XML eXtensible Markup Languaje

de etiquetas
nuestras propias etiquetas
no predefinidas
metalenguaje

XML eXtensible Markup Languaje

Utilizado para almacenar y transportar datos.

Es una recomendación del W3C.

Es una forma de almacenar, transportar y compartir datos, independiente del software y del hardware.

ejemplo01.xml:

```
<clientes>
	<cliente>
		<nombre>Maria</nombre>
		<tlfno>982342345</tlfno>
	</cliente>
	<cliente>
		<nombre>Manuel</nombre>
		<tlfno>676342546</tlfno>
		<tlfno>767893456</tlfno>
	</cliente>
	<cliente>
		<nombre>Inés</nombre>
	</cliente>
	<mejorcliente>
		<nombre>Manuel</nombre>
		<totalcompras>23.000</totalcompras>
	</mejorcliente>
</clientes>
```

Google: buscar “xml formatter”

Analizador XML

Un documento XML puede ser leído por personas pero el objetivo es que sea procesado por aplicaciones y procesos. Un analizador XML es un programa de bajo nivel que funciona entre una aplicación y los archivos XML. Su función es obtener información del archivo XML, controlando la sintaxis; algunos también pueden validar el archivo XML contra un documento (DTD o Schema).

La interface entre el analizador y la aplicación se basa en la lectura que el analizador hace del documento XML y en la interpretación que hace la aplicación de esa lectura. Puede estar basada en:

- **Eventos, SAX (Simple API for XML): Ejm: Aplicaciones que importan docs xml a BDs.**
- **Objetos, DOM (Document Object Model): Ejm: Exploradores y Editores.**

Analizador XML

Un analizador:

1. Lee el archivo.

2. Verifica la sintaxis.

3. Interpreta la estructura.

4. Hace una representación interna, (DOM ó SAX).

Ámbitos de aplicación de XML

- Un navegador web como IE, Mozilla, etc.
- Un procesador de textos como OpenOffice Writer que utiliza XML para estructurar los documentos.
- Un IDE como NetBeans que utiliza XML para estructurar los proyectos.
- Clientes gráficos como MySQL Query Browser o Workbench, o clientes web como PHPMyAdmin, que permiten importar y/o exportar datos de una base de datos MySQL en XML.
- Programas para la gestión de la formación de trabajadores como en el caso de la Fundación Tripartita para la Formación en el Empleo, que utiliza documentos XML validados por un Schema para unir la información que subministra cada empresa participante.
- Predicción del tiempo por localidades en Aemet.
- Páginas web escritas en XHTML.
- Programas escritos en Java, C o PHP.
- Una herramienta como Speccy que proporciona información sobre el sistema de un PC.
- Un programa de dibujo como Adobe Illustrator, que interpreta coordenadas de 2 dimensiones en XML.
- Una hoja de cálculo como Gnumeric, que analiza sintacticamente XML para buscar números y funciones usadas en un cálculo.
- Una aplicación como Google Earth que utiliza una gramática XML denominada KML para crear modelos y almacenamiento de función geográficas como por ejemplo puntos, lineas, imágenes, o polígonos y que permitirá compartir lugares y información sobre ellos.
- Protocolos estándar para servizos web: WSDL (Web Services Description Language) y SOAP (Simple Object Access Protocol).

Versiones

W3C define las versiones:

- **XML 1.0 Fifth Edition W3C Recommendation 26 November 2008 (http://www.w3.org/TR/REC-XML/) que modifica el original publicado el 10 de febreiro de 1998.**

- **XML 1.1 Second Edition W3C Recommendation 16 August 2006 (http://www.w3.org/TR/XML11/) que modifica el original publicado o 13 de diciembre de 2001.**

Las diferencias entre ellas en cuanto al contenido de los documentos XML es que la versión 1.0 es mucho más restrictiva en los caracteres permitidos para los nombres de las etiquetas, de los atributos y de los elementos identificados de forma única.

