Introducción a XML

Roberto Gómez Cárdenas
rogomez@itesm.mx
http://homepage.cem.itesm.mx/rogomez

Mitos sobre XML

- XML: eXtensible Markup Language
  - lenguaje de marcas extensible
- Es un lenguaje etiquetado
  - ¿es como HTML, pero con nuevas etiquetas?
- Es un protocolo
  - Dicen que se facilita la transmisión de datos
- ¿Es un lenguaje de programación?

Desmintiendo los mitos

- No es un lenguaje etiquetado.
  - Es un metalenguaje.
  - No es un lenguaje como HTML con una seria de etiquetas predefinidas (<body>, <table>, etc).
- No es un protocolo.
- No es un lenguaje de programación.
  - No puede ejecutarse.
  - No es posible obtener código ejecutable a partir del XML.

Lámina 3 Roberto Gómez C.

¿Entonces que és?

- XML es un metalenguaje etiquetado.
  - Un lenguaje para crear nuevos lenguajes.
  - Los lenguajes creados a partir de XML son lenguajes etiquetados.
    - Por ejemplo HTML.
    - WSBL.
- Con XML es posible crear “fácilmente” nuevos lenguajes etiquetados a la medida.
- A estos nuevos lenguajes se les suele llamar:
  - Dialecto XML.
  - Aplicación XML.

Lámina 4 Roberto Gómez C.

Algunas definiciones antes de empezar

- Scanner .
  - El analizador léxico o scanner, transforma el texto fuente en una secuencia a ordenada de elemento léxicamente válidos (tokens).
- Parseador (Parser).
  - Un parseador (parser) es un programa de computación que lleva a cabo un parseo.
- Parseo (Parcing).
  - Proceso de analizar una secuencia de símbolos a fin de determinar su estructura gramatical con respecto a una gramática formal dada.
  - Formalmente es llamado análisis de sintaxis.

Lámina 5 Roberto Gómez C.

Origen XML

- Versión actual 1.0
- Recomendación del W3C desde febrero de 1998.
- Basado en el anterior estándar SGML (Standard Generalized Markup Language, ISO 8879), que data de 1986.
  - A su vez basado en el GML creado por IBM.
- SGML proporciona un modo consistente y preciso de aplicar etiquetas para describir las partes que componen un documento.

Lámina 6 Roberto Gómez C.

XML vs SGML

- El problema que se atribuye a SGML es su excesiva dificultad; la recomendación ocupa unas 400 páginas.
- Usando SGML, por otro lado, se definió precisamente el HTML.
- En una primera aproximación se puede decir que mediante XML también se podría definir el HTML.

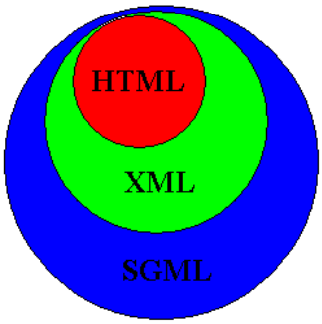


Lámina 7 Roberto Gómez C.

Ventajas XML

- Es un lenguaje prácticamente estándar.
  - Es fácil acceder a los datos contenidos en documentos XML (independientemente del dialecto creado) ya que existen mucho software para leer (parsear) documentos XML.
- Utilizar un formato ad hoc (o no estándar), implica que solamente los desarrolladores de este lo puedan usar.
- El utilizar XML permite intercambiar documentos de una forma muy fácil.

Lámina 8 Roberto Gómez C.

A notar

- XML es un lenguaje orientado al contenido.
- A diferencia de una etiqueta HTML (<h1> por ejemplo) una etiqueta XML no se visualiza de ninguna forma en especial,
- Es necesario aplicar estilos o transformaciones

Lámina 9 Roberto Gómez C.

Tipos documentos XML

- Dos tipos de documentos: válidos y bien formados.
- Documentos bien formados
  - son todos los que cumplen las especificaciones del lenguaje respecto a las reglas sintácticas.
  - sin estar sujetos a unos elementos fijados en un DTD
- Documentos válidos
  - Además de estar bien formados, siguen una estructura y una semántica determinada por un DTD
  - sus elementos y sobre todo la estructura jerárquica que define el DTD, además de los atributos, deben ajustarse a lo que el DTD dicte.

Lámina 10 Roberto Gómez C.

¿Y qué es un DTD?

- Document Type Definition,
  - Definición de Tipo de Documento
- Es una definición de los elementos que puede haber en el documento XML, y su relación entre ellos, sus atributos, posibles valores, etc.
- Es una especie de definición de la gramática del documento.
- Dos tipos de parsers:
  - Parsers no validadores: procesan documentos XML sin comprobar que siguen las reglas establecidas por un DTD (sólo comprueban que está bien formado).
  - Parsers validadores: comprueba que además de bien formado se atiene a su DTD y es válido.

Lámina 11 Roberto Gómez C.

Componentes de un documento XML bien formado

- Prólogo y declaración de tipo de documento
- Etiquetas

  <hola>
- Atributos

  <item id="33905">
- Entidades predefinidas

  < (<)
- Comentarios

  <!-- Comentario
- Componentes avanzados
  - Secciones CDATA
  - Instrucciones de procesamiento

Lámina 12 Roberto Gómez C.

