JAVA WEB SERVICES

eXTensible Markup Language

ERIC GUSTAVO CORONEL CASTILLO

www.desarrollasoftware.com

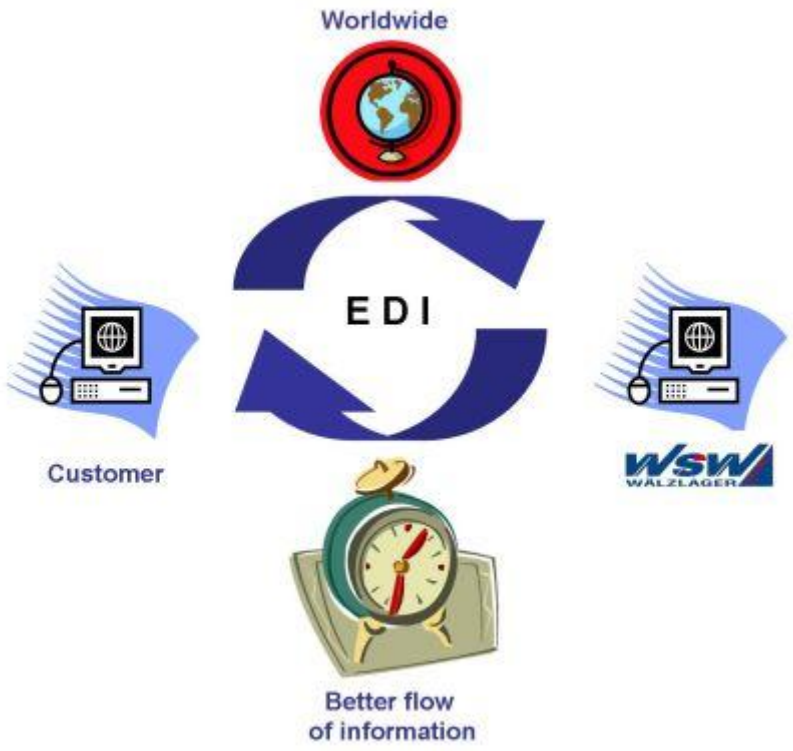
gcoronelc@gmail.com

Contenido

1 INTRODUCCIÓN

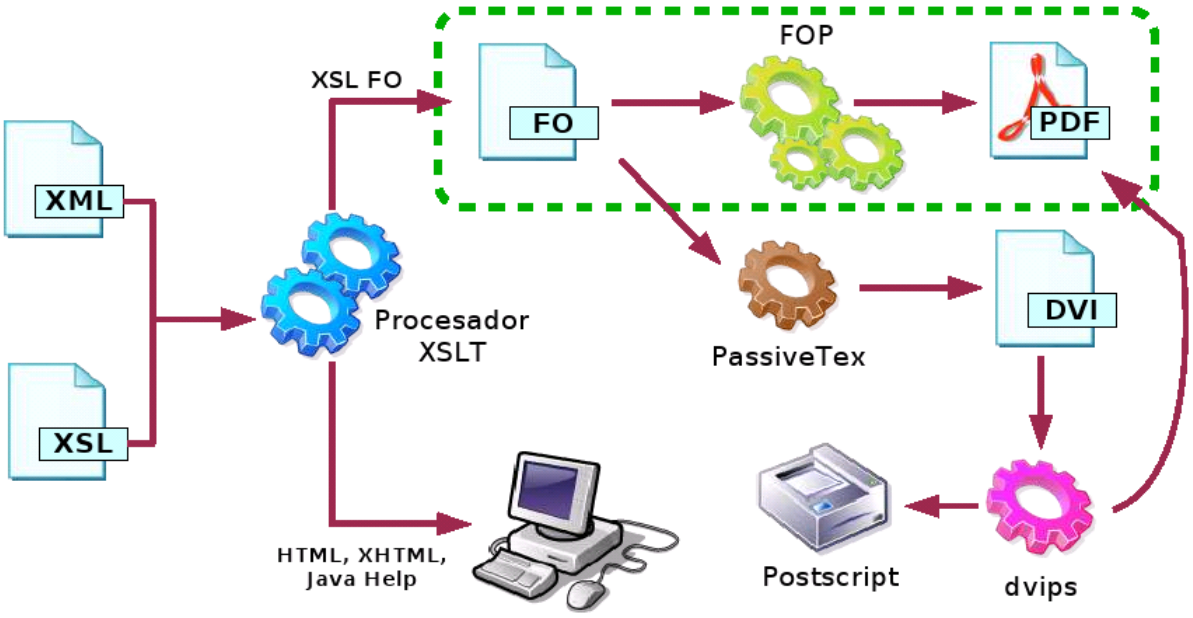
1.1 Tecnologías para el intercambio de datos

1.1.1 Electronic Data Interchange (EDI)



Es el intercambio electrónico de datos de computadora a computadora entre Socios Comerciales (B2B), Consumidores potenciales (B2C) todo esto para el soporte de ERP, CRM, BPM con la finalidad de ahorrar tiempo al eliminar los tradicionales métodos de preparación y envío de documentos a través de mensajería.

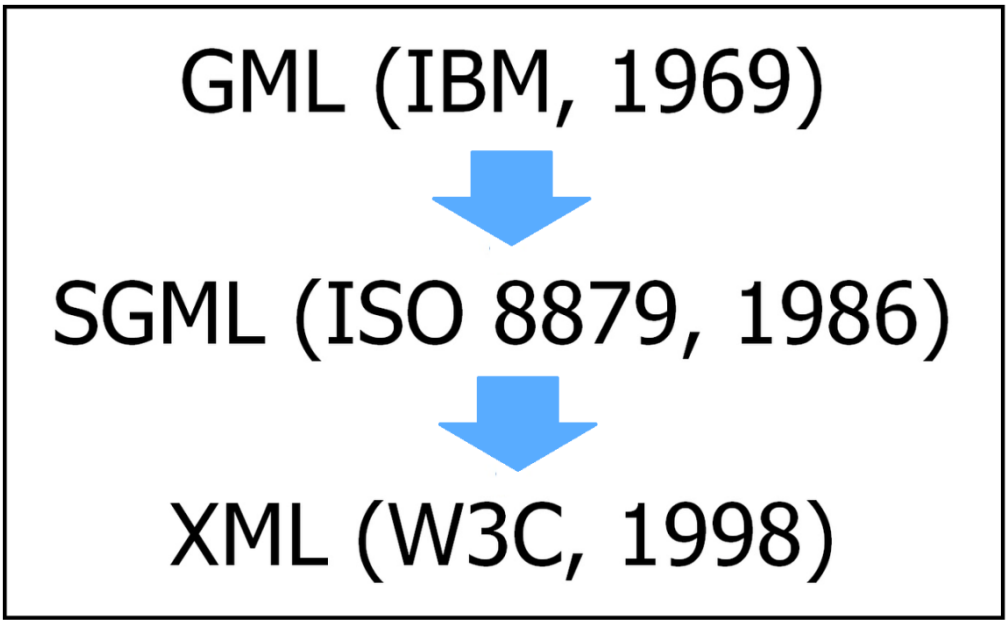
1.1.2 Extensible Markup Language (XML)



XML no ha nacido sólo para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa imaginable.

