Introducción al XML

Jaime E. Villate.
Universidad de Oporto
villate@fe.up.pt

5 de mayo de 2001

Resumen

Estas notas han sido preparadas para el Seminario sobre Programación en entorno GNU/Linux, en la universidad Rey Juan Carlos, Madrid, España. El estándar XML ha ganado mucha popularidad recientemente, debido a su gran utilidad para estructurar información y por ser un estándar abierto y bastante difundido.

1 Introducción

XML significa lenguaje de marcas generalizado (Extensible Markup Language). Es un lenguaje usado para estructurar información en un documento o en general en cualquier fichero que contenga texto, como por ejemplo ficheros de configuración de un programa o una tabla de datos. Ha ganado muchísima popularidad en los últimos años debido a ser un estándar abierto y libre, creado por el Consórcio World Wide Web, W3C (los creadores de la www), en colaboración con un panel que incluye representantes de las principales compañías productoras de software.

El XML fue propuesto en 1996, y la primera especificación apareció en 1998. Desde entonces su uso ha tenido un crecimiento acelerado, que se espera que continúe durante los próximos años; hoy en día parece que de repente todo el mundo está usando, o quiere usar, XML.

1.1 Ventajas del XML

Antes de ser lanzado el XML, ya existían otros lenguajes de marcas, como por ejemplo el HTML, basados en el lenguaje generalizado de marcas (SGML). El problema con el SGML es que por ser muy flexible y muy general, se torna difícil el análisis sintáctico de un documento y la especificación de la estructura (que como veremos mas adelante se incluye en otro documento llamado DTD). XML es más exigente que SGML en la sintaxis, lo que hace más fácil la construcción de librerías para procesarlo.

Comparado con otros sistemas usados para crear documentos, el XML tiene la ventaja de poder ser mas exigente en cuanto a la organización del documento, lo cual resulta en documentos

mejor estructurados. Por ejemplo en LaTeX existen también “marcas” que permiten estructurar un documento, por ejemplo identificando el nombre del autor y el título del documento – los comandos \author y \title – sin embargo no existe forma de obligar a los autores de documentos a que usen estas marcas y algunos de ellos pueden introducir el título de forma que aparezca visualmente igual a lo que se obtiene cuando se usa \author y \maketitle, sin usar esos comandos; esto conlleva a problemas cuando queremos extraer de forma automática el título de varios documentos.

Por ser posible exigir la estructura que deben tener un tipo determinado de documentos, se vuelve posible extraer información de varios documentos automáticamente, por ejemplo para crear bases de datos o listados con información sobre todos los documentos.

2 Conceptos básicos

Los ficheros XML son ficheros de texto, que en principio está en código Unicode, pero se pueden usar otros alfabetos como el latin-1. Existen cinco caracteres especiales en XML: los símbolos menor que, <, mayor que, >, las comillas dobles, ”, el apóstrofe ’ y el caracter &. Los símbolos mayor que y menor que se usan para delimitar las marcas que dan la estructura al documento. Cada marca tiene un nombre; veamos un ejemplo: la marca <figura>, que puede tener uno o más *atributos*: <figura fichero=”foto1.jpg”tipo=”jpeg”> tiene dos atributos, “fichero” y “tipo”. Los atributos toman valores que tienen que estar entre comillas o entre apóstrofes.

Cuando sea necesario usar uno de los 5 caracteres especiales en el texto, para evitar que sean interpretados de forma especial se usan las siguientes *entidades*: <, >, ", ', &, para <, >, ”, ’ y &, respectivamente. Esto explica también porque & es un caracter especial: se usa para representar entidades; una entidad es un caracter adicional que no hace parte del alfabeto usado por defecto en el texto (los caracteres especiales obviamente quedan excluidos del alfabeto usado para el texto) comienza por &, seguido del nombre de la entidad e inmediatamente un punto y coma[1].

Una diferencia importante con SGML, y en particular HTML, es que los nombres de las marcas y de sus atributos distinguen entre mayúsculas y minúsculas; <a> y <A> serian dos marcas diferentes. Normalmente se suelen usar únicamente minúsculas para los nombres de las marcas y de sus atributos. Otra diferencia sobresaliente con SGML es que en XML ninguna marca se puede dejar abierta; o sea, por cada marca, por ejemplo deberá existir una marca correspondiente que indica donde termina el contenido de la marca. En el siguiente ejemplo:

```
<refrán>El que mucho abarca, poco aprieta</refrán>
```

El contenido de la marca “refrán” esta claramente delimitado entre <refrán> y </refrán>. Si una marca cualquiera no contiene ningún texto, por ejemplo <hr></hr>, se puede abreviar de la siguiente forma: <hr/>, pero nótese que la primera forma también es válida, en cambio escribir únicamente <hr> o <hr> daría un error.

[1] Ya veremos más adelante que realmente existen otros tipos de entidades.

2.1 Definición del tipo de documento (DTD)

Las posibles marcas que pueden aparecer en un documento XML y los atributos que estas pueden tener, son definidos en un fichero llamado "Definición del Tipo de Documento" (en inglés "Document Type Definition") o simplemente DTD. Cada documento XML debe indicar al comienzo el DTD usado por medio de una marca <!DOCTYPE>; por ejemplo

```
<!DOCTYPE xbel
  PUBLIC "+//IDN python.org//DTD XML Bookmark Exchange Language 1.0//EN//XML"
   "http://www.python.org/topics/xml/dtds/xbel-1.0.dtd">
```

Esta marca indica que lo que viene a continuación en el fichero es una marca "xbel" (con todas sus posibles sub-marcas), que ha sido definida en un DTD que se llama "XML Bookmark Exchange Language 1.0". La palabra clave PUBLIC precede al nombre *oficial* que se le ha dado al DTD respectivo; en este caso esa indicación nos da alguna indicación adicional al nombre del DTD: el símbolo + indica que es un DTD reconocido por alguna entidad oficial, en este caso python.org como lo indica la palabra clave IDN, el lenguaje usado en el DTD es el inglés (EN) y la sintaxis usada es sintaxis XML. Realmente el nombre que viene entre comillas después de PUBLIC es algo arbitrario, pero como en cada sistema existe un catálogo SGML que identifica los DTD disponibles en el sistema, lo importante es usar exactamente el nombre que aparezca en el catálogo. Y para que documentos que usen el mismo DTD puedan ser transportables entre sistemas convienen usar la identificación exacta sugerida por el autor del DTD.

