Learning XML.
Un portal web para aprender XML en múltiples formatos.

Producto 3
Otros formatos, LaTeX

Grupo Create

Índice

1 Espacios web para aprender y practicar los lenguajes de marca

- (7 Ejercicios sobre documentos XML mal formados y corrección).
- (Gran cantidad de ejercicios sobre XML y lenguajes de marcas, incluido en las búsquedas grupales).
- (Ejercicios desde nivel básico sobre XML, sacado de las búsquedas grupales).
- (Ejercicios XML y lenguajes de marca desde nivel básico, sacado de las búsqueda grupales).
- (Examen de 25 preguntas sobre conocimiento general de XML).

2 Lenguajes de marcas

Diagrama sobre lenguajes de marcas

2.1 ¿Qué son los lenguajes de marcas?

Un **Lenguaje de Marcas**(Markup Language) es un modo de codificar (redactar) un documento donde, junto con el texto, se incorporan etiquetas (marcas o anotaciones) con información adicional relativa a la estructura del texto o su formato de presentación. Los Lenguajes de Marcas permiten hacer explícita la estructura de un documento, su contenido semántico o cualquier otra información lingüística o extralingüística que se quiera hacer patente.

El lenguaje de marcas más extendido es el HTML (HyperText Markup Language, lenguaje de marcado de hipertexto), fundamento del World Wide Web.

Los lenguajes de marcado suelen confundirse con lenguajes de programación. Sin embargo, no son lo mismo, ya que el lenguaje de marcado no

tiene funciones aritméticas o variables, como poseen los lenguajes de programación. Históricamente, el marcado se usaba y se usa en la industria editorial y de la comunicación, así como entre autores, editores impresores.Las principales características de los lenguajes de marcas son:

1. **Uso del texto plano**, para que cualquier usuario pueda leer y editar.
2. **Es un lenguaje compacto**, ya que se mezcla con el resto del texto del documento.
3. **Independencia del dispositivo**, para mostrarnos el contenido.
4. **Facilidad de procesamiento**, uso de diferentes lenguajes según el tipo de documento.
5. **Posibilidad de combinación** con otros lenguajes.

Los Lenguajes de Marcas se pueden clasificar de la siguiente manera:

1. **Lenguajes de presentación**: Define el formato (apariencia) del texto. Éstos suelen ocultar las etiquetas y mostrar al usuario solamente el texto con su formato.
2. **Lenguajes de procedimientos**: Orientado también a la presentación pero, además, el programa que representa el documento debe interpretar las etiquetas para realizar acciones en función de ellas.
3. **Lenguajes descriptivos o semánticos**: Describen las diferentes partes en las que se estructura el documento, es decir, definen su contenido, pero sin especificar cómo deben representarse.

2.2 Ejemplos de lenguajes de marcas

Dentro de los lenguajes de marcado descriptivos o semánticos hay que destacar un conjunto de lenguajes Especializados en un área concreta:

Tipo	Descripción	Ejemplo
Marcado de presentación	Indica el formato del texto, es decir cómo ha de presentarse el documento. La etiquetas de marcado suelen estar ocultas al usuario.	Microsoft Word.
Marcado de Procedimiento	También está enfocado hacia la presentación del texto, sin embargo, las marcas que formatean el texto son visibles para el usuario y permiten procesamiento (realizar un conjunto de acciones) según el tipo de etiqueta.	LaTeX, HTML.
Marcado descriptivo o semántico	Utiliza las marcas o etiquetas para describir los fragmentos de texto.	XML.

Área	**Lenguaje**	**Descripción /Ejemplo**
Matemática	MathML y OpenMath	Su objetivo es expresar notación matemática.
Geomática	Geography ML	Su objetivo es el modelaje, transporte y almacenamiento de información geográfica.
Aeronáutica	Spacecraft ML	
Multimedia	Synchronized Multimedia Integration Language	El lenguaje SMIL permite integrar audio, video, imágenes, texto o cualquier otro contenido multimedia.
Voz	VoiceXML	
Mensajería instantánea	XMPP	Con el protocolo XMPP queda establecida una plataforma para el intercambio de datos XML que puede ser usada en aplicaciones de mensajería instantánea.
Gráficos 3D	VRML/X3D, STEP	Formato de archivo normalizado que tiene como objetivo la representación de escenas u objetos interactivos tridimensionales.

3 eXtensible Markup language (XML)

