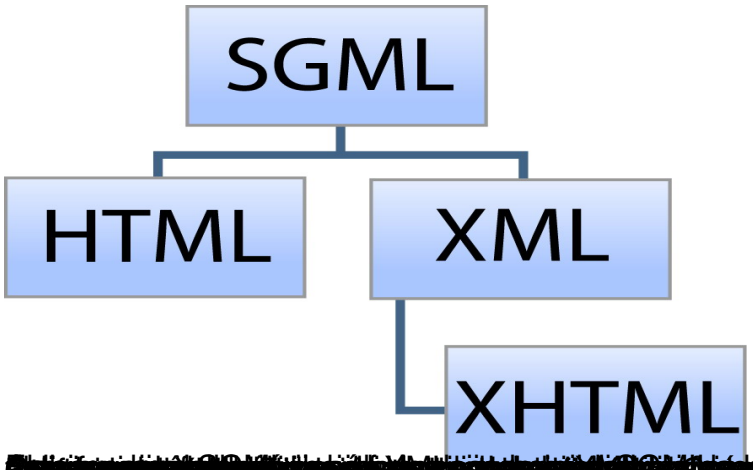
There are no translations available.
Descubre en este artículo, todo lo referente a este lenguaje de descripción de documentos...

Introducción

XML significa en Inglés e**X**tensible **M**arkup **L**anguage y es un lenguaje de descripción de documentos que no incluye ninguna información relativa al diseño de éstos.

HTML (H*yperText Markup Language*) es el lenguaje de marcas (etiquetas) mas conocido y utilizado para la creación de páginas web que permite la navegación tipo hipertexto. Pero XML es mas que un lenguaje, es un metalenguaje que permite definir otros lenguajes de marcas con objetivos diferentes. Por ese motivo se le llama 'eXtensible'.
Por lo tanto, XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes específicos. Un ejemplo de lenguaje que usa XML para su definición es
XHTML
(e
X
tensible,
H
ypertext
M
arkup
L
anguage)
, nueva versión de HTML que cumple
la especificación SGML
y cuyo objetivo es sustituirlo como estándar de páginas web.

XML aparece en 1997 como un subconjunto de **SGML** (**S***tructured **G**eneralized **M**arkup **L**angu age, ISO 8879*
)
, lenguaje que permite especificar las reglas de etiquetado de documentos. XML es algo así como
SGML simplificado
. Una aplicación no necesita comprender SGML completo para interpretar un documento XML. Los editores SGML, sin embargo, pueden comprender XML.



nados.

Características de XML.

Las características mas importantes de XML son las siguientes:

-

Permite la creación de etiquetas propias y permite asignar atributos a las etiquetas.

-

Trabaja con los llamados **DTD**s (Definición de Tipo de Documento) que en realidad son archivos de texto cuyo contenido son las definiciones de las etiquetas y sus atributos con los que se puede trabajar en un determinado documento. Es decir, el DTD contiene la estructura de los datos.

-

Como consecuencia de los puntos anteriores, XML permite la creación de nuevos DTDs. Por ese motivo, entre otros, este metalenguaje se llama **extensible**.

-

En un documento XML la estructura y el diseño están completamente separados.

-

Cada documento XML se puede validar ante un DTD, y si no es posible, se puede declarar

como documento '*bien formado*'[1].

-

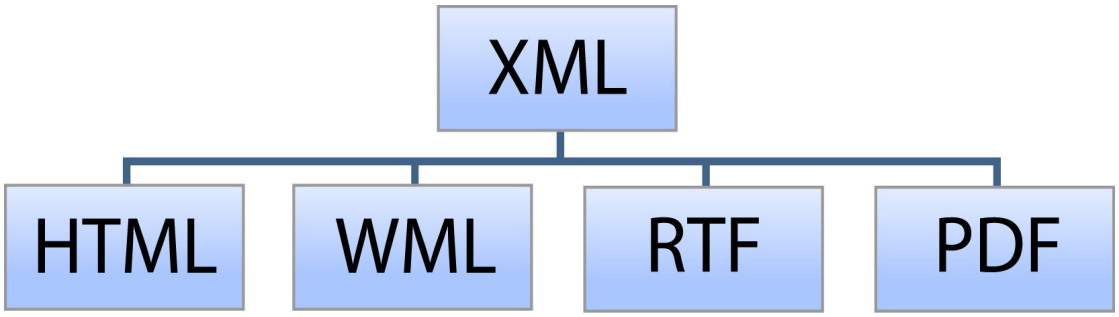
XML se almacena en formato texto (no binario) lo cual hace que los documentos sean directamente entendibles. Es decir, los documentos tienen una estructura entendible tanto por los ordenadores como por las personas.

-

Cada documento incluye metadatos[2] sobre sí mismo, lo cual facilita la tarea de los motores de búsquedas en la web, ya que devolverán respuestas más adecuadas y precisas.

-

Permite la exportabilidad a otros formatos de publicación de datos (HTML, PDF, texto enriquecido RTF, voz, etc).



-

XML es un estándar abierto no sujeto a ningún tipo de licencia (www.w3c.org).

-

XML permite la internacionalización, es decir puede trabajar con cualquier conjunto de caracteres, entre ellos el juego de caracteres UNICODE (utf-8).

-

XML utiliza reglas de generación concretas y, por tanto, los documentos son fácilmente procesables.

-

XML permite compartir información entre sistemas o fuentes de datos heterogéneas, por ejemplo, páginas web, distintas bases de datos, etc.

-

XML no es compatible con HTML, pero los documentos HTML v4.0 pueden ser convertidos a XML.

-

Se debe descartar la idea de que XML, al ser un lenguaje de marcas, se utiliza para la creación de páginas web.

En el ejemplo siguiente vemos el aspecto de un archivo XML:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE documento SYSTEM "documento.dtd">
<!-- Esto es un comentario -->
<documento><p>Mi Primer <negrita valor="1">documento XML</negrita></p>
```

