

LENGUAJES DE MARCAS Y SISTEMAS DE GESTIÓN DE INFORMACIÓN

Bloque XML: UD6: Schemas

XML Schemas

2

... the syntax of XML Schema was obviously produced by someone who grew up at the bottom of a deep well in the middle of a dark, wasteful moor where he was tortured daily by abusive giant squirrels and wishes to share his pain with the world.

--Robin Berjon
(co-author SVG 1.1 spec)

XML Schemas (XSD)

3

- XSD: **X**ML **S**chema **D**efinition.
- Alternativa a DTD basada en XML.
- Ventajas respecto a DTDs
 - Están escritos en XML
 - No hay que aprender una nueva sintaxis, Lenguaje DTD no intuitivo
 - Puedes usar el editor XML y el parser XML
 - Soportan tipos de datos
 - Más fácil validar distintos tipos de datos: fechas, números
 - Se pueden definir “restricciones”
 - límites dentro de los anteriores o patrones de datos,
 - o extensiones sobre los datos, tipos derivador de otros
 - Mejor soporte espacios de nombres
 -

XML Schema: ejemplo

4

- Tutorial XML Schema:
https://www.w3schools.com/xml/schema_intro.asp
- Ejemplo: esquema similar a note.dtd

```
<?xml version="1.0"?>  <!-- File: ejemplo.xsd -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="nota">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="de" type="xs:string"/>
        <xs:element name="a" type="xs:string"/>
        <xs:element name="tema" type="xs:string"/>
        <xs:element name="texto" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

El prefijo habitual es xs o xsd

Enlazar XML con XSD: método a)

5

- **No lo vamos a usar este curso...**
- Se asocia usando atributos en elemento raíz datos
- Varias posibilidades y atributos.
- Se usa **noNamespaceSchemaLocation** (se entiende que no se usa espacio de nombres en la instancia de datos)

```
<?xml version="1.0"?>  <!-- File nota.xml -->
<nota
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="nota.xsd">

  <de> ...
```

En los datos el prefijo habitual es xsi

Tranquilos, xmlcopyeditor lo genera automáticamente

XML y XSD con espacios de nombres: método b)

6

- Si se usan espacios de nombre en la instancia de datos

```
<?xml version="1.0" encoding="UTF-8"?> <!-- marcadores.xsd-->
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.abrirllave.com/marcadores"
  elementFormDefault="qualified">
  <xs:element name="marcadores">
```

```
<?xml version="1.0" encoding="UTF-8"?> <!-- marcadores.xml-->
<marcadores
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.abrirllave.com/marcadores
    marcadores.xsd"
  xmlns="http://www.abrirllave.com/marcadores">
<pagina> ...
```

+actual: Asociar con xml-model

7

- Se considera un sistema genérico para cualquier tipo de sistema de validación
- En VScode se genera con "Insert XML Schema Association" (<? y seleccionar)
 - también existe para DTD

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

```
<?xml-model href="nota.xsd" ... ?>
```

```
<nota>
```

```
  <de>Tove</de>
```

```
  ...
```

Ejercicio con xmlCE

8

- Descargar ejemplos de plataforma
- Asociar en xmlcopyeditor fichero nota.xml
 - con nota-sin-ns.xsd
 - con nota-con-ns.xsd
- Observar diferencias en forma de asociar
- Validar fichero XML
 - Modificar datos para que no supere validación
- Validar fichero XSD..
 - Atentos, es una opción distinta en xmlCE

Ejercicio

9

- Descargar ejemplos de plataforma
- Validar en CLI con xmlstarlet