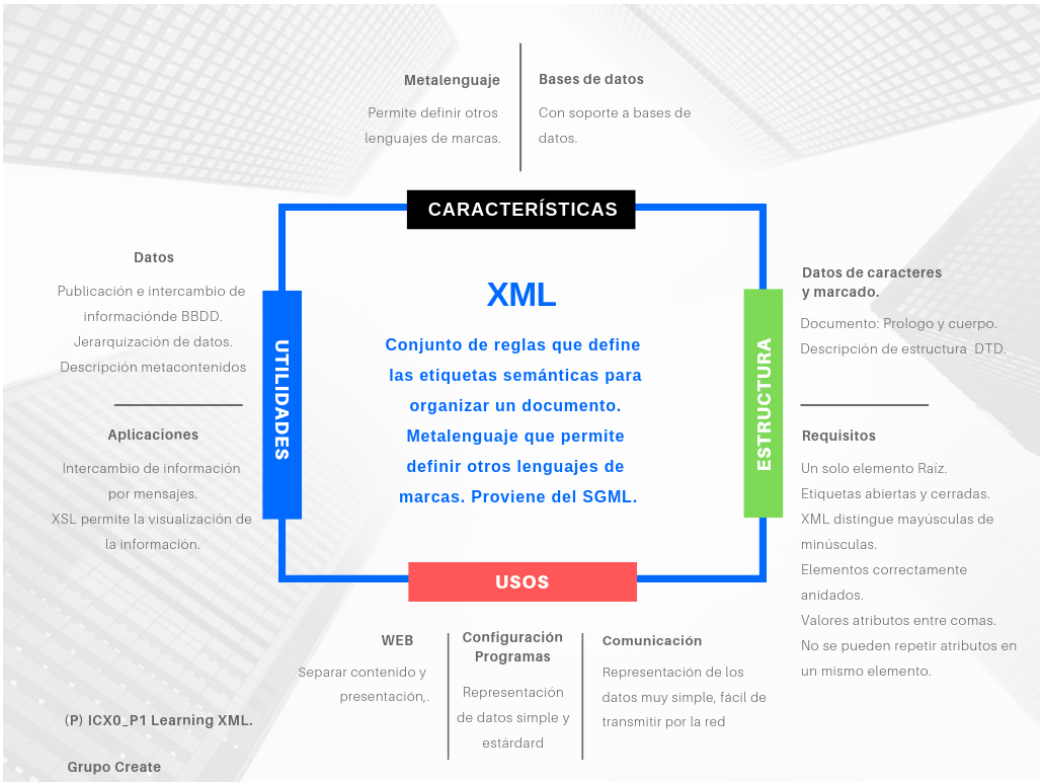


Diagrama sobre XML

3.1 Definición de XML

XML proviene de eXtensible Markup Language ("Lenguaje de Marcas Extensible"). Se trata de un metalenguaje (un lenguaje que se utiliza para decir algo acerca de otro) extensible de etiquetas que fue desarrollado por el Word Wide Web Consortium (W3C).

El XML es una adaptación del SGML (Standard Generalized Markup Language), un lenguaje que permite la organización y el etiquetado de documentos. Esto quiere decir que el XML no es un lenguaje en sí mismo, sino un sistema que permite definir lenguajes de acuerdo a las necesidades. El XHTML, el MathML y el SVG son algunos de los lenguajes que el XML tiene la capacidad de definir.

Las bases de datos, los documentos de texto, las hojas de cálculo y las

páginas web son algunos de los campos de aplicación del XML. El metalenguaje aparece como un estándar que estructura el intercambio de información entre las diferentes plataformas.**XML no es:**

- No es una una "*versión mejorada de HTML*"
- No es un lenguaje para hacer mejores páginas web.
- No es un lenguaje sustituto de HTML.
- No es difícil.

3.2 Utilidades de XML

Dado que, en gran parte, la utilidad de una herramienta depende de la creatividad de quien la utiliza, resulta imposible resumir todas las aplicaciones de XML. En pocas palabras, se puede decir que ofrece la posibilidad de estructurar y representar datos. En la actualidad, es común que los programas incluyan archivos de configuración en este formato; tal es el caso de Apache y de las aplicaciones creadas con la tecnología .NET (de Microsoft).

Cuando se desarrolla un programa con interfaz gráfica es necesario organizar todas las imágenes de manera que se vayan cargando a medida que se necesiten, y XML es de gran ayuda en estos casos: permite agruparlas, etiquetarlas, especificar su ubicación y relacionarlas con otros datos, según las necesidades de los diseñadores.

Pero además de facilitar la organización de los recursos y la configuración de un programa, XML cumple un papel muy importante que es, sin lugar a dudas, su punto fuerte: le permite comunicarse con otras aplicaciones, de diferentes plataformas y sin que importe el origen de la información en común. Se pueden tener, por ejemplo, un programa corriendo en Windows con una base de datos de SQL Server, y otro en Linux con Oracle, ambos compartiendo datos gracias a una estructura en XML.

Los servicios web, concepto muy común en esta era, son componentes de la Red que brindan la posibilidad de realizar una serie variada de operaciones, a través de métodos concretos que aprovechan el metalenguaje XML para sus comunicaciones, gracias a lo cual cualquier plataforma puede hacer uso de sus ventajas.En resumen:

- XML aplicado a los **sitios web**: permite separar contenido y presentación, y que los mismos datos se puedan mostrar de varias formas distintas sin demasiado esfuerzo.
- XML para la **comunicación** entre aplicaciones: 1, estándar. En los últimos tiempos este uso se está haciendo muy popular con el surgimiento de los Servicios web.
- XML para la **configuración** de programas: representación de los datos simple y estándar, en contraposición con los crípticos formatos propietarios.

4 Estructura de documentos XML

La estructura básica de XML se compone de un prólogo y un elemento raíz. El prólogo se refiere al resto del documento y contiene información sobre el mismo, tal como su versión, el código de caracteres usados y una descripción de la estructura del documento, para la que se suele usar DTD (Document Type Definition) o Schema XML.