```
<p>Comienza con la etiqueta &lt;documento&gt;</p>
<p>A continuación añadimos una etiqueta de elemento vacio</p>
<imagen archivo="imagen.png"/>
<p>Añadimos una etiqueta CDATA.</p>
<ejemplo>
<![CDATA[ Esto es una sección de datos en la que podemos escribir sin que el parser lo analice.]]> </ej
</documento>
```

Como vemos por las características descritas, las ventajas de la utilización del formato XML son muchas. Pero también tiene sus inconvenientes:

1. Al utilizar el formato texto los documentos ocupan mayor cantidad de espacio en disco para su almacenamiento. Aunque esto ya no es un gran problema debido al abaratamiento del hardware. Además permite compresión, con lo que se optimiza algo mas la ocupación en disco.

2. Ya que la estructura y el diseño están separados habrá que preparar el documento para su presentación. Existe un lenguaje de transformación especialmente diseñado para XML que es **XSL** que permite pasar documentos XML a otros formatos.

Leyendas sobre XML

XML es una extensión de HTML Falso

- SGML es un metalenguaje, es decir, un lenguaje para definir lenguajes. Y XML es una forma simplificada de SGML, pero NO una extensión de HTML. No hay que confundir.
- XML es tan popular como HTML, pero por diferente motivo. Un lenguaje creado con XML puede ser analizado sintácticamente por un *parser* tan pequeño que puede ir integrado en el propio navegador web.
- La presentación de un documento HTML se realiza mediante la utilización de hojas de estilo CSS solamente. Sin embargo la presentación de un documento XML tiene mas opciones disponibles.

XML puede ser manejado directamente por el navegador web Falso

- Las etiquetas XML no tienen significado en sí mismas.
- Al contenido de un documento XML hay que añadirle el tratamiento o transformación, ya sea mediante un programa o de forma declarativa con hojas de estilo.
- Se pueden utilizar hojas de estilo en cascada **CSS** (*Cascading Style Sheets) que le dan forma al documento,*
pero no permiten su transformación ni generar estructuras nuevas.

- *Normalmente se utiliza **XSL** (Extensible Style Language)* para dar forma a los documentos XML.

Tecnologías XML

Asociado a XML existe la:

-

Especificación propiamente -> **XML 1.0**

-

Definición de tipo de documentos -> **DTD**

-

Definición de estilos -> **XSL = XSLT+Xpath**

DTD (*Document Type Definition*): archivo que almacena la definición formal de un tipo de documento y especifica su estructura lógica. En general la utilización del DTD es opcional aunque conveniente. En función de si se utiliza o no DTD los documentos serán 'válidos' o sólo 'bien formados'.

XSL (*eXtensible Stylesheet Language*): establece el lenguaje de estilo del documento XML permitiendo modificar el aspecto del mismo. Permite la visualización de tablas, tipos y tamaños de letra diferentes. Es más potente que las hojas de estilo CSS. Utiliza:

-

XSLT (*XML Stylesheets Transformation Language*, o lenguaje de transformación basado en hojas de estilo) que es un lenguaje utilizado para convertir documentos XML en otros documentos XML o en otros formatos diferentes. Por ejemplo XSLT puede convertir un XML con un DTD a otro que siga un DTD diferente, convertirlo a un formato de presentación como

WML o HTML.

-

XPath (*XML Path Language*) que es un lenguaje que permite navegar dentro de un documento XML y, para ello, permite construir expresiones que recorren y procesan el documento. Algo así como expresiones regulares que buscan y seleccionan teniendo en cuenta la estructura jerárquica del documento XML. Por ejemplo con XPath se puede seleccionar y referenciar texto, elementos, atributos o cualquier información contenida en el documento XML.

-

XSL-FO (*XSL Formatting Objects*) que permite especificar el formato visual con el que se quiere presentar un documento XML. Se utiliza, sobre todo, para generar documentos PDF.

Docbook: es un lenguaje de creación de textos electrónicos que dispone de DTD propia y se utiliza sobre todo para generar documentación técnica relacionada con aplicaciones informáticas. El standar de Docbook es mantenido y actualizado por el grupo OASIS.

Documentos XML

Los documentos XML tienen una estructura lógica y una física.

-

La **estructura lógica** consta de un conjunto de declaraciones, elementos, comentarios, etc. incluidos en el documento mediante marcas concretas.

-

La **estructura física** consta de una serie de unidades llamadas entidades[3] que indican los datos que contendrá el documento.

Ambas estructuras, lógica y física deben encajarse correctamente.

Veamos un primer ejemplo de documento XML y su estructura básica. Para su creación necesitamos un editor de texto, tipo *Bloc de Notas* de Windows o *gedit* de GNU/Linux. En cualquiera de los dos editores introducir el código siguiente:

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

<!DOCTYPE chapter PUBLIC "-//OASIS//DTD DocBook XML V4.2//EN" "docbookx.dtd">

<!--Nuestro primer capítulo del libro-->

<chapter lang="es" id="capitulo_1">

<title>Introducción a XML.</title>

<sect1><title>Definición de XML</title>

<para><application>XML</application> es un lenguaje de descripción de documentos.</para>

</sect1>
```

```
</chapter>
```

Un capítulo en la realidad deberá tener mas contenido organizado en secciones. Esto es sólo un ejemplo muy concreto y en él observamos una serie de etiquetas e informaciones importantes de la estructura lógica que vamos a describir :

1ª línea:

Es el **prólogo** del documento y en él se establece que es xml mediante la etiqueta **<?xml .....?** >.
Dentro de esta etiqueta debe ir todo en minúsculas.

'*version*' indica la versión de xml actual utilizada, que es la **1.0.**

'*encoding*' indica el conjunto de caracteres utilizado. En el ejemplo es **utf-8** (unicode) que representa el conjunto de caracteres universal.

Otros ejemplos de prólogos son los siguientes:

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

```
<!DOCTYPE libro SYSTEM "libro.dtd">
```

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

```
<!DOCTYPE libro SYSTEM "http://www.pruebasxml/libro.dtd">
```

2ª línea:

Identifica el tipo de documento, es decir el DTD utilizado. En el ejemplo se trata de un capítulo de un libro (*book*) utilizando la DTD Docbook[4] que es pública (PUBLIC) es decir, con validez en cualquier sistema.

4ª línea y siguientes:

Constituyen el cuerpo del documento y en él se podrán utilizar aquellas etiquetas disponibles desde el DTD indicado, o creadas por el usuario e incluidas correctamente en el DTD. En nuestro caso utilizamos la etiqueta <chapter></chapter> indicando que es un capítulo de un libro y dentro de él definimos diferentes niveles de secciones (<sect1></sect1>; <sect2></sect2>;...). Los contenidos del capítulo propiamente son párrafos (<para></para>).

Hay que tener en cuenta que XML diferencia entre mayúsculas y minúsculas. Por ejemplo, XML tratará como etiquetas diferentes <sect1> y<Sect1>.

Etiquetas

Existen seis tipos de etiquetas: elementos, referencias a entidades, comentarios, instrucciones de procesamiento, secciones de datos y declaraciones de tipo de documento.

-

elementos: están delimitados por ángulos (<,>) e identifican el contenido que delimitan. Pueden tener atributos. Siguen la estructura:

<elemento atributo="valor">

<chapter lang="es" id="capitulo_1">

-

referencias a entidades: empiezan por "&" y acaban con "**;**". Las entidades predefinidas son las siguientes:

Entidad	Carácter
<	**< (menor)**
>	**> (mayor)**

&

& (ampersand)

'

' (apóstrofe)

"

“	**(dobles comillas)**

-

comentarios: no forman parte del texto. Siguen la estructura:

<!-- comentarios -->

```
<!--Nuestro primer capítulo del libro-->
```

- **instrucciones de procesamiento**: se utilizan para proporcionar información en un documento XML. El parser pasa esa información a la aplicación que realiza la llamada. Las instrucciones de procesamiento se utilizan principalmente para indicar a la aplicación el modo de administrar los datos del documento XML. Siguen la estructura:

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

En este ejemplo, la instrucción de procesamiento indica que el documento XML se debe mostrar utilizando una hoja de estilos XSL:

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

- **secciones de datos**: indica al *parser* que ignore el contenido de la sección. Siguen la estructura: **<![CDATA[Texto]]>**

```
<![CDATA[ Esto es una sección de datos en la que podemos escribir sin que el parser lo analice.]]>
```

DTD

La definición de tipo de documento (DTD) es una descripción de estructura y sintaxis de un documento XML[5]. Una DTD describe:

- **Elementos**: indican qué etiquetas son permitidas y el contenido de dichas etiquetas.
- **Estructura**: indica el orden en qué van las etiquetas en el documento.
- **Anidamiento**: indica qué etiquetas van dentro de otras.