```
$ xmlstarlet val -e -s nota.xsd nota.xml
```

nota.xml - valid
- Modificar datos en .xml para que no supere validación
- Asociar xml con xsd usando xml-model. Ver errores
- ¿Validar fichero XSD?
 - Ver errores en el editor o en CLI al usar xmlstarlet

Definir elementos simples

10

- Elementos sólo "*texto*": ni atributos ni hijos
`<xs:element name="xxx" type="yyy"/>`
 - xxx nombre del elemento, yyy su tipo de datos.
- tipos de datos predefinidos. Los más usados son:
 - xs:date (yy-mm-dd)
 - xs:string
 - xs:decimal
 - xs:integer
 - xs:boolean

El tipo de datos restringe los posibles contenidos. Si el tipo es "xs:date" y contiene "Hello", el elemento no será válido.

Esto NO era posible con las DTDs. Además se pueden añadir restricciones(facets) o forzar a que el texto siga un cierto patrón.

Elementos simples: Ejemplos

11

- Elementos XML:

```
<apellido>Rivera</apellido>  
<edad>36</edad>  
<fnacim>1970-03-27</fnacim>
```

- Sus definiciones en un esquema :

```
<xs:element name="apellido" type="xs:string"/>  
<xs:element name="edad" type="xs:integer"/>  
<xs:element name="fnacim" type="xs:date"/>
```

Formato date: YYYY-MM-DD

- Uso del valor por defecto o fijo

```
<xs:element name="color"  
            type="xs:string" default="rojo"/>  
<xs:element name="color" type="xs:string" fixed="azul"/>
```

Elementos complejos

12

- Contienen otros elementos y/o atributos
- Podemos clasificar en
 - Elementos que sólo contienen otros elementos, contenedores
 - Elementos que sólo contienen texto y atributo
 - Elementos vacíos
 - Elementos que contiene otros elementos y texto
 - ¡Mixtos, evitar!

Definir elementos contenedores

13

- Se consideran, a efectos de sintaxis, tipos complejos.

`<xs:complexType>`

- Dos posibilidades:

- **la correcta, usando tipos**

`<xs:complexType name="Tpersoninfo">`

`<xs:sequence>`

- la mejorable, definido el tipo en el propio elementos

`<xs:element name="employee">`

`<xs:complexType>`

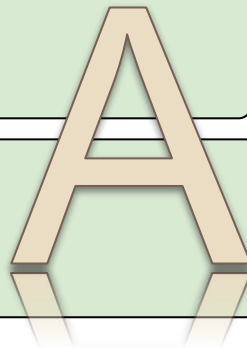
Tipos complejos: contenedores

14

- Más estructurado, reutilizar, legible
 - Se define un tipo y se le da un **name**
 - Se define el elemento con el atributo **type**: se refiere al **tipo definido por el usuario**.

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

```
<xs:element name="employee" type="Tpersoninfo"/>  
<xs:element name="student" type="Tpersoninfo"/>
```



Tipos complejos: contenedores

15

- Evitar..

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

Indicador sequence: los elementos deben aparecer en el mismo orden

B

Ejemplo tablón compraventa

16

- Realizar schema ejercicio tablón de anuncios
 - Simplificando
 - Sólo un anuncio
 - que no tenga atributos, etc...

```
<anuncio>  
  <fecha>12/12/2012</fecha>  
  <asunto>Vendo Bici paseo</asunto>  
  <texto>Bici marca RXcederTX en buen estado.... </texto>  
  <precio>100</precio>  
  <contacto>García, 956121212</contacto>  
</anuncio>
```


Ejemplo tablón compraventa

17

- Primero usar xs:string y probar
- Luego usar distintos tipos de datos para precio y fecha
 - Probar si es válido. Modificar tipos de datos para comprobar validez
- Importante: se puede comprobar **si el esquema**
 - está bien formado (editor o CLI): ¡¡¡ es un fichero XML !!!
 - y es válido como schema (editor o intentar validar con xmlstarlet)

Solución tablón (un anuncio..)

18

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:complexType name="Tanuncio">
    <xs:sequence>
      <xs:element name="fecha" type="xs:date"/>
      <xs:element name="asunto" type="xs:string"/>
      <xs:element name="texto" type="xs:string"/>
      <xs:element name="precio" type="xs:integer"/>
      <xs:element name="contacto" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>

  <xs:element name="anuncio" type="Tanuncio"/>
```

Avanzamos con tablón: más de un anuncio..