Dentro de elemento raíz, que es el principal, se encuentran el resto de los elementos que conforman el cuerpo del documento. Tiene una estructura jerarquizada, a modo que el primer elemento (raíz) se podría denominar como el "padre" del resto de elementos, los cuales derivan de él en forma ramificada y ordenada por jerarquía. Todos los elementos a su vez pueden tener sus propias derivaciones o elementos hijos.

```
<?xml version="1.0"?>
<!DOCTYPE MENSAJE SYSTEM "mensaje.dtd">
<mensaje>
  <remite>
    <nombre>Alfredo Reino</nombre>
    <email>alf@ibium.com</email>
  </remite>
  <destinatario>
    <nombre>Bill Clinton</nombre>
    <email>president@whitehouse.gov</email>
  </destinatario>
  <asunto>Hola Bill</asunto>
  <texto>
    <parrafo>¿'Hola qué tal? Hace <enfasis>mucho</enfasis> que no escribes. A ver si llamas y
↪ quedamos para tomar algo.</parrafo>
  </texto>
</mensaje>
```

Ejemplo de estructura de XML

Normas de diseño de un Documento XML Que marcas hay que incluir sus nombres la jerarquía, que debe de poseer, qué información ha de contener.

- ¿Que se ha de marcar?
- Organización y Estructura.
- Elementos o atributos.

consideraciones.

- Necesidades futuras.
- Revisión de nuestros requisitos.
- Nombres descriptivos.

Tipos de datos XML

Existen 19 tipos de datos simples predefinidos primitivos, que se pueden agrupar en 4 categorías:Tipos cadena

- string: secuencia de longitud finita de caracteres.
- anyURI: una uri estándar de Internet
- NOTATION: declara enlaces a contenido externo no-XML
- Qname: una cadena legal Qname (nombre con cualificador)

Tipos binario codificado

- boolean: toma los valores “true” o “false”.
- hexBinary: dato binario codificado como una serie de pares de dígitos hexadecimales
- base64Binary: datos binarios codificados en base 64

Tipos numéricos

- decimal: número decimal de precisión (dígitos significativos) arbitraria.
- float: número de punto flotante de 32 bits de precisión simple.

- double: número de punto flotante de 64 bits de doble precisión.

Tipos de fecha/hora

- duration: duración de tiempo
- dateTime: instante de tiempo específico, usando calendario gregoriano, en formato "YYYYMM-DDThh:mm:ss"
- date: fecha específica del calendario gregoriano, en formato "YYYY-MM-DD" *
- time: una instancia de tiempo que ocurre cada día, en formato "hh:mm:ss"
- gYearMonth: un año y mes del calendario gregoriano
- gYear: año del calendario gregoriano
- gMonthDay: día y mes del calendario gregoriano
- gMonth: un mes del calendario gregoriano
- gDay: una fecha del calendario gregoriano (día)

4.1 ¿Qué es DTD?

DTD

DEFINICIÓN DE TIPO DE DOCUMENTO

GRUPO CREATE

(P) ICX0_P1 LEARNING XML

DEFINICIÓN

Archivos de texto que encierran una definición formal de un tipo de documento y, tienen la función de especificar la estructura lógica de un archivo XML proporcionando los nombres de los elementos, atributos y entidades que utiliza. Suministra la información sobre la forma en que se pueden usar conjuntamente.

