Introduccion XML

Juan Ignacio Raggio

May 3, 2025

Contents

1 Documento

Un documento de cualquier tipo presenta 3 capas:

1. Estructura: Como estara organizada la informacion
2. Presentacion: La forma en la que se mostraran los datos
3. Contenido: Los datos y la informacion en concreto

1.1 Historia y origen del nombre

- 80': SGML (Standard Generalized Markup Language)
- 90': HTML (HyperText Markup Language)
- 1998: XML (`eXtensible Markup Language`)

2 Objetivos del diseño de XML

1. Capacidad de usar en internet
2. Soporte para un variado grupo de aplicaciones (capacidad de desplegarse en: celulares, bases de datos, computadoras, etc)
3. Compatibilidad con `SGML`
4. Debe ser sencillo escribir programas que procesen documentos XML
5. Las caracteristicas opcionales deben ser pocas, idealmente ninguna
6. Los documentos deben ser legibles para los humanos y razonablemente claros

7. El proceso de diseño debe ser rapido
8. El diseño de un XML debe ser formal y conciso
9. Los documentos XML deben ser facil de crear
10. La brevedad de los markups debe ser de minima importancia

2.1 Formacion de un documento XML

2.1.1 Importante

- Un documento XML `debe ser bien formado` (estrictamente) y `puede ser valido` (opcional)
- La `indentacion` es solo por claridad como en todos los lenguajes de programacion
- Los tags son case-sensitive, o sea que <cliente> no se puede cerrar con </Cliente>

2.1.2 Prologo (Indica la version y opcionalmente el encoding (UTF-8 por default))

2.1.3 Elementos TAG

- Empieza y termina con un tag
- Puede contener:
  - Solo texto
  - Elementos
  - Texto y elementos
  - Atributos
  - Nada

```
<cliente codigo=‘‘A3221’’> <- Codigo es un atributo y cliente contiene el Elemento nombre
        <nombre>ITBA</nombre> <- Solo texto
        <direccion>comercial
                <calle>Lavardén 315</calle>
                <ciudad>CABA</ciudad>
                <CP></CP>
        </direccion> <- Texto y elementos
```

```
        <telefono/>
</cliente>
```

1. Parsed character data (PCDATA) CARACTERES_NO_PERMITIDOS
   - Hay 5 caracteres no permitidos:
     (a) <
     (b) >
     (c) &
     (d) "
     (e) ’
   - Como incluirlos?
     – Opcion 1:

<	>	&	"	’
&lt	&gt	&amp	&quot	&apos

     – Opcion 2: <![CDATA[`Lo que se ingrese aqui no es interpretado, puede ingresarse cualquier caracter`]]>
2. Comentarios en documentos XML <!– Esto es un comentario –>
3. Atributos
   - Cada elemento puede tener uno o mas atributos
   - Las comillas son obligatorias
   - Se usa para `agregar informacion del elemento`

2.1.4 XML Bien formado

- Hay un solo elemento raiz (`raiz unica`)
- Todo elemento tiene `tag de comienzo y tag de fin` o es un `tag de cierre` (vacio)
- Los elementos forman una `jerarquia` (no pueden solaparse) El siguiente fragmento de xml `no cumple` con la jerarquia (el tag de ciudad deberia cerrarse antes)

```
<calle> Lavarden
<ciudad> Capital Federal
</calle>
</ciudad>
```

- Los valores de los `atributos` van entre `comillas dobles o simples`
- Un documento bien formado deberia poder representarse como un arbol rotulado en forma natural. Los atributos dentro de este arbol rotulado deben mostrarse con un @ seguido del nombre

2.2 XML vs HTML

- Los tags en HTML no dan informacion sobre el contenido de los datos, `solo indican el formato que va a tener el texto` (la presentacion)
- Mientras que los tags en XML muestran la `semantica`

XML es un `meta markup language` porque permite definir tipos de documentos. Fue diseñado para describir los datos y se enfoca en su semantica, HTML fue diseñado para mostrar los datos y se enfoca en como se ven

- Es `flexible`: Si una aplicacion ya usaba un documento XML y a este se le agregan cosas, la aplicacion no necesita ser cambiada
- `Extensible`: Cada diseñador genera sus propios tags personalizados
- `Simple`
- `Internacional`: Soporta unicode
- Hoy en dia es un verdadero estandar

3 Para que se usa el XML?

3.1 Document management

Permite que el mismo dato (raw data) se muestre en multiples formatos (HTML, WAP, PDF, etc). En este caso como el humano es el que va a ver los datos, es importante la capa de presentacion

3.2 Business to Business

Permite intercambiar datos entre sistemas distintos o aplicaciones incompatibles o con poco acoplamiento. Ejemplo: Una orden de compra entre dos procesos de negocios de distintas organizaciones

3.3 Separar los datos en documentos HTML

3.4 Intercambio de datos entre aplicaciones

3.5 Almacenamiento de datos

3.6 Ejemplos de uso mas especificos

- Maven es un estándar para construir proyectos java `https://maven.apache.org/what-is-`

maven.html. El Project Object Model (POM), es un documento XML y es la unidad de trabajo fundamental en Maven.

- Google Text-to Speech API se basa en el estándar SSML, `https://www.w3.org/TR/speech-`

synthesis/, Speech Synthesis Markup Language.

- SVG (Scalable Vector Graphics) : es un estándar para definir gráficos soportado por todos

los browsers.

- BPMN: Lenguaje basado en XML para modelar los procesos de negocios. `https://www.omg.org/spec/BPMN/2.0/`

- SAML (Security Assertion Markup Language): es un estándar abierto para la autenticación y

autorización de usuarios. `https://learn.microsoft.com/en-us/entra/identity-platform/single-` sign-on-saml-protocol

4 Parsers

Es un programa que analiza la estructura sintactica de un documento en base a ciertas reglas. Determina cuáles son las partes básicas de un documento basado en una gramática que permite analizarlo

1. Recibe el doc
2. Lee cada caracter
3. Decide si el XML esta bien formado

4.1 Existen dos parsers:

4.1.1 Document object model (DOM): Carga el contenido en memoria construyendola en formal de arbol (crea un DOM document) completamente en memoria -> Permite un acceso no secuencial al documento

- El resultado es un `arbol DOM` -> Necesita memoria para guardarlo
- La aplicacion puede usar la estructura que arma el parser
- Tipos de nodos: Document, Element, Attribute, Text, Comment, Processing instruction

4.1.2 Simple API for XML (SAX): Analiza secuencialmente, a medida que lo va leyendo. Solo crea el arbol de a pedazos en memoria -> Conveniente cuando un XML es muy grande o nuestros recursos son muy limitados y no se puede mantener integramente en memoria

- Facil
- Rapido
- No requiere memoria para alojar estructuras
- Util para `archivos grandes`

5 Atributo vs Elemento

- un atributo permite indicar exactamente sus valores posibles
- un atributo permite especificar valor default
- un atributo ID permite unicidad
- la lista de atributos NO tiene orden, si eso fuera importante....
- un atributo no puede contener nada más que su texto (no tiene hijos)...

6 Como armar un documento xml?

1. Notar `datos clave`
2. Definir una estructura jerarquica (muy similar a las clases)
3. Considerar relaciones y repeticiones