19

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

  ...
  <!-- añadimos a lo anterior -->
  <xs:complexType name="Ttablon">
    <xs:sequence>
      <xs:element name="anuncio" type="Tanuncio"
        minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="tablón" type="Ttablon"/>
```

maxOccurs y minOccurs: valor por defecto 1

Restricciones

20

- Se pueden aplicar restricciones sobre tipos de elementos y atributos
 - estas restricciones limitan los valores posibles en elementos y atributos
- Aplicadas a un tipo:
 - Se define un nuevo tipo simple y
 - dentro se define una restriction..
 - con atributo base que indica el tipo a restringir
 - Dentro de la restricción se aplica la sintaxis que corresponda a los distintos tipos de restricciones que existen

Restricciones 1. Rango enteros

21

- Limitando el rango de un entero

```
1 <xs:simpleType name="Tedad" > 3
  2 <xs:restriction base="xs:integer">
    4 <xs:minInclusive value="0"/>
      <xs:maxInclusive value="120"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="edad" type="Tedad" />
```

Recordamos definición tipo entero:

```
<xs:element name="edad" type="xs:integer"/>
```

Restricciones 2. Enumerados

22

```
<xs:element name="car" type="carType"/>
```

Ver. A

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

MÁS LEGIBLE, Y SE
PUEDE REUTILIZAR
EL TIPO

```
<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>
```

Ver. B

Restricciones 3. longitud

23

```
<xs:element name="password" type="Tpassword" />
```

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

Longitud
entre 5 y 8

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

Obligamos a
Longitud 8

Restricciones 4. Patrones (I)

24

- Una minúscula entre a y z:

```
<xs:restriction base="xs:string">  
  <xs:pattern value="[a-z]"/>  
</xs:restriction>
```
- Tres mayúsculas entre A y Z:

```
<xs:pattern value="[A-Z]{3}"/>
```
- Entre tres y cinco letras:

```
<xs:pattern value="[a-zA-Z]{3,5}"/>
```
- Tres letras al menos, sin límite superior

```
<xs:pattern value="[a-zA-Z]{3,}"/>
```


Restricciones 4. Patrones (II)

25

- Una única vocal minúscula
`<xs:pattern value="[aeiou]"/>`
- Cualquier caracter menos xyz
`<xs:pattern value="[^xyz]"/>`
- Abreviaturas para número de ocurrencias:
 - Cero o más letras o dígitos de la a la z:
`<xs:pattern value="([a-zA-Z0-9])*"/>`
 - Uno o más: 1 minúscula, se permiten espacios
`<xs:pattern value="([a-z])+"/>`
- Otros abreviaturas (bloques)
 - `\d` dígitos
 - `\w` Word chars: letras, dígitos
=> incluye Ñ, ñ, á, etc..
 - `\s` Espacio en blanco genéricos
 - `\p{P}` signos de puntuación

Restricciones 4. Patrones (III)

26

- Uno o más pares de letras, cada par compuesto de minúscula y mayúscula.
`<xs:pattern value="([a-z][A-Z])+"/>`
=> Ver diferencia con `([a-zA-Z])+`
- Enumerado con patrones:
`<xs:pattern value="paypal||tarjeta"/>`
- Texto con espacios.
`<xs:pattern value="([\w\s])*"/>`
- Mucho más sobre patrones
 - <https://regexone.com/>

Ejercicio restricciones: tablon

27

- Ejemplo tablón, añadir restricciones:
 - Asunto: longitud entre 8 y 22
 - Texto: Permitir espacios y letras, nada más.
longitud mínima 10
 - Precio: dos decimales.
 - Investigar tipos numéricos
 - investigar restringir n1 decimales
- Probar distintos valores si validan o no.
- Crear tipo para DNI formato: xxxxxxxx-L

XSD: Atributos

28

- Definición similar a los tipos simples.
`<xs:attribute name="xxx" type="yyy"/>`
- Tipos: los mismos que para elementos.
- Ejemplo:
 - Elemento XML:
`<lastname lang="EN">Smith</lastname>`
 - Definición de dicho atributo:
`<xs:attribute name="lang"
type="xs:string"/>`
- Falta definir dónde se incluye la definición del atributo

XSD Atributos (II)

29

- Los atributos son opcionales por defecto.
- Si quiero que sea obligatorio
`<xs:attribute name="lang" type="xs:string"
 use="required" />`
- Con valor por defecto:
`<xs:attribute name="lang" type="xs:string"
 default="EN" />.`
- Con valor fijo:
`<xs:attribute name="lang" type="xs:string"
 fixed="EN" />`

¿Dónde se incluye la definición del atributo?

30

- Para elementos contenedores es sencillo
 - al final de la definición
- Pero para elementos simples la cosa se complica..

Atributos: definición en contenedor (caso 1)

31

- ¿Dónde se coloca la definición del atributo en un elemento contenedor?
 - Dentro definición del tipo o *elemento*, al final de la misma.

- Ejemplo: prioridad y fechaAlta de un empleado