EJEMPLO GRAMÁTICA

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE Casas_Rurales [
<!ELEMENT Casas_Rurales (Casa)*>
      <!ELEMENT Casa (Dirección, Descripción, Estado,
Tamaño)>
      <!ELEMENT Dirección (#PCDATA) >
      <!ELEMENT Descripción (#PCDATA) >
      <!ELEMENT Estado (#PCDATA) >
      <!ELEMENT Tamaño (#PCDATA) >
]>
<Casas_Rurales>
...
</Casas_Rurales>
```

VALIDACIÓN

El procesador de XML utiliza DTD para verificar el documento.

Documento valido: se adapta al patrón DTD y cumple todas las reglas de XML.

Documento bien formado: cumple todas las reglas XML.

DECLARACIONES DE ENTIDADES

ENTIDAD: NOMBRE+ VALOR
Sintaxis entidad: (&nombreEntidad;)

Las entidades pueden ser internas o externas, a su vez estas pueden ser generales o parámetricas .

EJEMPLOS
Entidad interna general:
<!ENTITY nombreEntidad "valorEntidad">
Entidad externa general:
<!ENTITY nombreEntidad SYSTEM "uri">
Entidad externa NO texto:
<!ENTITY nombreEntidad SYSTEM "uri" NDATA tipo>
Entidad paramétrica:
<!ENTITY % nombreEntidad "valorEntidad">

DECLARACIÓN DE ELEMENTOS

Nombre del elemento: "nombreElemento"
Expresión que define el contenido: "(contenido)"
<!ELEMENT nombreElemento (contenido)>

EMPTY: el elemento no puede tener contenido
<!ELEMENT ejemplo EMPTY>
ANY: el elemento puede contener cualquier cosa
<!ELEMENT ejemplo ANY>
(#PCDATA): significa que el elemento puede contener texto
<!ELEMENT ejemplo (#PCDATA)>

DECLARACIÓN DE ATRIBUTOS

SINTAXIS:
<!ATTLIST nombreElemento nombreAtributo tipoAtributo valorInicialAtributo >
"nombreElemento": es el nombre del elemento
"nombreAtributo": es el nombre del atributo
"tipoAtributo": tipo de datos
"valorInicialAtributo: es el valor determinado del atributo

Se pueden otorgar varios atributos a un mismo elemento
<!ATTLIST nombreElemento nombreAtributo1 tipoAtributo1 valorInicialAtributo1>
<!ATTLIST nombreElemento nombreAtributo2 tipoAtributo2 valorInicialAtributo2>

DECLARACIONES DE NOTACIONES

No son analizadas por el procesador de XML

En este caso no usamos la notación habitual (&nombreEntidad;) sino el nombre de la entidad directamente.

Diagrama de DTD

DTD (definición de tipo de documento) es un documento que define la estructura del documento XML, ello comprende los elementos, atributos, notaciones, entidades y aquellos elementos que conforman el documento XML, enumerando las veces que pueden aparecer, su jerarquía, su orden, etc...

El procesador de XML utiliza DTD para verificar que el documento es válido y cumple las reglas, a este proceso se le denomina validación. Hay que hacer una distinción en este proceso entre documento válido y documento bien formado, ya que mientras el primero cumple todos los requisitos (se adapta al patrón DTD y cumple todas las reglas de XML) el segundo solo cumple la segunda premisa.

La DTD que debe utilizar el procesador XML para validar el documento XML se indica mediante la etiqueta DOCTYPE. La DTD puede estar incluida en el propio documento (interno), ser un documento externo o combinarse ambas.

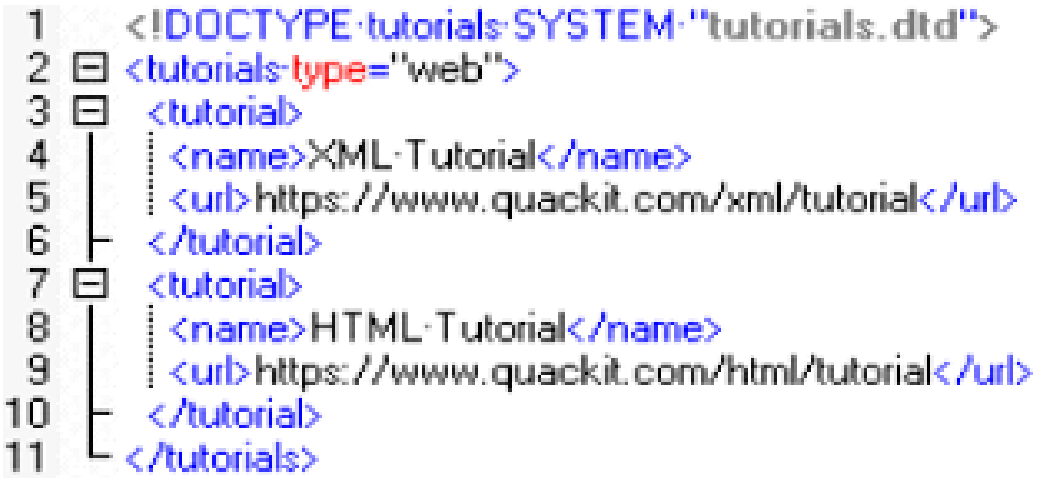
4.2 Ejemplos de gramáticas DTD

Este es un ejemplo de una DTD interna. Es interno porque el DTD está incluido en el documento XML de destino:

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE tutorials [
<!ELEMENT tutorials (tutorial)+>
<!ELEMENT tutorial (name,url)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT url (#PCDATA)>
<!ATTLIST tutorials type CDATA #REQUIRED>
]>
<tutorials type="web">
  <tutorial>
    <name>XML Tutorial</name>
    <url>https://www.quackit.com/xml/tutorial</url>
  </tutorial>
  <tutorial>
    <name>HTML Tutorial</name>
    <url>https://www.quackit.com/html/tutorial</url>
  </tutorial>
</tutorials>
```

Ejemplo de DTD interna

Aquí hay un ejemplo de un documento XML que utiliza una DTD externa. Hay que tener en cuenta que el standalone se establece en no. Esto se debe a que el documento se basa en un recurso externo (la DTD):



Ejemplo de DTD externa

4.3 Principales elementos de una gramática DTD

Las DTDs describen la estructura de los documentos XML mediante declaraciones.Hay cuatro tipos de declaraciones:

- **Declaraciones de entidades**. Para ello primero hemos de explicar que una entidad consta de un nombre y un valor. El procesador XML sustituye las referencias a entidades por sus valores antes de procesar el documento. Así que una vez definida la identidad (nombre+valor) se puede utilizar en el documento mediante una referencia a la misma. Esta referencia empieza con el carácter "", siguiendo con el nombre de la entidad y terminando con ";" (nombreEntidad;) Las entidades pueden ser internas (declaradas dentro de DTD) o externas (declaradas fuera de DTD) a su vez estas pueden ser generales (se deben declarar en el DTD antes de que se puedan usar en el documento XML) o parámetricas (permiten la creación de secciones de texto de recambio reutilizables).

- **Declaraciones de notaciones**. Hacen referencia a entidades externas que no va a analizar el procesador XML. En este caso no usamos la notación habitual (nombreEntidad;) sino el nombre de la entidad directamente.

- **Declaraciones de elementos**. Indican los elementos permitidos en

un documento y su contenido (que puede ser simplemente texto u otros elementos).