Una DTD no es mas que un archivo de texto con extensión **.dtd** cuyo contenido es un elemento raíz o tipo y una descripción de todos los elementos que intervienen en dicho elemento raíz.

Por ejemplo, para el caso del capítulo del libro se está haciendo uso de la DTD **docbookx.dtd** cuyo contenido está predefinido y es algo complejo. Pero un usuario puede definir su propia DTD.

Supongamos que queremos crear una DTD para el tipo libro. Se asume que un libro tiene título, autor y una serie de capítulos a su vez con título y texto. La DTD **libro.dtd** podría ser:

```
<!ELEMENT libro (autor,titulo,capitulo+)>

<!ELEMENT autor (#PCDATA)>

<!ELEMENT titulo (#PCDATA)>
```

```
<!ELEMENT capitulo (titulocap,texto)>

<!ELEMENT titulocap (#PCDATA)>

<!ELEMENT texto (#PCDATA)>
```

donde:

```

<!ELEMENT libro (autor,titulo,capitulo+)>
```

define la etiqueta *libro* que es el elemento raíz. Esta definición comienza con <!ELEMENT .... Los nombres encerrados entre paréntesis indican que un libro consta de un *autor*, un *titulo* y uno o más *capitulo* (+).
Forman un grupo de elementos separados sólo por (,) si se quiere indicar que deben aparecer todos ellos de forma obligatoria (y). Si se quiere indicar opcionalidad deben ir separados por (|).

Si un elemento aparece con (*?*) indica que puede aparecer o no ese dato en el *libro*.

El orden en el que aparecen los elementos establece su orden de utilización en el documento.

```
<!ELEMENT capitulo (titulocap,texto)>
```

La línea anterior define la etiqueta *capitulo* con el elemento *titulocap y texto.*

El resto de líneas definen los elementos de forma ordenada, indicando entre paréntesis **(#PCDATA)** que significa que el elemento puede contener datos de tipo carácter (**P**arser **C**haracter **Data**).

```
<!ELEMENT autor (#PCDATA)>

<!ELEMENT titulo (#PCDATA)>

<!ELEMENT titulocap (#PCDATA)>

<!ELEMENT texto (#PCDATA)>
```

Es posible también utilizar en la definición de los elementos el carácter (*) que indica que éste elemento se puede utilizar todas las veces que se quiera, pero no es obligatorio. Se diferencia del símbolo (+) en que éste indica que, al menos, se ha de utilizar una o varias veces[6].

El siguiente paso es referenciar el uso de nuestra DTD en el código XML. Esto se hace en la línea de identificación de tipo de documento:

```
<!DOCTYPE libro SYSTEM "libro.dtd">
```

En ella se indica cuál es el elemento raíz (*libro*) del documento y con **SYSTEM** se indica que la validez de la DTD es sólo local, para nuestros documentos XML.

libro.dtd es la DTD propiamente y debe incluirse, en general, con el path absoluto.

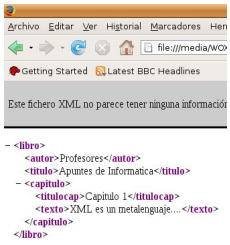
Ahora creamos un pequeño documento XML que utilice la DTD libro.dtd:

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

<!DOCTYPE libro SYSTEM "libro.dtd">
```

```
<libro>

<autor>Profesores</autor>

<titulo>Apuntes de Informatica</titulo>

<capitulo>

<titulocap>Capitulo 1</titulocap>

<texto>XML es un metalenguaje....</texto>

</capitulo>

</libro>
```

¿Qué aspecto tendría el documento escrito utilizando nuestra DTD? En el caso de utilizar el navegador Internet Explorer:

En el caso de abrir el archivo desde el navegador Mozilla Firefox:



En cualquiera de los dos navegadores lo que se muestra es el propio archivo .xml sin ningún tipo de interpretación o transformación.

Este sería el caso de utilización de una DTD externa, es decir, guardada en un archivo **.dtd**. También se puede incluir directamente la DTD en el archivo
.xml
de la siguiente forma:

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

<!DOCTYPE libro SYSTEM [

<!ELEMENT libro (autor,titulo,capitulo+)>

<!ELEMENT autor (#PCDATA)>
```

```
<!ELEMENT titulo (#PCDATA)>

<!ELEMENT capitulo (titulocap,texto)>

<!ELEMENT titulocap (#PCDATA)>

<!ELEMENT texto (#PCDATA)>

]>

<!--Nuestro primer capítulo del libro-->

<libro>

<autor>Profesores</autor>

<titulo>Apuntes de Informatica</titulo>

<capitulo>

<titulocap>Capitulo 1</titulocap>

<texto>XML es un metalenguaje....</texto>
```

```
</capitulo>

</libro>
```

En general, si se quiere reutilizar la DTD es preferible hacerla externa.

En el ejemplo del libro es posible tener el documento XML distribuido en varios archivos:

```

<!DOCTYPE libro SYSTEM "libro.dtd" [

<!ENTITY indice SYSTEM "indice.xml">

<!ENTITY capitulo1 SYSTEM "./capitulos/capitulo1.xml">

<!ENTITY capitulo2 SYSTEM "./capitulos/capitulo2.xml">

]>

<libro>

<autor>Profesores</autor>
```

```
<titulo>Libro de pruebas</titulo>

<cabecera>

&indice;

</cabecera>

<cuerpo>

&capitulo1;

&capitulo2;

</cuerpo>

</libro>
```

Lógicamente habría que redefinir el elemento raíz para dar cabida a los diferentes capítulos del libro e incorporar las nuevas etiquetas <cabecera></cabecera> y <cuerpo></cuerpo>.

En este ejemplo aparece el término **ENTITY** ... y ¿qué son las entidades?

Entidades

Las entidades son tipos de objetos que permiten representar caracteres que no se pueden incluir como texto, como caracteres especiales, etc.

Siguen la sintaxis siguiente: **<!ENTITY identificador "valor">**

Pero, además, las entidades permiten que, al ser referenciadas, el procesador reemplaze el identificador por el contenido asignado como valor, que puede incluir marcado.

Por ejemplo, si en un documento necesitamos hacer referencia a menudo a un enlace, como http://www.isftic.mepsyd.es/
podemos crear un entidad cuyo identificador sea
isftic
y su valor sea la propia referencia en HTML (etiqueta <a>, con atributo href). La entidad quedaría definida de la forma siguiente:

```
<!ENTITY isftic "<a href='http://www.isftic.mepsyd.es">isftic</a>">
```

Si necesitemos incluir muchas veces este enlace en un documento, en vez de escribir cada vez la estructura <a>, podemos referenciar la entidad creada de la forma &isftic;, que funcionará como abreviatura de la expresión completa. Cada vez que en el texto escribamos &isftic; se sustituirá por el link isftic.

Otro ejemplo de utilización de entidades es el ejemplo anterior en el que conseguimos incluir el contenido de un archivo completo a través de la entidad.