```
<xs:element name="empleado" type="Tpersoninfo" />
<xs:complexType name="Tpersoninfo">
  <xs:sequence>
    <xs:element name="nombre" type="xs:string"/>
    <xs:element name="apellido" type="xs:string"/>
  </xs:sequence>
  <xs:attribute name="prioridad" type="xs:string"/>
  <xs:attribute name="fechaAlta" type="xs:date"/>
</xs:complexType>
```

Atributos: definición en contenedor. Ejemplo tablón

32

- Añadir atributo al elemento raíz de tablon:
 - Fecha apertura tablón y categoría del tablón

```
<xs:complexType name="Ttablon">
  <xs:sequence>
    <xs:element name="anuncio" type="Tanuncio"
      maxOccurs="unbounded" minOccurs="1"/>
  </xs:sequence>
  <xs:attribute name="fechaApertura" type="xs:date"/>
  <xs:attribute name="categoria" type="xs:string"/>
</xs:complexType>
```


Atributos: definición en tipos simples (caso 2)

33

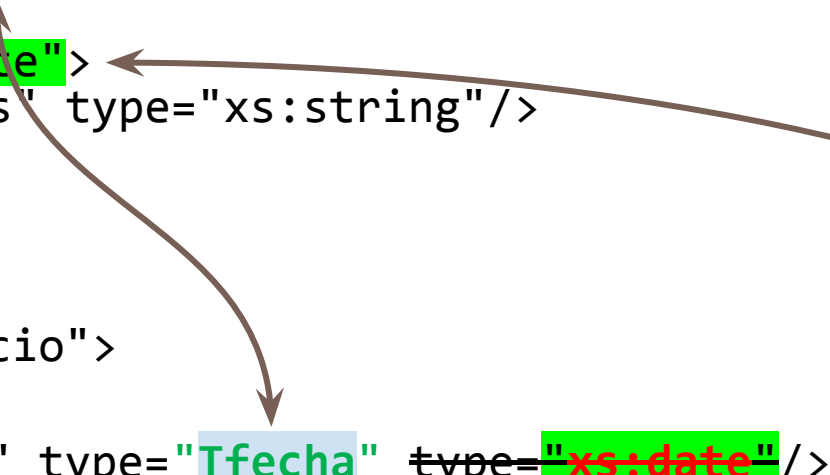
- A pesar de ser tipo simple, la cosa es más difícil.
- ¿Por qué? Los tipos simples no admiten atributos, *un elemento con atributos se considera complejo*
- Para definir un elemento simple con atributo hay que definir el elemento como tipo complejo
- Además implica sintaxis adicional nueva:
 - `simpleContent`
 - `extension` (o `restriction`)

Atributos en tipos simples

34

- Ejemplo: añadir atributo país al elemento fecha:
 - Definimos un nuevo tipo como **complexType** (Tfecha)
 - El contenido es simple: **simpleContent**.
 - Definimos **extensión**/restricción del tipo base **date** dentro de simpleContent
 - Dentro de la extensión se incluye **el atributo**

