencia el analizador XML al detectarlas se limita a pasar esa información a la aplicación que realiza la llamada, indicándole simplemente el modo de administrar los datos del documento.

Una IP se delimita mediante <? y ?>.

La declaración: <?xml...?> , a pesar de ir con acompañado de <? , ?> no es una IP ya que siempre debe ir al principio del prólogo y no se puede acceder mediante un analizador XML al contenido de la misma.

Las IPs empiezan con un identificador (acompañado de un valor) y denominado destino y objetivo que, siguiendo las mismas reglas de los nombres de elementos y atributos, se usa para identificar la aplicación a la cual se dirige. (PITarget).

Es posible incluir instrucciones que indican al procesador alguna acción a realizar
Sintaxis: **<?aplicación datos ?>**

En el ejemplo que sigue, la IP se utiliza para indicar a la aplicación que el documento se debe mostrar con una determinada hoja de estilo:

<?xml-stylesheet type="text/css" href="MiHojaDeEstilo.css" ?>
En este caso la información que la acompaña type especifica el tipo MIME de la hoja de estilo y href indica el URI donde se encuentra la hoja que se desea utilizar.

En realidad la declaración de documento es una instrucción de procesamiento
<?xml-stylesheet type="text/xsl" href="hoja.xsl"?>
Pueden utilizarse para asociar una hoja de estilos al documento:
...o para otros propósitos especiales

Una IP puede aparecer en cualquier lugar de un documento siempre que quede fuera de otra marca, aunque normalmente se colocan en el prólogo.

Es importante señalar que en el caso de que la aplicación no las use, éstas acaban por no tener efecto alguno, por lo que pueden mantenerse en un documento, en el caso en que éste vaya a trabajar con una aplicación distinta de la pensada, cuando fue creado.

ESPACIOS DE NOMBRES

1. ESPACIOS DE NOMBRES

Los espacios de nombres permiten definir la pertenencia de los elementos y los atributos de un documento XML a una familia de vocabulario XML. El nombre de un elemento puede pertenecer a varias familias sin que ello implique que poseen el mismo significado. El elemento visita de un sitio web no contendrá la misma información que el elemento visita de un museo. Sin embargo puede que se tenga que utilizar estos dos elementos dentro de un mismo documento XML.

Por lo general, los espacios de nombres permiten **definir un nombre único a cada elemento, indexándolos según el nombre de la familia de vocabulario adecuado**.
Un espacio de nombres se definen como referencia **URI(Uniform resource Identifier), que servirá para identificar los elementos que pertenecen a dicho espacios de nombres.** Otra forma de verlo es que los elementos tendrán un nombre compuesto por dos partes: una primera con su nombre y una segunda con el nombre de espacio de nombres. Este nombre compuesto permitirá identificar de forma unívoca al elemento en cuestión y de esta forma conocer siempre a que elemento se está refiriendo.
La construcción de estos nombres extendidos se realizan uniendo el nombre, el espacio de nombre y el nombre del elemento o atributo usando como conector el símbolo ":". Sin embargo, las referencias URI pueden ser largas, lo que va en detrimento de la legibilidad y claridad del documento, además de proporcionar que se comenta errores más fácilmente. Además los URIs pueden contener caracteres no válidos en XML. Para solucionar este problema, en XML se puede asignar un **sinónimo corto al espacio de nombres** de forma que este sinónimo corto sea el que se use a lo largo del documento. **El sinónimo se asigna el separador ":" y la etiqueta "xmlns". El atributo "xmlns" es un atributo reservado**.

EJEMPLO:
La clasificación de los datos en archivos XML para ser tratados por alguna aplicación puede ocasionar que en ciertas ocasiones sea necesario mezclar dos archivos XML distintos para obtener un nuevo archivo XML que los agrupe. Cuando esto ocurre es muy fácil que haya elementos con el mismo nombre en diferentes archivos y aparezcan

conflictos al unificarlos. Para evitar este conflicto se utilizan los **espacios de nombres,** cuya finalidad es la de **evitar estos conflictos estableciendo qué elemento corresponde a qué archivo original**.

