Extended Markup Language
XML

Horst H. von Brand
vonbrand@inf.utfsm.cl

Departamento de Informática
Universidad Técnica Federico Santa María

Contenido

Markup

En los inicios de la imprenta, autores entregaban sus textos manuscritos con marcas para indicar manejo particular de ciertos trozos: esto en negrilla, este es el título del capítulo, esto es una cita o un poema. Cuando estas tareas pasaron a manos de computadores, se popularizaron lenguajes que mezclan el texto a desplegar con este tipo de anotaciones, conocidos como *markup languages* en inglés.

Markup

Lamentablemente, aparecieron muchos lenguajes de este tipo, muy distintos. Algunos lenguajes tempranos servían simplemente para abreviar comandos a la fotocomponedora específica, otros apuntaban a ser independientes (más o menos) del dispositivo final.

Markdown

Una variante reciente de esta idea es *Markdown* (juego de palabras con *markup*), popular en Wikis y para escribir documentación simple. La idea es que el texto anotado sea inteligible sin conocer el formato (por ejemplo, enfatizar escribiendo `*énfasis*`, marcar encabezados subrayando la línea con una línea de = o -) pero puede ser procesado para dar formatos más atractivos.
Lamentablemente, nuevamente han florecido una variedad de dialectos, con extensiones en distintos ámbitos.

Otros formatos comunes

Ideas afines son el formato RTF (*Rich Text Format*, de Microsoft), reST (*reStructuredText*, usado para la documentación de Python) y el formato POD (por *Plain Old Documentation*, usado por Perl y Raku para documentación).

SGML

En 1986, ISO (*International Standards Organization*) definió un lenguaje de anotaciones generalizado, *Standard Generalized Markup Language*, SGML.
Parte de dos postulados:

Declarativo: Las anotaciones deben describir la estructura del documento y otros atributos, no definir el procesamiento a efectuar. Esto tiene menor probabilidad de entrar en conflicto con desarrollos o usos futuros.

Riguroso: Definición rigurosa permite aprovechar técnicas para manipular objetos rigurosamente definidos, como bases de datos o programas.

SGML

SGML fue diseñado para permitir compartir y procesar documentos electrónicos voluminosos en gobierno, leyes e industria. Muchos de tales documentos deben seguir siendo manejables por décadas, compartidos ampliamente y procesados automáticamente de distintas maneras.

SGML

En realidad, SGML no define *un* lenguaje, es un marco en el cual definir lenguajes especializados mediante una gramática de contexto libre. Ejemplos de lenguajes especializados definidos en SGML incluyen HTML (el lenguaje de las páginas web), y los lenguajes DocBook (diseñado para describir documentación técnica, como manuales) y LinuxDoc (una variante del anterior, adoptada por Linux para su documentación).

XML

En 1998, World Wide Web Consortium (W3C) adoptó SGML como base de *Extensible Markup Language* (XML). Especifican un subconjunto suficientemente expresivo y general. Objetivos de diseño fueron énfasis en simplicidad, generalidad y usabilidad en todo Internet. Aunque el diseño se centra en documentos de texto, se usa ampliamente para describir estructuras de datos arbitrarias. Los formatos de documento estándar de ISO (OASIS) son archivos XML comprimidos. Los nuevos formatos de Microsoft Office (DOCX, XLSX, PPTX) también son archivos XML comprimidos.

Formato general

La estructura básica de un archivo XML es una serie de estructuras de la forma <*tag*> ... </*tag*> anidadas, donde *tag* es una palabra arbitraria. Un archivo que cumple con este formato general se llama *bien formado* (en inglés *well formed*).

Formato general

Hay cinco caracteres que tienen significado especial, que deben expresarse como se indica:

"	&quot;
'	&pos;
<	&lt;
>	&gt;
&	&amp;

Formato general

Comentarios comienzan con −− y terminan en −−> al interior de <!. Por ejemplo, <!−−*Este es un comentario* −−>.