Prólogo y declaración de tipo de documento

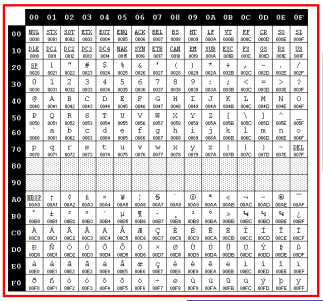
- No es obligatorio
- Línea que describe la versión de XML, el tipo de documento y otras cosas.
- La primera línea del prólogo o "declaración" permite especificar la versión de XML usada, hasta el momento, sólo existe la "1.0" y la codificación de carácter (US-ASCII, UTF-8, BIG5, ISO-8850-7,etc.).
- En general, para el castellano, usamos UTF-7 (código Unicode del que ASCII es un subconjunto) o ISO-8859-1.
- Ejemplo

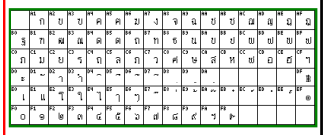
```
<?xml version="1.0" encoding="UTF-7" ?>
```

Lámina 13 Roberto Gómez C.

The ISO 8859 Alphabet Soup

- Tipos ISO
  - ISO-8859-1 (Latin 1)
  - ISO-8859-2 (Latin 2)
  - ISO-8859-3 (Latin 3)
  - ISO-8859-4 (Baltic)
  - ISO-8859-5 (Cyrillic)
  - ISO-8859-6 (Arabic)
  - ISO-8859-7 (Greek)
  - ISO-8859-8 (Hebrew)
  - ISO-8859-9 (Turkish)
  - ISO-8859-11 (Thai)
  - ISO-8859-10 (Latin6)
  - ISO-8859-15 (Latin 9)
- Referencia
  - http://czyborra.com/charsets/iso8859.html#ISO-8859-2





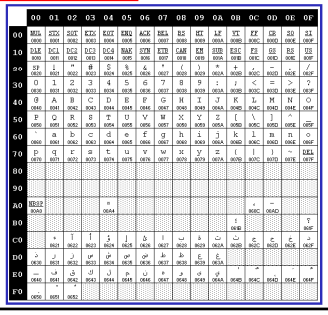


Lámina 14 Roberto Gómez C.

The UTF Alphabet Soup

- Unicode define dos métodos de "mapeo" o de localización de caracteres
  - La codificación **UTF** (Unicode Transformation Format) Formato de Transformación Unicode.
    - Número en nombre indica cantidad de bits de cada carácter
  - La codificación **UCS** (Universal Character Set) Juego de Caracteres Universal.
    - Número en nombre indica número de bytes por carácter
- Ejemplos
  - UTF-7
  - UTF-8
  - UTF-16/USC-2
  - UTF-32/USC-4

Para saber más:
http://es.debugmodeon.com/articulo/todo-lo-que-deberias-saber-de-unicode

Lámina 15 Roberto Gómez C.

Declaración de tipo

- La segunda línea, o "declaración de tipo de documento XML", define que tipo de documento estamos creando.
  - Definimos que Declaración de Tipo de Documento (DTD - Document Type Definition) cumple y define los datos que de contiene el documento XML.
- Por ejemplo:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE ejemplo SYSTEM "http://www.ejemplos.xml/ejemplo.dtd">
<nada> Ejemplo </nada>
```

Lámina 16 Roberto Gómez C.

Las reglas básicas

- XML es sensible a mayúsculas.
- Todas las etiquetas de inicio deben tener etiquetas de fin.
- Los elementos deben estar propiamente anidados.
- La declaración XML es el primer elemento.
- Cada documento debe contener al menos un elemento raíz.
- Algunos caracteres están reservados para parseo.

Lámina 17 Roberto Gómez C.

Etiquetas XML

- En XML una etiqueta empieza por el símbolo menor que < y termina con el símbolo mayor que >
- Después del símbolo < se encuentra el nombre de la etiqueta

<listas> <lista> <subs>

- Existen dos tipos de etiquetas
  - Non-Empty o contenedora
  - Empty o singular

Lámina 18 Roberto Gómez C.

Etiqueta non-empty

- También conocida como contenedora
- Contiene algo
- Tiene que haber una etiqueta de comienzo y una etiqueta de fin.
- En la etiqueta de fin el símbolo < siempre va seguido del símbolo /

```
<lista>
…
…
</lista>
```

Lámina 19 Roberto Gómez C.

Etiqueta empty

- También conocida como singular.
- No contiene nada.
- El símbolo > debe ir siempre precedido del símbolo /

```
<subs mail = “toto@live.com” />
```

- También se admite la forma:

```
<subs mail = “toto@live.com”> </subs>
```

Lámina 20 Roberto Gómez C.

Atributos

- Las etiquetas puede tener atributos.
- Estos atributos indican opciones de la etiqueta.
- Los atributos tienen la siguiente sintaxis

  nombre_atributo= valor
- Los atributos tienen que estar delimitado con comillas dobles (") o comilla simple (').
  - Cuando se usa uno para delimitar el valor del atributo, el otro se puede usar dentro.
- Ejemplo de atributo

  <subs mail = “toto@live.com” />

Lámina 21 Roberto Gómez C.

Mas de un atributo en una etiqueta

- Si hay más de un atributo, tienen que ir separados por espacios.
- Si la etiqueta es contenedora, los atributos solo se escriben en la etiqueta de comienzo.

Lámina 22 Roberto Gómez C.

Reglas etiquetas

- Toda etiqueta de comienzo debe tener etiqueta de fin.
- Las etiquetas no deben solaparse.
- Debe haber una única etiqueta en la raíz del documento (<listas>.
- Los valores de los atributos deben ir entre comillas (dobles o simples)
- Una etiqueta no puede tener dos atributos con el mismo nombre.

Lámina 23 Roberto Gómez C.

Comentarios

- Los comentarios pueden aparecer en cualquier punto del documento, fuera del resto de las marcas, es decir, fuera de las declaraciones etiquetas u otros comentaros.
- Tienen el mismo formato que los comentarios de HTML, por lo que comienzan con "<!--" y terminan con "-->".
- La cadena "--" no puede aparece dentro de un comentario.
- Ejemplo