Partimos de dos archivos XML originales, uno de clientes y otro de proveedores con el siguiente esquema:

Clientes:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<clientes>
  <cliente>
    <nombre>Belen Diaz </nombre>
    <direccion>Calle Nova 2</direccion>
  </cliente>
</clientes>
```

Proveedores:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<proveedores>
  <proveedor>
    <nombre>Ines Garcia</nombre>
    <direccion>Calle Flores 7</direccion>
  </proveedor>
</proveedores>
```

Si unificamos ambos ficheros en un único documento XML, los elementos <nombre> y <direccion> crearán conflictos al estar "repetidos". Para evitar esto **asignamos un identificador de espacio de nombres a cada uno de los elementos y/o atributos en el nuevo documento XML para que así, al combinarlos, no colisionen**. Podremos tener un elemento <direccion> para clientes y proveedores y no se confundirán puesto que irá precedido del identificador del espacio de nombres correspondiente.

Hay dos formas de asignar los espacios de nombres: mediante espacios de nombre por defectos por un atributo (a todo un elemento y su contenido) y mediante prefijos (a cada elemento particularmente).

2. ESPACIOS DE NOMBRE POR DEFECTO O PREDETERMINADO

Una espacio de nombres por defecto o predeterminado nos evita especificar el prefijo para todos los elementos que hacen referencia a ese espacio de nombres.

Se declara con un atributo **xmlns** sin prefijo y se considera que es válido, tanto en el elemento en el que ha sido declarado, como en todos los elementos dentro de su contenido.

Por ello en particular, para declarar un espacio de nombres por defecto en todo el documento basta con usar este atributo xmlns en el elemento raíz. Con el uso de un espacio de nombres por defecto, si no aparece ninguno, se usa este y todos los elementos del documento sin prefijo se tratarán como pertenecientes a él.

EJEMPLO:

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

<persona xmlns ="http://direccion/pers" >
  <nombre>
    <title>Sr</title>
    <npila>Juan</first>
    <apellido>Doelo</apellido>
  </nombre>
  <posicion>VicePresidente de Marketing</posicion>
</persona>
```

EJEMPLO RESUMEN:

Espacio de nombres por defecto (atributo)
Se define mediante un **atributo xmlns**. **El espacio de nombres se aplicará al elemento en el que se incluye el atributo xmlns y a todo su contenido** (siempre que no esté asociado a otro espacio de nombres). La sintaxis es la siguiente:

```
xmlns="URI del espacio de nombres"
```

En este caso el archivo XML Contactos, quedaría de la siguiente forma:

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

<contactos>

<clientes xmlns="http://www.servidor.com/clientes/">

  <cliente>

    <nombre>Belen Diaz </nombre>

    <direccion>Calle Nova 2</direccion>

  </cliente>

</clientes>

<proveedores xmlns="http://www.servidor.com/proveedores/">

  <proveedor>

    <nombre>Ines Garcia</nombre>

    <direccion>Calle Flores 7</direccion>

  </proveedor>

</proveedores>

</contactos>
```

Al usar un espacio de nombres por defecto, el documento es más claro y fácil de leer, aunque se deben tener precauciones con ello, pues cuando el archivo se importe o se incluya en otro documento, hay que asegurarse de que no se produzcan colisiones de nombres no previstas. Recalcar que los espacios de nombres por defecto y los que no lo son pueden combinarse sin problemas en un mismo documento. Así, el ejemplo del apartado anterior, declarando un espacio por defecto es equivalente a :

```
<?xml version="1.0" encoding="iso-8859-1"?>
<persona xmlns ="http://direccion/pers"
      xmlns:html="http://www.w3.org/1999/xhtml">
  <nombre>
    <title>Sr</title>
    <npila>Juan</npila>
    <apellido>Doelo</apellido>
  </nombre>
  <posicion>Vice Presidente de Marketing</posicion>
  <resumen>
    <html:html>
      <html:head><html:title>Resumen de Juan
Doelo</html:title></html:head>
      <html:body>
      <html:h1>Juan Doelo</html:h1>
      <html:p>Juan es un gran hombre</html:p>
      </html:body>
    </html:html>
  </resumen>
</persona>
```