La sintaxis en este caso es:

```
<!ENTITY identificador SYSTEM "nombre_archivo" >
```

Por ejemplo, si nuestro libro tiene un índice y dos capítulos definimos tres entidades y las referenciamos de la forma siguiente:

```
<!DOCTYPE libro SYSTEM "libro.dtd" [

<!ENTITY indice SYSTEM "indice.xml">

<!ENTITY capitulo1 SYSTEM "./capitulos/capitulo1.xml">

<!ENTITY capitulo2 SYSTEM "./capitulos/capitulo2.xml">

]>

<libro>
```

```
.....

<cabecera>

&indice;

</cabecera>

<cuerpo>

&capitulo1;

&capitulo2;

</cuerpo>

</libro>
```

En este caso las referencias &indice, &capitulo1 y &capitulo2 se sustituirán por los archivos .xml correspondientes.

Validación de documentos XML

En general un documento es válido si sigue el patrón establecido por la DTD. Si, por ejemplo, no establecemos DTD y se utiliza en vez de<capitulo>, <capitulos>, el archivo seguirá estando

disponible y podrá estar bien formado, pero no será válido.

Los programas que se encargan de comprobar o validar el código xml en función de las reglas establecidas se llaman *parsers validadores (analizador sintáctico)*. Existen muchos disponibles tanto para Windows como para GNU/Linux.

Si se decide no utilizar una DTD entonces se podrán generar documentos 'bien formados' pero no '*válidos*'. Un documento decimos que está '*bien formado*' cuando:

- Contiene información de la versión XML (normalmente la 1.0).
- Indica la codificación de caracteres utilizada (aunque es opcional): normalmente se utiliza UTF-8 o ISO-8859-1.
- Sólo existe **un elemento raíz** para cada documento.
- Todos los elementos están delimitados por una etiqueta inicial y otra final con el mismo nombre, excepto aquellas que corresponden a elementos vacíos, que acaban con />.
- Todos los valores de los atributos están entrecomillados.
- Todas las entidades están declaradas.

Los documentos '*válidos*' además de estar '*bien formados*' deben cumplir las reglas establecidas en la DTD. El proceso de validación realizada por un
parser
validador comprueba:

-

Qué elementos o atributos se permiten en el documento.

-

La estructura de los elementos y atributos (elementos anidados, atributos obligatorios u opcionales, etc.)

-

El orden de los elementos.

-

Los valores de los datos de los atributos y elementos, como por ejemplo, que no estén fuera de rango, valores nulos, formatos de fecha correctos, etc.

Aunque la situación ideal es utilizar un validador integrado en el editor XML es posible que, en determinadas circunstancias, interese disponer de algún programa validador específico.

Por ejemplo, en el caso de necesitar validar XML directamente desde Internet se puede acceder a sitios concretos que realizarán esta validación en línea. Es el caso de la web http://validator.w3.org
.

El validador disponible en http://www.stg.brown.edu/service/xmlvalid/ permite validar también pequeños documentos XML tanto si están o no en la web.

Editores XML

Una vez conocida la estructura de un documento XML y vistos algunos ejemplos vamos a utilizar editores XML específicos que facilitan la tarea de edición.

Los editores XML se pueden agrupar en:

1.

Editores de texto adaptados a XML

2.

Herramientas de edición gráfica

Editores de texto adaptados a XML

La funcionalidad básica que debe ofrecer un editor de texto para XML es la siguiente:

1.

Marcar con diferentes colores los elementos, y así se distingue rápidamente un elemento de otro y permite ver errores.

2.

Autocompletar cuando se comienza a escribir un elemento, dando a opción a seleccionar el elemento de la lista presentada que comienzan igual.

3.

Incorporar sangría de forma automática a elementos diferentes ya que, de esta forma, se repasa visualmente el código mejor.

4.

Validación de DTD dentro del mismo proceso de generación de código XML, que facilita la localización y corrección de errores.

Editor jEdit

Dentro de este grupo el editor mas conocido es **jEdit** que está escrito en Java y diseñado como entorno de desarrollo de programas. Tiene licencia GPL. La web oficial del proyecto es http://www.jedit.org.

Para trabajar con jEdit se debe disponer del entorno Java Runtime, descargable desde http://www.java.com/es/
(Descarga gratuita de Java).

A continuación hay que descargar e instalar la última versión estable de jEdit desde su página oficial. En el momento de escribir el artículo es la 4.2. Descargar y ejecutar jEdit.

Para ser utilizado como editor de XML requiere la instalación de algunos plugins específicos. Describimos su configuración desde una instalación hecha en Windows XP.

1.

Ir a la opción de Menú:

Plugins *Plugins Manager* *Install* *Download options*

Actualizar la lista de plugins disponibles y buscar e instalar uno llamado 'XML'. Este plugin arrastra a otros necesarios para trabajar con documentos XML. Indicar el mirror desde el cual se realiza la descarga.

Instalar también los plugins Error List y SideKick.

1.

Ir a *Plugins* *Plugins Options* y en 'Error List' seleccionar la opción 'Automatically display o-n error' que permitirá en una ventana ir viendo los errores a medida que se vayan produciendo.

2.

Ir a *Plugins* *Plugins Options* y en '*SideKick* *General*' seleccionar la opción 'Parse o-n Keystroke' que determina el tiempo transcurrido desde que se termina de escribir hasta que se inicia la revisión. Normalmente ½ o 1 segundo son suficientes.

3.

Ir a *Plugins* *Plugins Options* y en '*XML* *General*' comprobar que la opción 'Validate if DTD or schema available' está activada.

4.

Salir con OK.