- **Declaraciones de atributos**. que indican los atributos permitidos en cada elemento y el tipo o valores permitidos de cada elemento. Hay varias notaciones para los atributos según su tipo, CDATA, NMTOKEN, NMTOKENS, valores, ID, IDREF, IDREFS, ENTITY, ENTITIES y NOTATION.

5 Validación documentos XML

En primer lugar, los documentos XML deben basarse en la sintaxis definida en la especificación XML para ser correctos (documentos bien formados). Esta sintaxis impone cosas como la coincidencia de mayúsculas/minúsculas en los nombres de etiqueta, comillas obligatorias para los valores de atributo, etc. Sin embargo, para tener un control más preciso sobre el contenido de los documentos es necesario un proceso de análisis más exhaustivo.

La validación es la parte más importante dentro de este análisis, ya que determina si un documento creado se ciñe a las restricciones descritas en el esquema utilizado para su construcción. Controlar el diseño de documentos a través de esquemas aumenta su grado de fiabilidad, consistencia y precisión, facilitando su intercambio entre aplicaciones y usuarios. Cuando creamos documentos XML válidos aumentamos su funcionalidad y utilidad.

Es importante que un documento no solo cumpla las reglas XML si no que también ha de estar bien formado para trabajar con él. Esto es debido a que ha de mantener una homogeneidad en su estructura para que los servicios que se prestan a través de documentos XML mantengan su coherencia y los resultados de los mismos tengan sentido en su producto final.

- **Estar bien formado**: Cumple con la sintaxis XML.
- **Ser válido**: además de estar bien formado, cumple con determinadas reglas establecidas.

La validación se encarga de verificar:

- **La corrección de los datos**: aunque validar contra un esquema no garantiza al 100% que los datos son correctos, nos permite detectar formatos nulos o valores fuera de rango y por tanto incorrectos.

- **La integridad de los datos**: al validar, se comprueba que toda la información obligatoria está presente en el documento.
- **El entendimiento compartido de los datos**: a través de la validación se comprueba que el emisor y receptor perciban el documento de la misma manera, que lo interpreten igual.

El éxito de la validación dependerá de factores como:

- **Dónde se originan los documentos**: si son fuentes confiables o no.
- **Quién los crea**: si son creados por una aplicación automáticamente o por un usuario de forma manual.
- **Quién los manipula**: también es posible introducir errores involuntariamente durante la manipulación de los datos y documentos.
- **La calidad de los datos**: si los documentos se generan directamente de una base de datos de herencia, pueden no estar completos o correctos al 100%.
- **El rendimiento del procesador o aparato que realice la validación**: el procesado no es inmediato, necesita su tiempo. Si el rendimiento es crítico, se pueden aplicar diversas alternativas para reducir el coste computacional como limitar la validación a algunos aspectos, o crear un código específico para la aplicación particular que lo utiliza y se ejecute de forma más eficaz.

5.1 Herramientas de validación de documentos XML

Teniendo presente que existen diferentes métodos de validación para documentos XML (DTD, XML Schema, Relax NG y Schematron), que son por así decirlo, las reglas que han de interpretar las herramientas de validación, establecemos una interdependencia entre métodos y herramientas.

Para la creación de documentos XML, se puede trabajar de una forma, por así decirlo, manual, en la que vamos escribiendo el documento en un editor de texto, el cual no está preparado para advertirnos de los posibles errores que vayamos cometiendo, o se puede trabajar con una herramienta específica para XML, la cual nos puede avisar tanto de fallos en la sintaxis como en su estructura final. A esto le llamamos herramienta de validación, dentro

de esta categoría encontramos los **editores** de XML y los **analizadores** de XML.

Del primer grupo, los **editores**, son herramientas que nos avisan de los posibles errores de concepción del documento, que vayamos teniendo mientras escribimos (sintaxis, contenido, forma...). Existen numerosos editores XML, tanto de pago como open source. Cooktop ,
XML Copy Editor ,
XML pro ,
XMLSpy ,
Liquid XML ,
Turbo XML ,
XML Notepad ,
XMLwriter .

Cada una de estas herramientas tienen unas características determinadas y utilidades específicas, por ello hemos de analizarlas cuidadosamente para elegir aquella que se adapte mejor a nuestras necesidades.

El segundo grupo, los analizadores, son herramientas para procesar el texto una vez ya terminado, quiere decir que no nos ayudan durante el proceso de escritura del mismo, si no una vez ya terminado. Hay dos subgrupos dentro de los analizadores, los validantes, que comprueban que el documento está bien formado y lo validan (corrección sintáctica XML y cumplimiento de reglas específicas), y los no validadores, que sólo comprueban que el documento está bien formado pero no lo validan, por tanto procesan el documento a mayor velocidad. Esta diferencia entre unos y otros viene marcada por su utilización, mientras que los validantes se utilizan precisamente para validar, los no validantes se usan para documentos validados anteriormente antes de presentarlos. Dentro de los analizadores, las herramientas más usadas son: Xerces ,
XMLSpy ,
MSXML Parser ,
Stylus Studio XML ,
Xpat ,
XML Parser .

