Ontologías y Web Semántica

XML básico

Pablo R. Fillottrani

Depto. Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur

marzo 2019

XML básico

- XML es un acrónimo por eXtended Markup Language
- XML es una recomendación W3C desde febrero 1998; la última version aprobada es 1.1 de febrero 2004
- XML no es propio un lenguaje; es más bien una sintaxis en la que se pueden representar muchos lenguajes

- el consorcio W3C `www.w3c.org` fue fundado en 1994, teniendo como objetivo el desarrollo de protocolos comunes para la industria
- reune empresas, organismos de investigación, universidades, grupos de usuarios interesados en el desarrollo de Internet
- en W3C se puede encontrar
  - repositorio de información para desarrolladores y usuarios WWW
  - implementaciones de referencia de las diversas tecnologías
  - prototipos y aplicaciones que demuestran su uso

- el proceso de desarrollo de una nueva tecnología es arduo y requiere un consenso, pero generalmente el resultado gana en momentum
- al principio, alguien propone mediante una nota una nueva tecnología. La W3C puede decidir organizar un grupo de trabajo al respecto
- luego de mucha consulta y discusión, se prepara un borrador (working draft) al respecto
- cuando se logra cierto consenso el borrador se transforma en una propuesta de recomendación (proposed recommendation)
- finalmente con trabajo y más discusiones la tecnología puede lograr el status de recomendación (recommendation)
- W3C no es un órgano de policía, no controla el uso de las tecnologías diseñadas

- XML se usa no solo para representar datos, sino también para dar estructura a la información. De esta manera los datos acarrean información sobre si mismos
- XML caracteriza documentos que deben ser bien formados. Esto garantiza que todos los lenguajes basados en XML sean usados más fácilmente
- entonces XML provee los fundamentos para una nueva manera de comunicación a través Internet, favoreciendo la comunicación entre (las computadoras de) personas y empresas

- Objetivos de diseño de XML
  - XML debe poder ser usado directamente en Internet
  - XML debe ser soportado por múltiples aplicaciones
  - XML debe ser compatible con SGML
  - deber ser fácil escribir programas que procesen documentos XML
  - en lo posible, XML debe minimizar las características opcionales
  - los documentos XML deben ser leí bles por un humano
  - el diseño de XML debe ser formal y conciso

- un documento XML es una secuencia de caracteres Unicode que sigue ciertas reglas. Estas reglas proveen la forma para dar una estructura de árbol a los datos
- ciertas secuencias de caracteres representan datos; otras secuencias representan el marcado
- el marcado permite expresar la jerarquía lógica del documento al mismo tiempo que los datos
- el documento con la especificación de XML usa reglas EBNF para definir la sintaxis

- sintaxis de un documento XML

```
documento  ::=   prologo  element  misc*

misc  ::=   comment  | pi  | spaces

comment  ::=  '<!--' (( char - '-') |
              ('-' (char - '-')))* '-->'
```

- solo ciertos caracteres Unicode son válidos como `char`, no todos
- para interpretar un documento XML, es necesario saber cuál codificación entre UTF-8 y UTF-16 es usada en el documento

Prólogo de un documento XML

- un documento XML comienza con una sección de marcado denominada prólogo
- consiste de una declaración XML, o una declaración textual, seguido opcionalmente de un DTD, y luego misc
- ejemplo de declaración XML

```
<?xml version="1.1"
      enconding="UTF-8"
      standalone="no" ?>
```

- la versión xml es requerida; la codificación y la declaración de standalone son opcionales
- el prólogo de una entidad que no sea una entidad documento comienza con una declaración de texto similar, salvo que es opcional, siempre contiene la codificación y nunca contiene un atributo standalone