```
...
<!-- Que gran gato es Micifú -->
<gato raza="Persa" nombre="Micifú"/>
....
```

Lámina 24 Roberto Gómez C.

Entidades predefinidas

- En XML 1.0 se definen cinco entidades para representar caracteres especiales y que no se interpreten como marcas por el parser XML.

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

Lámina 25 Roberto Gómez C.

Secciones CDATA

- Permiten especificar datos, utilizando cualquier carácter, especial o no, sin que se interprete como una marca XML.
- De esta forma se puede leer más fácilmente el documento XML sin tener que descifrar los códigos de las entidades.
- Las secciones CDATA empiezan por la cadena "<![CDATA[" y terminan con la cadena "]]>" y sólo ésta última se reconoce como marca.
- No se pueden anidar secciones CDATA.

Lámina 26 Roberto Gómez C.

Ejemplo sección CDATA

```
<ejemplo>
    <![CDATA[
        <HTML>
          <HEAD>
            <TITLE>Rock & Roll</TITLE>
          </HEAD>
    ]]>
</ejemplo
```

Lámina 27 Roberto Gómez C.

Instrucciones de procesamiento

- Las instrucciones de procesamiento permiten a los documentos XML contener instrucciones para las aplicaciones, van entre **<?** y **?>** **.**
- No son parte del documento, pero deben ser pasadas a la aplicación.
- En el ejemplo, se le dice al cocoon (herramienta para publicar xml en la web) que la página XML es un XSLT.

```
<?cocoon-process type="xslt"?>
....
```

Lámina 28 Roberto Gómez C.

Ejemplo de un documento bien formado

```
<?xml version="1.0"?>
<!DOCTYPE MENSAJE SYSTEM "mensaje.dtd">
<mensaje>
<remite>
<nombre>Alfredo Reino</nombre>
<email>alf@ibium.com</email>
</remite>
<destinatario>
<nombre>Toto Cachafas</nombre>
<email>cuate@amigos.net</email>
</destinatario>
<asunto>Hola Toto</asunto>
<texto>
<parrafo>Hola ¿qué tal? ¿Vamos al cine? </parrafo>
</texto>
</mensaje>
```

Lámina 29 Roberto Gómez C.

Vista gráfica del documento

- El documento puede ser visto de forma gráfica, para comprender mejor la estructura de un documento XML.

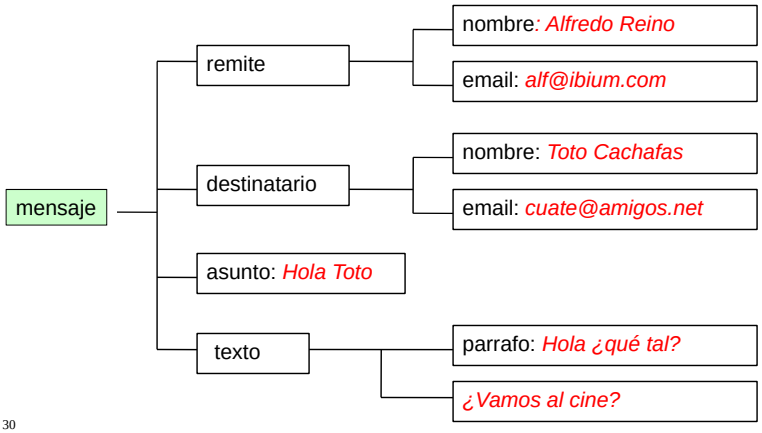


Lámina 30 Roberto Gómez C.

Documentos XML verificados

- Cualquier archivo que esta bien formado es parseable y puede considerarse un documento XML.
  - Podemos acceder a el desde un modelo de objetos.
  - Podemos aplicarle estilos.
  - Podemos aplicarle transformaciones.
- Sin embargo, a veces nos puede interesar definir formalmente el “dialecto XML”, para comprobar que el documento no sólo está bien formado, sino que además cumple las reglas de ese dialecto.

Lámina 31 Roberto Gómez C.

Parseando un documento XML

- Cuando se procesa cualquier información formateada mediante XML, lo primero es comprobar si está bien formada, y luego, si incluye o referencia a un DTD, comprobar que sigue sus reglas gramaticales.

Lámina 32 Roberto Gómez C.

Validando

```
<listas>
        <lista nombre="ESIDE-TIM" desc="Lista de TiM">
                <subs mail="1jolopez@rigel.deusto.es" />
                <subs mail="3jugonza@rigel.deusto.es" />
        </lista>
        <lista nombre="AMIGOS" desc="Lista de amigos">
                <subs mail="colegui@mixmail.com" />
                <subs mail="compi@hotmail.com" />
                <subs mail="amiguete@wanadoo.es" />
        </lista>
</listas>
```

DTD

Normas del Dialecto ListaML

- Hay tres tipos de etiquetas
  - `listas` (contenedora) RAIZ
  - `lista` (contenedora)
    - Atributo `nombre` (texto)
    - Atributo `desc` (texto)
  - `subs` (singular)
    - Atributo `mail` (texto)
- `Listas` puede contener 0 o más etiquetas `lista`
- `Lista` puede contener 0 o más etiquetas `subs`

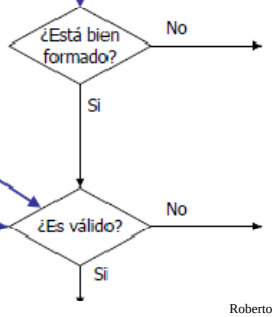


Lámina 33 Roberto Gómez C.

Validando

- ¿El documento anterior es válido?
  - SI -> cumple todas las normas del dialecto
- ¿Y el siguiente?

