

XML

Historia:

- HTML, un standard para modelado de datos en la web.

```
<html>
  <head>
    <title>Un documento</title>
  </head>
  <body>
    <p>Este es el <em>primer</em> parrafo del documento</p>
    <p>Este es el <em>segundo</em> parrafo del documento
  </body>
</html>
```

- La guerra de los browsers, extensiones a HTML, font, b,i, y otros tags

- . ~~HTML, un standard para modelado de datos en la web~~
- . HTML, un standard para la presentacion de datos en la web

```
<html>
  <head>
    <title>Un documento</title>
  </head>
  <body>
    <p><font face="arial" size="12">Este es el <b>rimer</i>
parrafo del documento</font></p>
    <p><font color="green">Este</font> es el <i>segundo</i>
parrafo del documento
    <blink>Peligro!</blink>
  </body>
</html>
```

Rumbo a un standard para el modelado de datos en la web

- El standard debe ser neutro con respecto a la presentacion de los documentos
- El standard debe ser extensible
- El standard debe ser facil de interpretar tanto por seres humanos como por programas

XML 1.0 (1998)

Tim Bray

Jean Paoli

C.M. Sperberg-McQueen

Eve Maler

Un documento XML

```
<?xml version="1.0"?>
<!-- Informacion sobre el curso 75.06 Organizacion de datos
-->
<curso codigo="7506" nombre="Organizacion de Datos">
  <docentes>
    <docente>
      <nombre DNI="...">Jorge A Saubidet</nombre>
      <cargo>Titular</cargo>
    </docente>
    <docente>
      <nombre DNI="12232654">Ramiro Vera</nombre>
      <cargo>JTP</cargo>
    </docente>
    ....
  </docentes>
  <horario>
    <dia1 tipo="teo-pra" obl="si" dia="Lunes" hora-
desde="19" hora-hasta="22" aula="200" />
    <dia2 tipo="teo-pra" obl="si" dia="Jueves" hora-
desde="19" hora-hasta="22" aula="200" />
  </horario>
  <vacantes>255</vacantes>
  <web>http://groups.yahoo.com/groups/datos</web>
</curso>
```

Especificacion XML 1.0

Todo documento XML debe estar bien-formado

Documentos

[1] document ::= prolog element Misc*

El prologo

[22] prolog ::= XMLDecl? Misc* (doctypeddecl Misc*)?

[23] XMLDecl ::= '<?xml' VersionInfo EncodingDecl?
SDDDecl? S? '?>'

[24] VersionInfo ::= S 'version' Eq (' VersionNum ' | "
VersionNum ")

[25] Eq ::= S? '=' S?

75.06 Organizacion de Datos: XML

[26] VersionNum ::= ([a-zA-Z0-9_.:] | '-') +

[27] Misc ::= Comment | PI | S

Comments

[15] Comment ::= '<!--' ((Char - '-') | ('-' (Char - '-'))) *
'-->'

Ejemplo:

<!-- Este es un comentario -->

Processing instructions

[16] PI ::= '<?' PITarget (S (Char* - (Char* '?')
Char*))) ? '>'

[17] PITarget ::= Name - (('X' | 'x') ('M' | 'm') ('L' | 'l'))

Ejemplo:

<?save file="foo.xml"?>

La declaracion del DTD

[28] doctypedec ::= '<!DOCTYPE' S Name (S ExternalID)? S?
1 ('[' (markupdecl | PEReference | S)* ']' S?)? '>'

[29] markupdecl ::= elementdecl | AttlistDecl | EntityDecl
| NotationDecl | PI | Comment

Elementos

Contenido de un elemento:

[43] content ::= (element | CharData | Reference | CDsect
| PI | Comment)*

Elementos

[39] element ::= EmptyElemTag

| STag content ETag

Tags

[40] STag ::= '<' Name (S Attribute)* S? '>'