DTD

- un DTD (document type description) describe la forma que deben tener el elemento raíz del documento, proveyendo instrucciones para controlar el marcado
- en un DTD se pueden definir también datos que serán usados para validar el documento (parámetros)
- si se hacer referencia a un DTD, se requiere al parser que compare la instancia del documento con el modelo del documento, esto se denomina chequeo de validez
- el DTD no es parte de los datos del documento, sino que contiene las declaraciones definidas por el usuarios que definien cuándo un documento es válido
- todos los documentos que cumplen las reglas de un DTD dado se consideran que son del mismo tipo. El chequeo de validez es opcional, pero generalmente más útil que el cheque de buen-formato

- la sintaxis de un DTD es la siguiente

```
dtd  ::=  '<!DOCTYPE' elementoRaiz
          [ uriDTDExterno ]
          [ '[' DTDInterno} ']' ] '>'
```

- la parte interna de un DTD se usa para aumentar y redefinir la parte externa
- generalmente, la parte externa de un DTD es compartida por varios documentos XML
- las dos partes son opcionales

- ejemplo DTD

```
<!DOCTYPE book
  PUBLIC "-//ORA/DTD DBLITE XML//EN"
  SYSTEM "/usr/local/prods/dblite.dtd"

[  <!ENTITY chap1 SYSTEM "ch01.xml">
   <!ENTITY chap2 SYSTEM "ch02.xml">
   <!ENTITY xml "<acronym>XML</acronym>">
]>
```

Misceláneos

- un documento XML puede ser decorado con comentarios, siempre cuando esten separados de otros marcados
- los parsers XML pueden ignorar los comentarios, pero también pueden tomar otras acciones no especificadas
- las instrucciones de procesamiento pi proveen un mecanismo de comunicación entre el parser y la aplicación por medio del documento. No forman parte de los datos del documento
- tienen la forma <?destino instruccion? >, donde el destino es un identificador significativo para la entidad, y la instrución es simplemente una cadena
- ejemplo:

```
<?xml-stylesheet href=''style.css''
   type=''text/css''?>
```

- no son usadas frecuentemente

Entidades

- un documento XML puede ser dividido en secciones llamadas entidades. Cada entidad puede existir en diferentes lugares (discos, computadoras)
- las entidades clasifican los datos de un documento en un estructura de árbol
- la entidad que contiene el cuerpo principal del documento se llama entidad documento, y constituye la raíz del árbol
- todas las demás entidades del documento son descendientes de esta raíz

- si una entidad consiste de datos XML (ie es un fragmento de un documento XML) entonces se denomina entidad parseable
- un parser XML lee la entidad documento y localiza, decodifica, e importa todas las otras entidades parseables como un texto de reemplazo por sus referencias

- una entidad que contiene datos no-XML (un archivo de imagen o sonido por ejemplo), no puede ser leida por un parser XML. Se denomina entidad no parseable
- un documento XML puede contener información sobre la ubicacón y el formato de una entidad no parseable. El parser no procesa el contenido de estas entidades, solo transmite esta información a la aplicación
- para que un documento sea válido, se requiere que la entidades esten declaradas en el DTD (document type definition) del documento
- el DTD es una parte de la estructura lógica del documento en donde se definen las restricciones de validez del mismo
- el DTD puede consistir de una parte interna, obligatoria, y una parte externa, opcional, existente en una entidad distinta a la entidad documento

- una entidad que sólo es usada en el documento es una entidad general
- una entidad que sólo es usada dentro de un DTD es una entidad parámetro. Se usan como macros, o como pseudo tipos de datos
- si la declaración de la entidad especifica su texto de reemplazo mediante su ubicación (URI), o si se trata de una entidad no parseable, la entidad se denomina externa
- si la declaración de la entidad incluye texto de reemplazo (un literal, una referencia, o ambos), entonces la entidad es interna

entidades
- general interna parseable
- parámetro interna parseable
- general externa parseable
- parámetro externa parseable
- general externa no parseable

- cuando una entidad parseable es declarada en un DTD, se le asigna un nombre, el cual constituye la base de las referencias a la entidad
- la sintaxis para referenciar una entidad es `&nombre;` para entidades generales, y `%nombre;` para entidades parámetros

