



| | | | | | | |
|---|------------|--|--------|-----|-------|------------|
|  | RAMA: | Informática | CICLO: | DAM | | |
| | MÓDULO | Lenguajes de marcas y sistemas de gestión de información | | | | CURSO: 1 |
| | PROTOCOLO: | Apuntes clases | AVAL: | 3 | DATA: | 05/05/2020 |
| | UNIDAD | COMPETENCIA | XML | | | |

Índice

| | | |
|-------|---|----|
| 1. | Conceptos generales | 3 |
| 2. | XML modelo de datos..... | 3 |
| 2.1. | Raíz (root) | 3 |
| 2.2. | Elementos..... | 3 |
| 2.3. | Atributos..... | 3 |
| 2.4. | Elementos VS Atributos | 4 |
| 3. | Elementos especiales | 4 |
| 3.1. | Elemento raíz o root | 4 |
| 3.2. | Elementos vacíos | 4 |
| 3.3. | Texto | 5 |
| 3.4. | Comentarios | 5 |
| 3.5. | Espacios de nombres | 5 |
| 3.6. | Instrucciones de procesamiento | 6 |
| 3.7. | Entidades predefinidas | 6 |
| 3.8. | Secciones CDATA | 7 |
| 3.9. | Secciones DTD | 7 |
| 4. | Nombres XML | 7 |
| 4.1. | Atributos..... | 8 |
| 5. | Documento XML bien formado | 8 |
| 6. | Documento XML válido | 9 |
| 7. | XML Schema | 9 |
| 8. | Elementos básicos de un esquema XML | 10 |
| 8.1. | xs:schema | 10 |
| 8.2. | xs:element | 10 |
| 8.3. | xs:attribute | 10 |
| 9. | Tipos de datos predefinidos | 10 |
| 9.1. | Datos numéricos..... | 11 |
| 9.2. | Datos de fecha y hora..... | 12 |
| 9.3. | Datos de texto | 12 |
| 9.4. | Binarios..... | 12 |
| 9.5. | Booleanos | 13 |
| 10. | Elementos simples..... | 13 |
| 10.1. | Atributos..... | 13 |
| 11. | Elementos complejos | 14 |

| | | | | | | |
|---|------------|--|--------|-----|-------|------------|
|  | RAMA: | Informática | CICLO: | DAM | | |
| | MÓDULO | Lenguajes de marcas y sistemas de gestión de información | | | | CURSO: 1 |
| | PROTOCOLO: | Apuntes clases | AVAL: | 3 | DATA: | 05/05/2020 |
| | UNIDAD | COMPETENCIA | XML | | | |

| | | |
|---------|---|----|
| 11.1. | Definición de elementos complejos | 14 |
| 11.2. | Elementos vacíos | 15 |
| 11.3. | Elementos con atributos..... | 15 |
| 11.4. | Elementos con hijos y con valor | 15 |
| 11.4.1. | Número de veces que aparece un element hijo | 16 |
| 11.4.2. | xs:all, xs:choice, xs:any | 16 |
| 12. | Reutilización de componentes de un esquema..... | 17 |
| 13. | Definición de elementos de tipo mixto | 17 |
| 14. | Facetas o restricciones | 18 |
| 14.1. | Tipos de facetas más usuales | 18 |
| 15. | Patrones..... | 19 |
| 15.1. | Cuantificadores..... | 19 |
| 15.2. | Aparición literal | 20 |

| | | | | | | |
|------------------------------|--------------------|--|--------|-----|--------|------------|
| COLEXIO VIVAS S.L. | RAMA: | Informática | CICLO: | DAM | | |
| | MÓDULO | Lenguajes de marcas y sistemas de gestión de información | | | CURSO: | 1 |
| | PROTOCOLO: | Apuntes clases | AVAL: | 3 | DATA: | 05/05/2020 |
| | UNIDAD COMPETENCIA | XML | | | | |

1. Conceptos generales

XML (eXtensible Markup Language) es un estándar diseñado para almacenar y transportar datos diseñados para ser legibles tanto por humanos como por máquinas.

XML utiliza etiquetas definidas por el usuario, estas etiquetas y sus atributos son metainformación, esto significa información sobre información (sobre datos), esta información nos permite estructurar el documento y facilitar su procesamiento, pero no son información en sí mismos.

En un documento XML, la información se organiza de forma jerárquica, de modo que los elementos del documento se relacionan entre sí mediante relaciones, padre, hijo, hermano, ascendente, descendente, etc.

Ejemplo:

El elemento <persona> es "padre" y como descendientes tiene los elementos <nombre> y <apellido> que son hermanos entre ellos.

```
<persona>
  <nombre> María </nombre>
  <apellido>Pérez </apellido>
</persona>
```

2. XML modelo de datos

Un documento XML consta de una estructura formada por los siguientes tipos de componentes o nodos:

2.1. Raíz (root)

Es obligatorio que los documentos XML contengan un elemento raíz que sea el padre único de todos los demás elementos

```
<raíz>
  <hijo>
    <nieto>..... </nieto>
  </hijo>
</raíz>
```

2.2. Elementos

Es la unidad básica de un documento XML

El contenido se define entre dos etiquetas, es necesario su apertura y su cierre, a menos que no contengan contenido, pero siempre se cerrarán.