Un espacio de nombre predeterminado se puede declarar para cualquier hijo y será válido en todos sus descendientes:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<persona xmlns ="http://direccion/pers" >
  <nombre>
    <title>Sr</title>
    <npila>Juan</npila>
    <apellido>Doelo</apellido>
  </nombre>
  <posicion>Vice Presidente de Marketing</posicion>
  <resumen xmlns ="http://www.w3.org/1999/xhtml">
    <html>
      <head><title>Resumen de Juan Doelo</title></head>
      <body>
      <h1>Juan Doelo</h1>
      <p>Juan es un gran hombre</p>
      </body>
    </html>
  </resumen>
</persona>
```

Dentro de resumen el espacio de nombres válido será el de xhtml.

Si queremos cancelar el espacio de nombres para un elemento determinado: xmlns="" dentro de la etiqueta del elemento.

Los atributos no funcionan como los elementos. De hecho para los atributos se supone que están asociados a los elementos a los que pertenecen.

3. ESPACIOS DE NOMBRES MEDIANTE PREFIJOS

Otra forma de especificar los espacios de nombres es usando prefijos en los elementos. En este caso la sintaxis general es la siguiente:

xmlns:prefix="URI del espacio de nombres"

El prefijo es tan sólo una forma corta de hacer referencia al identificador (la URI), que es la que realmente identifica al espacio de nombres. De esta forma, el documento "Contactos" quedaría de la siguiente manera:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<contactos xmlns:cli="http://www.servidor.com/clientes/"
xmlns:prov="http://www.servidor.com/proveedores/">
<clientes>
    <cliente>
    <cli:nombre>Ana Ridao</nombre>
     <cli:direccion>Calle Nova 2</direccion>
   </cliente>
</clientes>
<proveedores>
    <proveedor>
      <prov:nombre>Ines Garcia</nombre>
     <prov:direccion>Calle Flores 7</direccion>
    </proveedor>
</proveedores>
</contactos>
```

DOCUMENTOS BIEN FORMADOS Y DOCUMENTOS VÁLIDOS

El concepto de documento bien formado quiere decir que se siguen las normas y reglas básicas que se establecen en todos los documentos XML, esto implica, entre otras cosas que:

- Al principio del documento se debe declarar una cabecera (en formato etiqueta) que indique lo siguiente:
  - **Versión del documento XML**. Actualmente se puede indicar la versión 1.0 ó 1.1.

- **Tipo de codificación utilizada**. De esta manera se sabe que juego de caracteres se están utilizando. Lo normal es utilizar una codificación UTF-8, ISO-8859-15.
- **Si está basada en un documento de estructura externo** ((DTD o esquema). Si es así se indicará en el atributo **standalone="yes".** Por defecto es igual a no.
- **<?xml versión="1.0" enconding="UTF-8" standlone="no"?>**

Documento bien formado

Sigue las reglas sintácticas

Importante:

Contiene un único elemento raíz

Todas las etiquetas están correctamente anidadas

```
<pizzas>
 <pizza nombre="Margarita" precio="6">
  <ingrediente nombre="Tomate" />
  <ingrediente nombre="Queso" />
 </pizza>
</pizzas>
```

```
<pizzas>
 <pizza nombre="Margarita" precio="6">
  <ingrediente nombre="Tomate" >
</pizzas>
```

El concepto de documento bien formado implica, entre otras cosas que:

- **Sólo contiene un elemento raíz.**
- **Todos los elementos tienen etiqueta inicial y final con nombres idénticos.**
- **Los elementos XML no se entrecruzan por lo que están correctamente anidados.**
- **Todos los valores de atributo utilizan comillas.**
- **Un elemento no tiene dos atributos con el mismo nombre.**
- **No aparecen en el texto del documento los caracteres: < , > , &.**
- **Los comentarios y las IP (Instrucciones de procesamiento) no aparecen en etiqueta alguna.**
- **Aparecen sólo caracteres aceptados por la definición del documento, la cual fue definida en el atributo encoding del encabezado.**
- **Los nombres de elementos están compuestos usando letras, números y los signos -_.:, es relevante la presencia o no de caracteres con mayúscula.**

El siguiente es un ejemplo de documento XML bien formado en el que aparece el marcado XML cuyos elementos estructuran el documento de forma que el elemento raíz estado contiene los elementos nombre, presidente y provincias. Y donde además en la tercera línea se incluye una IP que indica que en caso de tener que presentarlo se debe usar una hoja de estilo determinada por los valores type="text/xsl" y href="formato_estados.xsl".

```
<?xml version = "1.0"?>
<!-- Uso de elementos y de atributos -- >
<?xml:stylesheet type = "text/xsl" href = "formato_estados.xsl"?>
<estado>
        <nombre> España </nombre>
        <presidente>
                <nombre>PPPPPP</nombre>
                <apellido>RRRRR</apellido>
```

```
        </presidente>
        <provincias>
                <provincia num = "1">Valencia</provincia>
                <provincia num = "2">Baleares</provincia>
                . . .
        </provincias>
</estado>
```

Escribir un documento XML bien formado generalmente no es suficiente para crear una aplicación por sencilla que sea, ya que de alguna forma se tiene que limitar o controlar en mayor menor medida el tipo de datos a incorporar.