Después del identificador público (lo que está entre comillas después de PUBLIC) puede venir un identificador del sistema que indica el camino y nombre del fichero donde se encuentra el DTD; en el ejemplo anterior el identificador del sistema es una URL que indica donde se puede encontrar el DTD usado.

Un fichero DTD define siempre una o más estructuras jerárquicas, con una marca principal, o padre, compuesta por otras marcas, o hijos. La figura 1 muestra la estructura de un DTD simple, con un elemento principal <article>. Dentro del elemento principal pueden aparecer otros elementos: <artheader>, <sect1> y <index>, y estos a su vez se componen de otros elementos.

El DTD puede ser diseñado de forma a hacer obligatorio el uso de algunos sub-elementos y limitar el número de veces que un elemento puede aparecer y el orden de los elementos. De esta forma el DTD puede ser bastante flexible o tan exigente como se desee, para forzar a los autores a ceñirse a un determinado estilo.

Un documento XML que especifique el DTD usado y siga las reglas en él definidas, se dice que es un documento XML válido. Se pueden también crear documentos que no especifiquen ningún DTD pero que sigan las reglas mínimas del XML; en este caso el documento XML se denomina "conforme" (en inglés conforming); existen programas para comprobar si un documento es válido (comprobar que el DTD existe y que la estructura del documento respeta las reglas definidas por el DTD).

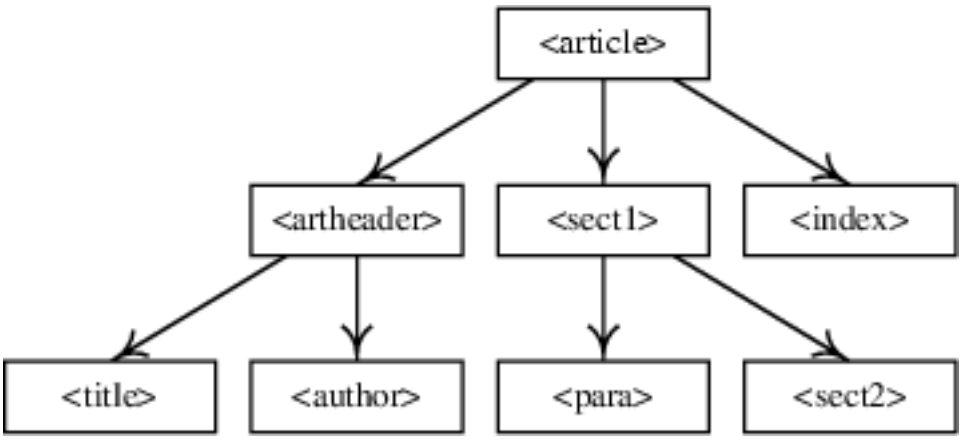

Figura 1: Estructura jerárquica de un DTD.

2.2 Entidades

En la sección anterior ya hablamos de un tipo de entidades que se usan para representar caracteres adicionales al alfabeto usado. La únicas entidades de ese tipo que están pre-definidas en XML son las cinco que ya mencionamos: <, >, ", &, '. Cualquier otra entidad adicional que queramos usar tendrá que estar definida previamente en el DTD usado. Por ejemplo, si estamos usando el alfabeto latin-1, que incluye el caracter ©, pero no sabemos como obtenerlo con el teclado, podremos definir una entidad ©. La definición se hace usando la marca ENTITY, de la siguiente forma

```
<!ENTITY  copy "&#169;">
```

El número decimal 169 es el código que le corresponde al caracter © en el alfabeto latin-1; también podríamos haber usado la representación hexadecimal #xA9;. La definición de una entidad como la anterior puede ya formar parte del DTD, o puede ser adicionada por el autor del documento XML, dentro de la propia declaración del DTD del documento. Por ejemplo la fuente de este manual que está leyendo es un fichero XML con la siguiente definición de tipo de documento

```
<!DOCTYPE article PUBLIC "-//laespiral.org//DTD LE-document 1.1//EN"
   "LE-document-1.1.dtd" [<!ENTITY  copy "&#169;">] >
```

Entre los paréntesis cuadrados pueden ir varias definiciones de entidades.

El valor de una entidad no está limitado a ser un caracter, sino que puede ser cualquier texto. Por ejemplo si definimos la entidad &qed; de la siguiente manera

```
<!ENTITY qed "Que es lo que queríamos demostrar">
```

Cada vez que en el documento se escriba &qed;, será substituido por el texto *Que es lo que queríamos demostrar*.

Una entidad se puede usar también para insertar el contenido completo de un fichero en un punto del documento, si se define de la siguiente forma

```
<!ENTITY nombre SYSTEM "fichero.txt">
```

El fichero puede incluir hasta marcas y cualquier otro texto que sea válido en el punto donde aparezca &nombre;.

Existen otro tipo de entidades internas, que no pueden ser usadas en un documento XML sino únicamente dentro de un DTD. Estas comienzan por el caracter especial % en vez de & en la sección sobre construcción de DTDs hablaremos más sobre ellas.