Ejemplo de elemento:

```
<etiqueta> Contenido </etiqueta>
```

2.3. Atributos

| | | | | | | |
|------------------------------|--------------------|--|--------|-----|-------|------------|
| COLEXIO VIVAS S.L. | RAMA: | Informática | CICLO: | DAM | | |
| | MÓDULO | Lenguajes de marcas y sistemas de gestión de información | | | | CURSO: 1 |
| | PROTOCOLO: | Apuntes clases | AVAL: | 3 | DATA: | 05/05/2020 |
| | UNIDAD COMPETENCIA | XML | | | | |

Al igual que los atributos HTML, son pares de nombre y valor que le permiten especificar datos adicionales para un elemento.

Están ubicados en la etiqueta de apertura del elemento y para asignar un valor a un atributo, se utiliza el signo igual. Todos los atributos se tratarán como texto y aparecerán entre comillas simples o dobles.

Ejemplos:

```
<distancia unidades="km">70</distancia>
<persona DNI="12345678A">...</persona>
```

Consideraciones:

Un número excesivo de atributos puede hacer que el documento XML sea menos legible, más difícil de mantener y difícil de extender.

Los atributos no pueden contener información en forma de árbol, es decir, no pueden contener otros elementos o atributos, pero sí contienen información accesible.

2.4. Elementos VS Atributos

Elementos:

- Representan jerarquías
- Se puede extender con otros elementos dentro
- El orden en que aparecen es representativo.
- Puede tener atributos
- Un elemento puede aparecer muchas veces

Atributos

- Están asociados con los elementos.
- Son modificadores de la información.
- El orden en que aparecen no es significativo
- No se puede extender con otros elementos dentro
- Un mismo atributo no puede aparecer varias veces dentro de un elemento.

3. Elementos especiales

3.1. Elemento raíz o root

Todo documento XML bien formado debe contener un único elemento raíz que contenga todos los demás (no tiene ascendentes ni hermanos).

3.2. Elementos vacíos

Son elementos que no tienen contenido entre etiquetas, o dicho de otra forma, que se pueden abrir y cerrar en una única etiqueta. Estos elementos pueden contener atributos.

Ejemplos:

```
<lápiz/>
```

| | | | | | | |
|------------------------------|------------|--|--------|-----|--------|------------|
| COLEXIO VIVAS S.L. | RAMA: | Informática | CICLO: | DAM | | |
| | MÓDULO | Lenguajes de marcas y sistemas de gestión de información | | | CURSO: | 1 |
| | PROTOCOLO: | Apuntes clases | AVAL: | 3 | DATA: | 05/05/2020 |
| | UNIDAD | COMPETENCIA | XML | | | |

<lápiz dureza="HB">

3.3. Texto

El texto como tipo de nodo que representa los datos de un documento XML sólo puede aparecer como contenido de un elemento, o como valor de un atributo.

Los espacios en blanco recibirán un tratamiento especial, existen 4 tipos de caracteres de espaciado en XML:

| | | | |
|------------------|----|------|----------------|
| Tabulador | \t | 	 | TAB |
| Nueva Línea | \n |
 | LF |
| Retorno de carro | \r |  | CR |
| Espacio | \s | | NO-BREAK SPACE |

Los espacios en blanco dentro del contenido textual de un elemento se mantienen como están.

Como valor de un atributo los espacios en blanco adyacentes se condensarán en uno solo.

| | | |
|-------------------------------------|---|------------------------------|
| <A> Dato | ≠ | <A>Dato |
| | = | |
| <A>más datos..... y más | = | <A>más datosy más |

Los espacios en blanco entre elementos se ignorarán

Ejemplo:

3.4. Comentarios

Al igual que en HTML, empiezan por caracteres <!-- y se cierran con los caracteres -->

Dentro de ellos se puede escribir cualquier signo menos el doble guión --

No pueden ubicarse dentro de una etiqueta de apertura ni en una etiqueta de cierre.

3.5. Espacios de nombres

Permite asignar nombres extendidos a los elementos para distinguir etiquetas cuando se mezclan documentos XML existentes y evitar conflictos de nombres.

Ejemplo:

Este XML contiene información de la tabla HTML:

```
<table>
<tr>
<td>Apples</td>
<td>Bananas</td>
</tr>
</table>
```

| | | | | | | |
|------------------------------|------------|--|--------|-----|--------|------------|
| COLEXIO VIVAS S.L. | RAMA: | Informática | CICLO: | DAM | | |
| | MÓDULO | Lenguajes de marcas y sistemas de gestión de información | | | CURSO: | 1 |
| | PROTOCOLO: | Apuntes clases | AVAL: | 3 | DATA: | 05/05/2020 |
| | UNIDAD | COMPETENCIA | XML | | | |

Este XML contiene información en inglés sobre una mesa (table):

```
<table>
  <name>African Coffee Table</name>
  <width>80</width>
  <length>120</length>
</table>
```

Si estos fragmentos XML se agregaran juntos, habría un conflicto de nombres. Ambos contienen un elemento <table>, pero los elementos tienen diferente contenido y significado.

Un usuario o una aplicación XML no sabrá cómo manejar estas diferencias, para evitarlo se utiliza un **prefijo**

Los conflictos de nombres en XML se pueden evitar fácilmente utilizando un prefijo de nombre.

Este XML contiene información sobre una tabla HTML y un mueble(table):

```
<h:table>
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>

<f:table>
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
```

En el ejemplo anterior, no habrá conflicto porque los dos elementos <table> tienen prefijos diferentes.