6 Transformación de estructuras XML

Un documento XML ha de disponer de ciertas "reglas complementarias" para que pueda generar un resultado, como por ejemplo, en una aplicación. Estas "reglas" que en realidad son un código de un programa, van a transformar el documento original XML para que esta aplicación lo pueda procesar y presentar el resultado requerido. Este paquete de "reglas complementarias", viene dado en un documento XSLT y se ejecuta (documentos XML+XSTL) en un **procesador XLST**, tras lo cual ya obtendremos el resultado final deseado.

Es interesante advertir que a un mismo documento XML, se le pueden aplicar diferentes **hojas de estilo XSTL**, obteniendo así desde ese documento inicial diferentes resultados según la transformación que ejecutemos con estas hojas de estilo.Una transformación puede tener lugar en una de tres ubicaciones:

- En el servidor
- En el cliente (por ejemplo, su navegador web)
- Con un programa independiente.

6.1 ¿Qué es el lenguaje de transformación XSLT?

Es un estándar de la organización W3C que presenta una forma de transformar documentos XML en otros e incluso a formatos que no son XML.

Las hojas de estilo XSLT - aunque el término de hojas de estilo no se aplica sobre la función directa del XSLT - realizan la transformación del documento utilizando una o varias reglas de plantilla. Estas reglas de plantilla unidas al documento fuente a transformar alimentan un procesador de XSLT, el que realiza las transformaciones deseadas poniendo el resultado en un archivo de salida, o, como en el caso de una página web, las hace directamente en un dispositivo de presentación tal como el monitor del usuario.

Actualmente, **XSLT** es muy usado en la edición web, generando páginas HTML o XHTML . La unión de XML y **XSLT** permite separar contenido y presentación , aumentando así la productividad.Las principales características que nos ofrece son:

- Seleccionar solo los elementos o atributos que nos interesan.

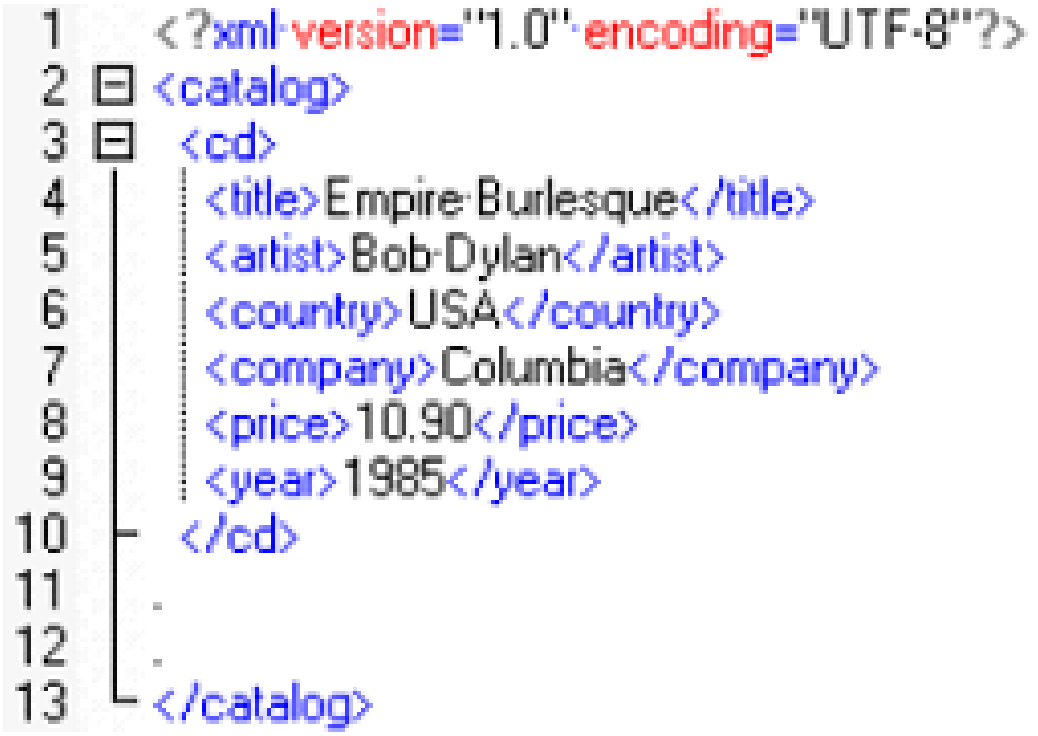
- Ordenar elementos.
- Realizar comparaciones.
- Establecer condiciones.

Así, partiendo de un único documento XML podemos obtener infinidad de documentos resultantes diferentes simplemente cambiando la hoja de estilos. Viene a ser como aplicar una hoja de estilos CSS a una página HTML para manipular la visualización pero en este caso lo que se manipula es la información.El proceso de transformación sigue las siguientes pautas:

- El procesador analiza el documento y construye una estructura para el mismo en forma de árbol.
- El procesador va recorriendo todos los nodos desde el nodo raíz, aplicando a cada nodo una plantilla, sustituyendo el nodo por el resultado.
- Cuando se ha hecho todo el recorrido de todos los nodos, se da por finalizada la transformación.