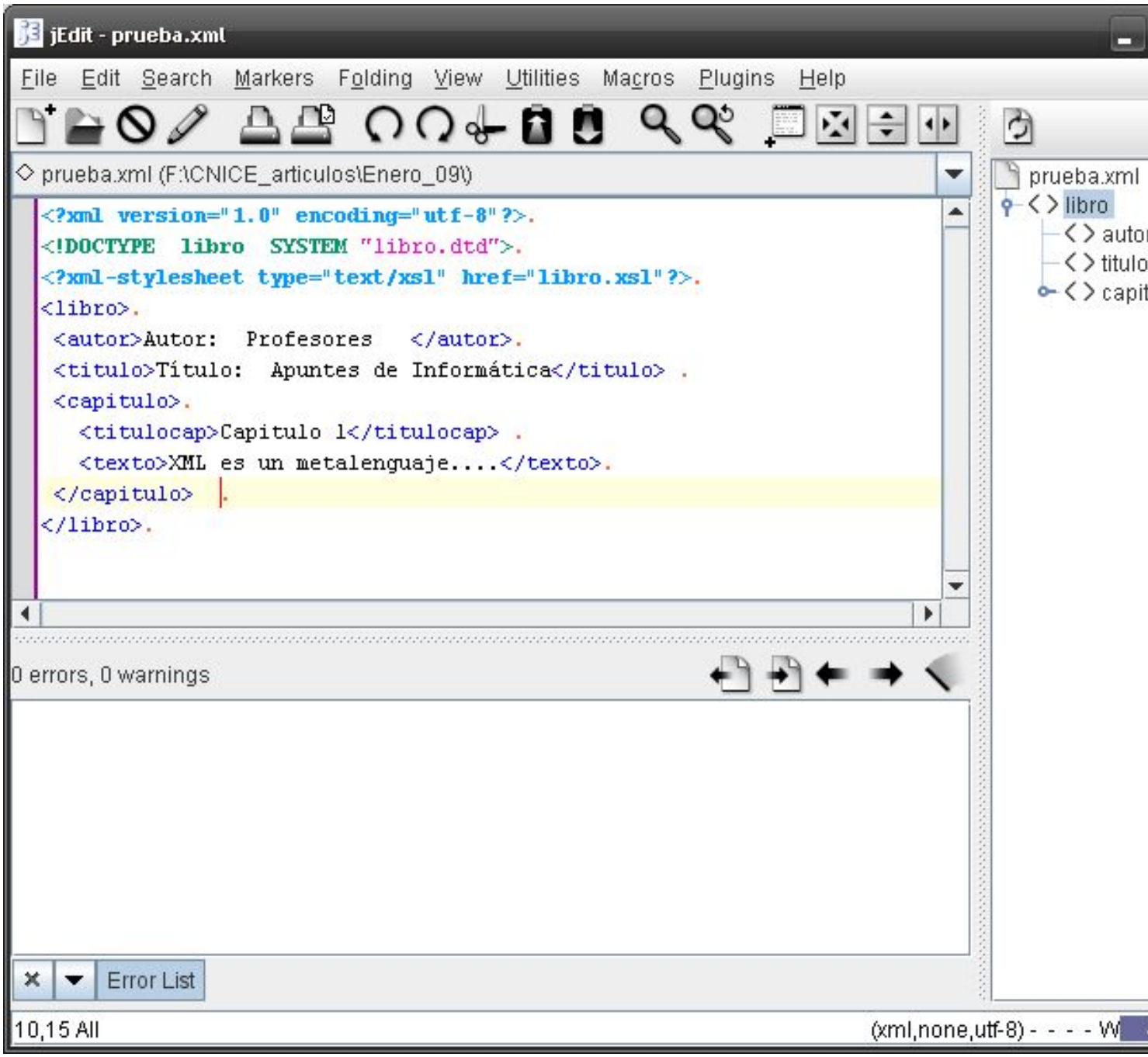
5.

Ir a *Plugins* *ErrorList* *ErrorList*. Esto abre una pequeña ventana que hay que ajustar en jEdit. Pulsar en la marca negra situada en la esquina superior izquierda y seleccionar la opción 'Dock at Bottom' para situar dicha ventana en la parte inferior de jEdit.

6.

Ir a *Plugins* *SideKick* *Structure Browser* y la ventana mostrada ajustarla a la izquierda o derecha de jEdit.

Ahora ya está preparado el entorno para trabajar con documentos XML.



Herramientas de edición gráfica

Las herramientas de edición gráfica no muestran las marcas XML y representan el documento utilizando estructuras en árbol o cajas anidadas, texto con colores, etc.

Existen muchos editores XML gráficos y en general todos ellos permiten verificar que el documento está '*bien formado*' y es '*válido*' para una DTD.

Los siguientes son algunos de los editores gráficos XML más importantes:

-

Open Source: **butterflyXML** (http://sourceforge.net/projects/butterflyxml) y **conglomerate** (http://www.conglomerate.org/), ambas con soporte para Docbook. El problema de ambas herramientas es que las últimas versiones estables son algo antiguas.

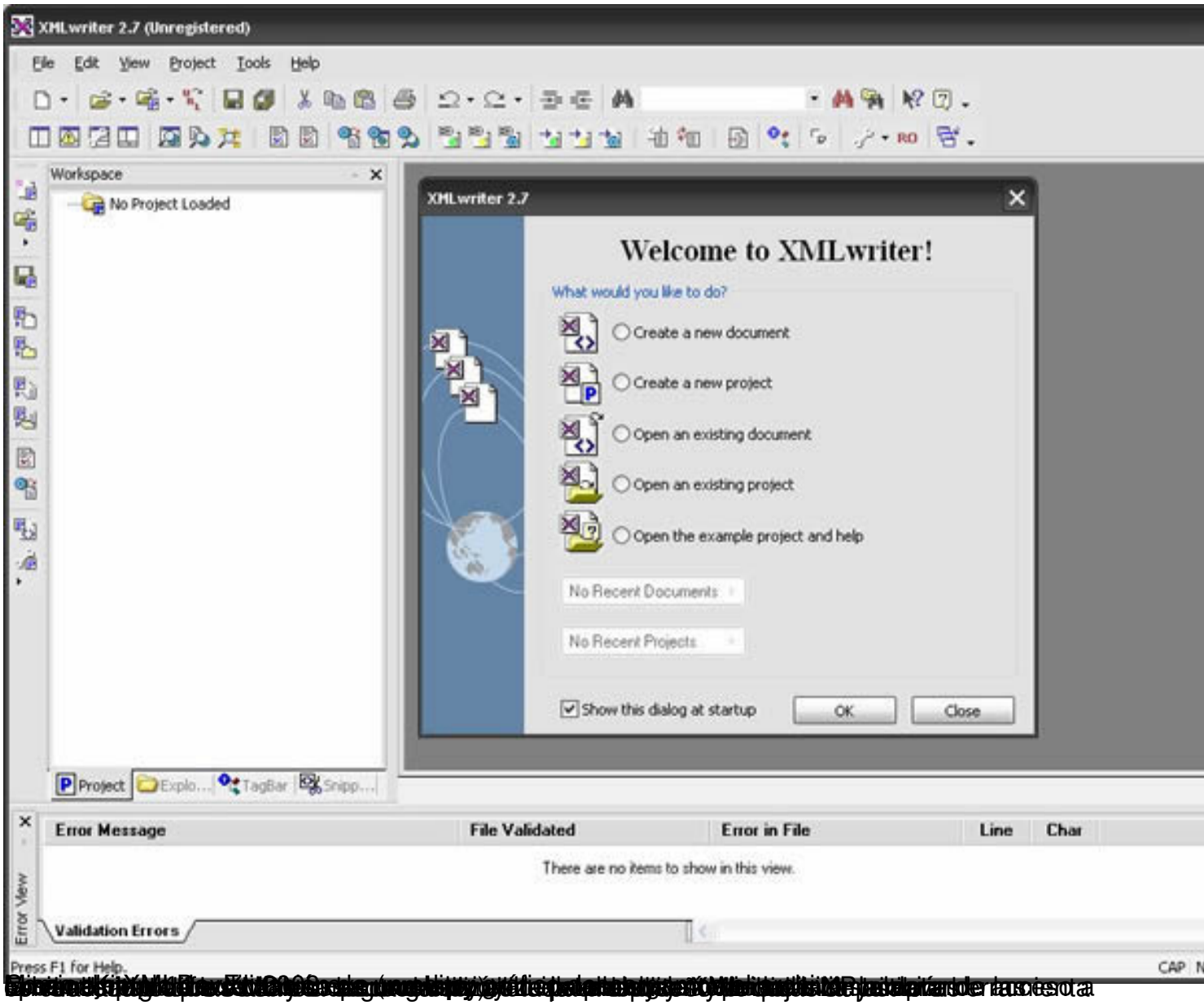
También está **Quanta+**, que es una herramienta de desarrollo de páginas web para KDE y forma parte del paquete kdewebdev. Es ampliable mediante plugins.