75.06 Organizacion de Datos: XML

[41] $\text{Attribute} ::= \text{Name Eq } \underline{\text{AttValue}}$
e

[42] $\text{ETag} ::= '</' \text{Name } S? '>'$

Elemento vacio

[44] $\text{EmptyElemTag} ::= '<' \text{Name } (S \text{ Attribute})^* S? '/>'$

Ejemplos:

`<foo />`

`<foo name="hola" id="1232" />`

Elementos, tags, atributos

`<elemento atr1="value" atr2="value">....</elemento>`

Reglas:

- . Todos los tags deben cerrarse
- . No pueden solaparse tags
- . No puede haber dos atributos con el mismo nombre
- . Todo atributo debe tener un valor
- . Los valores de los atributos deben estar entre comillas (dobles o simples)

Nombres validos para elementos y atributos

[4] NameChar ::= Letter | Digit | '.' | '-' | '_' | ':'
| CombiningChar | Extender

[5] Name ::= (Letter | '_' | ':') (NameChar)*

[6] Names ::= Name (S Name)*

[7] Nmtoken ::= (NameChar)+

[8] Nmtokens ::= Nmtoken (S Nmtoken)*

CharData

[14] CharData ::= [^<&]* - ([^<&]* ']]>' [^<&]*)

Referencias

[67] Reference ::= EntityRef | CharRef

[68] EntityRef ::= '&' Name ';'

Ejemplos:

&

>

<

"

'

&pepe;

Referencias a Character

[66] CharRef ::= '&#' [0-9]+ ';'

| '&#x' [0-9a-fA-F]+ ';'

CDATA sections

[18] CDsect ::= CDstart CData CDEnd

[19] CDstart ::= '<![CDATA['

[20] CData ::= (Char* - (Char* ']]>' Char*))

[21] CDEnd ::= ']]>'

Ejemplo:

```
<![CDATA[<html><head><title>Hola</title><body>foo</body><
/html>]]>
```

Documentos bien formados

Resumen de reglas para que un documento este bien-formado

- . Debe tener un unico elemento raiz
- . Los elementos deben estar correctamente anidados
- . Todos los elementos deben tener un tag de comienzo y fin
- . No se pueden repetir nombres de atributos en un mismo elemento
- . Los nombres de los atributos y elementos deben seguir la definicion dada
- . Los atributos deben tener un valor
- . Los valores de los atributos deben estar entre comillas

Namespaces

```
<foo xmlns:cars="http://cars.org"
xmlns:persons="http://persons.org">
  <report>
    <cars:name>Toyota</cars:name>
    <persons:name>Juan Perez</persons:name>
  </report>
</foo>
```

Scope

Default namespace

DTDs

Documentos validos, un concepto nuevo

Un documento es valido cuando cumple con la definicion establecida en un DTD

DTDs: Sintaxis

Elementos

Declaración de tipo de elemento

```
[45] elementdecl ::= '<!ELEMENT' S Nombre S
                      contentspec S? '>'
```

```
[46] contentspec ::= 'EMPTY' | 'ANY' | Mixed
                      | children
```

```
<!ELEMENT name ANY>
<!ELEMENT name EMPTY>
<!ELEMENT name (e1,e2)>
<!ELEMENT name (e1|e2)>
<!ELEMENT name (e1?,e2*,e3+)>
```

Algunos ejemplos:

```
<!ELEMENT foo (a,b*,c|a,d?,c+)>
<!ELEMENT foo ((a|b,c)+,c,(a|d*)?)>
```

Declaración de contenido mixto

```
[51] Mixed ::= '(' S? '#PCDATA' (S? '|' S? Nombre)*
                S? ')*'
```

75.06 Organizacion de Datos: XML

```
| '(' S? '#PCDATA' S? ')' |
```

Atributos

Declaración de lista de Atributos

[52] AttlistDecl ::= '<!ATTLIST' S Nombre AttDef* S? '>'