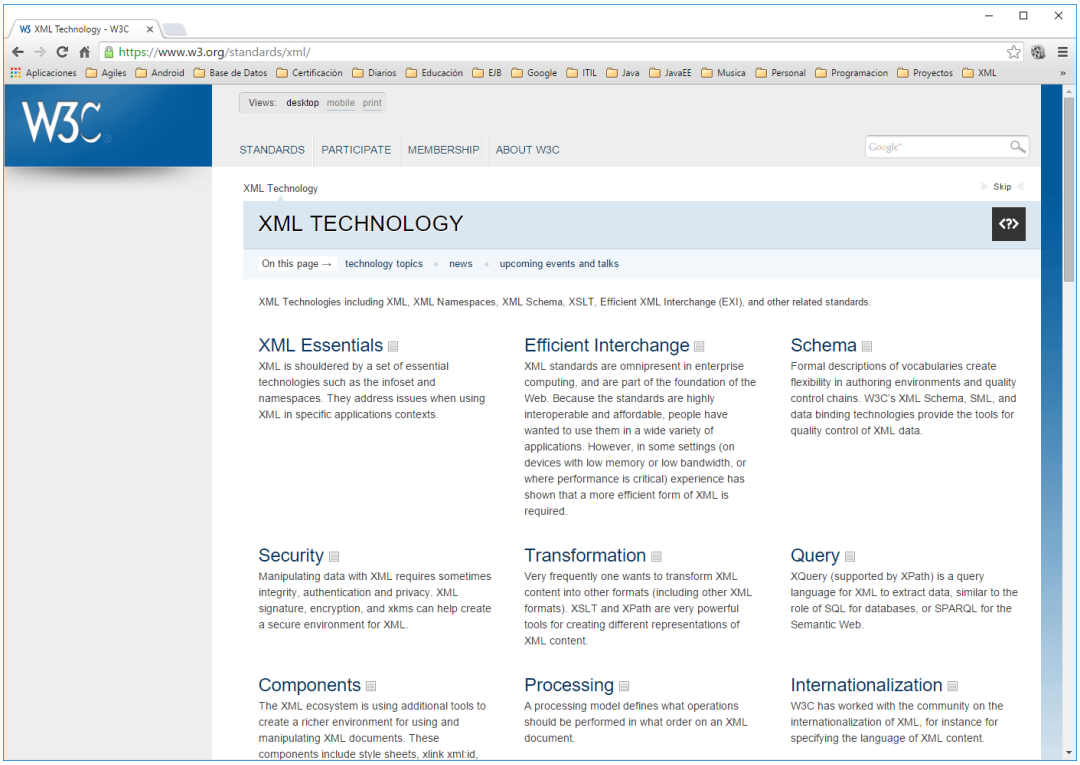
En la actualidad todo intercambio de información implementa XML, entonces podemos afirmar que los sistemas B2B, B2C, ERP, CRM, BPM y muchos sistemas de información de cualquier plataforma y de cualquier lenguaje usa XML.

1.2 Introducción a XML



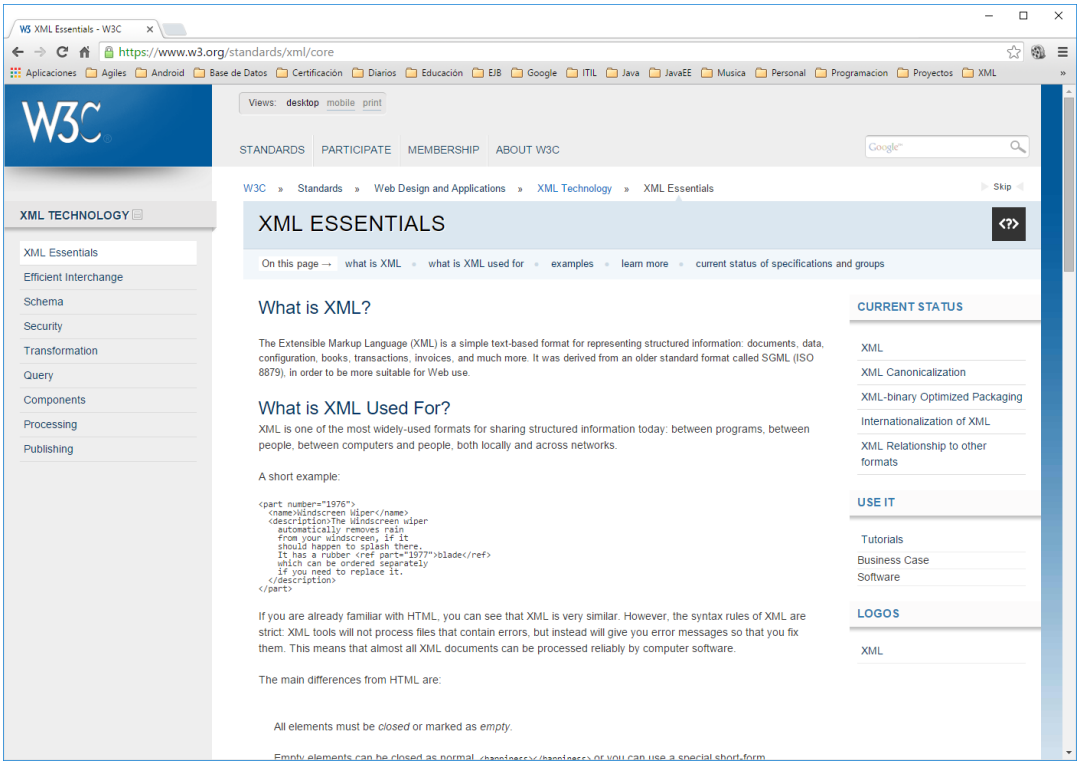
XML proviene de un lenguaje inventado por IBM llamado GML (Generalized Markup Language), este lenguaje fue normalizado por la ISO en 1986 bajo el nombre de SGML (Standard Generalized Markup Language).

A partir de SGML la W3C crea XML y es aprobado en febrero de 1998, en la actualidad es increíble la cantidad de información, aplicaciones y software que se han ido generando alrededor de este estándar.



Algunas características de XML:

- Es un método para introducir datos estructurados en un fichero de texto.
- Se parece al HTML pero no es HTML.
- XML es gratis, independiente de la plataforma y ampliamente distribuida



En el sitio oficial tienes toda una sección dedicada a XML, donde encontraras toda la información necesaria sobre XML.

1.3 Diferencia entre GML, SGML, HTML y XML

1.3.1 GML

```
:h1.Chapter 1:  Introduction
:p.GML supported hierarchical containers, such as
:ol
:li.Ordered lists (like this one),
:li.Unordered lists, and
:li.Definition lists
:eol.
as well as simple structures.
:p.Markup minimization (later generalized and formalized in SGML),
allowed the end-tags to be omitted for the "h1" and "p" elements.
```

En este lenguaje las etiquetas se indican mediante dos puntos (:), y definen los elementos de un documento.

- Encabezado

  ```
  :h1.Chapter 1:  Introduction
  ```

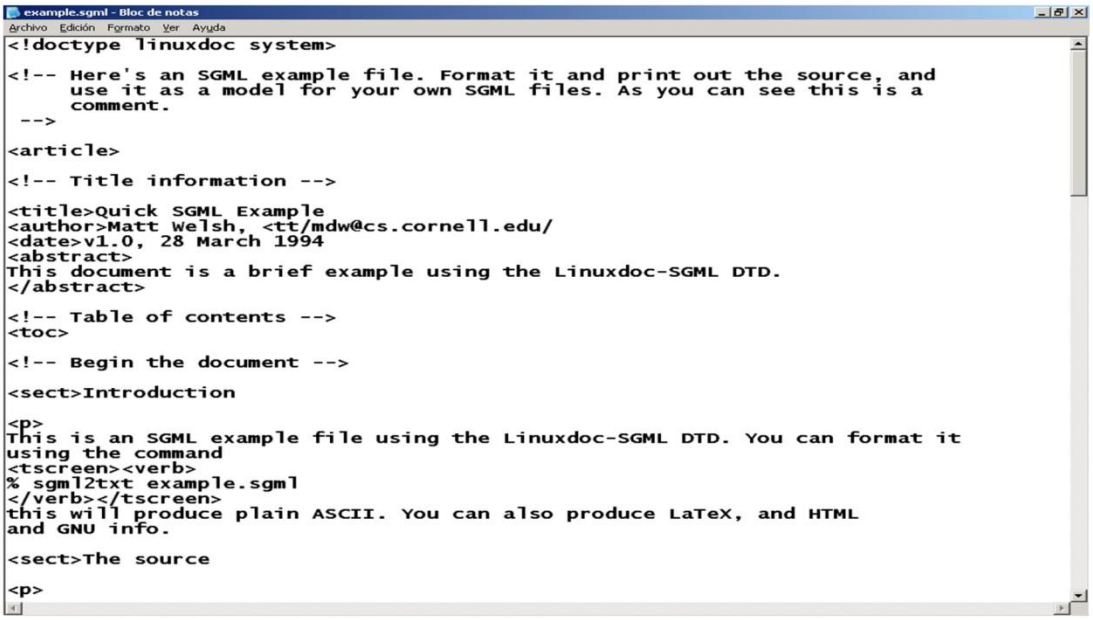
- Parrafo

  ```
  :p.GML supported hierarchical containers, such as
  ```