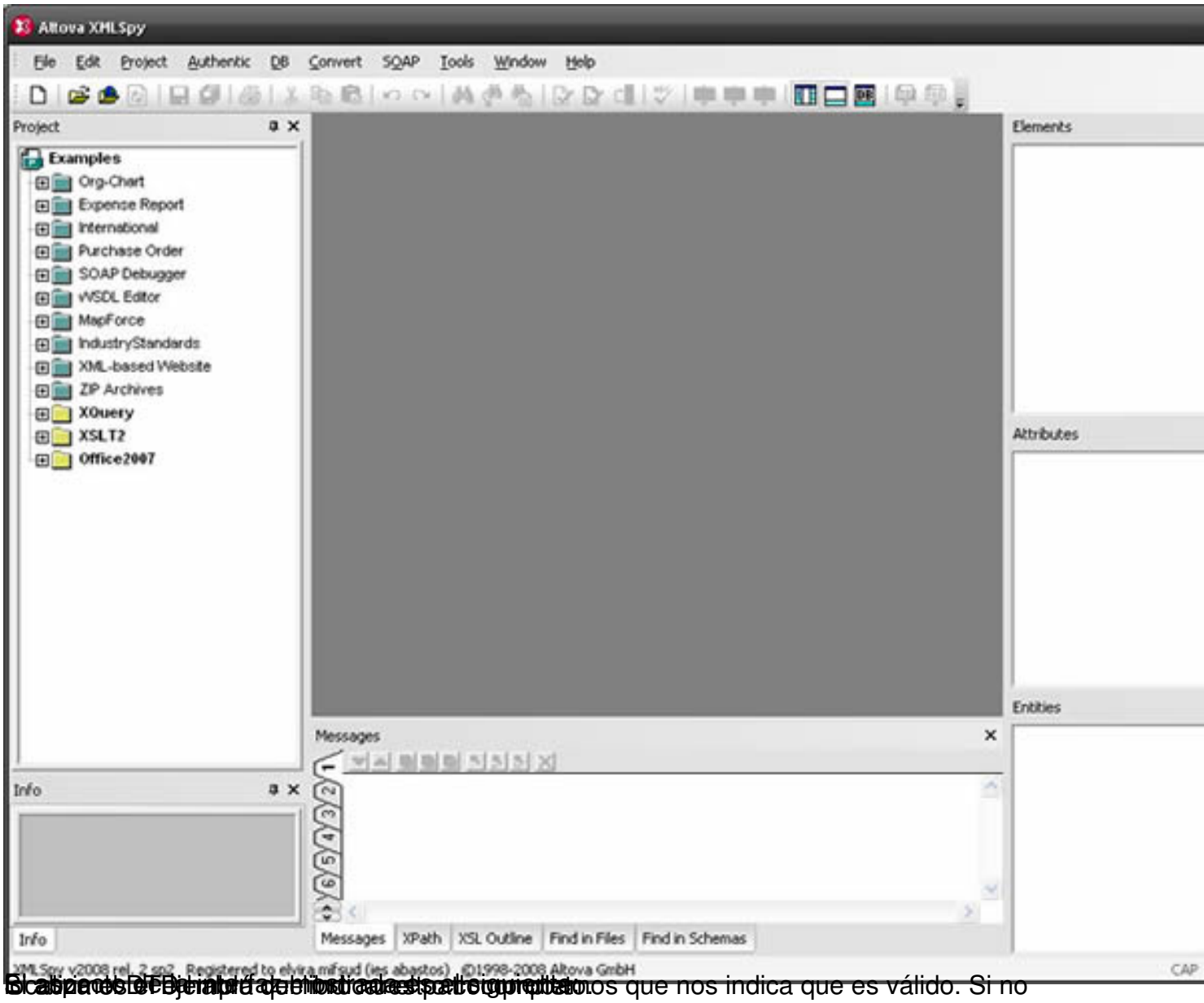
Otra opción libre es el editor **XMLcopyeditor** disponible tanto para GNU/Linux como Windows.

-

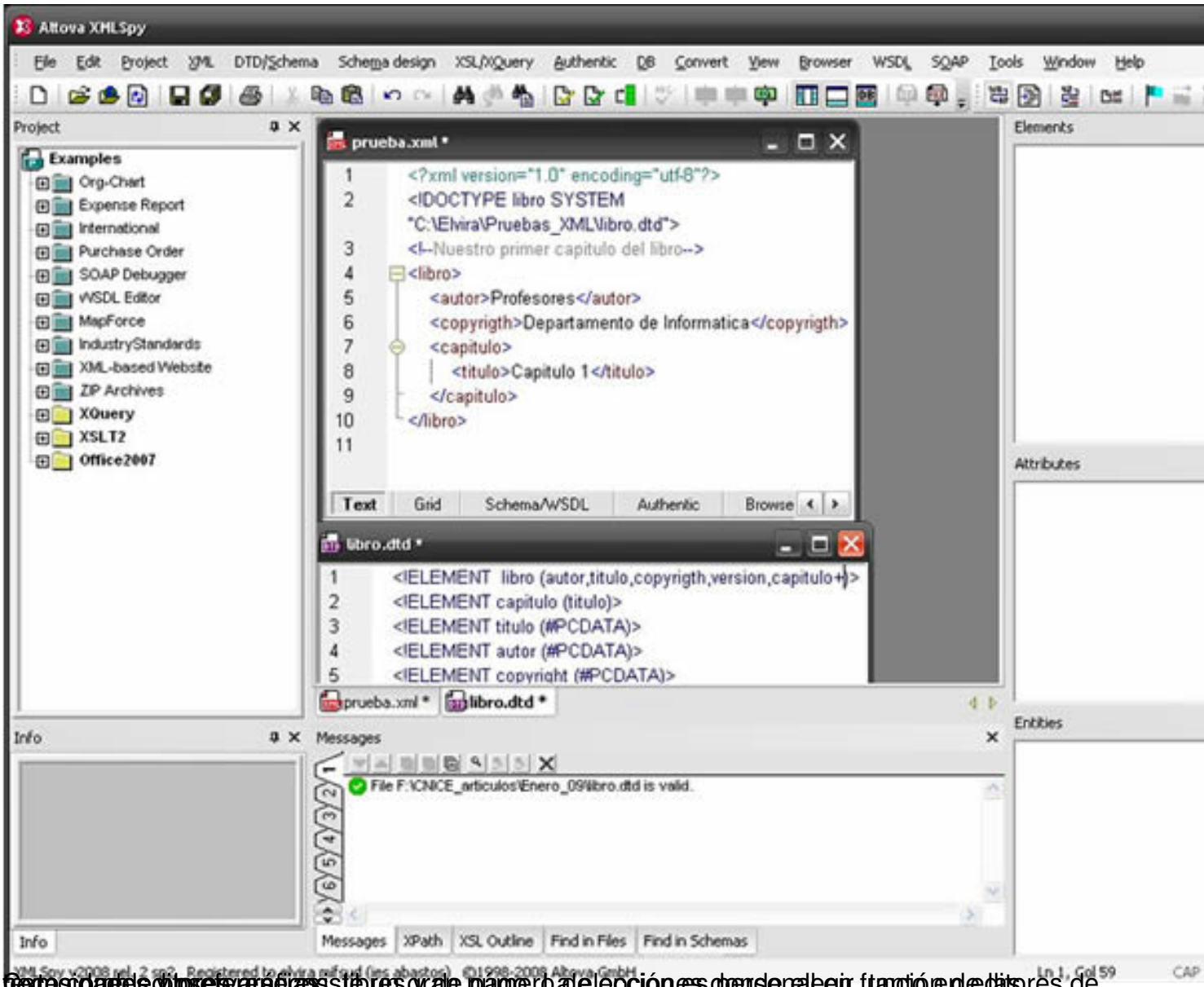
De pago: **xmlspy** (http://www.altova.com) es uno de los editores XML mas utilizados. Permite la edición gráfica de documentos XML, validación de documentos, editor de DTDs, editor CSS y XSL y otras muchas opciones relacionadas con un entorno de desarrollo.