Se puede incluir una declaración del tipo de documento

```
<?xml version="1.0"?>
<!DOCTYPE poema SYSTEM "poema.dtd">
<poema fecha="Abril de 1915"
        lugar="Granada">

<titulo>Alba</titulo>

<verso>Mi corazón oprimido</verso>
<verso>siente junto a la alborada</verso>
<verso>el dolor de sus amores</verso>
<verso>y el sueño de las distancias. </verso>
</poema>
```

poema.dtd

```
<!ELEMENT poema (titulo,verso*)>
<!ELEMENT titulo (#PCDATA)>
<!ELEMENT verso (#PCDATA)>
<!ATTLIST   poema fecha CDATA #REQUIRED
                   lugar CDATA #IMPLIED>
```

Documento válido

Está bien formado y

La estructura encaja con la declaración del tipo de documento

Por ejemplo, si tuviéramos una aplicación para manejar información de facturas en el que apareciera algo como lo que sigue :

```
<factura>
        <importe>5000</importe>
        <cliente>Vicente Sanchez</cliente>
        <cliente>Maria del Carmen Perez</cliente>
        <cliente>Juan Estevez</cliente>
        <fecha>edad media</fecha>
</factura>
```

Sería, informativamente inútil. Ya que estando el documento bien formado, presenta evidentes errores de sentido común, como que una factura debe pertenecer a un único cliente y que una fecha debe especificarse en algún tipo de formato del tipo día/mes/año. Por ello, hay que ir más allá de lo bien formado y validar la estructura usada, respecto a unas reglas, con el objeto de tener **asegurada tanto su integridad como su formato**. Nótese que este problema no es específico de los documentos XML, un 70% de las líneas de código que se escriben están orientadas a comprobar la integridad de los datos, con cosas aparentemente tan elementales como si donde se supone que hay un entero, efectivamente lo hay, etc...

Ante esta situación y siguiendo la forma de proceder utilizada en Bases de Datos para definir restricciones, **en XML se recurre también a ficheros de definición o esquemas que especifican la distribución de datos y los contenidos que están permitidos en un documento.** Concretamente, un esquema es una forma de especificar restricciones sintácticas al marcado al objeto de definir su estructura. La necesidad de contar con un mecanismo como este es evidente, ya que si se considera :

<importe>5000</importe>

El elemento importe además de poder validar que tiene un contenido, es necesario asegurarse que este contenido es numérico. Por ello sin medidas apropiadas, la expresión:

<importe>hola</importe>

Se considera aceptable y las aplicaciones que usaran un documento con este marcado no tendrían otro remedio que comprobar que el contenido del importe es numérico, y tomar la acción apropiada si no lo fuera, con toda la laboriosidad que ello supondría. Por ello, en el esquema que se use en XML hay que expresar de alguna forma que el dato del elemento importe puede ser descrito como numérico, con lo que 5000 quedaría validado, mientras que con hola fallaría. En XML si un documento está de acuerdo con un esquema se dice que es válido respecto a él, y en caso contrario se declara como no válido.

EJEMPLO DE DOCUMENTO BIEN FORMADO. Puede ser interpretado por el navegador de Internet

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes" ?>
<directorio>
<entrada>
 <apellido>Guevara</apellido>
 <nombre>Ernesto</nombre>
</entrada>
</directorio>
```

EJEMPLO DE DOCUMENTO MAL FORMADO. Genera un error de interpretación por parte del navegador de Internet

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<directorio>
<entrada>
 <apellido>Guevara
 <nombre>Ernesto</nombre>
</entrada>
```

```
</directorio>
```

En XML existen distintas formas para describir estos esquemas, de los cuales nos interesan dos :

- Las DTDs,

- Los Esquemas XML del W3C .

Aunque a un documento XML no se le exige que incorpore ningún esquema su uso es más que recomendable ya que hay que tener un mecanismo para asegurar la conformidad del documento cuando vaya a ser intercambiado, ya que si no se especifica un esquema, por un lado, las aplicaciones que utilicen los archivos tendrán que detectar cuál es la estructura que define al fichero y por otro lado se correrá un serio riesgo de generar inconsistencias, al carecer la aplicación de las especificaciones que guiaron el esquema del documento origen.