Otras construcciones especiales comienzan <!, vea por ejemplo la definición de un DTD más adelante. Esto permite comentar al interior de construcciones, así:
<!**ELEMENT** −−*comment* −−>
es equivalente a:
<!**ELEMENT**>

Se usa <? ... ?> para indicar opciones o configuración.

Formato general

Puede definirse qué *tag* pueden usarse en el documento, y cómo, mediante un DTD (*Document Type Definition*, definición de tipo de documento). Un DTD es esencialmente una gramática de contexto libre, expresada en un formato un tanto inusual. Un documento XML que está bien formado y cumple con la estructura definida por un DTD se llama *válido*.

Una nota en XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE note SYSTEM "Note.dtd">
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Recordatorio</heading>
  <body>
    Llama sin falta este fin de semana.
  </body>
</note>
```

La primera línea identifica la versión de XML y la codificación empleada.
La segunda referencia al DTD que describe el formato del documento.

DTD para notas en XML

```
<!DOCTYPE note [
  <!ELEMENT note (to, from, heading, body)>
  <!ELEMENT to (#PCDATA)>
  <!ELEMENT from (#PCDATA)>
  <!ELEMENT heading (#PCDATA)>
  <!ELEMENT body (#PCDATA)>
]>
```

La primera línea identifica el símbolo (rótulo) de partida. Las demás son producciones. #PCDATA es *Parsed Character Data*, texto arbitrario que puede incluir construcciones de XML a ser verificadas únicamente por bien formadas.

Nota con DTD incrustado

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Internal DTD -->
<!DOCTYPE note [
  <!ELEMENT note (to,from,heading,body)>
  <!ELEMENT to (#PCDATA)>
  <!ELEMENT from (#PCDATA)>
  <!ELEMENT heading (#PCDATA)>
  <!ELEMENT body (#PCDATA)>
]>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Recordatorio</heading>
  <body>Llama este fin de semana.</body>
</note>
```

Producciones en DTD

Lados derechos especiales de producciones incluyen EMPTY (vacío, nuestro ε) y ANY (cualquier cosa).
Otros lados derechos de producciones se escriben entre paréntesis, permitiendo las construcciones tomadas de expresiones regulares | (alternativa), * (cero o más de la anterior), + (una o más de la anterior) y ? (la anterior es opcional). Otras posibilidades incluyen definir atributos.

Alternativas a DTD

Hay varias propuestas de *XML Schemas*, descripciones alternativas a los DTD para documentos, que requieren mayor rigurosidad en el contenido de caracteres entre otras. Son relevantes en aplicaciones de procesamiento de datos, para texto o documentas simples bastan DTDs.

Simple API for XML

Simple API for XML (abreviado SAX) es un esquema provisto para una variedad de lenguajes de programación. En realidad solo verifica que el documento esté bien formado, no adherencia a un DTD. El usuario define eventos de interés (se abre/cierra una construcción particular) a los que asocia acciones.
Es simple de proveer y usar, rápido y consume poca memoria. Es engorroso si se requiere navegar la estructura del documento, por ejemplo, se requiere manejar de distinta forma una construcción dada según su contexto en el árbol de derivación.

Document Object Model

Document Object Model (DOM) representa el documento como un árbol en memoria, donde puede navegarse desde el programa. Es una interfaz independiente de la plataforma y lenguaje de programación.

Resumen

Es común escuchar que XML refleja la semántica del documento. Una gramática de contexto libre solo da estructura sintáctica. El significado debe establecerse externamente.

- Dimos una rápida mirada a Markup en general.
- Dimos un vistazo a XML, una de las técnicas más populares para definir formatos de datos.
- Mostramos el formato de un DTD, comentamos de XML Schema.
- Distinguimos documentos *bien formados* y *válidos*.
- APIs para trabajar con XML incluyen SAX y DOM.