3.6. Instrucciones de procesamiento

Empiezan por <? y acaban por ?> No forman parte del contenido del documento XML Se utilizan para dar información a las aplicaciones que procesan el documento XML.

Ejemplo:

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

- version: Indica la versión de XML usada en el documento. Es obligatorio ponerlo, a no ser que sea un documento externo a otro que ya lo incluía.
- encoding: La forma en que se ha codificado el documento. Se puede poner cualquiera, y depende del parser el entender o no la codificación. Por defecto es UTF-8, aunque podrían ponerse otras, como UTF-16, US-ASCII, ISO-8859-1, etc..
- standalone: Indica si el documento va acompañado de un DTD("no"), o no lo necesita("yes"); no hay por qué ponerlo, porque luego se indica el DTD si se necesita.

3.7. Entidades predefinidas

En XML existen símbolos reservados del lenguaje, para poder representarlos es necesario usar su entidad predefinida:

| Entidad | Caracter |
|---------|----------|
| & | & |
| < | < |

| | | | | | | |
|------------------------------|------------|--|--------|-----|--------|------------|
| COLEXIO VIVAS S.L. | RAMA: | Informática | CICLO: | DAM | | |
| | MÓDULO | Lenguajes de marcas y sistemas de gestión de información | | | CURSO: | 1 |
| | PROTOCOLO: | Apuntes clases | AVAL: | 3 | DATA: | 05/05/2020 |
| | UNIDAD | COMPETENCIA | XML | | | |

| | |
|--------|---|
| > | > |
| ' | ' |
| " | " |

3.8. Secciones CDATA

Permiten marcar un texto para que no sea analizado por el procesador, la definición de estas secciones permite agilizar el análisis del documento.

Dentro de ellas se pueden introducir libremente caracteres como < y &

Sintaxis <![CDATA[contenido]]>

No pueden aparecer antes del elemento raíz ni después de su cierre. Y pueden contener cualquier carácter excepto el signo delimitador de final de sección CDATA]]>

3.9. Secciones DTD

Permite definir reglas que fuercen restricciones sobre la estructura de un documento XML.

Su existencia no es obligatoria, pero sirve para validar documentos XML.

Debe aparecer en la segunda línea del documento XML, entre la instrucción de procesamiento inicial y el elemento raíz.

Ejemplo:

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE document SYSTEM "elementos.dtd">
3  <!-- Esto es un comentario-->
4  <?xml-stylesheet type="text/css" href="elementos.css"?>
5  <elementos>
6      <elemento>
7          <nombre>Agua</nombre>
8          <color>Azul</color>
9      </elemento>
10     <elemento>
11         <nombre>Fuego</nombre>
12         <color>Rojo</color>
13     </elemento>
14 </elementos>
```

Inclusión DTD

Instrucción de procesamiento que vincula al documento XML una hoja de estilos llamada elementos.css

4. Nombres XML

Para definir correctamente los nombres de los elementos utilizaremos una serie de reglas. Un nombre XML:

- Puede empezar con una letra (con o sin tilde), de cualquier alfabeto, subrayado o dos puntos (Los dos puntos se desaconsejan ya que se reservan para el espacio de nombres).
- Los siguientes caracteres pueden ser letras, dígitos, subrayados, guiones bajos, comas y dos puntos.
- Los nombres que empiezan por XML en cualquier combinación de mayúsculas y minúsculas se reservan para estandarización.
- No puede contener ningún carácter de espaciado.
- No puede contener ningún carácter de puntuación que los ya citados como válidos, esto incluye comillas simples o dobles, signo de dólar acento circunflejo, signo de porcentaje, barra inclinada, punto y coma.

| | | | | | | |
|---------------------------------|------------|--|--------|-----|--------|------------|
| COLEXIO VIVAS S.L. | RAMA: | Informática | CICLO: | DAM | | |
| | MÓDULO | Lenguajes de marcas y sistemas de gestión de información | | | CURSO: | 1 |
| | PROTOCOLO: | Apuntes clases | AVAL: | 3 | DATA: | 05/05/2020 |
| | UNIDAD | COMPETENCIA | XML | | | |

Ejemplos:

Nombres incorrectos:

<O'Donnell> General </O'Donnell>

<fecha nacimiento>2013-12-09</fecha nacimiento>

<día/mes/año>2/02/2014</día/mes/año>

Nombres correctos:

<primerApellido>Pérez</primerApellido>

<número_seguridad_social>33-806016</número_seguridad_social>

<Ωδ∅>ΣΔΤ</Ωδ∅>

4.1. Atributos

Los elementos XML pueden tener atributos, al igual que HTML.

Los atributos están diseñados para contener datos relacionados con un elemento específico.

Los valores de los atributos siempre se deben citar. Se pueden usar comillas simples o dobles. Para el género de una persona, el elemento <persona> podría escribirse:

<persona género="femenino">

ó

<persona género='femenino'>

5. Documento XML bien formado

Un documento XML está bien formado cuando cumple las reglas establecidas por el W3C en las especificaciones para XML.

Estas reglas son numerosas y en ocasiones difíciles de entender, así que se citarán las más significativas:

- Un documento XML puede contener solo un elemento raíz. El elemento raíz de un documento xml es un elemento que está presente solo una vez en un documento xml y no aparece como un elemento hijo dentro de ningún otro elemento. El documento XML más simple y bien formado contendrá un único elemento raíz sin descendientes.
- En un documento XML bien formado todas las etiquetas tienen que tener cierre.

<nombre> María </ nombre>

<apellido> Pérez </apellido>

Los elementos vacíos deben cerrarse con />

<sombreros />

- La anidación de elementos entre sí en un documento XML debe ser adecuada.
<alfa><beta><gamma>...</gamma></beta> </alfa> es una forma correcta de anidar pero <alfa><beta> <gamma> ... </alfa> </beta> </gamma> no lo es.
- Los nombres de los elementos y atributos distinguen entre mayúsculas y minúsculas. <Persona> es diferente de <persona>
- El valor de los atributos debe aparecer entre comillas simples o dobles, pero ambos del mismo tipo.
- No puede haber dos atributos con el mismo nombre asociado con el mismo elemento.
- No se puede haber instrucciones de procesamiento o comentarios dentro de las etiquetas de apertura y cierre de los elementos.

| | | | | | | |
|------------------------------|------------|--|--------|-----|--------|------------|
| COLEXIO VIVAS S.L. | RAMA: | Informática | CICLO: | DAM | | |
| | MÓDULO | Lenguajes de marcas y sistemas de gestión de información | | | CURSO: | 1 |
| | PROTOCOLO: | Apuntes clases | AVAL: | 3 | DATA: | 05/05/2020 |
| | UNIDAD | COMPETENCIA | XML | | | |

- No puede haber nada antes de la instrucción de procesamiento `<? xml ...?>`
- No puede haber texto antes o después del elemento del documento.
- Los signos `<>` "y & no pueden aparecer en el contenido textual de elementos o atributos.

Si un documento XML está bien formado, se mostrará como un árbol de nodos en un navegador.

6. Documento XML válido

Un documento XML es válido si hay reglas de validación asociadas, por lo que el documento debe cumplir con estas reglas.

Todos los documentos XML válidos están bien formados, pero un documento bien formado puede no ser un documento válido.

Para la validación de documentos XML se utilizan DTDs, esquemas o combinaciones de los dos, dado que los DTDs están prácticamente en desuso y las restricciones de tiempo que tenemos, nos centraremos únicamente en los esquemas XML.

7. XML Schema

Los esquemas se definen como documentos XML, en un documento separado con extensión .xsd. Debe **incluirse una referencia** del archivo xsd en documentos XML basados en ese esquema.

Un ejemplo de una estructura de esquema XML para limitar el valor de un elemento simple llamado edad entre 0 y 100 sería:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="edad">
    <xs:simpleType>
      <xs:restriction base="xs:integer">
        <xs:minInclusive value="0"></xs:minInclusive>
        <xs:maxExclusive value="100"></xs:maxExclusive>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
</xs:schema>
```