2.3 Ficheros XML

a los ficheros XML se les suele dar un nombre terminado en .xml para identificarlos como xml. Esto es simplemente una convención para los usuarios; el estándar XML 1.0 indica que para identificar un fichero como XML es necesario que la primera línea tenga el siguiente contenido

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

Dentro de esta marca puede ir otra información adicional. El alfabeto usado por defecto en los ficheros XML es el Unicode; para documentos en español será mas conveniente usar el alfabeto latin-1, lo cual se logra usando el atributo *encoding* de la marca xml

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

3 Construcción de DTD's

El DTD suele estar dentro de un fichero con extensión dtd, pero puede incluso ser definido dentro de la propia marca DOCTYPE en el documento XML. Veamos un ejemplo muy simple de un fichero XML que incluye también el DTD

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<!DOCTYPE cd[
           <!ELEMENT cd (titulo, artista, pista+)>
           <!ATTLIST cd fecha CDATA #IMPLIED>
           <!ELEMENT titulo (#PCDATA)>
           <!ELEMENT artista (#PCDATA)>
           <!ELEMENT pista (#PCDATA)>
                ]>
<cd fecha="2001">
  <titulo>Los poyitos dicen</titulo>
```

```
  <artista>Los niños cantores del Tirol</artista>
  <pista>Pio, Pio, Pio.</pista>
  <pista>Pio, Pio, Pio (versión instrumental).</pista>
  <pista>Pio, Pio, Pio (versión extendida).</pista>
</cd>
```

El elemento principal definido en el DTD es cd, el cual tiene que tener inicialmente una marca titulo, seguida de una marca artista y finalmente seguida de una o mas marcas pista; el símbolo más al lado de la marca pista, en la definición del elemento cd, indica que tiene que aparecer por lo menos una vez. Otros modificares usados son *, que significa cualquier número de veces incluyendo cero, y ? que indica que puede no aparecer o aparecer a lo sumo una vez; si no aparece ningún modificador, la marca respectiva debe aparecer exactamente una vez.

Si queremos que el orden de los sub-elementos título y artista de cd pueda ser arbitrario podemos usar la siguiente construcción

```
<!ELEMENT cd ((titulo | artista)*, pista+)>
```

Pero en este caso estaríamos permitiendo que aparezca más de un título o autor (o ninguno). El elemento cd acepta un atributo llamado fecha. Para indicar que el contenido de un atributo o elemento puede ser una combinación de caracteres del alfabeto usado, empleamos la palabra clave CDATA, en el caso de los atributos, y #PCDATA en el caso de los elementos; otras posibilidades para el tipo de datos de los atributos son NMTOKEN, cuando solo puedan tener valores numéricos, ID cuando sea un código de identificación que tenga un valor único, e IDREF cuando tenga que ser una referencia a un código de identificación ya existente. Los elementos también pueden incluir la palabra clave EMPTY cuando se trate de elementos que no pueden tener ningún contenido.

La palabra clave #IMPLIED indica que el atributo es opcional; si fuera obligatorio se usaría en vez #REQUIRED, y si quisiéramos especificar una lista de posibles valores, se pondrían entre paréntesis, separados por barras verticales, y después de los paréntesis se escribiría el valor por defecto.

Como normalmente estaremos interesados en crear varios documentos con estructura semejante, es mejor colocar el DTD en un fichero separado. El DTD del ejemplo anterior, dentro de un fichero aparte, quedaría así:

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


<!ELEMENT cd (titulo, artista, pista+)>
<!ATTLIST cd fecha CDATA #IMPLIED>
<!ELEMENT titulo (#PCDATA)>
<!ELEMENT artista (#PCDATA)>
<!ELEMENT pista (#PCDATA)>
```

Dentro del fichero del DTD se pueden usar entidades para simplificar su escritura. Por ejemplo, una sección del DTD LE-document-1.1.dtd es la siguiente

```
<!ENTITY % listtype  " itemizedlist | orderedlist | variablelist |
                      simplelist | programlisting | figure | form | table " >

<!ELEMENT article (artheader, (para | sect1 | %listtype;)*, bibliography?) >
<!ATTLIST article lang CDATA #IMPLIED
                  xreflabel CDATA #IMPLIED
                  id ID #IMPLIED
                  parentbook IDREF #IMPLIED>
```

El DTD completo se puede ver en http://www.laespiral.org/xml/styles/LE-document-1.1.dtd

4 DTDs disponibles

La creación de un DTD es una tarea complicada, no por la parte del código que se tiene que escribir, sino porque el diseño de la estructura jerárquica y las marcas usadas es crucial en el éxito de un DTD. en algunas aplicaciones se necesita mucha experiencia para tomar las decisiones acertadas sobre el diseño del DTD. Afortunadamente existen muchos grupos de expertos trabajando en la creación de DTDs públicos que pueden ser usados y libremente.

Un líder en el campo de creación de DTDs es el propio consorcio W3C, que ya tiene varios DTDs disponibles. Algunos de ellos son: SVG para gráficos vectoriales, MathML para ecuaciones matemáticas y XHTML que es una versión XML del DTD de HTML. Otros DTDs importantes creados por otros grupos son DocBook (originalmente en SGML, pero ya en versión XML) para escribir libros, especialmente manuales de software, BioML y BSML para biología, CML para química, AML y AIML para astronomía y TMX para traducciones.

5 Lenguajes de páginas de estilo

Un elemento importante para poder estructurar la información de un documento es separar el contenido del documento de su formatación. Quien esté familiarizado con LaTeX sabrá que una de sus principales ventajas es que permite a los autores concentrarse en el contenido del documento, sin tener que preocuparse mucho con la forma como será presentado. El formato que se usa para presentar el contenido está definido en otro fichero, que define la *documentclass*, que ha sido preparado por un experto, de manera que cualquier autor puede producir documentos de elevada calidad tipográfica sin mucho esfuerzo.

En HTML y XML ha habido también un esfuerzo por separar el contenido de la formatación y dar la posibilidad de reutilizar un formato pre-definido. El formato usado lo define otro fichero llamado una hoja de estilo (en inglés Style Sheet) usando un lenguaje propio para páginas de estilo.