[53] AttDef ::= S Nombre S AttType S DefaultDecl

<!ATTLIST elemento attributeName type modifier>

Tipos:

CDATA
(en1|en2...)
ID
IDREF
IDREFS
NMTOKEN
NMTOKENS
ENTITY
ENTITIES

Modifiers:

value
#REQUIRED
#IMPLIED

#FIXED value

Ejemplos:

```
<!ATTLIST foo color (verde|azul)>
<!ATTLIST persona dni CDATA #REQUIRED>
<!ATTLIST combo visible (yes|no) yes>
<!ATTLIST foo a CDATA #IMPLIED>
```

Entidades

Internas:

```
<!ENTITY name valor>
```

Externas:

```
<!ENTITY entity-name SYSTEM "URI/URL">
```

[69] PEReferencia ::= '%' Nombre ';' ;

```
<!ENTITY % name valor>
```

Documentos validos

Resumen de reglas para que un documento sea valido

- Debe cumplir las especificaciones del DTD mediante el cual se valida
- El nombre en la declaracion del tipo de documento debe ser igual al nombre del elemento raiz
- No se puede declarar mas de una vez un elemento
- No puede aparecer mas de una vez el mismo elemento en las declaraciones de elementos mixtos
- No es un error declarar atributos para elementos no declarados
- Si existen varias declaraciones de atributos para un mismo elemento, son concatenadas.
- Si existen varias definiciones de atributos para el mismo atributo se utiliza la primera definicion encontrada
- Si se declara mas de una vez una entidad, la

primera es utilizada

DTDs: problemas

Carencia de tipos de datos

El modelo de elementos es pobre, ausencia de ALL

No admite herencia ni derivacion de tipos

No permite restricciones

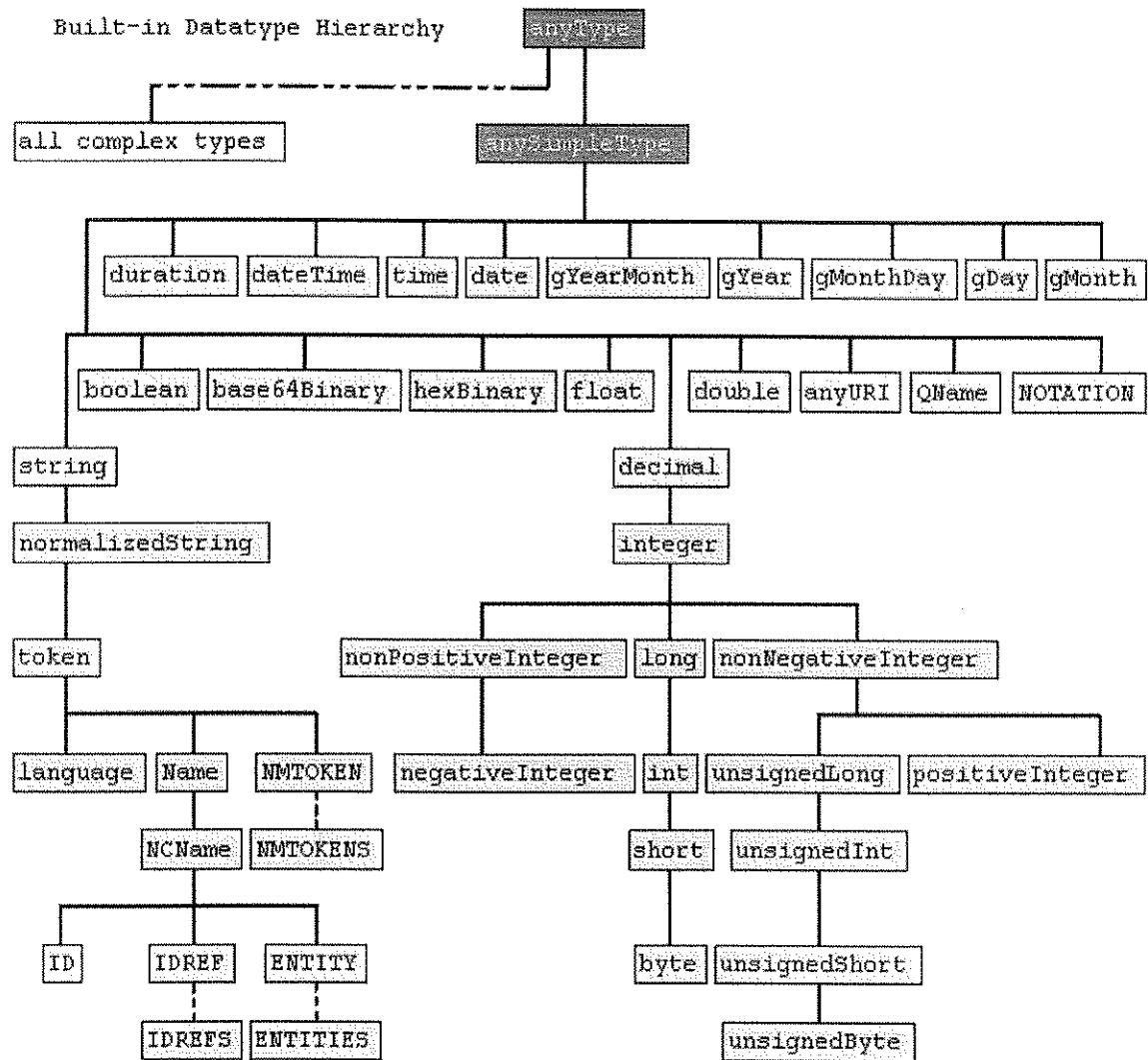
Ausencia de valores defaults para elementos