Para vincular el esquema con el documento XML, usaremos `xmlns:xsi` y `xsi:noNamespaceSchemaLocation`, para el ejemplo anterior tendremos:

```
?xml version="1.0"?>

<edad xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:noNamespaceSchemaLocation="prueba.xsd"> 22</edad>
```

De forma general será algo como:

```
<etiqueta xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:noNamespaceSchemaLocation="nombredelesquema.xsd">
```

| | | | | | | |
|------------------------------|--------------------|--|--------|-----|-------|------------|
| COLEXIO VIVAS S.L. | RAMA: | Informática | CICLO: | DAM | | |
| | MÓDULO | Lenguajes de marcas y sistemas de gestión de información | | | | CURSO: 1 |
| | PROTOCOLO: | Apuntes clases | AVAL: | 3 | DATA: | 05/05/2020 |
| | UNIDAD COMPETENCIA | XML | | | | |

8. Elementos básicos de un esquema XML

Los componentes imprescindibles son:

- xs:schema
- xs:element
- xs:attribute

8.1. xs:schema

Elemento schema es el elemento raíz del documento en el que se define el esquema.

8.2. xs:element

Representa la existencia de un elemento en el documento XML. Sus atributos principales son:

- name
- ref
- type
- default
- fixed
- minOccurs
- maxOccurs

8.3. xs:attribute

Representa la existencia de un atributo de un elemento en el documento XML. Sus atributos principales son:

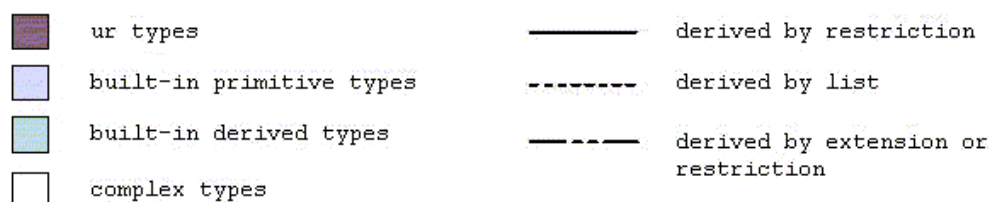
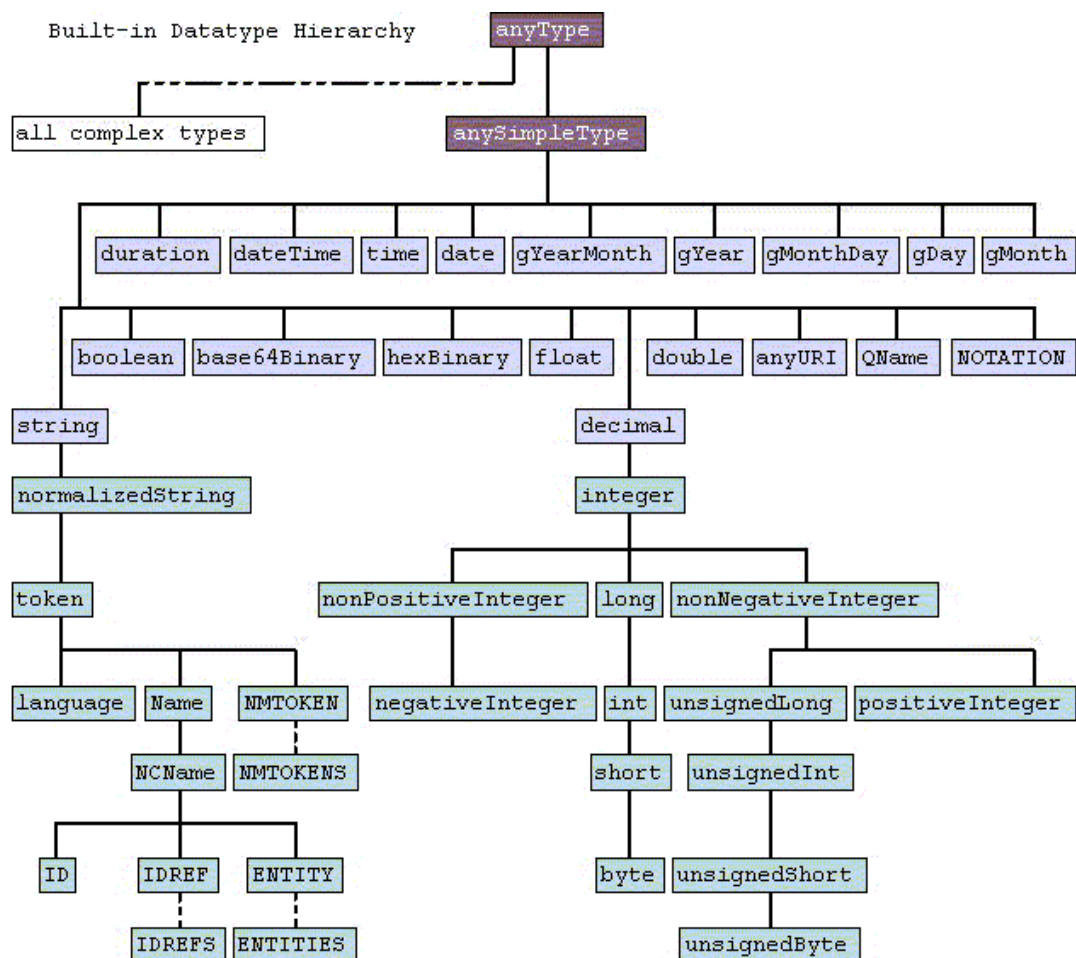
- name
- ref
- type
- use (optional | required | prohibited)
- default
- fixed

9. Tipos de datos predefinidos

Existen 44 tipos de datos predefinidos

Se organizan de forma de relación jerárquica de forma que cada tipo será igual que su tipo padre más alguna particularidad que lo distinga.

Existe un tipo predefinido especial xs:anyType que se encuentra en la raíz de la jerarquía de tipos y del que derivan todos los demás tipos.



9.1. Datos numéricos

| Tipos de datos | Descripción |
|--------------------|--|
| float | Número en punto flotante de precisión simple (32 bits) |
| double | Número en punto flotante de precisión doble (64 bits) |
| decimal | Números reales que pueden ser representados como potencias de 10 |
| integer | Números enteros (positivos y negativos) |
| nonPositiveInteger | Números negativos más el 0 |
| negativeInteger | Números enteros menores que 0 |
| nonNegativeInteger | Números enteros positivos más el 0 |
| positiveInteger | Números enteros mayores que 0 |
| unsignedLong | Números enteros positivos (64 bits de representación) |
| unsignedInt | Números enteros positivos (32 bits de representación) |

| | | | | | | |
|------------------------------|--------------------|--|--------|-----|--------|------------|
| COLEXIO VIVAS S.L. | RAMA: | Informática | CICLO: | DAM | | |
| | MÓDULO | Lenguajes de marcas y sistemas de gestión de información | | | CURSO: | 1 |
| | PROTOCOLO: | Apuntes clases | AVAL: | 3 | DATA: | 05/05/2020 |
| | UNIDAD COMPETENCIA | XML | | | | |

| | |
|---------------|---|
| unsignedShort | Números enteros positivos (16 bits de representación) |
| unsignedByte | Números enteros positivos (8 bits de representación) |
| long | Números enteros positivos y negativos representados con 64 bits |
| Int | Números enteros positivos y negativos representados con 32 bits |
| short | Números enteros positivos y negativos representados con 16 bits |
| byte | Números enteros positivos y negativos representados con 8 bits |

9.2. Datos de fecha y hora

| Tipo de datos | Descripción |
|---------------|---|
| duration | Duración en años+meses+días+horas+minutos+segundos |
| dateTime | Fecha y hora en formato aaaa-mm-dd T hh:mm:ss (Especificaciones norma ISO 8601) |
| date | Solamente fecha en formato aaaa-mm-dd |
| time | Solamente hora en formato hh:mm:ss |
| gDay | Día en formato dd (la g viene de Gregoriano) |
| gMonth | Mes |
| gYear | Año en formato aaaa |
| gYearMonth | Año y mes en formato aaaa-mm |
| gMonthDay | Mes y día en formato mm-dd |