Estructura de un documento XML

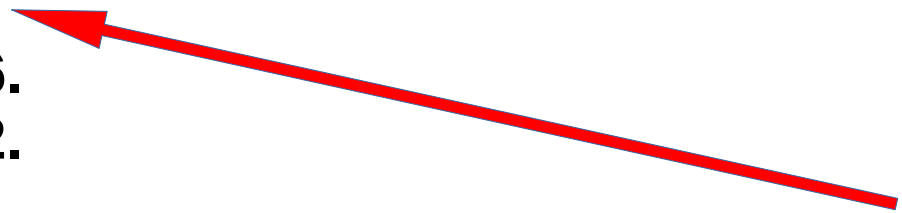
El documento XML está formado por una serie de caracteres que tienen que cumplir unas normas de sintaxis básicas para estar bien formados, almacenados según un sistema de codificación.

Consta de dos partes: prólogo y cuerpo.

Estructura de un documento XML

Posibles sistemas de codificación:

ASCII (American Standard Code for Information Interchange).
ASCII estendido.
Unicode.
- **UTF-8.**
- **UTF-16.**
- **UTF-32.**



El que usaremos nosotros

Prólogo

El prólogo es la parte en la que se declara el documento XML, en la que también se puede declarar el DTD o Schema utilizado para validar, y en la que se colocan las posibles instrucciones de procesamiento.

Prólogo

Sintaxis:

```
<?xml version="versionXML" encoding="codificacion" standalone="yes|no"?>
```

- **version: versión de XML utilizada, habitualmente (y su valor por defecto) 1.0.**
- **encoding: sistema de codificación utilizado. Se aconseja indicar siempre el sistema de codificación aunque no es obligatorio.**
- **standalone="yes | no" : informa de si el documento es independiente de declaraciones externas como por ejemplo un DTD. Su valor por defecto es "no".**

Ejemplo:

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

En la versión 1.1 es obligatorio indicar por lo menos la versión XML. Por defecto se usará la versión 1.0.

Tipo de documento

Declara las normas de sintaxis para validar el documento XML mediante un DTD interno o externo. Esto se verá con detalle más adelante.

La sintaxis para un DTD interno sería:

```
<!DOCTYPE elementoraiz[
...definición do DTD...
]>
```

Instrucciones de procesamiento

En el prólogo y en general en cualquier lugar del documento XML, pueden aparecer instrucciones de procesamiento que pasan información a las aplicaciones que van a leer el documento.

Sintaxe:

```
<?aplicación instrucións?>
```

Cuerpo

El cuerpo está compuesto por elementos y los elementos están formados por etiquetas.

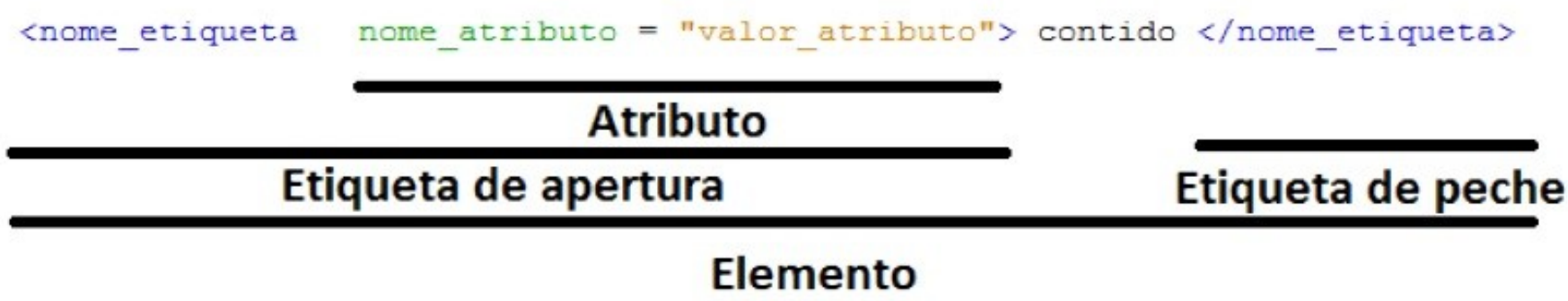
Las etiquetas son texto entre los símbolos < y > y pueden tener atributos y/o contenido.