No es XML!

Alternativas a los DTDs

- . XML Schema
- . RelaxNG
- . Schematron

XML Schema: Tipos



- | | | | |
|---|--------------------------|---|-------------------------------------|
|  | ur types |  | derived by restriction |
|  | built-in primitive types |  | derived by list |
|  | built-in derived types |  | derived by extension or restriction |
|  | complex types | | |

XML Schema

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.w3schools.com"
xmlns="http://www.w3schools.com"
elementFormDefault="qualified">
...
...
</xs:schema>
```

Elementos simples, atributos:

```
<xs:element name="xxx" type="yyy"/>
<xs:element name="color" type="xs:string" default="red"/>
<xs:element name="color" type="xs:string" fixed="red"/>
<xs:attribute name="lang" type="xs:string" default="EN"/>
<xs:attribute name="lang" type="xs:string" use="optional"/>
<xs:attribute name="lang" type="xs:string" use="required"/>
```

Restriccion por valor:

```
<xs:element name="age">
  <xs:simpleType>
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="0"/>
      <xs:maxInclusive value="100"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Restriccion por un conjunto de valores

```
<xs:element name="car">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="Audi"/>
      <xs:enumeration value="Golf"/>
      <xs:enumeration value="BMW"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Restriccion por patrones

```
<xs:element name="letter">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[a-z]"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>

<xs:element name="gender">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="male|female"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Restriccion por longitud

```
<xs:element name="password">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:minLength value="5"/>
      <xs:maxLength value="8"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Elementos compuestos

```
<xs:element name="employee">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="employee" type="personinfo"/>
<xs:element name="student" type="personinfo"/>
<xs:element name="member" type="personinfo"/>
<xs:complexType name="personinfo">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

Sequence, All, choice

```
<xs:element name="person">
  <xs:complexType>
    <xs:all>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
    </xs:all>
  </xs:complexType>
</xs:element>
```

```
<xs:element name="person">
  <xs:complexType>
    <xs:choice>
      <xs:element name="employee" type="employee"/>
      <xs:element name="member" type="member"/>
    </xs:choice>
  </xs:complexType>
</xs:element>
```

```
<xs:element name="person">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Derivacion por extension