```
<listas>
    <subs mail="toto@acme.com" />
<listas>
```

- NO
  - La etiqueta listas no puede contener subs.
  - Solo puede contener list

Lámina 34 Roberto Gómez C.

El DTD

- Recopilación de normas que definen cómo debe ser el documento.
- Se compone con una sintaxis concreta.
- El DTD del ejemplo anterior sería:

```
<!ELEMENT lista (subs*)>
<!ATTLIST lista
        nombre CDATA #REQUIRED
        desc CDATA #REQUIRED
>
<!ELEMENT listas (lista *)>
<!ELEMENT subs  EMPTY>
<!ATTLIST subs
        mail CDATA #REQUIRED
>
```

Normas del dialecto ListaML

- Hay tres tipos de etiquetas
  - listas (contenedora) RAIZ
  - lista (contenedora)
    - Atributo nombre (texto)
    - Atributo desc (texto)
  - subs (singular)
    - Atributo mail (texto)
  - Listas puede contener 0 o más etiquetas lista
  - Lista puede contener 0 o más etiquetas subs

Lámina 35 Roberto Gómez C.

Elementos DTD

- Las declaraciones DTD son las líneas que empiezan con "<!ELEMENT" y se denominan declaraciones de tipo elemento.
  - También se pueden declarar atributos, entidades y anotaciones para una DTD.
- Las declaraciones de tipo de elemento deben empezar con "<!ELEMENT" seguidas por el identificador genérico del elemento que se declara.
- Ejemplo: <!ELEMENT receta (titulo, ingredientes, procedimiento)>

```
<receta>
<titulo>...</titulo>
<ingredientes>...</ingredientes>
<procedimiento>...</procedimiento>
</receta>
```

válido

```
<receta>
<parrafo>Esto es un párrafo</parrafo>
<titulo>...</titulo>
<ingredientes>...</ingredientes>
<procedimiento>...</procedimiento>
</receta>
```

inválido

Lámina 36 Roberto Gómez C.

Especificación contenido

- Puede ser de cuatro tipos
  - EMPTY
  - ANY
  - Mixed
  - Element

Lámina 37 Roberto Gómez C.

EMPTY

- Puede no tener contenido.
- Suele usarse para los atributos.
- Ejemplo

```
<!ELEMENT salto-de-pagina   EMPTY>
```

Lámina 38 Roberto Gómez C.

ANY

- Puede tener cualquier contenido.
- No se suele utilizar, ya que es conveniente estructurar los documentos XML.
- Ejemplo

```
<!ELEMENT loquesea ANY>
```


Mixed

- Puede tener caracteres de tipo datos o una mezcla de caracteres y sub-elementos especificados en la especificación de contenido mixto.
- Ejemplo

```
<!ELEMENT enfasis (#PCDATA)>
<!ELEMENT parrafo (#PCDATA|enfasis)*>
```

  - El primer elemento definido en el ejemplo (<enfasis>) puede contener datos de carácter (#PCDATA).
  - El segundo (<parrafo>) puede contener tanto datos de carácter (#PCDATA) como sub-elementos de tipo <enfasis>.

Element

- Sólo puede contener sub-elementos especificados en la especificación de contenido.

```
<!ELEMENT mensaje (remite, destinatario, texto)>
```

- Para declarar que un tipo de elemento tenga contenido de elementos se especifica un modelo de contenido en lugar de una especificación de contenido mixto o una de las claves ya descritas.

Lámina 41 Roberto Gómez C.

Modelos de contenido

- Es un patrón que establece los sub-elementos aceptados, y el orden en que se acepta.
- Un modelo sencillo puede tener un solo tipo de sub-elemento:

```
<!ELEMENT aviso (parrafo)>
```

- Esto indica que <aviso> sólo puede contener un solo <parrafo>.

```
<!ELEMENT aviso (titulo, parrafo)>
```

- La coma, en este caso, denota una secuencia. Es decir, el elemento <aviso> debe contener un <titulo> seguido de un <parrafo>.

```
<!ELEMENT aviso (parrafo | grafico)>
```

Lámina 42 Roberto Gómez C.

Modelos de contenido

<!ELEMENT aviso (parrafo | grafico)>

- La barra vertical "|" indica una opción. Es decir, <aviso> puede contener o bien un <parrafo> o bien un <grafico>.
- El número de opciones no está limitado a dos, y se pueden agrupar usando paréntesis.

<!ELEMENT aviso (titulo, (parrafo | grafico))>

- En este último caso, el <aviso> debe contener un <titulo> seguido de un <parrafo> o de un <grafico>.
- Además, cada partícula de contenido puede llevar un indicador de frecuencia, que siguen directamente a un identificador general, una secuencia o una opción, y no pueden ir precedidos por espacios en blanco.

Lámina 43 Roberto Gómez C.

Indicadores de frecuencia

Indicador	Significado
?	Opcional (0 o 1 vez)
*	Opcional y repetible (0 o más veces)
+	Necesario y repetible (1 o más veces)

<!ELEMENT aviso (titulo?, (parrafo+, grafico)*)>

Lámina 44 Roberto Gómez C.

Declaración lista atributos

- Los atributos permiten añadir información adicional a los elementos de un documento.
- La principal diferencia entre los elementos y los atributos, es que los atributos no pueden contener sub-atributos.
- Se usan para añadir información corta, sencilla y desestructurada.
- Ejemplo

```
<mensaje prioridad="urgente">
<de>Alfredo Reino</de>
<a>Hans van Parijs</a>
<texto idioma="holandés">
Hallo Hans, hoe gaat het?
...
</texto>
</mensaje>
```

Lámina 45 Roberto Gómez C.

Definiendo atributos en el DTD

- La clausula ATTLIST permite indicar qué atributos puede tener un ELEMENT, de qué tipo, si son o no obligatorio.
- Sintaxis:

```
<!ATTLIST elemento atributo tipo-atributo valor-defecto>
```

- Ejemplo:

```
<!ATTLIST pago metodo CDATA “contra-reembolso” >
```

- Su uso en XML

```
<pago metodo=“contra-reembolso” />
```

Lámina 46 Roberto Gómez C.

Tipos de atributos

Valor	Descripción
CDATA	El valor son caracteres alfanuméricos
(v1 | v2 | …)	El valor será uno de la lista explicitada
ID	El valor será un identificador único
IDREF	El valor es el ID de otro elemento
IDREFS	El valor es una lista de ID otros elementos
NMTOKEN	El valor es un nombre XML válido
NMTOKENS	El valor es una lista de nombres XML válidos
ENTITY	El valor es una entidad
ENTITIES	El valor es una lista de entidades
NOTATION	El valor es el nombre de una notación
xml	El valor es un valor XML predefinido
Valor	El valor por defecto del atributo
#REQUERIDED	El valor del atributo debe aparecer obligatoriamente en el elemento
#IMPLIED	El atributo no tiene por qué ser incluido
#FIXED valor	El valor del atributo es fijo

Lámina 47 Roberto Gómez C.

Ejemplos

- Ejemplo valor por defecto

```
<!ELEMENT pago EMPTY>
<!ATTLIST pago metodo CDATA “contra-reembolso” >
<pago />
```

  - En este caso, donde no especificamos valor para método, éste contendrá el valor por defecto de contra-reembolso.

- Ejemplo #IMPLIED

```
<!ELEMENT pago EMPTY>
<!ATTLIST pago metodo CDATA #IMPLIED >
```

  - Validará correctamente el siguiente XML:

```
<pago metodo=“tarjeta” />
<pago />
```

Lámina 48 Roberto Gómez C.

Ejemplos

- Ejemplo #REQUIRED

```
<!ELEMENT pago EMPTY>
<!ATTLIST pago metodo CDATA #REQUIRED >
```

- Validará correctamente el siguiente XML:

```
<pago metodo="tarjeta" />
```

- Pero no validará:

```
<pago  />
```

Lámina 49 Roberto Gómez C.

Especificando archivo DTD

- Además antes de que empiece el documento XML, es necesario especificar el archivo donde se encuentra especificado el DTD

```
<!DOCTYPE listas SYSTEM "listaml.dtd">
```

  - El elemento raíz es listas
  - EL DTD esta almacenado en el archivo listaml.dtd
    - Esta archivo debe residir en el mismo directorio que el documento XML
  - Con <!DOCTYPE> se le indica al parseador que la validez del archivo , se tiene que hacer usando el DTD indicado.

Lámina 50 Roberto Gómez C.

Posible definir DTD dentro del mismo archivo XML

```
<?xml version="1.0" encoding="us-ascii"?>
<!DOCTYPE Libros [
<!ELEMENT Libros (libro+)>
<!ELEMENT libro (Titulo, Autor*, Editorial)>
<!ELEMENT Titulo (#PCDATA)>
<!ELEMENT Autor (#PCDATA)>
<!ELEMENT Editorial (#PCDATA)>
]>
<Libros>
        <libro>
                <Titulo>Fundamentos de XML Schema</Titulo>
                <Autor>Allen Wyke</Autor>
                 <Autor>Andrew Watt</Autor>
                 <Editorial>Wiley</Editorial>
        </libro>
        <libro>
                <Titulo>El maestro de esgrima</Titulo>
                <Autor>Arturo Perez</Autor>
                <Editorial>Alfaguara</Editorial>
        </libro>
</Libros>
```

Lámina 51 Roberto Gómez C.

Un último ejemplo

```
<!DOCTYPE elemento-raiz [
  ……
  ……
]>
```

```
<!DOCTYPE elemento-raiz SYSTEM “archivo” >
```

```
<menu>
  <entrada> Foie Gras </entrada>
   <plato> Confit de Pato</plato>
   <postre> Isla Flotante</postre>
</menu>
```

```
<!ELEMENT menu (entrada, plato, postre)>
<!ELEMENT entrada (#PCDATA)>
<!ELEMENT plato (#PCDATA)>
<!ELEMENT postre (#PCDATA)>
```

```
<menu>
  <entrada> Foie Gras </entrada>
   <plato> Confit de Pato</plato>
   <postre> Isla Flotante</postre>
   <postre> Crepes Suzette </postre>
</menu>
```

```
<!ELEMENT menu (entrada, plato, postre+)>
<!ELEMENT entrada (#PCDATA)>
<!ELEMENT plato (#PCDATA)>
<!ELEMENT postre (#PCDATA)>
```

```
  <menu>
    <entrada> Foie Gras </entrada>
     <plato tipo=fria> Confit de Pato</plato>
     <postre> Isla Flotante</postre>
     <postre> Crepes Suzette </postre>
   </menu>
```