Otra opción de pago para Windows es **XMLwriter** (http://xmlwriter.net) que proporciona validación de documentos XML contra un DTD, conversión de XML a HTML usando hojas de estilo XSLT o bien combinación de CSS con XML para formateo directo de datos XML. También puede configurarse para ejecutar validadores externos desde dentro del programa. Existe una versión free. El aspecto de la interfaz es el siguiente:


XMLwriter 2.7 (Unregistered)
File Edit View Project Tools Help
Workspace
No Project Loaded
XMLwriter 2.7
Welcome to XMLwriter!
What would you like to do?
Create a new document
Create a new project
Open an existing document
Open an existing project
Open the example project and help
No Recent Documents
No Recent Projects
Show this dialog at startup
OK
Close
Project
Explo...
TagBar
Snipp...
Error Message
File Validated
Error in File
Line
Char
There are no items to show in this view.
Error View
Validation Errors
Press F1 for Help.
CAP



Si abrimos el ejemplo que nos indica que es válido. Si no



Como podemos observar, tras crear un nuevo documento podemos elegir entre varias opciones para generar el mismo: texto, gráfico, tablas, ... En nuestro caso elegiremos la opción de texto.

Ejemplos de utilización de XML

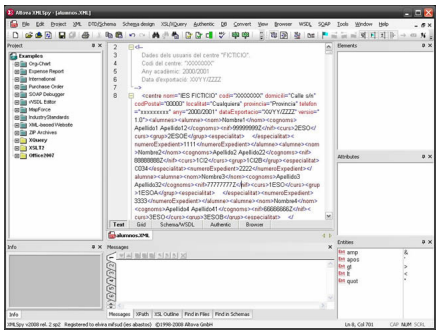
Un ejemplo clásico de documento XML es el relativo a los datos de un mensaje de correo electrónico. Suponemos definida una DTD propia correo.dtd cuya referencia incluimos en el documento:

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

```
<!DOCTYPE correo SYSTEM "correo.dtd">

<correo>

<remite>

<nombre>Profesor de Informática</nombre>

<email>profesor@gmail.com</email>

</remite>

<destinatario>

<nombre>Alumnos 3º ESO</nombre>

<email>alumnos3ESO@gmail.com</email>

</destinatario>

<asunto>Fecha de examen</asunto>

<texto>
```

```
<parrafo>El próximo día XX a las YY horas tendremos el examen de evaluación...</parrafo>

</texto>

</correo>
```

Otro ejemplo típico puede ser el archivo XML generado por la exportación desde una BBDD para ser importado por otra BBDD. En este ejemplo se trata de los datos de la matrícula de un centro hipotético[7] que se necesitan para dar de alta, de forma automática en otra aplicación, a dichos alumnos. El archivo se muestra desde la interfaz de xmlspy.



Utilización de CSS

El objetivo de este punto es aprender a dar formato a los documentos XML para su correcta visualización en un navegador. Para ello utilizamos CSS.

Utilizaremos como ejemplo base el del libro.

```
<libro>

<autor>Profesores</autor>

<titulo>Apuntes de Informatica</titulo>

<capitulo>

<titulocap>Capitulo 1</titulocap>

<texto>XML es un metalenguaje....</texto>

</capitulo>

</libro>
```

El elemento raíz es libro y contiene las etiquetas *titulo*, *autor* y *capitulo*.

1. A todo el documento queremos asignar el tipo de letra Arial y al elemento *titulo* el tipo de letra Courier New y negrita. ¿Cómo indicamos ésto?. Creamos un archivo
libro.css
y en él escribimos:

```
libro {

font-family: Arial;

}

titulo {

font-family: Courier New;

font-weigth: bold;

}
```

2. Ahora enlazamos el archivo CSS con el documento XML. En el archivo libro.xml añadimos la línea segunda:

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

3. En el navegador actualizamos la vista y comprobamos que el texto mostrado tiene tipos de letras diferentes en función de la etiqueta. En general, todo el archivo se visualizará con Arial excepto aquellas etiquetas que han sido redefinidas con otro tipo de letra u otras características adicionales.

En términos CSS las etiquetas *libro* y *titulo* se llaman *selectores* y todos los atributos asignados a la presentación de los datos contenidos en las etiquetas correspondientes deben ir entre llaves. En el caso de que dos o mas etiquetas requieran el mismo estilo indicar de la forma:

```
autor, titulo {

font-family: Courier New;

font-weigth: bold;

color: green;

}
```

En este caso hemos añadido un color para el texto.

De esta forma podemos establecer el estilo de cada una de las etiquetas y para ello disponemos de una gama de atributos aplicables cuya descripción no son objeto de este

artículo y que pueden consultarse en diferentes webs.

En el caso de que el documento utilice una DTD, el enlace a la hoja de estilos utilizada debe colocarse siempre debajo de dicha referencia:

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

<!DOCTYPE libro SYSTEM "libro.dtd">

```

Utilización de XSL

Aunque no hemos entrado con detalle en CSS si que podemos decir que sus posibilidades son limitadas y que para visualizar los documentos en el navegador como realmente queremos a menudo CSS se queda corto.

En estos casos la solución suele ser utilizar conjuntamente CSS y HTML, para lo cual necesitamos utilizar otro estándar llamado **XSL** (*eXtendible Stylesheet Languaje*, Lenguaje extensible de hojas de estilo) que, desde 2001 es la recomendación oficial de la tripleW (*World Wide Web Consortium*
). Dentro de este estándar nos interesa, especialmente para aplicar transformaciones (formatear) a documentos XML, el subconjunto de dicha especificación, llamado **XSLT**
.

XSLT nos permitirá transformar el documento XML a HTML. También transforma XML a WML (

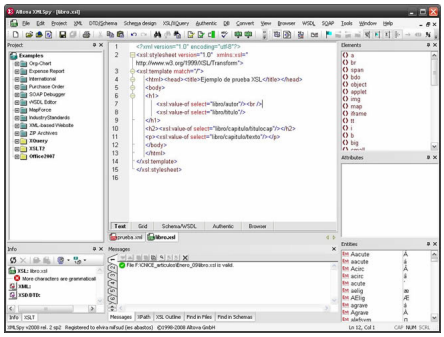
Wireless Markup Language
) para móviles con WAP; a SVG (
Scalable Vector Graphics
) y utilizando
XSL-FO
(
XSL
F
ormatting
O
bjects
) a PDF.

Para ello abrimos el documento XML desde el editor *xmlspy* sobre él hemos de generar una salida en XHTML.

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

<libro>

<autor>Profesores</autor>

<titulo>Apuntes de Informatica</titulo>

<capitulo>

<titulocap>Capitulo 1</titulocap>
```

```
<texto>XML es un metalenguaje....</texto>

</capitulo>

</libro>
```

El archivo de estilo generado **libro.xsl** es el que se muestra abajo.