9.3. Datos de texto

| Tipos de datos | Descripción |
|------------------|--|
| string | Cadenas de texto |
| normalizadString | Cadenas de texto en las que se convierten los caracteres tabulador, nueva línea y retorno de carro en espacios simples |
| token | Cadenas de texto sin los caracteres tabulador, nueva línea, retorno de carro ni espacios anteriores o posteriores |
| language | Valores válidos para xml:lang |
| NMTOKEN | Tipo de datos para atributo según XML 1.0, compatible con DTD |
| NMTOKENS | Lista separada por espacios de NMTOKEN, compatible con DTD |
| Name | Tipo de nombre según XML 1.0 |
| QName | Nombre cualificado del espacio de nombres XML |
| NCName | QName sin el prefijo ni los dos puntos |
| anyURI | Cualquier URI |
| ID | Tipo de datos para atributo según XML 1.0. Han de ser únicos, compatible con DTD |
| IDREF | Tipos de datos para atributo según XML 1.0, compatible con DTD |
| IDREFS | Lista separada por espacios de IDREF, compatible con DTD |
| ENTITY | Tipo de datos para atributo en XML 1.0, compatible con DTD |
| ENTITIES | Lista separada por espacios de ENTITY, compatible con DTD |
| NOTATION | Tipo de datos para atributo en XML 1.0, compatible con DTD |

9.4. Binarios

| Tipo de datos | Descripción |
|---------------|------------------------------------|
| hexBinary | Secuencia de dígitos hexadecimales |
| base64Binary | Secuencia de dígitos en base 64 |

| | | | | | | |
|-------------------------------|------------|--|--------|-----|--------|------------|
| COLEXIO VIVAS .S.L. | RAMA: | Informática | CICLO: | DAM | | |
| | MÓDULO | Lenguajes de marcas y sistemas de gestión de información | | | CURSO: | 1 |
| | PROTOCOLO: | Apuntes clases | AVAL: | 3 | DATA: | 05/05/2020 |
| | UNIDAD | COMPETENCIA | XML | | | |

9.5. Booleanos

| Tipo de datos | Descripción |
|---------------|---|
| boolean | Puede tener 4 valores: 0,1,true y false |

10. Elementos simples

Es un elemento que sólo puede contener texto (cualquier tipo de dato), pero no a otros elementos ni atributos. Su sintaxis es:

```
<xs:element name="xxx" type="yyy"/>
```

Ejemplos

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

Los tipos de datos más utilizados son:

- xs:string
- xs:decimal
- xs:float
- xs:integer
- xs:boolean
- xs:date
- xs:time

Un elemento simple puede tener un valor por defecto y un valor “fijo”. Esto se indica mediante los atributos default y fixed.

Ejemplo:

```
<xs:element name="color" type="xs:string" default="red"/>
<xs:element name="iva" type="xs:float" fixed="0.23"/>
```

10.1. Atributos

Los atributos se declaran de forma similar a los “elementos simples”. Un atributo se declara de la siguiente forma:

```
<xs:attribute name="xxx" type="yyy"/>
```


Ejemplo:

```
<xs:attribute name="idioma" type="xs:string"/>
```

Si un elemento simple va acompañado de atributos, el elemento se deberá declarar como un elemento “complejo”

Los atributos llevan asociados un tipo de datos, los más habituales son:

- xs:string

| | | | | | | |
|---|--------------------|--|--------|-----|-------|------------|
|  | RAMA: | Informática | CICLO: | DAM | | |
| | MÓDULO | Lenguajes de marcas y sistemas de gestión de información | | | | CURSO: 1 |
| | PROTOCOLO: | Apuntes clases | AVAL: | 3 | DATA: | 05/05/2020 |
| | UNIDAD COMPETENCIA | XML | | | | |

- xs:decimal
- xs:integer
- xs:boolean
- xs:date
- xs:time

Los atributos pueden tener valores por defecto y valores fijos:

```
<xs:attribute name="idioma" type="xs:string" default="ES"/>
```

Para indicar que un atributo debe ser obligatorio, se debe añadir a su declaración "use"

```
<xs:attributename="lang" type="xs:string" use="required"/>
```

El atributo use puede tomar el valor "optional" si el atributo no es obligatorio (opción por defecto).

11. Elementos complejos

Los elementos complejos contienen a otros elementos hijos, o que tienen atributos. Se dividen en 4 tipos:

- Elementos vacíos
- Elementos no vacíos con atributos
- Elementos con elementos hijos
- Elementos con elementos hijos y con "texto" o valor propio.

Ejemplos de elementos complejos:

- Elemento vacío:

```
<productpid="1345"/>
```

- Elemento con atributos:

```
<comida tipo="postre">Choco bomba</postre>
```

- Elemento con elemento hijos:

```
<description>Sucedió el <date>03.03.99</date> ....</description>
```

- Elemento con hijos y con valor

```
<empleado>
  <nombre>Juan </nombre>
  <apellido>Pérez</apellido>
</empleado>
```

11.1. Definición de elementos complejos

Para declarar un elemento complejo, utilizaremos la secuencia:

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

Dentro de esta definición podremos utilizar tres tipos de declaración para los elementos hijos:

- **xs:sequence** indica que los elementos anidados en él deben aparecer en un **orden** determinado.

| | | | | | | |
|-------------------------------|------------|--|--------|-----|--------|------------|
| COLEXIO VIVAS .S.L. | RAMA: | Informática | CICLO: | DAM | | |
| | MÓDULO | Lenguajes de marcas y sistemas de gestión de información | | | CURSO: | 1 |
| | PROTOCOLO: | Apuntes clases | AVAL: | 3 | DATA: | 05/05/2020 |
| | UNIDAD | COMPETENCIA | XML | | | |

- **xs:all** Indica que los elementos que contiene pueden aparecer en cualquier orden, pero **como máximo sólo una vez**.
- **xs:choice** indica que **sólo puede aparecer uno** de los elementos que contiene. (Sería análogo a un select sin opción múltiple).