```
<xs:complexType name="Tfecha">
  <xs:simpleContent>
    <xs:extension base="xs:date">
      <xs:attribute name="pais" type="xs:string"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
...
<xs:complexType name="Tanuncio">
  <xs:sequence>
    <xs:element name="fecha" type="Tfecha" type="xs:date"/>
```



¿extensión? parte de tipo ya existente, y añade atributos o/y elementos.

Extensión de tipo

35

- Mejora XSD: Permite, definido un tipo, definir otro basado en el primero al que le añade información (elementos o atributos)
- Utiliza los elementos
 - **xs:complexContent** / **xs:simpleContent**
 - **xs:extension**
- Ejemplo extender con atributos: elementos simples con atributos => el apartado previo.
- Ejemplo extender elementos: partir del tipo `personInfo` (nombre, apellidos) y crear tipo `fullpersoninfo` que añade elementos dirección y ciudad
 - solución más legible y fácil de modificar/mantener

Ejemplo: extensiones de tipos con elementos

36

```
<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:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

```
<xs:element name="employee" type="fullpersoninfo"/>
```

Atributo a elemento con tipo restringido (caso 3)

37

- Caso: añadir atributo moneda a precio
 - precio tiene un tipo "restringido" (Tprecio)
 - con dos decimales y mínimo de 1
 - y le intentamos añadir un atributo
- **Problema:** De acuerdo con la norma, *"Each complex type definition is either*
 - *a restriction of a complex base type definition or*
 - *an extension of a simple or complex ...*
- Tprecio ya era una restricción, no lo podemos extender.
=> Nos obliga a definir el tipo final en dos pasos:
 1. tipo simple inicial con la restricción
 2. tipo complejo derivado del inicial (**extensión**) con atributo



Solución final:

atributo moneda de precio..

38

1

```
<xs:simpleType name="Tprecio_base">  
  <xs:restriction base="xs:decimal">  
    <xs:fractionDigits value="2" />  
    <xs:minInclusive value="1" />  
  </xs:restriction>  
</xs:simpleType>
```

2

```
<xs:complexType name="Tprecio">  
  <xs:simpleContent>  
    <xs:extension base="Tprecio_base">  
      <xs:attribute name="moneda" type="xs:string"/>  
    </xs:extension>  
  </xs:simpleContent>  
</xs:complexType>
```

```
<xs:element name="precio" type="Tprecio" />
```

Elemento complejo **mixto**: texto y otros elementos

39

- No recomendable..
- Ejemplo

Único cambio: se añade atributo
mixed por defecto false

```
<letter>
```

```
  Dear Mr.<name>John Smith</name>.
```

```
  Your order <oid>1032</oid> will be shipped on  
2001-07-13. </letter>
```

```
---
```

```
<xs:element name="letter" type="lettertype"/>
```

```
<xs:complexType name="lettertype" mixed="true">
```

```
  <xs:sequence>
```

```
    <xs:element name="name" type="xs:string"/>
```

```
    <xs:element name="oid" type="xs:integer"/>
```

```
  </xs:sequence>
```

```
</xs:complexType>
```

Indicadores

40

- **De orden:** definen el orden de los elementos.
 - **xs:all:** los elementos pueden aparecer en cualquier orden, y cada uno debe aparecer una única vez.
 - **xs:choice:** sólo uno de los que se enumeran.
 - **xs:sequence:** en ese orden.
- **De ocurrencia:** definen cuántas veces puede aparecer un elemento.
 - **maxOccurs y minOccurs**

```
<xs:element name="destino" minOccurs="1" maxOccurs="5" />
```

Sin límite: unbounded
- De grupo: no los estudiamos, para agrupar elementos o atributos

Ejemplo choice

41

```
<xs:complexType name="Tanuncio">
  <xs:sequence>
    <xs:element type="xs:date" name="fecha"/>
    <xs:choice>
      <xs:element type="xs:string" name="asunto"/>
      <xs:element type="xs:string" name="texto"/>
    </xs:choice>
    <xs:element type="xs:float" name="precio"/>
    ...
  </xs:sequence>
</xs:complexType>
```