5.1 CSSL

CSSL significa Cascading Style Sheet Language, y es el lenguaje para páginas de estilo desarrollado por el consorcio W3 para ser usado en páginas HTML. Un documento XML puede también hacer uso de una página de estilo CSS, en forma semejante a como se hace en una página HTML. Pero no entraremos en detalles aquí, para concentrarnos en otro estándar mas reciente para páginas de estilo (XSL).

5.2 XSL

El lenguaje de páginas de estilo que ha sido desarrollado por el consorcio W3C, para dar formato a los documentos XML, se llama XSL, que es el acrónimo de Extensible Style-sheet Language. Una página de estilo XSL permite modificar un documento XML, produciendo varios un resultado que puede estar en varios formatos diferentes incluyendo el propio XML y HTML.

Una página de estilo XSL es también un documento XML que usa el DTD xsl:stylesheet. El comienzo de una página usada para producir HTML podría tener el siguiente contenido

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<!DOCTYPE xsl:stylesheet  [<!ENTITY nbsp " ">]>
<xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" encoding="iso-8859-1"
 doctype-public="//w3c//dtd html 4.0 transitional//en"/>
```

En este caso se ha definido una entidad adicional que será usada en el HTML producido, y se ha especificado la información que deberá aparecer en la marca DOCTYPE del fichero HTML que se genere. El *espacio de nombres* usado; o sea la especificación de marcas usadas en XSL, se ha definido con el atributo xmlns:xsl.

Todas las marcas de XSL comienzan con la secuencia xsl:. La marca básica que realiza el procesamiento del fichero XML, es la marca <xsl:template> que define la plantilla que se debe usar para producir la salida de datos. Veamos un ejemplo

```
<xsl:template match="itemizedlist">
  <ul>
  <xsl:apply-templates/>
  </ul>
</xsl:template>
```

Esta plantilla será aplicada cada vez que aparezca una marca <itemizelist> en el documento XML; al texto que aparezca entre <itemizelist> y la correspondiente </itemizelist>. En este caso se usará la marca que crea listas de items en HTML: <ul>. La marca <xsl:apply-templates/> hace que el procesamiento continúe, aplicando todas las otras plantillas que sean relevantes al texto que se ha seleccionado (el contenido de <itemizelist>). Algo importante que se debe tener en cuenta es que aunque queramos que en el fichero HTML de salida aparezca una marca vacía

como por ejemplo
, en la plantilla se debe escribir
, pues la plantilla hace parte de un documento XML; en la salida aparecerá
 pues el resultado se presenta en HTML.

Si la página de estilo tuviera que producir LaTeX en vez de HTML, la plantilla correspondiente a la anterior sería

```
<xsl:template match="itemizedlist">
\begin{itemize}
  <xsl:apply-templates/>
\end{itemize}
</xsl:template>
```

Las páginas XSL usadas para producir las versiones LaTeX y HTML de este documento pueden ser obtenidas en http://www.laespiral.org/xml/styles/

6 Herramientas GNU/Linux para XML

Existen varias herramientas disponibles en GNU/Linux para trabajar con ficheros XML. Muchos programas ya usan XML también como un medio de intercambiar información o como especificación para escribir los ficheros de configuración.

6.1 Catalogo SGML

En un sistema en el que se use XML para crear documentos, conviene que exista un catálogo de los DTD disponibles y la información de donde encontrarlos en el sistema. En Debian GNU/Linux, las herramientas para crear y mantener el catálogo vienen dentro del paquete *sgml-base*; este paquete incluye el programa **install-sgmlcatalog** que será usado por otros paquetes que instalen DTDs, para actualizar el catálogo, que se encuentra localizado en `/etc/sgml.catalog`.

Si por ejemplo quisiéramos instalar el DTD contenido en el fichero LE-document-1.1.dtd, usaríamos el comando

```
install-sgmlcatalog --install catalogo le-document
```

Donde el fichero `catalogo` contendría la siguiente información

```
  -- SGML catalog for LE-document --
PUBLIC "-//laespiral.org//DTD LE-document 1.1//EN" "dtd/LE-document-1.1.dtd"
```

la última parte en el comando install-sgmlcatalog, le-document, es un identificador que nos permitirá después desinstalar el DTD por medio del comando

```
install-sgmlcatalog --remove le-document
```

Con la información anterior introducida en el catálogo, tendríamos que copiar el fichero LE-document-1.1.dtd en /usr/lib/sgml/dtd/, que es donde suelen estar los otros DTDs. Otro paquete importante en Debian es el paquete sgml-data que instala varios DTDs conocidos.

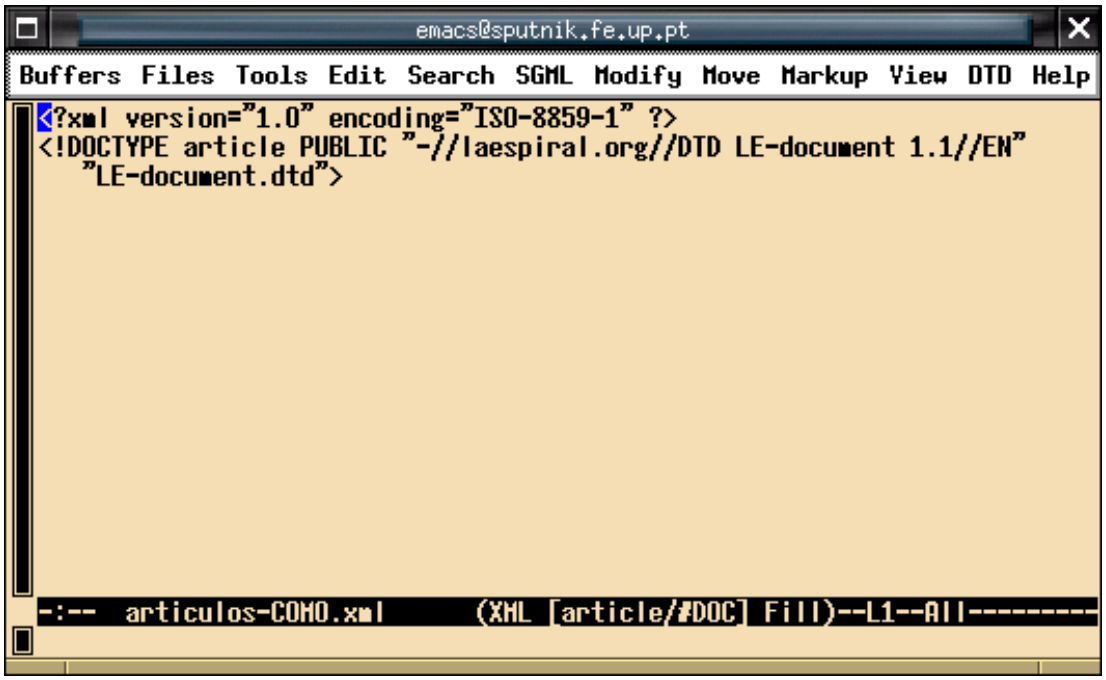

Figura 2: Comienzo de un documento XML usando LE-document.dtd.

6.2 Edición de ficheros XML con Emacs

Existe un paquete psgml que define un modo XML para el editor Emacs. Emacs junto con psgml es bastante útil para editar ficheros XML. Para comenzar a escribir un documento, el primer paso es crear un fichero con extensión xml y con el siguiente contenido en las dos primeras líneas:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE article PUBLIC "-//laespiral.org//DTD LE-document 1.1//EN"
   "http://www.laespiral.org/xml/styles/LE-document-1.1.dtd">
```