```
<!ELEMENT menu (entrada, plato, postre+)>
<!ELEMENT entrada (#PCDATA)>
<!ATTLIST tipo (frio | caliente) “frio”)>
<!ELEMENT plato (#PCDATA)>
<!ELEMENT postre (#PCDATA)>
```

Lámina 52 Roberto Gómez C.

¿Y para que sirve XML?

- Posibilidad de almacenar información en un formato con las siguientes características
  - Portable
  - Extensible
  - Accesible desde lenguajes de programación
    - Existen diferentes APIs
- En un formato universal de datos
  - Es como almacenar los datos en el “Esperanto de Internet” e intercambiarlos fácilmente.
- Posible usar varias tecnologías XML, en especial las de transformación
  - Con XSLT es posible transformar un documento XML de un dialecto a otro, o un formato no-XML si es necesario.

Lámina 53 Roberto Gómez C.

Ejemplo uso XML

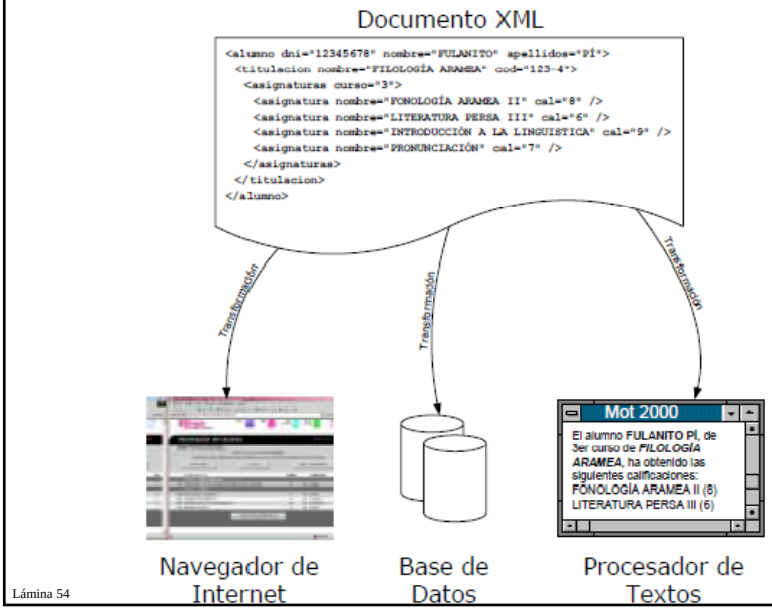


Lámina 54 Roberto Gómez C.

Dialectos XML existentes

- SMIL: Synchronized Multimedia Integration Language
- SVG: Scalable Vector Graphics
- CML: Chemical Markup Language
- MathML: Mathematical Markup Language
- XHTML: eXtensible Hypertext Markup Language
- SOAP: Simple Object Access Protocol

Lámina 55 Roberto Gómez C.

El PCDATA

- Todo el texto de un documento es “parseado”.
- El texto dentro de CDATA es ignorado por el parser.
- PCDATA: Parser Character Data
- Se refiere al texto encontrado entre las etiquetas de principio y de final de un elemento XML.
- Ejemplo de un elemento con PCDATA

```
<elemento> Caracteres datos dentro de este elemento son PCDATA </elemento>
```

Lámina 56 Roberto Gómez C.

Etiquetas, texto y elementos

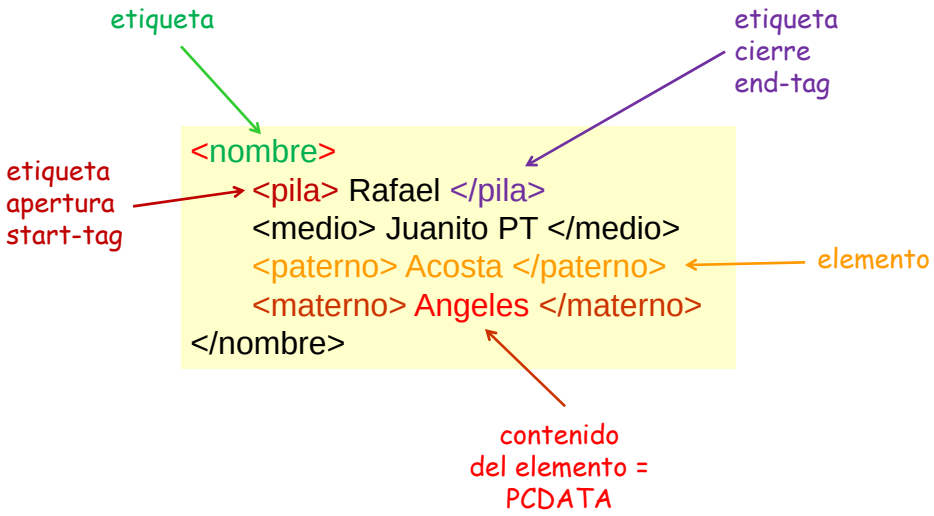


Lámina 57 Roberto Gómez C.

Espacios

- Permitido

```
<first  >John</first>

<first  >John</first  >

<first
>John</first>
```

- No permitido

```
<  first  >John<  /first  >

<  first  >John<  /  first  >
```

Lámina 58 Roberto Gómez C.

Reglas para los elementos

- Cada etiqueta de inicio debe contar con una etiqueta de fin.
- Las etiquetas no pueden sobreponerse.
- Documentos XML solo pueden contar con un elemento raíz.
- Elementos deben seguir las convenciones de nombres .
- XML es sensible a mayúsculas.
- XML respeta los espacios en blanco en su PCDATA.

Lámina 59 Roberto Gómez C.

Etiqueta inicio con etiqueta final

```
<html>
<body>
<p>Here is some text in an HTML paragraph.
<br>
Here is some more text in the same paragraph.
<p>And here is some text in another HTML paragraph. </p>
</body>
</html>
```

Lámina 60 Roberto Gómez C.

Elementos

- XML es estrictamente jerárquico.
- Poner atención para cerrar los elementos hijos antes de cerrar los de su parientes.
- Ejemplo HTML

```
<p>Some <strong>formatted <em>text</strong>, but
</em> no grammar no good!</p>
```

  - ¿ es un hijo de o es un hijo de ?
  - ¿ o los dos son “hermanos” e hijos de <p> ?

Lámina 61 Roberto Gómez C.

Ejemplo anidado