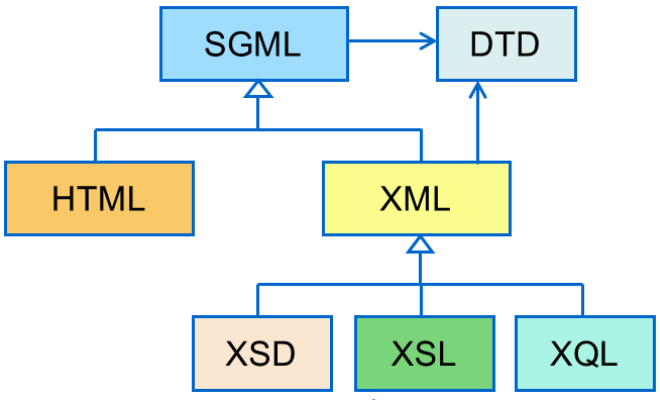
- Listas

  ```
  :ol
  :li.Ordered lists (like this one),
  :li.Unordered lists, and
  :li.Definition lists
  :eol.
  ```

1.3.2 SGML



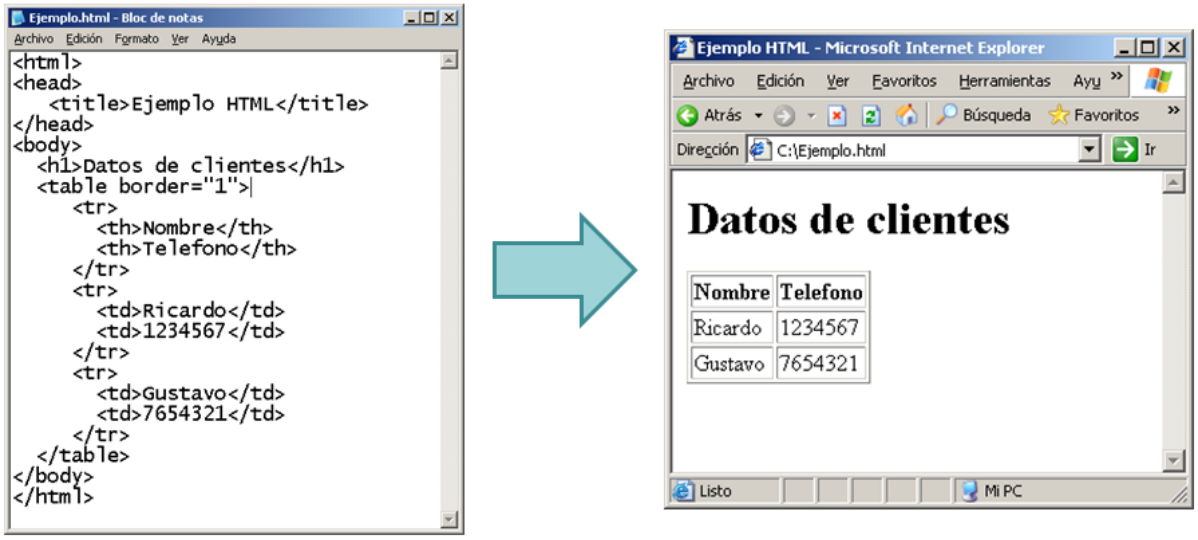
Es un metalenguaje normalizado por ISO en 1986, dada su complejidad se han desarrollado sub-lenguajes de etiquetados basados en SGML, tales como HTML y XML, se considera a SGML un lenguaje de etiquetado para definir otros lenguajes de etiquetado que cumplan propósitos específicos.



Características:

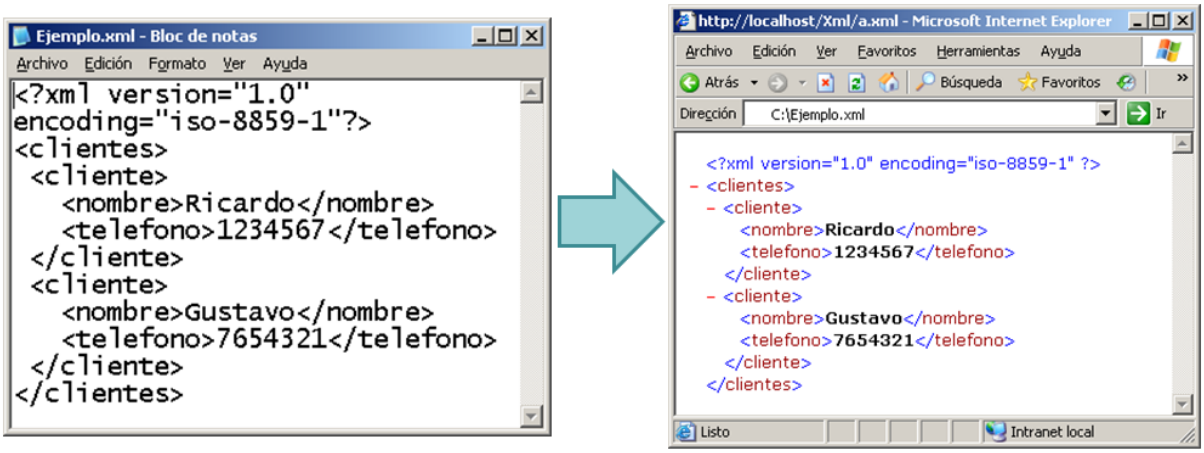
- Se utilizan etiquetas.
- Se utilizan los símbolos menor que (<) y mayor que (>) para delimitar las etiquetas.
- Se definen varias etiquetas que son heredadas por otros lenguajes.

1.3.3 HTML



Es un lenguaje de especificación de contenidos, usa un conjunto de etiquetas para representar la información en un navegador o browser.

1.3.4 XML



Es un lenguaje para definir estructuras de los datos y permite el intercambio de datos estandarizado, no es un lenguaje que reemplace a HTML, porque en XML el usuario define sus propias etiquetas.

1.4 Rol del W3C

W3C: Es una asociación internacional formada por organizaciones miembro del consorcio, personal y el público en general, que trabajan conjuntamente para desarrollar estándares Web, cuyo propósito es guiar la Web hacia su máximo potencial a través del desarrollo de protocolos y pautas que aseguren el crecimiento futuro de la Web.

El futuro expuesto por la W3C es: Web para todos, en cualquier cosa, en cualquier lugar, navegar a través de la vista, el oído, la voz y el tacto.

Tim Berners-Lee, quien inventó la World Wide Web en 1989 fundó junto con otros el W3C en 1994.

1.5 Ejemplo de Uso

1.5.1 Aplicación XML

- Se denomina **Aplicación XML** a cualquier lenguaje de marcado basado en XML.
- En este contexto Aplicación no se refiere a un programa que utilice XML.
- Aplicación significa el uso de XML para un dominio específico.