En este caso vamos a usar el DTD “LE-document”, y vamos a utilizar caracteres latin-1. El elemento principal en el documento será <article>.

Si se abre con emacs un fichero con extensión xml que tenga las dos líneas anteriores, el resultado será como el de la figura 2

La extensión xml del nombre del fichero ha hecho que emacs entre en el modo XML de *psgml*, como se puede ver en el centro de la *línea de estado* (la línea negra en la parte inferior) y por la aparición de varios menús adicionales para trabajar con XML. El modo XML se ha encargado también de leer la línea que define el DTD y ha cargado el fichero `LE-document-1.1.dtd` (si existe una copia local del DTD, se puede substituir la URL por el camino completo de esa copia).

En la línea de estado en la figura 2 se puede ver que el DTD ya ha sido leído y analizado, pues ya ha sido identificado el elemento principal del documento: “article”; también puede ver alguna

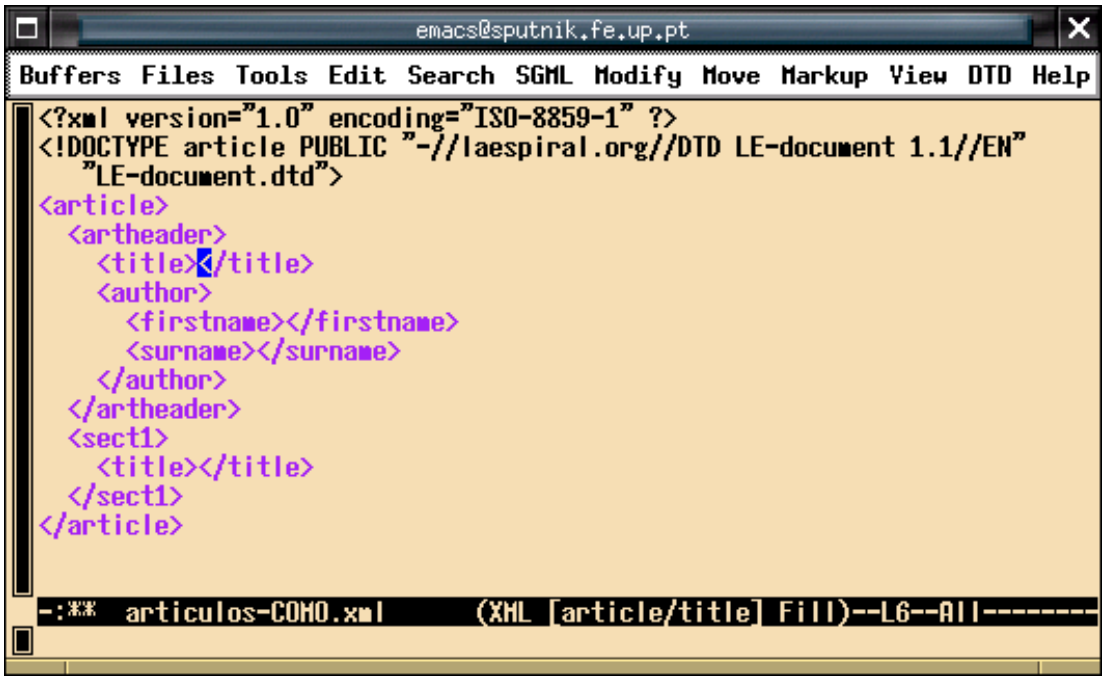

Figura 3: La estructura mínima de un artículo en LE-document.dtd.

información sobre el DTD y los elementos que define, en el menú DTD que presenta psgml en Emacs. Si no aparece esa información, por ejemplo si comenzó a escribir las dos primeras líneas en un fichero vacío, tendrá primero que asegurarse de que está usando el modo XML, con el comando "M-x xml-mode", y después seleccionar la opción "Parse DTD" en el menú DTD (o si prefiere puede usar la secuencia "C-c C-p").

Después de estar en modo XML y de haber seleccionado un DTD, se puede usar una opción de menú muy útil que nos permite escribir el documento rápidamente; se trata de la opción **Insert Element**, en el menú "Markup". Esta opción nos muestra una lista de los elementos que son permitidos en el punto donde se encuentra el cursor; escogiendo un elemento en la lista, son introducidas las etiquetas exigidas por ese elemento y si existe alguna información adicional obligatoria, será pedida en el *mini-búfer* (la última línea en la pantalla).