```
<p>Some <strong>formatted <em>text</strong>, but </em> no
grammar no good!</p>
```

```
<p>
    Some
    <strong>
        formatted
        <em>
            text
        </em>
    </strong>
    <em>
        , but
    </em>
    no grammar no good!
</p>
```

Lámina 62 Roberto Gómez C.

Solo puede existir un elemento raíz.

- Mal formado

```
<nombre> Carmela </nombre>
<nombre> Rafael </nombre>
```

- Bien formado

```
<nombres>
    <nombre> Carmela </nombre>
    <nombre> Rafael </nombre>
</nombres>
```

- Bien formado

```
<nombre></nombre>
```

Lámina 63 Roberto Gómez C.

Elementos deben obedecer convenciones XML

- Nombres pueden empezar con letras (incluyendo caracteres no-latinos) o el guión (-) , pero no con números u otro caracteres de puntuación.
- Después del primer carácter, números , guiones y puntos son permitidos.
- Nombres no contienen espacios.
- Nombres no pueden contener el carácter punto y coma (;)
- Nombres no pueden empezar con las letras XML

Lámina 64 Roberto Gómez C.

Ejemplos

- Ejemplos nombres válidos

```
<nombre.apellido>
<profesión>
```

- Ejemplos nombres inválidos

```
<xml-tag>
<123>
<fun=xml>
<mi etiqueta>
```

Lámina 65 Roberto Gómez C.

Sensibilidad a Mayúsculas

- XML es sensible a mayúsculas.
- Inválido:

```
<P></p>
```

- Algunas convenciones

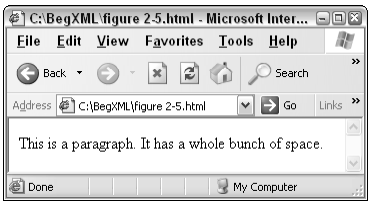
```
<nombre_apellido>
<nombreApellido>
<nombre-apeliido>
<NombreApellido>
```

Lámina 66 Roberto Gómez C.

Espacios en blanco dentro PCDATA

- Ejemplo HTML

```
<p>This is a paragraph.        It has a whole bunch
                of space.</p>
```




```
<p>This is a paragraph.     It has a whole
bunch<br>  of space.</p>
```

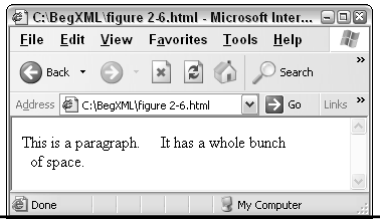


Lámina 67

Roberto Gómez C.

Espacios

- Ejemplo XML

```
<pre>This is a paragraph.        It has a whole bunch
                of space.</pre>
```

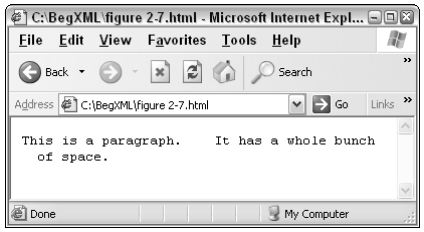


Lámina 68

Roberto Gómez C.

Atributos

- Son pares nombre/valor asociados con un elemento.
- Se encuentran atados a la etiqueta de inicio pero no a al del final.
- Estos atributos indican opciones de la etiqueta.
- Los atributos tienen la siguiente sintaxis

  nombre_atributo= valor
- Los atributos tienen que estar delimitado con comillas dobles (") o comilla simple (').
  - Cuando se usa uno para delimitar el valor del atributo, el otro se puede usar dentro.

Lámina 69 Roberto Gómez C.

Ejemplo atributos

```
<personaje apodo="Juanito PT">
    <nombre> Rafael </nombre>
    <paterno> Acosta </paterno>
    <materno> Angeles </materno>
</personaje>
```

- Valido

```
<input checked='true'>

<input checked="true">
```

- No valido

```
<input checked="true'>
```

Lámina 70 Roberto Gómez C.

Ejemplo sin atributo

```
<ListaCurso>
   <alumno>
      <matricula> 463148 </matricula>
      <nombre> Roberto </nombre>
      <apellido> Sotelo</paterno>
   </alumno>
   <alumno>
      <matricula> 466822 </matricula>
      <nombre> Daniel </nombre>
      <apellido> Cedillo </paterno>
   </alumno>
   <alumno>
      <matricula> 463148 </matricula>
      <nombre> Karen </nombre>
      <apellido> Sánchez </paterno>
   </alumno>
</ListaCurso>
```

Lámina 71 Roberto Gómez C.

Ejemplo con atributo

```
<ListaCurso>
   <alumno  cont="1">
      <matricula> 463148 </matricula>
      <nombre> Roberto </nombre>
      <apellido> Sotelo</apellido>
   </alumno>
   <alumno cont="2">
      <matricula> 466822 </matricula>
      <nombre> Daniel </nombre>
      <apellido> Cedillo </apellido>
   </alumno>
   <alumno cont="3">
      <matricula> 463148 </matricula>
      <nombre> Karen </nombre>
      <apellido> Sánchez </apellido>
   </alumno>
</ListaCurso>
```

Lámina 72 Roberto Gómez C.

Comentarios

- Los comentarios pueden aparecer en cualquier punto del documento, fuera del resto de las marcas, es decir, fuera de las declaraciones etiquetas u otros comentaros.
- Tienen el mismo formato que los comentarios de HTML, por lo que comienzan con "<!--" y terminan con "-->".
- La cadena "--" no puede aparecer dentro de un comentario.
- Ejemplo

```
<alumno  cont="1">
<!-- El alumno es becado  -->
      <matricula> 463148 </matricula>
      <nombre> Roberto </nombre>
      <apellido> Sotelo</apellido>
</alumno>
```

Lámina 73 Roberto Gómez C.

¿Qué no se puede hacer con comentarios?

- No se puede comentar dentro de una etiqueta

```
<middle></middle <!--John lost his middle name in a fire--> >
```

- No se puede usar la cadena de doble hash, dentro de un comentario

```
<!--John lost his middle name -- in a fire-->
```

Lámina 74 Roberto Gómez C.

Elementos vacíos

- Algunas veces un elemento no contiene PCDATA.