- existen cinco entidades generales internas parseables pre-definidas: `&`, `<`, `>`, `&quot;`, `&apos;`
- un parser XML que no está validando un documento no necesita leer las entidades externas. Si el documento se declara como `standalone`, entonces necesariamente no debe hacer referencias a declaraciones de marcado externas
- además de las referencias a las entidades, existen también referencias a caracteres en un documento XML
- la sintaxis es la misma que para entidades generales, excepto que el nombre es reemplazado por el código Unicode
- el término entidad caracter no es definido en la especificación XML; pero como existen las referencias a caracteres, se lo puede entender como una entidad que tiene un texto de reemplazo dado por un sólo caracter

- los datos textuales pueden ser de dos formas: parseables o no parseables
- si son parseables, entonces se codifican con una sección PCDATA (sin tags) en donde las entidades caracteres pre-definidas pueden usarse
- ejemplo: `1 &amp; 2 are &lt; 3`
- si los datos textuales no son parseables, entonces se usa una sección CDATA que debe estar encerrada en un marcado
- ejemplo:

```
<![CDATA[1 &amp; 2 are &lt; 3]]>
```

- es sólo una conveniencia para ahorrarse secuencia de caracteres de escape

Elementos y Atributos

- los datos textuales se dividen a su vez en elementos y atributos
- aunque la especificación no define ninguna diferencia entre estas estructuras, se entiende que los elementos definen una jerarquía tipo árbol, y que los atributos sólo son un par (nombre, valor)
- todo elemento o atributoxample puede tener un nombre que empieza con una letra, guión bajo o coma; y que además contiene sólo ciertos otros caracteres
- los nombres de un elemento son su tipo. Todos los elementos con el mismo nombre son del mismo tipo
- no pueden existir dos atributos con el mismo nombre en el mismo elemento

- un elemento es un contenedor para su contenido, que puede ser datos, más elementos, o ambos
- un documento XML debe tener exactamente un elemento raíz, que se denomina elemento documento
- todos los datos textuales y otros elementos en el documento deben existir contenidos en el elemento raíz
- es posible definir una relación padre-hijos entre un elemento y sus elementos contenidos

- si un elemento no tiene contenido, entonces es vacío, y se denota con un marcado especial

```
<elementoVacio/>
```

- si el elemento tiene contenido, entonces su contenido está delimitado por un marcado de inicio y un marcado final

```
<nombreElto>contenido textual</nombreElto>
```

- siempre que haya otros elementos contenidos en un elemento, los marcados deben estar perfectamente anidados

- un atributo que está asociado a un elemento se coloca en el marcado de inicio del mismo, a continuación del nombre
- se debe dar el nombre, seguido del valor encerrado entre comillas dobles o simples (si son necesarias en el valor, se puede usar secuencias de escape en el valor)
- ejemplo:

$$<\texttt{saludotipo} = \texttt{"informal"} > \texttt{Hola!} </\texttt{saludo}>$$

- los valores de los atributos no son los lugares más apropiados para incluir datos textuales, ya que el parser normaliza su contenido

- si un documento respeta todas las restricciones de la codificación de caracteres y las reglas sintácticas (en EBNF) de la especificación XML, entonces se dice bien formado
- si el documento además cumple con las reglas definidas en el DTD y las restricciones de validez de la especificación XML, entonces se dice válido

- un parser validador controla que un documento sea bien formado, y reporta violaciones de la validez en el documento completo, incluyendo entidades parseables externas y el DTD completo
- para un parser no validador se requiere que controle que el documento sea bien formado solamente en la entidad documento, y el subconjunto interno del DTD, no reportando violaciones de la validez
- también, para un parser no validador no se requiere que obtenga texto de reemplazo de entidades parseables externas, pero debe informar a la aplicación que existen tales referencias

- la sintaxis XML para DTDs es heredada de SGML, siendo bastante complicada. La única buena referencia es la propia especificación
- ejemplos:

```
<!ENTITY entidadGenInterna  ''Hola''>
<!ENTITY entidadGenExterna1
                        SYSTEM ''archivo.txt''>
<!ENTITY % entidadParExternal  SYSTEM
          ''http://foo.net/miDTD.xml''>
```