Ejemplos all

42

```
<xs:complexType name="Tpersona">
```

```
  <xs:all>
```

```
    <xs:element name="nombre" type="xs:string"/>
```

```
    <xs:element name="apell" type="xs:string"/>
```

```
    <xs:element name="dni" type="Tdni" minOccurs="0"/>
```

```
  </xs:all>
```

```
</xs:complexType>
```

- Deben aparecer todos los valores, una única vez, pero en cualquier orden
- Se puede modificar número de ocurrencias, pero sólo 0 o 1 => solo permite minOccurs=0, maxOccurs=0 no tiene sentido, y 1 es el default.

Elemento vacío (y con atributo)

43

- No se indica nada del tipo del elemento

XML Schema empty types are defined implicitly, there is no explicit keyword for defining an empty type. if a type has no content model inside it, it is empty (it still may have attributes)

```
<xs:complexType name="Tbr">  
</xs:complexType>  
<xs:element name="br" type="Tbr"/>
```

- Puede llevar atributos

```
<xs:complexType name="Tpedido">  
  <xs:attribute name="codigo" type="xs:integer"/>  
</xs:complexType>  
<xs:element name="pedido" type="Tpedido"/>
```

+ tipos: fecha, hora..

44

- xs:boolean
 - Valores false/true
- xs:anyURI:
 - Para URL
- xs:time
 - "hh:mm:ss" (todos necesarios)
- xs:duration (intervalo de tiempo)
 - Formato: "PnYnMnDTnHnMnS"
 - Ejemplos:
 - <period>P5Y2M10D</period>
 - <period>P5Y2M10DT15H</period>
 - <period>PT15H</period>

+ tipos: numéricos

45

Numeric Data Types

Note that all of the data types below derive from the Decimal data type (except for decimal itself)!

Name	Description
byte	A signed 8-bit integer
decimal	A decimal value
int	A signed 32-bit integer
integer	An integer value
long	A signed 64-bit integer
negativeInteger	An integer containing only negative values (..,-2,-1)
nonNegativeInteger	An integer containing only non-negative values (0,1,2,..)
nonPositiveInteger	An integer containing only non-positive values (..,-2,-1,0)
positiveInteger	An integer containing only positive values (1,2,..)
short	A signed 16-bit integer
unsignedLong	An unsigned 64-bit integer
unsignedInt	An unsigned 32-bit integer
unsignedShort	An unsigned 16-bit integer
unsignedByte	An unsigned 8-bit integer

+ tipos: derivados de string (I)

46

- language
- NMTOKEN
 - letras, dígitos, punto, guión, subrayado y dos puntos.
- NMTOKENS
 - lista de NMTOKEN separados por espacio.
- NCName
 - similar al tipp name, pero sin “dos puntos”, nombre no colonizado.
 - empezar con letra o subrayado y puede contener letras, atributos,, subrayado, guión y punto.
 - similar al tipp name, pero sin “dos puntos”

+ tipos: derivados de string (II)

47

- ID => NCName con restricciones:
 - Se aplica a atributos
 - valores únicos en la instancia XML
 - y alguna restricción más que nos vemos
- IDREF, IDREFS...
 - se aplica a atributos
 - similar a su uso en DTDs

http://www.datypic.com/sc/xsd/t-xsd_string.html

Resumen

48

- Se aplican espacios de nombres: xs, xsi..
- Nueva forma de validar: schema
 - Ventajas/inconvenientes frente a DTD
 - XML !!
 - Tipos de datos
- Definir esquemas:
 - elementos, atributos, secuencias,..
 - Restricciones, extensiones, tipos, etc.