```
<alumno  cont="1">
<!-- El alumno es becado  -->
      <matricula> 463148 </matricula>
      <nombre> Roberto </nombre>
      <apellido> </apellido>
</alumno>
```

- Sintaxis elemento vacío: `<apellido/>`
- Sintaxis válidas:

```
<apellido  />
```

- Sintaxis inválidas

```
<apellido/  >
<apellido  /  >
```

Lámina 75 Roberto Gómez C.

Declaraciones XML

```
<?xml version='1.0' encoding='UTF-16' standalone='yes'?>
<alumno  cont="1">
 <!-- El alumno es becado  -->
      <matricula> 463148 </matricula>
      <nombre> Roberto </nombre>
      <apellido> Sotelo</apellido>
</alumno>
```

- Empieza con caracteres <?xml y termina con ?>
- Si se incluye una declaración, se debe incluir el atruib versión, pero los campos encoding y standalone son opciones.
- Los atributos version, encoding y standalone deben ir en ese orden.
- Debe ser lo primero que aparezca en el archivo.
  - El primer carácter en el archivo debe ser: <

Lámina 76 Roberto Gómez C.

Atributos declaración XML

- Version
  - 1.0 ó 1.1
- Encoding
  - Método de codificación de caracteres.
  - UTF-16 o UTF-8 (Unicode)
- Standalone
  - yes: el documento existe por su cuenta
  - no: el documento puede depender de un documento DTD

Lámina 77 Roberto Gómez C.

Instrucciones Procesamiento

- No es muy común su uso.
- Posible incluir instrucciones especificas a alguna aplicación en la información que afectará la forma en que será procesada.
  - Processing Instructions o PI
- No son parte de los datos del documento, son pasadas a la aplicación.

Lámina 78 Roberto Gómez C.

Ejemplo

```
<?xml version='1.0' encoding='UTF-16' standalone='yes'?>
<alumno  cont="1">
 <!– El alumno es becado
      <matricula> 463148 </matricula>
      <nombre> Roberto </nombre>
      <?nameprocessor PRINT cont?>
      <apellido> Sotelo</paterno>
</alumno>
```

Es el PITarget

El texto del PI (la instrucción)

Lámina 79 Roberto Gómez C.

Características PI

- No existen muchas reglas para las PI.
- Se componen de:
  - un <? seguido del nombre de la aplicación que se supone recibirá el PI
    - se conoce como PITarget
  - el resto es la instrucción en sí
  - termina en ?>

Lámina 80 Roberto Gómez C.

Caracteres PCDATA ilegales

- Existen algunos caracteres que no se pueden incluir como PCDATA ya que son parte de la sintaxis XML.
- Para usarlos es necesario usar un equivalente.

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

Lámina 81 Roberto Gómez C.

Definiendo un documento HTML dentro de XML

- Ejemplo:

```
<texto>
    Un documento HTML básico:
     <html>
     <head>
          <title> Ejemplo </title>
     </head>
     <body>
        Esto es el texto.
     </body>
     </html>
</texto>
```

Lámina 82 Roberto Gómez C.

Pasando a formato XML

- Se tendría que convertir a

```
<texto>
    Un documento HTML básico:
     &lt;html&gt;
     &lt;head&gt;
          &lt;title&gt;  Ejemplo &lt;/title&gt;
     &lt;/head&gt;
     &lt;body&gt;
        Esto es el texto.
     &lt;/body&gt;
     &lt;/html&gt;
</texto>
```

Lámina 83 Roberto Gómez C.

Usando CDATA

- Para eso existe el CDATA

```
<texto>
    Un documento HTML básico:
     <![CDATA[
        <html>
        <head>
              <title> Ejemplo </title>
        </head>
        <body>
           Esto es el texto.
        </body>
        </html>
     ]]>
</texto>
```

Lámina 84 Roberto Gómez C.

Secciones CDATA

- Permiten especificar secciones, utilizando cualquier carácter, especial o no, sin que se interprete como una PCDATA.
- De esta forma se puede leer más fácilmente el documento XML.
- Las secciones CDATA empiezan por la cadena "<![CDATA[" y terminan con la cadena "]]>" y sólo ésta última se reconoce como marca.
- No se pueden anidar secciones CDATA.
- CDATA es un termino heredado de SGML.

Lámina 85 Roberto Gómez C.

Ejemplo sección CDATA

```
<ejemplo>
    <![CDATA[
        <HTML>
          <HEAD>
            <TITLE>Rock & Roll</TITLE>
          </HEAD>
    ]]>
</ejemplo
```

Todo lo que encuentra después <![CDATA[de y antes]]> de es ignorado por el parser, y pasado a la aplicación tal cual.

Lámina 86 Roberto Gómez C.

Pero…

- Sintaxis CDATA introduce otra complejidad a XML
- La secuencia]]> no se permite en la sección CDATA o fuera de esta.
- Si es necesario usar estos caracteres en ese orden, debe usarse:

```
]]&gt;
```

Lámina 87 Roberto Gómez C.

Introducción a XML

Roberto Gómez Cárdenas
rogomez@itesm.mx
http://homepage.cem.itesm.mx/rogomez

Lámina 88 Roberto Gómez C.