- el texto entre comillas simples o dobles es parseado reemplazando referencias a parámetros y a caracteres

- se recuerda que si un DTD, o parte de un DTD es una entidad externa, el texto de reemplazo debería empezar con una declaración de texto
- las entidades externas no parseables se declaran de una manera similar, pudiéndoso agregar información adicional para la aplicación

```
<!ENTITY entidadGenExterna2
    SYSTEM "foto.jpg" NDATA formatoJPEG>
<!NOTATION formatoJPEG
    SYSTEM "http://www.jpeg.org">
```

- la presencia de `NDATA` indica que se trata de una entidad no parseable
- la notación es sólo un indicio a la aplicación para que detecte cómo tratar esos datos

- el único uso en un documento XML de una referencia a una entidad no parseable es en el valor de un atributo declarado de tipo `ENTITY` o `ENTITIES`
- la declaración del tipo de un elemento se hace enumerando el contenido posible de ese elemento, en un lenguaje de expresiones regulares
- ejemplo:

```
<!ELEMENT saludo (#PCDATA | nombre )*>
<!ELEMENT nombre (#PCDATA)>
```

- en principio, los espacios dentro de un elemento son significativos. Esto puede cambiarse con el uso del atributo `xml:space`

- la declaración de atributos es más complicada. Ejemplo:

```
<!ATTLIST saludo
    tipo (formal | informal ) #REQUIRED
    longitud CDATA #IMPLIED>
```

- el tipo de los atributos puede ser:
  - `CDATA` datos textuales parseables
  - `NMTOKEN` una secuencia de caracteres que cumple con ciertas reglas
  - `ID` una secuencia de caracteres que cumple la regla para `ID` y que no aparece repetida en el documento
  - `IDREF` una secuencia ID que es la misma que un atributo `ID` definido en algúna otra parte en el documento
  - `ENTITY` el nombre de una entidad no parseable declarada en otra parte en el DTD
  - `NMTOKENS`, `IDREFS`, `ENTITIES`, secuencias de los respectivos anteriores separadas por espacios

- es posible declarar valores por defecto para los atributos colocando el valor en lugar del `REQUIRED` o el `IMPLIED`
- si el valor es precedido por `#FIXED` entonces el atributo siempre existe en los elementos de ese tipo, aún cuando no está declarado
- la especificaciń XML incluye dos atributos con significado especial: `xml:space` y `xml:lang`
- los posibles valores `CDATA` para `xml:space` son `preserve` o `default`
- los posibles valores `CDATA` para `xml:lang` están determinados por los códigos de lenguajes RFC 1766

Ejemplo - Documento XML

```
<?xml version="1.0"?>
<!DOCTYPE PARTS SYSTEM "parts.dtd">
<?xml-stylesheet type="text/css" href="xmlpartsstyle.css"?>
<PARTS>
   <TITLE>Computer Parts</TITLE>
   <PART>
      <ITEM>Motherboard</ITEM>
      <MANUFACTURER>ASUS</MANUFACTURER>
      <MODEL>P3B-F</MODEL><COST> 123.00</COST>
   </PART>
   <PART>
      <ITEM>Video Card</ITEM>
      <MANUFACTURER>ATI</MANUFACTURER>
      <MODEL>All-in-Wonder Pro</MODEL>
      <COST> 160.00</COST>
   </PART>
</PARTS>
```

Ontologías y Web Semántica
└Validez de un documento XML

Ejemplo - DTD

```
<!ELEMENT PARTS (TITLE?, PART*)>
<!ELEMENT TITLE (#PCDATA)>
<!ELEMENT PART (ITEM, MANUFACTURER, MODEL, COST)+>
<!ATTLIST PART
   type (computer|auto|airplane) #IMPLIED>
<!ELEMENT ITEM (#PCDATA)>
<!ELEMENT MANUFACTURER (#PCDATA)>
<!ELEMENT MODEL (#PCDATA)>
<!ELEMENT COST (#PCDATA)>
```