1.5.2 Ejemplos

- Chemical Markup Language

  Lenguaje basado en XML para expresar información molecular.

```
<?xml version="1.0" ?>
<cml xmlns="http://www.xml-cml.org/schema/cml2/core"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-
   instance"
        xsi:schemaLocation="http://www.xml-
   cml.org/schema/cml2/core/cmlCore.xsd">
<molecule title="Water">
   <atomArray>
        <atom id="a1" elementType="H" hydrogenCount="0" />
        <atom id="a2" elementType="O" hydrogenCount="2" />
        <atom id="a3" elementType="H" hydrogenCount="0" />
   </atomArray>
   <bondArray>
        <bond atomRefs2="a1 a2" order="1" />
        <bond atomRefs2="a2 a3" order="1" />
   </bondArray>
</molecule>
</cml>
```

- DocBook

Desarrollado por OASIS. Es un lenguaje de marcado utilizado en la documentación técnica. Especialmente utilizado para documentar programas informáticos.

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE book PUBLIC "-//OASIS//DTD DocBook XML V4.2//EN"
       "http://www.oasis-open.org/docbook/xml/4.2/docbookx.dtd">
<book lang="es" id="simple_libro">
      <title>Un libro muy simple</title>

       <chapter id="capitulo_1">
            <title>Capitulo 1</title>
            <para>Hola mundo!</para>
            <para>¡Yo espero que tu día sea bueno!</para>
       </chapter>

       <chapter id="capitulo_2">
            <title>Capitulo 2</title>
            <para>Hola otra vez, mundo!</para>
       </chapter>
</book>
```

- Dublin Core

Promueve el desarrollo de vocabularios de metadatos especializados en la búsqueda y localización de recursos, para permitir sistemas más eficientes que permitan buscar y ubicar recursos de una manera más eficiente.

Cuenta con 15 definiciones semánticas descriptivas que pretenden transmitir un significado semántico a las mismas.

Se emplea en muchos contestos, por ejemplo, páginas web, bibliotecas, bases de datos y otros.

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

<metadata
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:dc="http://purl.org/dc/elements/1.1/">

  <dc:title>JAVA WEB SERVICE</dc:title>
  <dc:creator>GUSTAVO CORONEL</dc:creator>
  <dc:subject>INFORMATICA</dc:subject>
  <dc:description>Creación de servicios web con Java.</dc:description>
  <dc:publisher>SISTEMAS UNI</dc:publisher>
  <dc:date>ENE-2016</dc:date>
  <dc:language>ESPAÑOL</dc:language>

</metadata>
```

- Keyhole Markup Language

  Es un lenguaje de marcado que se utiliza para mostrar información geográfica en aplicaciones como Google Earth y Google Maps.

  Fue desarrollado por Google.

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
<Document>
<Placemark>
    <name>New York City</name>
    <description>New York City</description>
    <Point>
        <coordinates>-74.006393,40.714172,0</coordinates>
    </Point>
</Placemark>
</Document>
</kml>
```

- Mathematical Markup Language

  Su objetivo es expresar formulas y ecuaciones matemáticas de forma que sea entendible tanto por los humanos como por las computadoras.

  Se puede usar de forma combinada con XHTML en páginas web.

  También se emplea para el intercambio de información entre programas de tipo matemático, como Maple, Matlab, etc.

  Es desarrollado por W3C.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE math PUBLIC "-//W3C//DTD MathML 2.0//EN"
          "http://www.w3.org/Math/DTD/mathml2/mathml2.dtd">
<math xmlns="http://www.w3.org/1998/Math/MathML">
  <mrow></mrow>
  <mrow>
    <mi>a</mi>
    <mo>&InvisibleTimes;</mo>
    <msup>
      <mi>x</mi>
      <mn>2</mn>
    </msup>
    <mo>+</mo>
    <mi>b</mi>
    <mo>&InvisibleTimes; </mo>
    <mi>x</mi>
    <mo>+</mo>
    <mi>c</mi>
  </mrow>
</math>
```

  La expresión que representa es:

$$ax^2 + bx + c$$

- Scalable Vector Graphics

  Desarrollado por la W3C, se utiliza para representar gráficos vectoriales.

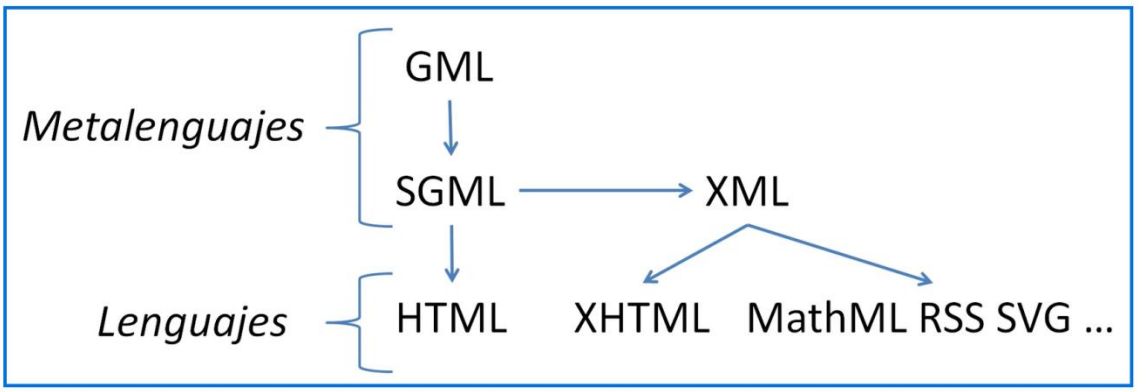
```
<?xml version="1.0"?>
<svg xmlns="http://www.w3.org/2000/svg"
     width="12cm" height="8cm">
  <title>The pink triangle!!!</title>
  <text x="10" y="15">This is SVG!</text>
  <polygon style="fill: pink"
  points="0,311 180,0 360,311" />
</svg>
```

1.6 Resumen

- Intercambio de información entre negocios EDI y XML, pero en la actualidad XML es un estándar y ofrece muchas características que no incluye EDI.
- HTML y XML nace de SGML y la W3C es la organización encargada de dar las últimas especificaciones para estas tecnologías.
- HTML para la presentación de los datos en la Web tiene etiquetas definidas y XML para estructurar, almacenar e intercambiar uno crea sus propias etiquetas.
- XSD, XSL, XQS, entre otros son tecnologías basados en XML y trabajan en conjunto para satisfacer otros requerimientos.

2 XML

2.1 ¿Qué es XML?



XML es un meta-lenguaje de marcas, es decir, permite que el usuario diseñe sus propias marcas (tags) y les dé un significado, con tal de que siga un modelo coherente.

XML define un conjunto de reglas para que cualquiera pueda definir su propio conjunto de etiquetas y atributos, y las relaciones que existen entre esas etiquetas.

El conjunto de reglas del lenguaje es la gramática del lenguaje.

GML, SGML y XML son metalenguajes, estos 3 permiten crear lenguajes.

HTML, XHTML, MathML, RSS, SVG, etc. son lenguajes.

Ejemplo 1

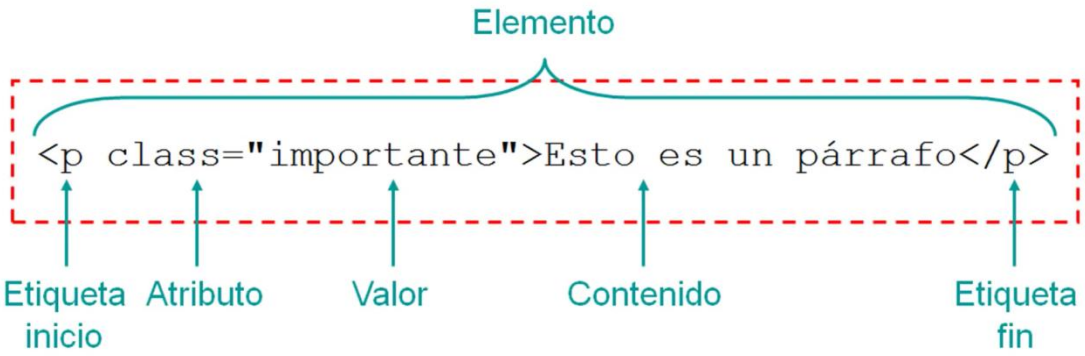
```
<!-- ejm01.xml -->
<Catalogo>
  <Articulo>
    <Nombre>Monitor</Nombre>
    <Precio>120.00</Precio>
  </Articulo>
  <Articulo>
    <Nombre>Teclado</Nombre>
    <Precio>24.00</Precio>
  </Articulo>
  <Articulo>
    <Nombre>Mouse</Nombre>
    <Precio>18.00</Precio>
  </Articulo>
</Catalogo>
```