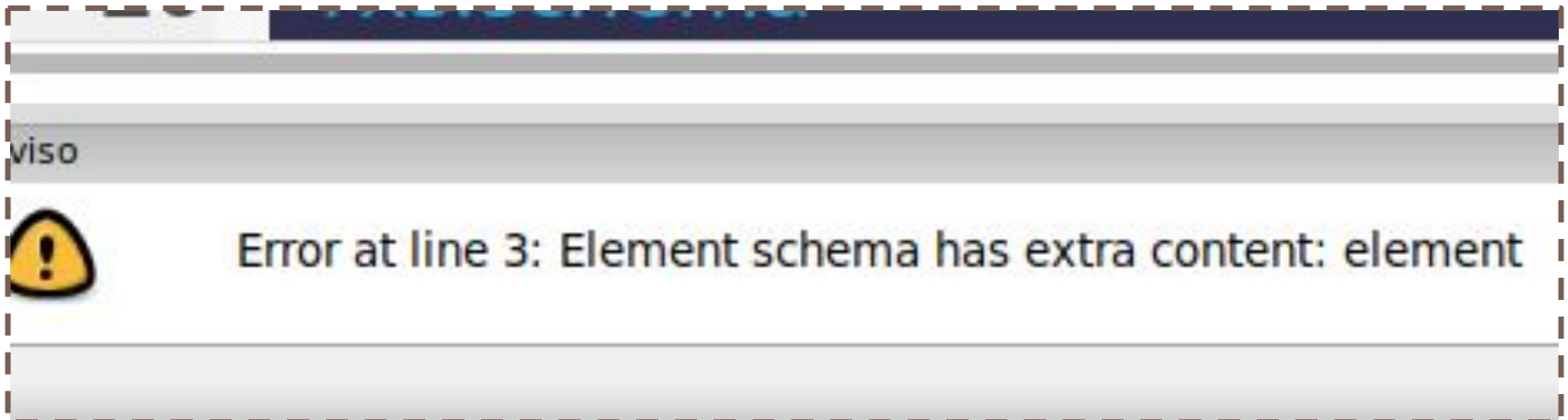
Aviso: si errores al validar XSD en editor un poco "crípticos" probar en CLI

```
xmlstarlet val -e -s f.xsd f.xml
```

```
xmlLint --noout --schema f.xsd f.xml
```


Salida xmlCE vs CLI

49



```
usuario@xlmint19$ xmlstarlet val -e -s t03-all.xsd t03-all.xml
t03-all.xsd:17.0: Element
'{http://www.w3.org/2001/XMLSchema}element': Invalid value for
maxOccurs (must be 0 or 1).
usuario@xlmint19$
```

Ejercicio tipo examen: examen.xml (00, mismo DTD)

50

Ejercicio 1: El ejercicio consiste en realizar tareas relacionadas con ficheros XML que almacenan la información de un proyecto.

- 1) examen.dtd: fichero utilizado para validar el fichero `examen.xml` (que deberá ser un fichero válido), y que además deberá cumplir:
 - La colección puede tener uno o más de un propietario
 - El precio de venta es opcional.
 - El número de películas puede ser cero, aunque el elemento películas debe existir siempre.
 - El título original es opcional, y puede tener un atributo de nombre lang con los valores en, es o fr.
 - El cartel es un elemento sin contenido que tiene un atributo fuente obligatorio.
 - Cada película tiene un atributo código que debe ser un identificador único.
 - La duración se mide en minutos
 - El director es uno y sólo uno.
 - El reparto está compuesto de al menos un actor.
 - La sinopsis tiene al menos un párrafo.

Ejercicio XSD tipo examen: proyecto.xml (01)

51

- Enunciado y fichero en plataforma
- proyecto: atributo lang es, en, fr, ge
- proyecto: 1 título, uno o más autores
- puede tener dedicatoria, y si la hay al menos 1 párrafo
- Fecha publicación válida
- Número de páginas entre 100 y 200.
- bibliografía obligatoria con al menos una referencia.
- Cada apartado: título, y número variable de secciones. Y un atributo obligatorio que no se puede repetir
- Cada sección tiene un título y cero o más párrafos.
- Párrafo tiene atributo estilo, con valores cita, codigo, revisar o normal. Valor por defecto normal.