En XML, hay que distinguir entre documento válido y documento bien formado, en este último no existen limitaciones sobre el número o tipo de contenidos, mientras que en un documento válido, además de estar bien formado se deben respetar las restricciones establecidas por la definición externa de un esquema (DTD o Esquema XML) en cuanto a cuestiones relacionadas con los elementos y atributos aceptados, su cantidad, el orden en el cual pueden aparecer, etc... Estar bien formado es un requisito básico, mientras que la validez es una condición más fuerte que no se requiere en XML para procesar documentos.

Cuando se valida un documento XML se comprueban las siguientes cuestiones:

- **Sólo se pueden utilizar elementos o atributos definidos en el DTD o esquema.** Cualquier elemento no declarado hará que el documento no pase el proceso de validación (análisis léxico).
- **Que los elementos y atributos estén en el orden definido en el DTD o esquema.** Además se comprueba que los atributos o elementos obligatorios estén presentes dentro del documento (análisis sintáctico).
- Si se declaran valores que deben ser únicos, como por ejemplo, atributos ID, debe asegurarse que en todo el documento XML se cumple esta condición.
- Si se define un tipo concreto para los atributos o elementos, los valores que tomen estarán tipados a la definición anterior. No es lo mismo declarar un tipo entero que un tipo string.

De acuerdo con ello, los procesadores XML se dividen en validadores y no validadores. Ambos dan cuenta de donde falla formalmente el documento que procesan y si además es validador, debe indicar cualquier violación a las restricciones impuestas a la estructura por una DTD o un Esquema.

En cualquier caso, la decisión de rechazar o no documentos no válidos es la de la aplicación, no del procesador validador que se limita a informar a la aplicación de ello y continúa el análisis.

A continuación, se muestra un ejemplo de declaración de documento para inicio.xml que ahora contiene un referencia a una DTD externa (inicio.dtd) :

```
<?xml version = "1.0?">
<!-- Uso de un subconjunto externo -->
```

```
<! DOCTYPE miCurso SYSTEM "inicio.dtd">
<miCurso>
<Curso>Curso de XML </Curso>
</miCurso>
```

Suponiendo que en inicio.dtd se hubiera especificado que el elemento miCurso contiene un único elemento hijo llamado curso, y que en el documento se hubiera omitido este elemento, como por ejemplo en :

```
<?xml version = "1.0"?>
<! DOCTYPE miCurso SYSTEM "intro.dtd">
<!-- Al elemento raíz le falta el elemento hijo mensaje -->
<miCurso>
</miCurso>
```

Se tendría una estructura de documento inconsistente con la DTD (sería un documento bien formado y no válido) y si procesáramos este código tal como está, obtendríamos un mensaje parecido a : "error por ausencia de una etiqueta de cierre".

HERRAMIENTAS QUE COMPRUEBAN LA BUENA FORMACIÓN

Es conveniente, al finalizar la creación del documento XML, comprobar que está bien formado. Para ello, hay una gran variedad de aplicaciones disponibles. Quizás las aplicaciones más sencillas de utilizar sean los navegadores para la Red que utilizamos diariamente. Si el documento se almacenó en un fichero de texto plano, podemos intentar abrirlo con el explorador Netscape Navigator o Microsoft Internet Explorer, Mozilla, si está bien formado podremos ver el árbol asociado al documento.

Para comprobar sintaxis podemos utilizar: nuestro analizador XML (sólo IE), o el analizador en línea de RUWF.

En caso de querer utilizar otras aplicaciones, en las siguientes direcciones podemos utilizar analizadores en línea:

- http://www.w3.org/2001/03/webdata/xsv
- http://validator.w3.org/
- http://www.gotdotnet.com/services/xsdvalidator/
- http://tools.decisionsoft.com/schemaValidate.html
- http://www.asahi-net.or.jp/~eb2m-mrt/onLineValidation.html
- http://www.stg.brown.edu/service/xmlvalid/
- http://www.cogsci.ed.ac.uk/~richard/xml-check.html
- http://msdn.microsoft.com/downloads/samples/internet/xml/xml_validator/
- http://www.xmlvalidation.com: Permite insertar en un formulario un fichero XML con los DTD incluidos o con un esquema externo y verifica la validez del documento.

Aplicaciones:

Serna Free: editor de documentos XML que permite analizar la validez y aplicar los estilos previamente definidos. http://www.syntex.com/products/serena/.

TotalEdit: es un editor multilenguaje que puede ser utilizado también para tratar los documentos XML. http://www.codertools.com.

XML Notepad : editor de Microsoft que facilita el trabajo con documentos XML.
http:// www.microsoft.com.

oXygen: http://www.oxigenxml.com, etc……..