A continuación se tiene el resultado de realizar un chequeo con NetBeans:

```
XML checking started.
Checking file:/H:/Tema04/EjemplosXML/src/xml/ejm01.xml...
XML checking finished.
```

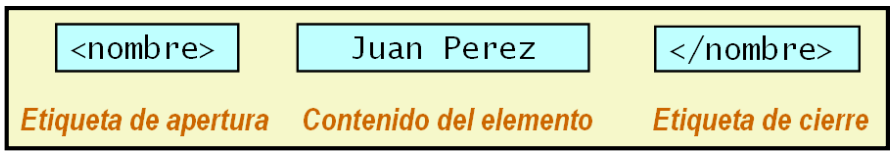
2.2 Sintaxis XML

- XML usa etiquetas para definir el contexto de los datos
- La unidad básica de información es el elemento



- Nombres de los elementos
  - Primer carácter:
    `[A-Z] | "_" | [a-z]`
  - Resto:
    `[A-Z] | "_" | [a-z] | "-" | "." | [0-9]`

- XML es sensible a las mayúsculas



- Los elementos pueden ser anidados

```
<empleado>
   <nombre>Juan Perez</nombre>
   <sueldo>3500</sueldo>
</empleado>
```

- Elementos vacíos
  - `<LIBRO></LIBRO>`
  - `<LIBRO/>`

2.3 Contenido básico de un documento XML

Instrucción de Procesamiento	<?xml version="1.0"?>
Comentario	<!-- Esto es un comentario -->
Elemento raíz	<empleado>
Elemento hijo	<nombre>Juan Perez</nombre>
Elemento vacío	<cargo />
Elemento con atributo	<sueldo moneda="Soles"> 3500 </sueldo>
Fin de elemento raíz	</empleado>

2.4 Prologo

Los documentos XML deben empezar con unas líneas que describen la versión de XML, el tipo de documento, y otras cosas.

La primera o "declaración XML", define la versión de XML usada. Además, en la "declaración XML" se especifica la codificación del documento, que puede ser, por ejemplo, US-ASCII (7 bits) o UTF-8 (código Unicode del que el ASCII es un subconjunto), ISO-8859-1 hasta ISO- 8859-7, etc. En general, y para uso con lenguajes europeos (incluyendo el juego de caracteres especiales del castellano, usamos UTF-7 o ISO-8859-1). También, se puede incluir una declaración de documento autónomo (standalone), que controla qué componentes de la DTD son necesarios para completar el procesamiento del documento.

La segunda o "declaración de tipo de documento", define qué tipo de documento estamos creando para ser procesado correctamente. Es decir, definimos que Declaración de Tipo de Documento (DTD o Document Type Definition) valida y define los datos que contiene nuestro documento XML.

En ella se define el tipo de documento, y dónde encontrar la información sobre su Definición de Tipo de Documento, mediante un identificador público (PUBLIC) que hace referencia a dicha DTD, o mediante un Identificador Universal de Recursos (URI) precedido por la palabra SYSTEM.

Ejemplo 2

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mensaje SYSTEM "mensaje.dtd">
```

Ejemplo 3

```
<?xml version="1.0" encoding="UTF-7"?>
<!DOCTYPE LABEL SYSTEM "http://www.empresa.com/dtds/label.dtd">
```

2.5 Documentos XML bien Formados

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- ejm06.xml -->
```

Prólogo

```
<direccion>
  <nombre>
    <titulo>Mrs.</titulo>
    <nombres>Mary</nombres>
    <apellidos>McGoon</apellidos>
  </nombre>
  <calle>1401 Main Street</calle>
  <ciudad>Anytown</ciudad>
  <estado>NC</estado>
  <codigo-postal>34829</codigo-postal>
</direccion>
```

Cuerpo del Documento

Al inicio de un documento XML se debe incluir la declaración XML, que tiene la siguiente sintaxis:

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
```

Por partes, lo que se especifica es lo siguiente:

- **version:** Indica la versión de XML que se está empleando.
- **encoding:** Especifica el juego de caracteres con el que se ha codificado el documento. Por defecto es UTF-8, así que, a menos que se especifique algún otro como ISO-8859-1, es opcional.
- **standalone:** Especifica si la validez del documento depende de otro documento externo —bien una DTD, bien un esquema—, en cuyo caso el valor es no, o si depende de una DTD incluida en el mismo documento, en cuyo caso se especifica yes. Este atributo no tiene valor por defecto, por lo que si no se especifica, el documento XML no puede validarse.

Si no se incluye la declaración de documento se utilizan los siguientes valores por defecto:

- **Versión:** 1.0
- **Encoding:** UTF-8
- **Entidades externas:** "yes"

Adicionalmente, debe cumplir con los siguientes requisitos:

- Un único elemento raíz
- Es una estructura jerárquica
- Las etiquetas de apertura y cierre deben coincidir
- Los caracteres deben ser consistentes (case sensitive)
- Elementos correctamente anidados
- Valores de atributos entre comillas
- No debe haber atributos repetidos en un elemento
- No se permite solapamiento de los elementos

2.6 Documentos XML Válidos

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- ejm06.xml -->
<!DOCTYPE direccion SYSTEM "ejm05.dtd">
```

Prólogo

```
<direccion>
  <nombre>
    <titulo>Mrs.</titulo>
    <nombres>Mary</nombres>
    <apellidos>McGoon</apellidos>
  </nombre>
  <calle>1401 Main Street</calle>
  <ciudad>Anytown</ciudad>
  <estado>NC</estado>
  <codigo-postal>34829</codigo-postal>
</direccion>
```

Cuerpo del Documento

Un documento XML válido en primer lugar debe estar bien formado, y luego debe cumplir con la definición concreta del lenguaje que se está empleado, por ejemplo, que los nombre de las etiquetas y atributos sean los correctos.

Para especificar un lenguaje se utilizan DTD, XML Schema, RelaxNG.

3 Document Type Definition - DTD

3.1 Introducción

XML es un metalenguaje, porque permite crear lenguajes.

En concreto, XML proporciona una serie de reglas para que cualquiera pueda definir su propio conjunto de etiquetas y atributos, y pueda también definir las relaciones entre ese conjunto de etiquetas.

El conjunto de reglas de un lenguaje es la gramática del lenguaje.

Existen diferentes maneras de definir la gramática de un lenguaje basado en XML:

- DTD
- XML Schema
- RelaxNG
- Schematron

3.2 Document Type Definition – DTD

3.2.1 Definición

Describe la estructura y la sintaxis de un documento XML, es decir de su gramática.

Un problema de los DTD es que no es XML, este problema se resuelve con XML Schema.

3.2.2 Función

La función del DTD es describir la estructura de un documento XML, con el fin de mantener una estructura común y la consistencia entre todos los documentos que utilicen el misma DTD.

El DTD permite de esta manera validar los documentos XML, puede comprobarse que son correctos, y pueden ser compartidos entre diferentes usuarios.