Otras formas de seleccionar está opción a partir del teclado es por medio de la secuencia "C-c C-e" (TAB mostrará la lista de posibilidades), o oprimiendo simultáneamente la tecla de mayúsculass y el botón derecho del ratón: aparece un cuadro con la lista de elementos válidos; por ejemplo en la figura 2 se acabó de pulsar "Mayúsculas+botón derecho" y ha aparecido una lista con un único elemento, **article**, que es el único elemento que se puede seleccionar inicialmente. Seleccionando este elemento, aparece toda la información que se muestra en la figura 3.

El contenido de la figura 3 es la estructura mínima que debe tener un artículo que use el DTD *LE-document-1.1*. El cursor ha sido desplazado al primer lugar donde se puede comenzar a escribir texto: el título del artículo. Después de escribir el título del artículo, se puede usar la

opción **Next data field**, en el menú **Move** (o con el teclado: "C-c C-d"), para desplazarnos al próximo campo que debe ser rellenado.

6.3 Analizadores sintácticos de XML/XSL

Existen varios analizadores sintácticos de XML que permiten determinar si un documento XML es válido. Algunos ejemplos son *nsgmls* y *rx*. También existen programas que aplican páginas de estilo XSL para transformar documentos XML en otros formatos como por ejemplo HTML; tres ejemplos son *Sablotron*, *Xalan* y *Libxslt*. Por ejemplo, Sablotron ha sido usado para producir versiones LaTeX y HTML de este documento:

```
sabcmd LE-document.xsl curso.xml >curso.html
sabcmd LE-document-latex.xsl curso.xml >curso.tex
```

Usando dos páginas de estilo que se encuentran disponibles en http://www.laespiral.org/xml/styles/[2]. La librería libxsltk, trae el programa **xsltproc** que tiene básicamente las mismas funcionalidades de **sabcmd**.

6.4 Cocoon

El proyecto Apache tiene un grupo dedicado exclusivamente al desarrollo de herramientas XML. Ya han desarrollado programas Java para analizar y transformar XML (Xerces y Xalan) y un servlet llamado Cocoon, que procesa documentos XML y les aplica las transformaciones indicadas por una hoja de estilo XSL para producir HTML. De esta forma se puede configurar el servidor http de Apache para que genere código HTML dinámicamente, a partir de ficheros fuente XML.

6.5 DOM y SAX

Han sido desarrollados dos métodos de analizar sintácticamente un documento XML. En el primer método, DOM (Document Object Model), se lee el documento completo y se identifica su estructura jerárquica. El segundo método, SAX (Standard API for XML), consiste en ir identificando las marcas a medida que se va leyendo el documento. El segundo método es obviamente más rápido y consume menos recursos, pero tiene la desventaja de que cada vez que aparece una marca se debe decidir que hacer con ella, y no se puede regresar para atrás en el documento.

SAX ha sido desarrollado con aplicaciones de servidor en mente; el servidor debe suministrar rápidamente el resultado de transformar un documento XML. DOM fue desarrollado con aplicaciones de cliente en mente; por ejemplo un editor de XML necesita poder navegar en cualquier dirección la estructura del documento; en este caso el método SAX no sería muy útil.

Existen varias librerías disponibles que implementan un u otro método en varios lenguajes de programación diferentes. Veamos un ejemplo de un programa perl que usa el módulo XML::Dom para sacar información de un fichero XML:

[2] Realmente se ha usado un script le2html que modifica algunos caracteres especiales de LaTeX, antes de aplicar la página de estilo XSL usando sabcmd.

```
#! /usr/bin/perl
    use XML::DOM;
    my $fichero = 'fichero.xml';
    my $parser = new XML::DOM::Parser;
    my $doc = $parser->parsefile ($fichero);
    my $titulo = &extraer($doc->getElementsByTagName ("titulo"));
    my $autor = &extraer($doc->getElementsByTagName ("autor"));

sub extraer
{
    my (@elementos) = @_;
    my $elemento = $elementos[0]->toString;
    $elemento =~ s/^[^>]*>\n?//;
    $elemento =~ s/\n?\s*<[^<]*$//;
    return $elemento;
}
```

Este programa lee `fichero.xml` y extrae la información de las marcas titulo y autor. La subrutina extraer elimina las marcas que delimitan cada elemento.

7 Algunas aplicaciones

Son muchísimas las aplicaciones del XML; y con la existencia de librerías para producir, analizar y transformar XML, disponibles para varios lenguajes de programación, cada día se usa más el XML en varios campos muy diversos. En esta sección veremos algunas aplicaciones

7.1 Preparación de documentos

El XML es un sistema muy útil para preparar documentos. Como ya se ha dicho en secciones anteriores, este manual ha sido escrito en XML, al cual se le ha aplicado una hoja de estilo XSL para generar un fichero LaTeX. Existen varias ventajas de usar XML en vez de producir directamente un fichero LaTeX.

- El DTD orienta al autor en los pasos que debe seguir. Como vimos en la sección en que hablamos del uso de Emacs para editar ficheros XML, al introducir el elemento inicial, nos aparece la estructura mínima que debe tener el documento; y en cada parte del documento podemos consultar una lista de las posibles marcas que se pueden usar en esa sección.
- El XML permite ser mas exigente respecto a la estructura del documento, lo que permite una mayor uniformidad entre diferentes documentos.
- Existen programas que permiten revisar un documento XML rápidamente y descubrir errores de sintaxis o de la estructura del documento. en LaTeX se puede revisar la sintaxis, pero descubrir fallas en la estructura es más difícil. Si por ejemplo el autor definió una sub-sección antes de haber definido alguna sección, o si repitió el título en el medio del

documento, el resultado continua siendo un fichero LaTeX válido y ese tipo de errores son difíciles de descubrir de forma automática.

- XML es más fácil de transformar en otros formatos. Por ejemplo, pasar de XML a HTML puede ser hecho fácilmente con una página de estilo XSL.