```
<xs:element name="employee" type="fullpersoninfo"/>
<xs:complexType name="personinfo">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="fullpersoninfo">
  <xs:complexContent>
    <xs:extension base="personinfo">
      <xs:sequence>
        <xs:element name="address" type="xs:string"/>
        <xs:element name="city" type="xs:string"/>
        <xs:element name="country" type="xs:string"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Ocurrencias

```
<xs:element name="person">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="full_name" type="xs:string"/>
      <xs:element name="child_name" type="xs:string"
        maxOccurs="10" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

XML Schema: ejemplo

```
<?xml version="1.0" encoding="UTF-8" ?>
<schema targetNamespace="http://www.example.com/IPO"
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:ipo="http://www.example.com/IPO">

  <complexType name="Address">
    <sequence>
      <element name="name" type="string"/>
      <element name="street" type="string"/>
      <element name="city" type="string"/>
    </sequence>
  </complexType>

  <complexType name="USAddress">
    <complexContent>
      <extension base="ipo:Address">
        <sequence>
          <element name="state" type="ipo:USState"/>
          <element name="zip" type="positiveInteger"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>

  <simpleType name="USState">
    <restriction base="string">
      <enumeration value="AK"/>
      <enumeration value="AL"/>
      <enumeration value="AR"/>
      <enumeration value="PA"/>
    </restriction>
  </simpleType>

</schema>
```


Metadatos

Metadatos son datos utilizados para describir otros datos / Data about Data

<pre><cdcollection> <cd id="1"> <title>Canciones infantiles</title> <artist>M.E Walsh</artist> ... </cd> ... </cdcollection></pre>	<pre>CREATE TABLE cds(title varchar(200), author varchar(240), cdId integer(12) not null, primary key(cdId));</pre>
--	--

Los metadatos se utilizan para describir la “**semantica**” de un dato

- *El titulo del CD cuyo id es 1 es “Canciones infantiles”*
- *El artista que interpreta el CD cuyo id es 1 es “M.E Walsh”*

Muchos terminos para una misma semantica:

- “interprete, artista, cantante”
- “creador, compositor, autor”

Estandarizacion de semanticas: Dublin Core

Dublin Core es un grupo de trabajo que ha establecido el significado de ciertos terminos que pueden usarse para definir metadatos

Elementos Dublin Core:

Title	Type
Creator	Format
Subject	Identifier
Description	Source
Publisher	Language
Contributor	Relation
Date	Coverage

Usando Dublin Core en XML

```
<cdcollection xmlns:dc="http://purl.org/dc/elements/1.1/">
  <cd id="1">
    <dc:title></dc:title>
    <dc:creator></dc:creator>
  </cd>
</cdcollection>
```

Estandarizacion de semanticas

<pre><cdcollection xmlns:dc="http://purl.org/dc/elements/1.1/"> <cd dc:identifier="1"> <dc:title>Canciones infantiles</dc:title> <dc:creator>M.E Walsh</dc:creator> ... </cd> ... </cdcollection></pre>	<pre>CREATE TABLE cds(http://purl.org/dc/elements/1.1/title varchar(200), http://purl.org/dc/elements/1.1/creator varchar(240), http://purl.org/dc/elements/1.1/identifier integer(12) not null, primary key(http://purl.org/dc/elements/1.1/identi fier));</pre>
---	--

Taxonomias

Una taxonomía establece una colección de valores standard para una o mas propiedades de forma tal que los valores que se atribuyen a dichas propiedades puedan ser comparables.

Ejemplo:

<code><dc:creator>M.E Walsh</dc:creator></code>	<code><dc:creator>Maria E Walsh</dc:creator></code>
---	---

¿Que es un meta-dato?

Nombre-valor

dc:creator = "M.E Walsh"

dc:title = "Canciones Infantiles"

Objeto-nombre-valor

CD-creator-M.E. Walsh

CD-title-Canciones Infantiles