Esta es la dirección para consultar los DTDs de XHTML:

https://www.w3.org/TR/xhtml1/dtds.html

A continuación tenemos como ejemplo la definición de los atributos **img, map y area**:

```
<!ELEMENT img EMPTY>
<!ATTLIST img
  %attrs;
  src          %URI;          #REQUIRED
  alt          %Text;         #REQUIRED
  longdesc     %URI;          #IMPLIED
  height       %Length;       #IMPLIED
  width        %Length;       #IMPLIED
  usemap       %URI;          #IMPLIED
  ismap        (ismap)        #IMPLIED
  >

<!ELEMENT map ((%block; | form | %misc;)+ | area+)>
<!ATTLIST map
  %i18n;
  %events;
  id           ID             #REQUIRED
  class        CDATA          #IMPLIED
  style        %StyleSheet;   #IMPLIED
  title        %Text;         #IMPLIED
  name         NMTOKEN        #IMPLIED
  >

<!ELEMENT area EMPTY>
<!ATTLIST area
  %attrs;
  %focus;
  shape        %Shape;        "rect"
  coords       %Coords;       #IMPLIED
  href         %URI;          #IMPLIED
  nohref       (nohref)       #IMPLIED
  alt          %Text;         #REQUIRED
  >
```

3.2.3 Uso de un DTD

Un DTD define los elementos que son permitidos y el contenido de dichos elementos.

También define los atributos que se pueden utilizar en cada elemento, así como también cuales son los valores validos a los atributos.

Un DTD define la estructura valida de los elementos que pueden aparecer en un documento XML, el orden en el cual pueden aparecer, cómo pueden estar anidados y otros detalles de la estructura del documento XML.

Ejemplo 4

Analicemos el siguiente ejemplo que se encuentra en:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- personal.dtd -->
<!ELEMENT lista_de_personas (persona*)>
<!ELEMENT persona (nombre, fechanacimiento?, sexo?,
numeroseguridadsocial?)>
<!ELEMENT nombre (#PCDATA) >
<!ELEMENT fechanacimiento (#PCDATA) >
<!ELEMENT sexo (#PCDATA) >
<!ELEMENT numeroseguridadsocial (#PCDATA)>
```

Observándolo línea a línea este DTD se puede afirmar lo siguiente:

- Si sintaxis es diferente a la de un documento XML.
- Se puede afirmar que se está definiendo 6 elementos: lista_de_personas, persona, nombre, fechanacimiento, sexo, numeroseguridadsocial.
- **lista_de_personas** es un nombre de elemento válido. El "*" indica que puede haber 0 o más elementos de persona.
- **persona** es un nombre de elemento válido. Éste contiene obligatoriamente el elemento **nombre** mientras que el resto son opcionales. Y lo son porque lo indica el símbolo "?".
- **nombre** es un nombre de elemento válido. Contiene caracteres.
- **fechanacimiento** es un nombre de elemento válido. Contiene caracteres.
- **sexo** es un nombre de elemento válido. Contiene caracteres.
- **numeroseguridadsocial** es un nombre de elemento válido. Contiene caracteres.

A continuación tienes un ejemplo de documento XML que hace uso del DTD:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- personal.xml -->
<!DOCTYPE lista_de_personas SYSTEM "personal.dtd">
<lista_de_personas>
  <persona>
    <nombre>Gustavo Coronel</nombre>
    <sexo>Varón</sexo>
    <numeroseguridadsocial>34768923</numeroseguridadsocial>
  </persona>
</lista_de_personas>
```

3.3 Estructura de un DTD

3.3.1 Declaración

Un DTD está compuesto por declaraciones, su sintaxis es:

```
<!   . . .   >
```

3.3.2 Tipos de Declaraciones

- ELEMENT: Declaración de elemento.
- ATTLIST: Declaración de atributo.
- ENTITY: Declaración de entidad.
- NOTATION: Declaración de notación.

En este documento se estudiaran las dos más importantes: ELEMENT y ATTLIST.

3.3.3 Declaración de Elemento

La declaración de elemento define un elemento valido en el documento XML y su posible contenido.

Su sintaxis es:

```
<!ELEMENT nombre contendido >
```

El nombre del elemento:

- Debe comenzar con una letra, "_", ":"
- El resto de caracteres puede ser letra, "_", ":", dígitos, "." o "-"

El contenido puede ser:

- EMPTY: Indica que el elemento no incluye contenido.
- ANY: Que indica que cualquier contenido es posible, puede ser otros elementos o texto.
- #PCDATA: Que indica que solo puede contener texto.
- Modelo de grupo: Indica que el elemento incluye un elemento hijo.
- Modelo Mixto: Permite que se mezcle texto con otros elementos.

3.3.4 Modelo de Grupo

Representa una secuencia concreta de elementos que definen el contenido de un elemento.

Puede estar formado por:

- Lista:
  - ✓ Secuencia: Indica una secuencia de elementos que deben aparecer en el orden indicado.
  - ✓ Elección: Indica una lista de elementos excluyentes de los que se debe elegir solo uno.
- Cuantificador: Indica el números de ocurrencias sucesivas del elemento que puede aparecer.
  - ✓ ?: Puede aparecer 0 o 1 vez.
  - ✓ +: Debe aparecer por lo menos 1 vez.
  - ✓ *: Puede aparecer 0 o más veces.
  - ✓ Si un elemento no lleva cuantificador, significa que debe aparecer una vez en la posición indicada.

3.3.5 Modelo de Contendido Mixto

Permite que se mezcle texto con otros elementos.

En este modelo no se puede restringir ni el orden de aparición, ni el número de ocurrencias.

Su definición debe ser siempre de la siguiente forma:

- El primer elemento debe ser #PCDATA, que representa una cadena de caracteres.
- Los distintos elementos se deben separar con la barra vertical (|).
- Y finalmente, todo el grupo debe ser repetible cero o más veces (*).

Por ejemplo, la siguiente declaración:

```
<!ELEMENT p( #PCDATA | strong | em ) * >
```

Indica que el elemento "p", puede ser un párrafo de texto, puede contener texto y los elementos "strong" y "em".

Este sería un ejemplo de contenido XML que cumple con la declaración "p":

```
<p>
Este <strong>elemento</strong> tiene
<em>contenido mixto</em>
</p>
```

3.3.6 Declaración de Atributo

Define los posibles atributos que se pueden emplear en un elemento.

Su sintaxis es:

```
<!ATTLIST element
   atributo1 contenido1 defecto1
   atributo2 contenido2 defecto2 >
```

Para cada atributo se define su nombre, su tipo de dato o una enumeración de sus posibles valores, y su valor por defecto.

El contenido o tipo de dato de un atributo puede tomar estos posibles valores:

- CDATA: Cualquier cadena de texto.
- NMTOKEN: Parecido a CDATA, pero solo acepta los caracteres válidos para nombrar cosas: letras, números, punto, guion, subrayado y los dos puntos (:). No acepta el espacio en blanco
- NMTOKENS: Lista de NMTOKEN separados por espacios en blanco.
- ENTITY: Nombre de una entidad externa.
- ENTITIES: Lista de nombres de entidades externas separadas por espacios en blanco.
- ID: Indica un valor único en el documento. Identifica al elemento al que esta asociado.
- IDREF: Referencia a un ID valido en el mismo documento.
- IDREFS: Lista de referencias a ID separadas por espacio en blanco.
- NOTATION: Su valor se ajusta a una notación declarada previamente. Veamos el siguiente ejemplo:

```
<!NOTATION ISODATE SYSTEM
   "http://www.iso.ch/cate/d15903.html">
<!NOTATION USDATE SYSTEM
   "http://tf.nist.gov/timefreq/general/enc-d.htm#date">

<!ATTLIST FECHA
   FORMATO NOTATION (ISODATE | USDATE) #IMPLIED>
```

  En este ejemplo se está declarando una notación estandarizada de tipo fecha que se emplea para el atributo formato.

- Grupo de valores enumerados: Una lista de posibles valores separados por la barra vertical, veamos el siguiente ejemplo:

```
<!ATTLIST img
  print (yes | no) "yes">
```

  El atributo "print" del elemento "img" puede tomar los valores "yes" o "no", siendo el valor por defecto "yes".