Los atributos están pensados para guardar información adicional o descripción acerca de las etiquetas, se especifican en la etiqueta de inicio y tienen que tener un único valor entre comillas simples o dobles.

Las etiquetas pueden almacenar varios datos e incluso extenderse mediante estructuras anidadas de varios elementos.

El contenido puede no existir o estar formado por elementos anidados.

Cuerpo



Exemplo de etiquetas:

```
<root>
        <titulo>O meu primeiro libro de pesca</titulo>
        <cantidade unidade="kg">7</cantidade>
        <asignatura numHoras="12">Programación en Java</asignatura>
        <becario></becario>
</root>
```

Validación

Se dice que un documento xml es válido, cuando cumple con las siguientes características:

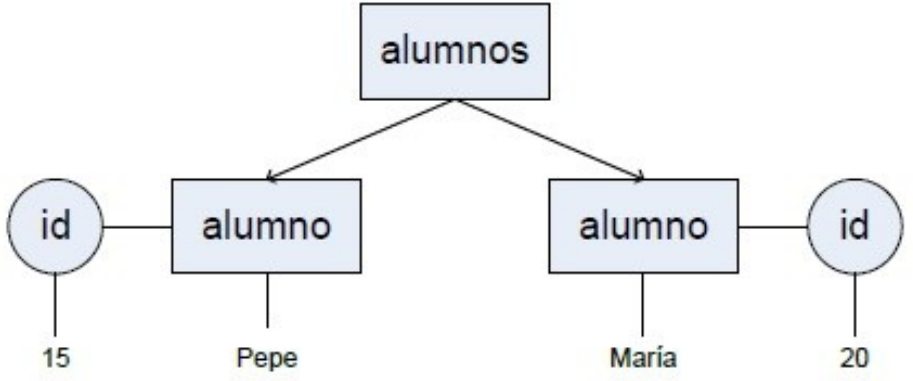
- **Está bien formado, (well formed). Cuando satisface todas las condiciones básicas de sintaxis.**
- **Validado: Se comporta de acuerdo a lo establecido en un DTD o Schema.**

Esto último se verá más adelante...

Para que sea “bien formado”:

Todos los elementos del documento se anidan en forma de árbol, existiendo un único elemento raíz, del que descienden todos los demás.

```
<?xml version="1.0" encoding="UTF-8"?>
<alumnos>
    <alumno id="15">Pepe</alumno>
    <alumno id="20">María</alumno>
</alumnos>
```



Para que sea “bien formado”:

Toda etiqueta de apertura tendrá su correspondiente etiqueta de cierre (símbolo / antes del nombre de la etiqueta), excepto si no tiene contenido, y podrá indicarse como <nombre_etiqueta />.

```
<becario></becario>
<beca>1200</beca>
<disfruta_beca/>
```

Para que sea “bien formado”:

Los elementos se pueden anidar, pero las etiquetas se cerrarán de forma estructurada, (ordenada), es decir, en orden inversa a su apertura.

```
<observacions>O solar está <importante>pendente de revisión
</importante> no Concello</observacions>
```

Para que sea “bien formado”:

Un atributo, si existe, siempre tiene que tener un valor entre comillas simples o dobles y no puede aparecer más de una vez en una etiqueta, aunque se podrá especificar en cualquier orden.

```
<modulo horas="125" idioma="galego">LMSXI</modulo>
<modulo idioma="catalán" horas="200" >CODE</modulo>
<modulo>BADA</modulo>
```

Para que sea “bien formado”:

Los documentos XML pueden contener comentarios. Estos, aparecerán siempre entre las marcas de inicio y cierre de comentario, <!-- y --> respectivamente.
Dentro de un comentario se podrá añadir cualquier carácter excepto dos guiones consecutivos (--).
No puede haber comentarios en las etiquetas.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Document:       proba.xml
    Created on:     7 de xaneiro de 2016, 11:00
    Author:         profesora
    Description:    Probas para introdución á XML.
-->
<root>
    <modulo horas="125" idioma="galego">LMSXI</modulo>
    <modulo idioma="catalán" horas="200" >CODE</modulo>
    <modulo>BADA</modulo>
</root>
```

Para que sea “bien formado”:

Nombre de etiquetas y atributos:
– Comenzará siempre por una letra (puede llevar tilde y se puede utilizar la ñ), un guión bajo (_) o dos puntos (:) seguido de cualquier combinación de los caracteres anteriores, números, guion (-) o punto(.). Por lo tanto, no se pueden utilizar espacios en blanco.
– No puede comenzar por XML ni variantes de mayúsculas/minúsculas (xml, Xml, ...).
– Se distingue entre mayúsculas y minúsculas (case-sensitive).
– El contenido de los elementos y el valor de los atributos puede tener cualquier carácter siempre que no interfiera en la sintaxis propia de XML. Por ejemplo, los carácteres & y < no pueden aparecer como parte del contenido, salvo en los comentarios, instrucciones de procesamiento y secciones CDATA. Lo mismo sucede con las comillas (simples o dobles): no pueden formar parte del valor de un atributo delimitado mediante el mismo tipo de comillas. Para solucionar esto, se utilizarán referencias, entidades o secciones CDATA.

Referencias, entidades y secciones CDATA

Los caracteres especiales como &, < , dobles comillas (“) y apóstrofe ('), pueden forma parte del contenido de un elemento o como valor de un atributo, utilizando una de las siguientes formas:

- **Empleando una referencia a un carácter Unicode. Las referencias siempre empiezan por &# (o por &#x si empleamos hexadecimal) y acaban en punto y coma (;). Por ejemplo, los caracteres & y < se podrían substituir respectivamente por las referencias & y < (en hexadecimal serían & y <).**

- **Empleando entidades. Las entidades empiezan por &, van seguidas del nombre de la entidad y terminan en punto y coma (;). Pueden crearse nuevas entidades como se verá más adelante y existen entidades predefinidas como por ejemplo:**

Referencias, entidades y secciones CDATA

Carácter	Entidade
&	&amp;
<	&lt;
>	&gt;
"	&quot;
'	&apos;

Referencias, entidades y secciones CDATA

El uso de cualquiera de las dos formas anteriores puede resultar pesado si hay que escribirlo muchas veces, y además dificulta mucho la lectura a las personas, por lo que la solución puede ser colocar el texto dentro de una sección CDATA.

Sintaxis:

```
<![CDATA[
texto
]]>
```

- **texto no puede llevar ningún símbolo que interfiera en la sintaxis.**

Referencias, entidades y secciones CDATA

Por ejemplo, para poder ver en un navegador un elemento con el contenido siguiente:

<códigoFuente>
El operador de concatenación es &. Los operadores de comparación son < y >
</códigoFuente>

Se podría escribir:

<códigoFuente>
<![CDATA[
El operador de concatenación es &.
Los operadores de comparación son < y >
]]>
</códigoFuente>:

Ó:

```
<códigoFuente>
El operador de concatenación es &amp;.
Los operadores de comparación son &lt; y &gt;
</códigoFuente>
```

Atributos especiales

Aunque cada documento XML tiene sus propios elementos, atributos y estructura ajustada a su finalidad, existen dos atributos especiales con un significado ya definido que viene recogido en la especificación de XML.

xml:lang

Cuando queramos indicar el idioma en el que está escrito un determinado contenido. Su valor será un código de dos letras que indicará el idioma, (es, en, fr, etc). Se le aplicará al elemento en el que se inserte, incluyendo el valor de otros atributos si los hubiese, y a todos sus hijos siempre y cuando no tengan un atributo xml:lang propio.

```
<descripción xml:lang="es">Color rojo</descripción>
```

Atributos especiales

xml:space

Xml define como espacios en blanco: espacio en blanco (32), tabulador (9), salto de línea (10) y retorno de carro (13) que permiten hacer el código más legible, sin embargo si quisiéramos que esos espacios fueran preservados, tendríamos que indicarlo mediante:

xml:space (default | preserve)

Espacios de nombres

Anotación al margen: *“no está de más echarle un ojo, pero de momento, no los vamos a usar”*.

Herramientas

IDE’s …

Visual Studio Code

+ Extensión XML Autor: Red Hat

+ XML Tools Autor: Josh Johnson

NetBeans Xpath Utility…

XML Copy Editor