Sin embargo cuando se trata de documentos con bastante contenido matemático, aún no existe ningún DTD que permita escribir ecuaciones con la facilidad y el poder disponibles en LaTeX y TeX.

7.2 Creación de páginas web

Escribir una página HTML es una tarea fácil. Pero cuando queremos construir un site completo, la labor es mucho mas ardua porque es necesario tener un buen sistema de navegación entre las páginas y tentar crear una imagen de marca que sea consistente en todas las páginas. A medida que el site crece, se va volviendo mas complicado su mantenimiento, y un pequeño cambio puede implicar tener que actualizar varias páginas.

Algunas soluciones adoptadas frecuentemente son php o SSI para introducir información de forma dinámica en las páginas. De esta forma si hay información que es actualizada frecuentemente, esta puede entrar en forma automática en las páginas. Sin embargo, no es muy conveniente convertir todas las páginas en dinámicas, pues cuando el site puede llegar a tornarse muy lento debido a una sobrecarga del servidor. Las páginas dinámicas también pueden ser difíciles de modificar porque pueden ser auténticos programas que quien no esté muy familiarizado con su funcionamiento no los podrá modificar fácilmente.

El XML puede ayudar a resolver estos problemas. Separando el contenido de la presentación, se puede mantener la información mínima en necesaria en las páginas, convirtiéndolas mas fáciles de modificar y actualizar. La presentación y las barras de navegación puede estar toda contenida en una página de estilo general. Con un simple comando make se pueden generar páginas estáticas HTML a partir de las fuentes XML, cada vez que existan cambios.

La dificultad de este método está en la construcción de un DTD adecuado para páginas web, mas por suerte el trabajo ya está hecho: se trata de XHTML, que es un DTD XML que describe básicamente todas las mismas funcionalidades del lenguaje HTML 4.0, pero con sintaxis XML. Quien esté acostumbrado a trabajar con HTML, solo tendrá que tener en cuenta unas pocas reglas para hacer la transferencia a XHTML:

- Nunca usar mayúsculas en los nombres de las marca
- No dejar ninguna marca abierta; es necesario cerrarlas con la respectiva marca </...>, o si se trata de una marca vacía, se puede usar por ejemplo
- Usar siempre comillas para delimitar el valor de los atributos

Un ejemplo de un site donde hemos usado este método, son las páginas de la Free Software Foundation Europe (http://www.fsfeurope.org). La figura 4 muestra la página principal; el fichero

Figura 4: Página principal de la FSF Europe.

fuente incluía únicamente la información encerrada en el cuadro blanco principal; todo el resto fue agregado pro la página de estilo XSL. La propia página tiene un par de enlaces, abajo y al lado izquierdo, que permiten ver la fuente XHTML y la página de estilo usada.

7.3 Organización de información usando RDF

RDF es un DTD orientado a la descripción de recursos. Con el rápido crecimiento de la www, la cantidad de información disponible en campos muy variados es bastante amplia. Un problema complicado es como clasificar al información disponible para poder encontrarla cuando se necesaria.

Han existido intentos de crear meta-catálogos de documentos disponibles en la web, pero han resultado ser una tarea muy complicada ya que las páginas web aparecen y desaparecen con mucha facilidad. Otro enfoque han sido los motores de búsqueda que recorren la web clasificando información de forma automática. La dificultad existente es que sin un buen resumen de lo que contiene un documento, una búsqueda automática puede no ser muy útil. Han habido intentos de facilitar la labor de clasificación de los robots, usando por ejemplo las marcas META en el encabezado de las páginas web.

El RDF ha sido concebido con este tipo de problemas en mente. Su objetivo es la descripción

de recursos disponibles; y normalmente se destina a al intercambio de información entre sistemas, más que a suministrar contenido. Un ejemplo de aplicación es en los servidores de noticias, como por ejemplo Slashdot y Barrapunto. El listado siguiente muestra lo que se obtiene si seleccionamos una sección de Barrapunto, en este caso la sección "La Espiral" y accedemos a la URL http://barrapunto.com/laespiral.rdf (solo mostraremos la parte inicial del listado, apenas para dar una idea de como es)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns="http://my.netscape.com/rdf/simple/0.9/">
  <channel>
    <title>Barrapunto: La Espiral</title>
    <link>http://barrapunto.com/index.pl?section=laespiral</link>
    <description><La información que te interesa!</description>
  </channel>

  <image>
    <title>Barrapunto</title>
    <url>http://barrapunto.com/images/topics/topicslash.gif</url>
    <link>http://barrapunto.com</link>
  </image>

  <item>
    <title><Woody se congela!</title>
    <link>http://barrapunto.com/article.pl?sid=01/02/17/094222</link>

  </item>

  <item>
    <title>Entrevista a Wichert Akkerman en Debian Planet</title>
    <link>http://barrapunto.com/article.pl?sid=01/02/16/179236</link>
  </item>
```

Como se puede ver, el resultado es un documento XML que usa el DTD RDF; realmente es una pequeña implementación de RDF, llamada RSS (RDF Site Summary), pues RDF puede ser mucho mas complexo. Este fichero RDF será muy fácil de manipular con los procesadores de XML/XSL y podrá ser usado para extraer por ejemplo los titulares de las noticias en la sección.

Existen varios programas que ayudan a crear o procesar RSS. En Debian el paquete libxml-rss-perl trae un módulo perl que puede ser usado para ese propósito.

7.4 Ficheros de configuración

XML es una muy buena opción para escribir ficheros de configuración de programas. La existencia de librerías optimizadas para extraer información y modificar documentos XML facilita la tarea del programador.

Un ejemplo típico son los ficheros de configuración usados en Glade. Glade es un programa para construir interfaces gráficas de usuario (GUI) que usen las librerías gráficas GTK+. Glade tiene una interfaz gráfica fácil de usar, donde se pueden definir las ventanas que usará el programa que se va a construir y se le pueden ir colocando diferentes "widgets". El resultado después de definir el GUI del programa se resume en un fichero XML con el nombre del programa y con extensión .glade. Si mas tarde se quiere modificar la interfaz gráfica del programa, glade leerá ese fichero y analizándolo recuperará toda la información que necesita para volver a representar la interfaz gráfica del programa.

Veamos un ejemplo. Glade trae un ejemplo que consiste en un editor de texto. Las primeras líneas del fichero `editor.glade` son así:

```
<?xml version="1.0"?>
<GTK-Interface>

<project>
  <name>Glade Text Editor</name>
  <program_name>glade-editor</program_name>
  <directory></directory>
  <source_directory>src</source_directory>
  <pixmaps_directory>pixmaps</pixmaps_directory>
  <language>C</language>
  <gnome_support>False</gnome_support>
  <gettext_support>True</gettext_support>
```

Nótese que no ha sido usado ningún DTD. Cuando el fichero XML es creado y modificado por un programa, normalmente no es necesario validarlo contra un DTD, pues si el programa ya ha sido depurado los ficheros producidos tendrán siempre la estructura esperada.

7.5 Bases de datos

Los documentos XML son una buena interfaz para proporcionar datos a una base de datos, o para almacenar copias de partes del contenido de la base de datos, en ficheros de texto. Cada campo en una tabla de la base de datos se puede hacer corresponder al contenido de alguna marca XML. Veamos un ejemplo de una subrutina perl en la que se usa DOM para extraer información de un fichero XML para actualizar una base de datos SQL:

```
sub actualizarbd
{
    use DBI;
    use XML::DOM;
    my ($id,$fichero) = @_;
    my $bd = 'laespiral';
    my $dbmaster = 'villate';
    my $mibd = DBI->connect("dbi:Pg:dbname=$bd",$dbmaster,'')
        or return "Base de datos inaccesible\n$DBI::errstr\n";
    my $sql  = $mibd->prepare(q{update recetas set titulo=?, autor=?, }
```

```
                        . q{fecha=?, contenido=? where id=?});
    my $parser = new XML::DOM::Parser;
    my $doc = $parser->parsefile ($fichero);
    my $titulo = &extraer($doc->getElementsByTagName ("titulo"));
    my $autor = &extraer($doc->getElementsByTagName ("autor"));
    my $fecha = &extraer($doc->getElementsByTagName ("fecha"));
    my $contenido = &extraer($doc->getElementsByTagName ("contenido"));
    $sql->execute($titulo,$autor,$fecha,$contenido,$id)
        or return "recetas::actualizar: Error actualizando receta $id\n"
            . "$DBI::errstr\n";
    $mibd->disconnect;
    return "La base de datos de recetas ha sido actualizada.\n";
}

sub extraer
{
    my (@elementos) = @_;
    my $elemento = $elementos[0]->toString;
    $elemento =~ s/^[^>]*>\n?//;
    $elemento =~ s/\n?\s*<[^<]*$//;
    return $elemento;
}
```

7.6 Procesamiento distribuido usando SOAP

SOAP es el acrónimo de Simple Object Access Protocol. Es un protocolo usado para ejecutar comandos en servidores remotos. La información enviada al servidor remoto y el resultado de la ejecución del comando se envían en ficheros XML.

8 Bibliografía

1. *The World Wide Web Consortium (W3C)*, http://www.w3.org/ . Los creadores de los estándares HTML, XML, XSL, XHTML y muchos otros; en esta página se encuentran las versiones mas recientes de los estándares así como información adicional y enlaces a otras fuentes de información.

2. *Apache XML Project*, http://xml.apache.org/ . Este proyecto actualmente se subdivide en siete grupos muy importantes: Xerces, Xalan, Coccon, FOP, Xang, SOAP, Batik y Crimson, con el objetivo de producir una solución livre y completa basada en XML y el servidor http de Apache.

3. *The XML C library for Gnome*, http://www.xmlsoft.org/ . La página de las librerías XML creadas por el proyecto Gnome; incluye el procesador de XML/XSL xslt.

4. *Sablotron 0.52*, http://www.gingerall.cz/ga/Sablot-0-52.html . Manual de Sablotron 0.52, un procesador de XML+XSL programado en C++.

5. *Programación en XML: el nuevo lenguaje de Internet*. Los apuntes de un curso bastante completo sobre XML, realizado en Granada, http://geneura.ugr.es/cursos/xml/programa.shtml .

6. *xml.com*, http://www.xml.com . La página de O'Reilly dedicada al XML; allí se encuentra mucha información sobre XML y XSL.

7. *The XML Cover Pages*, http://www.oasis-open.org/cover/sgml-xml.html . Un servidor del grupo OASIS, lleno de información útil sobre XML/XSL y software.

8. *Página personal de Norman Walsh*, http://www.nwalsh.com . Una fuente muy buena de información; bastante útiles especialmente las transparencias sobre XSL.

9. St. Lauren, S. y Biggar, R.. *Inside XML DTD's*, McGraw-Hill, 1999, ISBN: 0-07-134621-X.

10. Navarro, A., White, W. y Burman, L. *Mastering XML*, SYBEX, 2000, ISBN: 0-7821-2266-3.

11. Walsh, N. y L. Muellner. *DocBook: The Definitive Guide*, 1a edic., O'Reilly & Associates, Inc. octubre de 1999, ISBN: 156592-580-7. (disponible en http://www.docbook.org)

12. Villate, J. E.. *Creación de artículos y páginas web usando XML en Linux*, http://www.laespiral.org/documentacion/articulos/articulos-COMO/ La Espiral, 22 de noviembre de 2000.

    Manual de le-docxml, el sistema XML/XSL usado para producir este documento.

13. Lerner, R. M.. *Introducing SOAP*, The Linux Journal, **83**, marzo de 2001.

14. *TMX Format*, http://www.lisa.unige.ch/tmx/tmx.htm . Página del DTD Translation Memory Exchange, para ayudar en la traducción de documentos.