El valor por defecto de un atributo puede tomar los siguientes valores:

- #REQUIRED: El atributo es obligatorio.
- #IMPLIED: El atributo es opcional.
- #FIXED "valor": El atributo tiene un valor fijo, y se indica ese valor.
- "valor": Un valor encerrado entre comillas.

A continuación se tiene un ejemplo de declaración de atributos:

```
<!ATTLIST img
   src CDATA #IMPLIED
   align (left | center | right) "left"
   size NMTOKEN #REQUIRED>
```

- El atributo **src** se definido como #CDATA, es decir una cadena de caracteres y es opcional.
- El atributo **align** es un valor enumerado, que puede tomar el valor **left**, **center** o **right**. Su valor por defecto es **left**.
- El atributo **size** es de tipo **NMTOKEN** y es obligatorio.

En este otro ejemplo, se definen los atributos para el elemento **alumno**:

```
<!ATTLIST alumno
   nombre CDATA #REQUIRED
   matricula ID #REQUIRED
   curso CDATA #FIXED "Primero">
```

3.4 Uso de un DTD

La declaración del DTD debe estar al inicio, en el prólogo del documento, y su definición puede estar en propio documento XML o se puede hacer referencia a un archivo externo.

3.4.1 DTD Interno

En este caso la definición del DTD se realiza en el mismo documento XM, tal como se ilustra en el Ejemplo 5.

Ejemplo 5

A continuación se tiene un ejemplo de un documento XML un DTD interno.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- ejm05.xml -->
<!DOCTYPE Mensaje
  [
    <!ELEMENT Mensaje (Frase,Arenga)>
    <!ELEMENT Frase (#PCDATA)>
    <!ELEMENT Arenga (#PCDATA)>
  ]
>
<!-- este es un comentario -->
<Mensaje>
  <Arenga>PERU es el campeón</Arenga>
  <Frase>Somos PERÚ</Frase>
</Mensaje>
```

A continuación tenemos el resultado de la validación en NetBeans:

```
XML validation started.

Checking file:/Tema04/EjemplosXML/src/xml/ejm05.xml...
El contenido del tipo de elemento "Mensaje" debe coincidir con
"(Frase,Arenga)". [14]

XML validation finished.
```

Se tiene un error porque se ha invertido el orden de los elementos Frase y Arenga.

3.4.2 DTD Externo

En este caso, el DTD está en un archivo aparte, que es lo más lógico, porque es normal que se utilice el mismo DTD para crear varios documentos XML.

A continuación se tiene la sintaxis para usar DTD externos:

- PUBLIC: global, un estándar

```
<!DOCTYPE elementoRaiz PUBLIC "nombreDTD“>
<!DOCTYPE elementoRaiz PUBLIC "nombreDTD" "urlDTD">
```

- SYSTEM: local, definido por el usuario (privado)

```
<!DOCTYPE elementoRaiz SYSTEM "urlDTD">
```

A continuación se tiene el ejemplo de uso de un DTD público que se utiliza en página web:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

El siguiente ejemplo ilustra el uso de un DTD propio o local, donde el elemento raíz en NOTICIA en mayúsculas:

```
<!DOCTYPE NOTICIA SYSTEM "noticia.dtd">
```

También es posible combinar DTD interno con DTD externo.

En el Ejemplo 6 se ilustra el uso de un DTD externo propio.

Ejemplo 6

Archivo: ejm06.dtd

```
<!-- ejm06.dtd -->
<!ELEMENT direccion (nombre, calle, ciudad, estado, codigo-postal)>
<!ELEMENT nombre (titulo, nombres, apellidos)>
<!ELEMENT titulo (#PCDATA)>
<!ELEMENT nombres (#PCDATA)>
<!ELEMENT apellidos (#PCDATA)>
<!ELEMENT calle (#PCDATA)>
<!ELEMENT ciudad (#PCDATA)>
<!ELEMENT estado (#PCDATA)>
<!ELEMENT codigo-postal (#PCDATA)>
```

A continuación se tiene la validación del DTD con NetBeans:

```
DTD checking started.
Checking file:/H:/Tema04/EjemplosXML/src/xml/ejm06.dtd...
DTD checking finished.
```

A continuación se tiene el documento XML.

Archivo: ejm06.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- ejm06.xml -->
<!DOCTYPE direccion SYSTEM "ejm06.dtd">
<direccion>
  <nombre>
    <titulo>Mrs.</titulo>
    <nombres>Mary</nombres>
    <apellidos>McGoon</apellidos>
  </nombre>
  <calle>1401 Main Street</calle>
  <ciudad>Anytown</ciudad>
  <estado>NC</estado>
  <codigo-postal>34829</codigo-postal>
</direccion>
```

A continuación se tiene el resultado de la validación del documento XML utilizando NetBeans:

```
XML validation started.
Checking file:/H:/ Tema04/EjemplosXML/src/xml/ejm06.xml...
Referenced entity at "file:/H:/ Tema04/EjemplosXML/src/xml/ejm06.dtd".
XML validation finished.
```

Ejemplo 7

En este ejemplo queremos representar un documento que está conformado por título, autor y varias secciones, a su vez, cada sección está conformada por un tema y uno o más párrafos.

El DTD que se ha diseñado es el siguiente:

```
<!-- documento.dtd -->
<!ELEMENT documento (titulo, autor, seccion+)>
<!ELEMENT titulo (#PCDATA)>
<!ELEMENT autor (#PCDATA)>
<!ELEMENT seccion (tema,parrafo+)>
<!ELEMENT tema (#PCDATA)>
<!ELEMENT parrafo (#PCDATA)>
```

Y el documento de prueba es el siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- documento.xml -->
<!DOCTYPE documento SYSTEM "documento.dtd">
<documento>
  <titulo>Fundamentos de XML</titulo>
  <autor>Gustavo Coronel</autor>
  <seccion>
    <tema>Introducción</tema>
    <parrafo>Parrafo 1</parrafo>
    <parrafo>Parrafo 2</parrafo>
  </seccion>
  <seccion>
    <tema>XML Validos</tema>
    <parrafo>Parrafo 1</parrafo>
    <parrafo>Parrafo 2</parrafo>
  </seccion>
</documento>
```

Ejercicio 1

Definir el DTD para el siguiente documento XML:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<curso>
  <titulo>Spring Framework</titulo>
  <unidad titulo="Spring Core">
    <lecciones>
      <leccion titulo="Introducción">En esta lección ...</leccion>
      <leccion titulo="Uso de XML">En esta lección ...</leccion>
      <leccion titulo="Uso de Anotaciones">En esta lección ...</leccion>
    </lecciones>
  </unidad>
  <unidad titulo="Spring MVC">
    <lecciones>
      <leccion titulo="Configuración">En esta lección ...</leccion>
      <leccion titulo="El Controlador">En esta lección ...</leccion>
      <leccion titulo="AJAX y JSON">En esta lección ...</leccion>
    </lecciones>
  </unidad>
</curso>
```