Además, podemos utilizar el modelo de contenido ANY, que permite incluir elementos no declarados inicialmente en el esquema.

A continuación veremos varios ejemplos de definición de elementos complejos.

11.2. Elementos vacíos

Para declarar un elemento vacío con atributos, utilizaremos el siguiente la siguiente sintaxis:

Ejemplo: `<producto num="1345" />`

```
<xs:element name="producto">
  <xs:complexType>
    <xs:attribute name="num" type="xs:positiveInteger"/>
  </xs:complexType>
</xs:element>
```

11.3. Elementos con atributos

Para declarar un elemento no vacío con atributos, y sin elementos hijos, se utilizará la siguiente sintaxis:

Ejemplo `<moneda pais="ESP">Euro</moneda>`

```
<xs:element name="moneda">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:integer">
        <xs:attribute name="pais" type="xs:string" />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

11.4. Elementos con hijos y con valor

Para el caso de empleado tendríamos:

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

| | | | | | | |
|-------------------------------|------------|--|--------|-----|--------|------------|
| COLEXIO VIVAS .S.L. | RAMA: | Informática | CICLO: | DAM | | |
| | MÓDULO | Lenguajes de marcas y sistemas de gestión de información | | | CURSO: | 1 |
| | PROTOCOLO: | Apuntes clases | AVAL: | 3 | DATA: | 05/05/2020 |
| | UNIDAD | COMPETENCIA | XML | | | |

```

    <xs:element name="apellido" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
</xs:element>

```

11.4.1. Número de veces que aparece un element hijo

minOccurs, maxOccurs se utilizan para indicar el número máximo y mínimo de veces que puede aparecer un elemento hijo de un elemento complejo.

El atributo maxOccurs puede tomar el valor “unbounded”, que indica que no existe ningún límite.

11.4.2. xs:all, xs:choice, xs:any

xs:all

Nombre y apellido pueden aparecer en cualquier orden pero como máximo una vez.

```

<xs:element name="persona">
  <xs:complexType>
    <xs:all>
      <xs:element name="nombre" type="xs:string"/>
      <xs:element name="apellido" type="xs:string"/>
    </xs:all>
  </xs:complexType>
</xs:element>

```

xs:choice

En persona sólo puede aparecer o nombre o apellido

```

<xs:element name="persona">
  <xs:complexType>
    <xs:choice>
      <xs:element name="nombre" type="xs:string"/>
      <xs:element name="apellido" type="xs:string"/>
    </xs:choice>
  </xs:complexType>
</xs:element>

```

xs:any

```

<xs:element name="persona">
  <xs:complexType>
    <xs:sequence>

```


| | | | | | | |
|-------------------------------|--------------------|--|--------|-----|--------|------------|
| COLEXIO VIVAS .S.L. | RAMA: | Informática | CICLO: | DAM | | |
| | MÓDULO | Lenguajes de marcas y sistemas de gestión de información | | | CURSO: | 1 |
| | PROTOCOLO: | Apuntes clases | AVAL: | 3 | DATA: | 05/05/2020 |
| | UNIDAD COMPETENCIA | XML | | | | |

```

<xs:element name="nombre" type="xs:string"/>
<xs:element name="apellidos" type="xs:string"/>
<xs:any minOccurs="1"/>
</xs:sequence>
</xs:complexType>
</xs:element>

```

12. Reutilización de componentes de un esquema

Si queremos reutilizar la definición de los elementos hijos en varios elementos padre, podremos hacerlo definiendo un “tipo” para estos elementos.

Ejemplo.

Definición del tipo “**personinfo**” para definir los hijos nombre y apellido dentro de los elementos empleado y estudiante.

```

<xs:element name="empleado" type="personinfo"/>
<xs:element name="estudiante" type="personinfo"/>
<xs:complexType name="personinfo">
  <xs:sequence>
    <xs:element name="nombre" type="xs:string"/>
    <xs:element name="apellido" type="xs:string"/>
  </xs:sequence>
</xs:complexType>

```

13. Definición de elementos de tipo mixto

Para declarar un elemento con contenido “mixto”, basta con añadir un atributo “mixed” al elemento `xs:complexType`

Ejemplo:

```

<carta>Estimado cliente:<nombre>Juan Pérez</nombre>. Su pedido número
<orderid>1032</orderid> se enviará el día
<fechaenvio>2013-07-13</fechaenvio>.</carta>

```

Esquema:

```

<xs:element name="carta">
  <xs:complexType mixed="true">
    <xs:sequence>
      <xs:element name="nombre" type="xs:string"/>
      <xs:element name="orderid" type="xs:positiveInteger"/>
      <xs:element name="fechaenvio" type="xs:date"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

| | | | | | | |
|------------------------------|------------|--|--------|-----|--------|------------|
| COLEXIO VIVAS S.L. | RAMA: | Informática | CICLO: | DAM | | |
| | MÓDULO | Lenguajes de marcas y sistemas de gestión de información | | | CURSO: | 1 |
| | PROTOCOLO: | Apuntes clases | AVAL: | 3 | DATA: | 05/05/2020 |
| | UNIDAD | COMPETENCIA | XML | | | |

14. Facetas o restricciones

Las facetas o restricciones permiten restringir el valor que se puede dar a un elemento o atributo XML, así podemos indicar que un valor debe estar comprendido en un rango determinado, que debe ser un valor de una lista de valores “cerrada”, que debe ser mayor o menor que otro valor, que debe tener una determinada longitud...