6.2 Ejemplos de transformaciones XSLT

Ejemplo 1 Comenzamos con un documento XML sin procesar.Queremos transformar el siguiente documento XML ("cdcatalog.xml") en XHTML:



Ejemplo de fichero XML

Luego creamos una hoja de estilo XSL (”cdcatalog.xsl”) con una plantilla de transformación:

```
1   <?xml version="1.0" encoding="UTF-8"?>
2
3   <xsl:stylesheet version="1.0"
4   xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
5
6   <xsl:template match="/">
7     <html>
8     <body>
9     <h2>My CD Collection</h2>
10    <table border="1">
11      <tr bgcolor="#9acd32">
12        <th>Title</th>
13        <th>Artist</th>
14      </tr>
15      <xsl:for-each select="catalog/cd">
16      <tr>
17        <td><xsl:value-of select="title"/></td>
18        <td><xsl:value-of select="artist"/></td>
19      </tr>
20      </xsl:for-each>
21    </table>
22    </body>
23    </html>
24  </xsl:template>
25
26  </xsl:stylesheet>
```

Ejemplo de hoja de estilo XSLT

Agregamos la referencia de la hoja de estilo XSL a su documento XML ("cdcatalog.xml"):

```xml
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="cdcatalog.xsl"?>
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
  .
  .
</catalog>
```

Ejemplo de fichero XML con referencia a fichero XSLT

Ejemplo 2 Partiendo de este código XML:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<biblioteca>
  <libro>
    <titulo>La vida está en otra parte</titulo>
    <autor>Milan Kundera</autor>
    <fechaPublicacion año="1973"/>
  </libro>
  <libro>
    <titulo>Pantaleón y las visitadoras</titulo>
    <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
    <fechaPublicacion año="1973"/>
  </libro>
  <libro>
    <titulo>Conversación en la catedral</titulo>
    <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
    <fechaPublicacion año="1969"/>
  </libro>
</biblioteca>
```

Ejemplo de fichero XML

Si aplicamos la siguiente plantilla XSLT:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">

  <xsl:template match="autor">
  </xsl:template>

</xsl:stylesheet>
```

Ejemplo de hoja de estilo XSLT

Obtendremos como resultado:

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

  La vida está en otra parte

  Pantaleón y las visitadoras

  Conversación en la catedral
```

Ejemplo de resultado de la transformación

6.3 Principales instrucciones XSLT

value-of Esta instrucción extrae el contenido del nodo seleccionado.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">

  <xsl:template match="libro">
    <xsl:value-of select="autor"/>
  </xsl:template>

</xsl:stylesheet>
```

Ejemplo de value-of

Resultado:

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

  Milan Kundera
  Mario Vargas Llosa
  Mario Vargas Llosa
```

Resultado de value-of

apply-templates Esta instrucción hace que se apliquen a los subelementos las reglas que les sean aplicables.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">

  <xsl:template match="/">
    <html>
      <h1>Autores</h1>
      <xsl:apply-templates />
    </html>
  </xsl:template>

  <xsl:template match="libro">
    <p><xsl:value-of select="autor"/></p>
  </xsl:template>

</xsl:stylesheet>
```

Ejemplo de apply-templates

Resultado:

```
<?xml version="1.0" encoding="UTF-8"?>
<html><h1>Autores</h1>
  <p>Milan Kundera</p>
  <p>Mario Vargas Llosa</p>
  <p>Mario Vargas Llosa</p>
</html>
```

Resultado de apply-templates

text Esta instrucción permite generar texto que no se puede generar simplemente añadiendolo (saltos de líneas y espacios en blanco, por ejemplo).

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">

  <xsl:template match="/">
    <html>
      <xsl:text>&#10;  </xsl:text>
      <h1>Autores</h1>
      <xsl:apply-templates />
    </html>
  </xsl:template>

  <xsl:template match="libro">
    <p><xsl:value-of select="autor"/></p>
  </xsl:template>

</xsl:stylesheet>
```

Ejemplo de text

Resultado:

```
<?xml version="1.0" encoding="UTF-8"?>
<html>
  <h1>Autores</h1>
  <p>Milan Kundera</p>
  <p>Mario Vargas Llosa</p>
  <p>Mario Vargas Llosa</p>
</html>
```

Resultado de text

strip-space Esta instrucción permite indicar si los elementos que contienen única-mente espacios en blanco se incluyen en la transformación o no.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:strip-space elements="*" />

  <xsl:template match="/">
    <html>
      <h1>Autores</h1>
      <xsl:apply-templates />
    </html>
  </xsl:template>

  <xsl:template match="libro">
    <p><xsl:value-of select="autor"/></p>
  </xsl:template>

</xsl:stylesheet>
```

Ejemplo de strip-space

Resultado:

```
<?xml version="1.0" encoding="UTF-8"?>
<html>
  <h1>Autores</h1>
  <p>Milan Kundera</p>
  <p>Mario Vargas Llosa</p>
  <p>Mario Vargas Llosa</p>
</html>
```

Resultado de strip-space

attribute Esta instrucción permite generar un atributo y su valor. Se utiliza cuando el valor del atributo se obtiene a su vez de algún nodo.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">

  <xsl:template match="licencia">
    <p><img>
    <xsl:attribute name="src">
      <xsl:value-of select="imagen" />
    </xsl:attribute>
    </img>
    </p>
  </xsl:template>

</xsl:stylesheet>
```

Ejemplo de attribute

Resultado:

```
<?xml version="1.0" encoding="UTF-8"?>
<p>
  <img src="cc-bysa-88x31.png"/>
</p>
```

Resultado de attribute

6.4 Herramientas de transformación con XSLT

En el mercado existen diversas herramientas de transformación. Hemos elegido para destacar las siguientes porque son gratuitas.

- Cooktop Herramienta gratuita, bajo Windows solo.
- XML Copy Editor Herramienta libre, bajo Windows, Mac OS y GNU/Linux.

7 Ejemplos de lenguajes basados en XML

Tal como habíamos indicado en la definición de XML, este es un metalenguaje, es decir, que es un lenguaje que sirve como expresión para definir otros lenguajes. De esta manera, XML, se ha convertido en el generador de multitud de lenguajes utilizados para tareas muy dispares, ya que utilizando XML, se puede estructurar cualquier tipo de información (documentos, registros, bbdd, audio, video...). Podemos encontrar lenguajes basados en XML especializados en estos aspectos, así por ejemplo tenemos SMLI, que se encarga de aspectos multimedia, SSML, que trata con voz al igual que VoiceXML o SGRS. Para imágenes tenemos VML y también SVG. Para matemáticas existe MathML.

Entre estos lenguajes está xHTML, que es la adaptación del lenguaje HTML a XML. HTML es un lenguaje de marcado al igual que XML, pero en este caso es de hipertexto para la realización de páginas Web. El caso que nos atañe xHTML es una solución buscada por el Consorcio World Wide Web con el objetivo de conseguir una Web semántica, donde la información y la forma de presentarla, discurran por separado.

7.1 xHTML

7.1.1 ¿Que es xHTML?

xHTML es una adaptación del lenguaje HTML para que sea compatible con el lenguaje XML. De esta forma, mantiene la mayoría de características del HTML, aunque con elementos del XML.

El xHTML, por lo tanto otorga mayor robustez y capacidad de adaptación, resultando importante para el desarrollo de la Web 3.0[2] gracias a su tipo de codificación.

7.1.2 Utilidades

XHTML se utiliza para marcar contenido como texto, imágenes y enlaces en forma de hipervínculos para crear una cierta estructura que puede ser mostrada por los navegadores. Los documentos pueden ser estructurados con XHTML para hacerlos legibles para un analizador. El analizador interpreta los elementos de marcado especificados en las definiciones del lenguaje XHTML y reproduce el contenido de estos elementos de una manera específica.

1. Los documentos creados con el mismo resulta que ofrecen un estupendo rendimiento.

2. Se mantiene de manera muy sencilla.

3. No menos notable es el hecho de que permite una gran facilidad a la hora de acometer lo que es la edición directa del código en cuestión.

4. Es un lenguaje compatible con distintos estándares.

5. Hacer uso del XHTML permite emplear herramientas actuales que ofrecen un mejor rendimiento que las de otros lenguajes.

7.1.3 Declaración

1. Un prólogo XML indicando la versión.
2. Un declaración DOCTYPE.
3. Un elemento HTML raíz en el que deberemos referenciar el espacio de nombre "http://www.w3.org/1999/xhtml".

Además, dentro de head la etiqueta title es obligatoria.

7.2 RSS / Atom

7.2.1 ¿Qué es RSS o Atom?

RSS es un sublenguaje XML. Por tanto, un archivo RSS es un documento de texto compuesto por etiquetas acotadas entre los símbolos mayor y menor que similares a las utilizadas en el XHTML.

RSS corresponde a Rich Site Summary o Really Simple Syndication, y está diseñado para la distribución (syndication en inglés) de noticias o información tipo noticias contenidas en sitios web y weblogs.

Los archivos RSS comúnmente se llaman feeds RSS o canales RSS y contienen un resumen de lo publicado en el sitio web de origen. Se estructura en uno o más ítems. Cada ítem consta de un título, un resumen de texto y un enlace a la fuente original en la web donde se encuentra el texto completo.

Atom También es un sublenguaje XML. No se corresponde ni se basa en ninguna versión de RSS, pero es un formato muy similar a éste y que sobre todo tiene el mismo objetivo: permitir la distribución de contenidos y noticias de sitios web. La versión más actual es la 0.3 de febrero de 2004. Las mejoras que supone respecto a RSS (en cualquiera de sus versiones) hacen que su uso se extienda rápidamente a pesar de ser algo más complicado. Un documento Atom puede contener más información (y más compleja) y es más consistente que un documento RSS.

7.2.2 Utilidades

Los archivos RSS, a diferencia de los archivos XHTML, no son interpretados por los navegadores web y al abrirlos lo que hacen es mostrar el código XML que compone el archivo RSS. Los feeds o canales en formato Atom son parcialmente visualizados por los navegadores, pero mostrando un aviso

de que el documento corresponde a un feed y no a una página web. Para visualizar directamente un feed RSS es necesario utilizar un programa lector o agregador de feeds.

7.3 Otros lenguajes basados en XML

- **Extensible Stylesheet Language (XSL)**. El Lenguaje de Hoja de Estilo Extensible (eX-tensible Stylesheet Language, XSL) es una familia de lenguajes que permiten describir como los archivos codificados en XML serán formateados (para mostrarlos) o transformados. Hay tres lenguajes en esta familia: XSL Transformations (XSLT), XSL Formatting Objects (XSL- FO) y XML Path Language.

- **Lenguaje de enlace XML (XLINK)**. XLink es una aplicación XML que intenta superar las limitaciones que tienen los enlaces de hipertexto en HTML. XLink 1.1 es ya una recomendación W3C.