4 XML y CSS

4.1 Referencia de una Hoja de Estilo

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/css" href="estilo.css" ?>

<!-- El documento XML -->
```

4.2 Ejemplo

Archivo XML: curso.xml

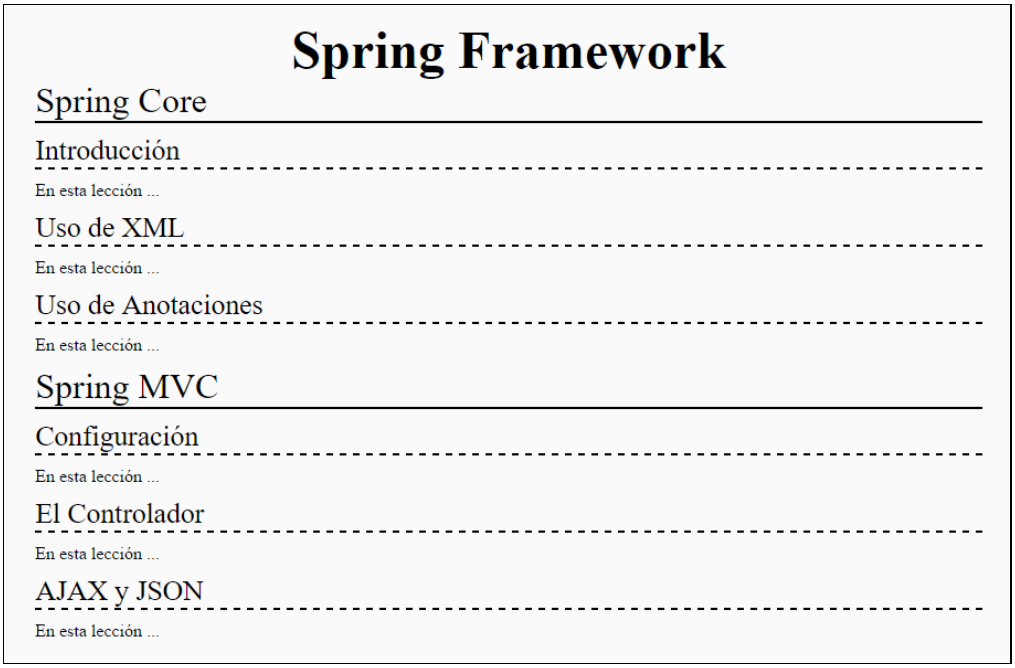
```
<?xml version="1.0" encoding="UTF-8"?>
<curso>
  <titulo>Spring Framework</titulo>
  <unidad titulo="Spring Core">
    <lecciones>
      <leccion titulo="Introducción">En esta lección ...</leccion>
      <leccion titulo="Uso de XML">En esta lección ...</leccion>
      <leccion titulo="Uso de Anotaciones">En esta lección ...</leccion>
    </lecciones>
  </unidad>
  <unidad titulo="Spring MVC">
    <lecciones>
      <leccion titulo="Configuración">En esta lección ...</leccion>
      <leccion titulo="El Controlador">En esta lección ...</leccion>
      <leccion titulo="AJAX y JSON">En esta lección ...</leccion>
    </lecciones>
  </unidad>
</curso>
```

Hoja de Estilo: curso.css

```
curso, titulo, unidad, unidad:before,
lecciones, leccion, leccion:before{
  display: block;
}

curso{
  width: 80%;
  margin: 20px auto;
```

```
  background-color: #FAFAFA;
}

titulo{
  text-align: center;
  font-weight: bold;
  font-size: 3em;
}

unidad:before{
  content: attr(titulo);
  margin-bottom: 10px;
  border-bottom: 2px solid #000;
  font-size: 2em;
}

leccion:before{
  content: attr(titulo);
  margin-bottom: 10px;
  border-bottom: 2px dashed #000;
  font-size: 1.6em;
}

leccion{
  margin-bottom: 10px;
}
```

Resultado:



5 Uso de NameSpaces

5.1 Introducción

XML Namespaces, los espacios de nombre de XML, permite crear distintos espacios de nombres en un documento XML y que, por tanto, los elementos y atributos definidos en distintos módulos no entre en conflicto entre sí.

XML se ha desarrollado con el objetivo de que un único documento XML pueda contener elementos y atributos (vocabulario de etiquetado, markup vocabulary) que han sido definidos por y son usados por múltiples módulos distintos. La motivación de este objetivo es la modularidad: si ya existe un vocabulario de etiquetado ampliamente aceptado para un propósito concreto, es mucho mejor reutilizarlo que tener que reinventarlo.

Sin embargo, en un documento XML donde se emplean múltiples vocabularios de etiquetado se pueden dar problemas de colisión de nombres y, por tanto, de interpretación: dos módulos distintos que emplean los mismos nombres para elementos o atributos.

La solución a este problema sería que los vocabularios de etiquetado tuviesen nombres universales, cuyo ámbito se extendiese más allá del documento donde están definidos. Este mecanismo es el que proporciona XML Namespaces.

El concepto de Espacios de nombres o Namespaces surge de la posibilidad de combinar diferentes vocabularios XML evitando colisiones entre los nombres de los vocabularios.

> ***Los XML NAMESPACES es un método para obtener nombres únicos en los vocabularios de etiquetado.***

Ejemplo 8

En este ejemplo ilustraremos el error de declarar la misma etiqueta en contextos diferentes.

Supongamos que queremos representar en un documento XML valido los datos de una venta, el cual debe contener los datos del cliente y los artículos, para lo cual se ha diseñado el siguiente DTD:

```
<!-- venta.dtd -->
<!ELEMENT venta (cliente, articulos)>
<!ELEMENT cliente (nombre, direccion)>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT direccion (#PCDATA)>
<!ELEMENT articulos (articulo+)>
<!ELEMENT articulo (nombre, precio, cantidad)>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT precio (#PCDATA)>
<!ELEMENT cantidad (#PCDATA)>
```

El ejemplo de cómo se utilizaría el DTD lo tenemos a continuación:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- venta.xml -->
<!DOCTYPE venta SYSTEM "venta.dtd">
<venta>
   <cliente>
      <nombre>Gustavo Coronel</nombre>
      <direccion>Av. Arenales 2670</direccion>
   </cliente>
   <articulos>
      <articulo>
         <nombre>Laptop 3i</nombre>
         <precio>2890.00</precio>
         <cantidad>3</cantidad>
      </articulo>
      <articulo>
         <nombre>Impresora</nombre>
         <precio>578.00</precio>
         <cantidad>2</cantidad>
      </articulo>
   </articulos>
</venta>
```

Cuando validamos este documento con NetBeans tenemos el siguiente resultado:

```
XML validation started.
Checking file:/H:/Tema04/EjemplosXML/src/xml/venta.xml...
Referenced entity at "file:/H:/Tema04/EjemplosXML/src/xml/venta.dtd".
El tipo de elemento "nombre" no debe declararse más de una vez. [8]
XML validation finished.
```

El resultado de la validación nos indica que hay un error con el elemento **nombre**, debido a que se está declarando más de una vez en el DTD, esto se conoce como colisión de nombres.

5.2 Caso 1

En este ejemplo todos los elementos se han escrito con su correspondiente espacio de nombre.

```
<h:html xmlns:xdc="http://www.xml.com/books"
        xmlns:h="http://www.w3.org/HTML/1998/html4">
 <h:head><h:title>Revisiones de libros</h:title></h:head>
 <h:body>
  <xdc:bookreview>
   <xdc:title>XML en 10 minutos</xdc:title>
   <h:table>
    <h:tr align="center">
     <h:td>Autor</h:td><h:td>Precio</h:td>
     <h:td>Páginas</h:td><h:td>Fecha</h:td></h:tr>
    <h:tr align="left">
     <h:td><xdc:author>Sergio Luján Mora</xdc:author></h:td>
     <h:td><xdc:price>31.98</xdc:price></h:td>
     <h:td><xdc:pages>352</xdc:pages></h:td>
     <h:td><xdc:date>2003</xdc:date></h:td>
    </h:tr>
   </h:table>
  </xdc:bookreview>
 </h:body>
</h:html>
```

5.3 Caso 2

En este caso se está utilizando un espacio de nombre por defecto.

```
<html xmlns="http://www.w3.org/HTML/1998/html4"
      xmlns:xdc="http://www.xml.com/books">
 <head><title>Revisiones de libros</title></head>
 <body>
  <xdc:bookreview>
   <xdc:title>XML en 10 minutos</xdc:title>
   <table>
    <tr align="center">
     <td>Autor</td><td>Precio</td>
     <td>Páginas</td><td>Fecha</td></tr>
    <tr align="left">
     <td><xdc:author>Sergio Luján Mora</xdc:author></td>
     <td><xdc:price>31.98</xdc:price></td>
     <td><xdc:pages>352</xdc:pages></td>
     <td><xdc:date>2003</xdc:date></td>
    </tr>
   </table>
  </xdc:bookreview>
 </body>
</html>
```