Donde:

1ª línea: parte obligatoria de xml.

2ª línea: <xsl:stylesheet.... es elemento raíz del archivo .xsl que se cierra en la línea 15. xmlns:xsl es un atributo del elemento raíz donde *xmlns* es el espacio de nombres[8].

3ª línea: <xsl:template.... hace referencia a la plantilla utilizada en la transformación. En el caso de este ejemplo tan sencillo se hace referencia al elemento raíz del documento XML.

4-13 líneas: es un esqueleto HTML típico y dentro de él incluimos sintaxis de XPath para indicar el camino a los elementos XML.

14-15 líneas: cierre de las etiquetas *template* y *stylesheet*.

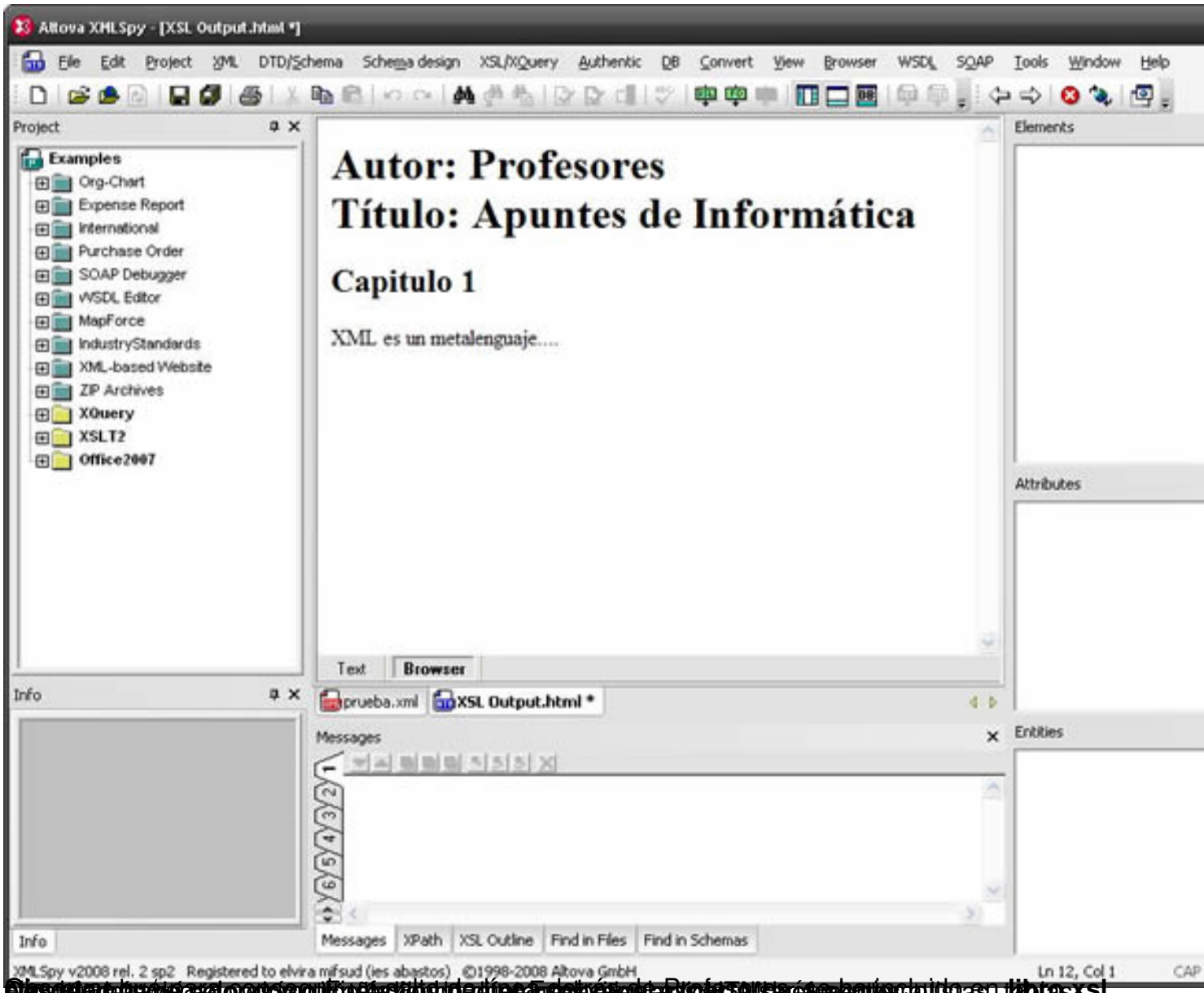
Para validar el documento **libro.xsl** pulsar F8.

El código siguiente será el documento **prueba.xml** con la línea de estilo incluida (libro.xsl) y la referencia a la DTD:

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

<!DOCTYPE libro SYSTEM "libro.dtd">

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

<          libro          >

<          autor          >          Autor: Profes </          autor          >

<          titulo          >          Título: Apunt </          titulo          >
```

<	capitulo	>				
<	titulocap	>	Capitulo 1	</	titulocap	>
<	texto	>	XML es un π	</	texto	>
</	capitulo	>				
</	libro	>				

El aspecto que mostraría un navegador con el estilo asociado se obtiene pulsando F10 (XSL Transformation) es el siguiente:



Conclusión

En realidad con este artículo sobre XML sólo hemos abierto una puerta al mundo de los metalenguajes de marcas. Hay vida mas allá del HTML y es interesante conocer estos estándares que nos permiten crear archivos entendibles por humanos y procesables por las máquinas independientemente de la plataforma.

Pero estamos hablando de estándares del año 1997, es decir diez años en los que han surgido nuevas propuestas que están dejando en evidencia algunos de los problemas del XML. Entre ellos cabe destacar el hecho de que el código XML no es directamente utilizable desde ningún lenguaje de programación.

Sabemos que XML fue una revolución en sus orígenes ya que puso orden dentro del mundo de los lenguajes de marcas. Pero desde hace algún tiempo lenguajes como JSON están tirando fuerte ya que presentan todas las ventajas de XML y además es código directamente utilizable desde Javascript, por ejemplo.

No sabemos ahora mismo cuál será el futuro de XML y si será desplazado por JSON. En este momento avanzan en paralelo, XML como generador de lenguajes ya establecido y de probada eficiencia y JSON con muy buenas perspectivas y resultados en su corta andadura.

Notas

[1] Documento bien formado (*well-formed*): es el que cumple la especificación XML 1.0 (es sintácticamente correcto).

Documento válido (*valid*): es el que cumple una estructura predefinida en un DTD.

[2]Metadatos: datos que describen el contenido de otros datos.

[3]Las entidades se explican con detalle en el punto 3.2

[4]Docbook → es una DTD SGML/XML utilizada principalmente para documentación técnica.

[5]Definición obtenida de http://es.wikipedia.org

[6]Recordar que el sentido de estos caracteres es paralelo a los comodines utilizados en las expresiones regulares.

[7]Los datos reales se han eliminado por cuestiones legales.

[8]Espacio en nombres XML: se utiliza para diferenciar entre elementos XML con el mismo nombre o par