Al definir una faceta o restricción tenemos:

- Atributos obligatorios:
base: nombre del tipo base a partir del cual se construirá el nuevo tipo.
- Atributos optativos:
id: identificador único del componente.

Ejemplo de restricción:

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

14.1. Tipos de facetas más usuales

| | |
|-----------------------------|---|
| enumeration | Establece una lista de valores “aceptados” |
| fractionDigits | Número de cifras decimales |
| length | Número de caracteres obligatorios |
| maxExclusive y maxInclusive | Valor máximo en un rango |
| minExclusive y minInclusive | Valor mínimo en un rango |
| maxLength y minLength | Número máximo y mínimo de caracteres permitidos |
| pattern | Define una secuencia de caracteres permitida |
| totalDigits | Número exacto de dígitos permitidos |
| whiteSpace | Indica cómo se deben de tratar los espacios en blanco |

Ejemplo: Realiza un esquema XML para un elemento llamado <edadLaboral> que será un entero no negativo de valor mínimo incluido 16 y valor máximo no incluido 70.

Solución 1.

```
<xs:element name="edadLaboral">
  <xs:simpleType>
    <xs:restriction base="xs:nonNegativeInteger">
      <xs:minInclusive value="16"/>
      <xs:maxExclusive value="70"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

| | | | | | | |
|-------------------------------|--------------------|--|--------|-----|-------|------------|
| COLEXIO VIVAS .S.L. | RAMA: | Informática | CICLO: | DAM | | |
| | MÓDULO | Lenguajes de marcas y sistemas de gestión de información | | | | CURSO: 1 |
| | PROTOCOLO: | Apuntes clases | AVAL: | 3 | DATA: | 05/05/2020 |
| | UNIDAD COMPETENCIA | XML | | | | |

```

</xs:restriction>
</xs:simpleType>
</xs:element>

```

Solución 2.

Si quisiéramos definir un tipo de datos al que poder referenciar repetidas veces y declarar un elemento <edadLaboral> de ese tipo, sería:

```

<xs:simpleType name="TipoedadLaboral">
  <xs:restriction base="xs:nonNegativeInteger">
    <xs:minInclusive value="16"/>
    <xs:maxExclusive value="70"/>
  </xs:restriction>
</xs:simpleType>

<xs:element name="edadLaboral" type="TipoEdadLaboral">

```


15. Patrones

Los patrones sirven para marcar normas de comportamiento dentro del esquema XML, a continuación, se detallan los de más uso:

| Patrón | Significado | Ejemplo |
|-----------|---|---------|
| . | Cualquier caracter | ; |
| \w | Cualquier letra | M |
| \d | Un dígito | 7 |
| \D | Cualquier carácter no-dígito | j |
| \s | Cualquier carácter de espaciado (tabulador, espacio...) | |
| \S | Cualquier carácter distinto a espacio | S |
| \d{n} | n dígitos exactamente (Ej \d{4}) | 1234 |
| \d{n,m} | De n o m dígitos (Ej \d{2,3}) | 98 |
| \d{n,} | n o más dígitos (Ej \d{3,}) | 3500 |
| [xyz] | uno de los tres caracteres x, y o z (en minúsculas) | z |
| [A-Z] | Uno de los caracteres de la A a la Z (en mayúsculas) | D |
| [^abc] | Negación de un grupo de caracteres | d |
| [F-J-[H]] | Sustracción de un carácter de un rango | G |
| (a b) | a ó b | b |
| b? | Sucesión de 0 o una ocurrencia de una cadena | b |
| 1* | Sucesión de 0 o más ocurrencias de una cadena | 111 |
| (cd)+ | Sucesión de 1 o más ocurrencias de una cadena | cdcdcd |

15.1. Cuantificadores

| Cuantificador | Significado |
|---------------|---------------------|
| ? | 0 o 1 ocurrencia |
| * | 0 o más ocurrencias |

| | | | | | | |
|---|------------|--|--------|-----|-------|------------|
|  | RAMA: | Informática | CICLO: | DAM | | |
| | MÓDULO | Lenguajes de marcas y sistemas de gestión de información | | | | CURSO: 1 |
| | PROTOCOLO: | Apuntes clases | AVAL: | 3 | DATA: | 05/05/2020 |
| | UNIDAD | COMPETENCIA | XML | | | |

| | |
|-------|-----------------------------------|
| + | 1 o más ocurrencias |
| {n} | n ocurrencias exactas |
| {n,m} | De n a m ocurrencias (siendo n<m) |
| {n,} | n o más ocurrencias |

15.2. Aparición literal

Se antepone el carácter de

escape \ Ejemplo:

[ab\+] → a ó b ó el carácter +