

UNIDAD 5

XML: creación, validación y utilización

Objetivos

- Identificar las tecnologías relacionadas con la definición de documentos XML.
- Crear descripciones de documentos XML.
- Utilizar descripciones en la elaboración y validación de documentos XML.
- Utilizar herramientas para definición y validación de documentos.

```
<?xml version="1.0"?>
<quiz>
  <qanda seq="1">
    <question>
      Who was the forty-second
      president of the U.S.A.?
    </question>
    <answer>
      William Jefferson Clinton
    </answer>
  </qanda>
  <!-- Note: We need to add
  more questions later.-->
</quiz>
```

XML

Índice

1. Introducción
2. Estructura y sintaxis de XML
3. Validación de XML
4. XML aplicado

5.1 Introducción

eXtensible Markup Language

- Es el estándar para definir la estructura interna de documentos
- Es un metalenguaje empleado para definir otros lenguajes
- Desarrollado por W3C (World Wide Web Consortium) desde 1998
- Proviene de SGML (Standard Generalized Markup Language)
- Permite definir el sentido o semántica de los datos.
- Utiliza etiquetas (marcas) para representar las partes del documento.

<CAPITULO>, </CAPITULO>, ...

5.1 Introducción

Ejemplo:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<animal>
  <nombre>Tiburón blanco</nombre>
  <nombrecientifico>Carcharodon carcharias</nombrecientifico>
  <familia>Lamnidae</familia>
  <estadodeconservacion>Vulnerable</estadodeconservacion>
</animal>
```

5.2 Estructura y Sintaxis de XML

Los elementos de un documento XML deben seguir una estructura de “árbol” (estrictamente jerárquica). Y está compuesto por dos partes fundamentales:

- **Prólogo:** Donde se declara la versión XML y la codificación empleada para representar los caracteres. Es opcional. Si existe, debe ser la primera línea del documento. En él se indica:
 - ✓ Versión de XML
 - ✓ Encoding: Codificación utilizada en el documento
 - ✓ standalone.: indica si el documento es independiente o si existe un DTD externo para la validación.
- **Cuerpo.** Recoge los elementos con la información. Los elementos deben estar correctamente anidados. Todo el contenido del documento debe de estar incluido en el llamado elemento **raíz**. En el siguiente ejemplo el elemento cine es el elemento raíz.

prólogo →

cuerpo

```
<?xml version="1.0" encoding="iso-8859-1"?>
<cine>
  <película>
    <director>Señor X</director>
    <duración>60</duración>
    <título> Las aventuras de los SGML</título>
  </película>
</cine>
```

5.2 Estructura y Sintaxis de XML

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

```
<libro>
```

```
  <autor>El ingenioso...</autor>
```

```
  <titulo>Miguel de Cervantes</titulo>
```

```
  <capitulo number="1">
```

```
    <titulo>Que trata...</titulo>
```

```
    <parrafo>En un lugar...</parrafo>
```

```
    ...
```

```
  </capitulo>
```

```
  <capitulo number="2">...</capitulo>
```

```
  ...
```

```
</libro>
```

**Existe un elemento raíz
que engloba todo el
documento**

5.2 Estructura y Sintaxis de XML

El cuerpo está compuesto por elementos. Un elemento es un grupo formado por una etiqueta de apertura, otra de cierre y el contenido que hay entre ambas. La distribución de los elementos está jerarquizada según una estructura de árbol, lo que implica que es posible anidarlos pero no entrelazarlos.

Las etiquetas de apertura tienen un nombre y , opcionalmente, uno o más atributos separados por espacios. La etiqueta de cierre debe tener el mismo nombre que la de apertura, pero este estará precedido por el símbolo /.

Las **reglas para la asignación de nombres** de elemento son las siguientes:

- Diferencian entre mayúsculas y minúsculas
- Deben comenzar con un letra o un guion bajo
- Los nombres de elementos tienen que ser idénticos en la etiquetas de apertura y de cierre
- Los nombres pueden estar formados por caracteres alfanuméricos, guiones, guiones bajos y puntos
- Los nombres no pueden contener espacios(interpreta como nombre del elemento la primera palabra y el resto como parámetros).

5.2 Estructura y Sintaxis de XML

Los nombres de las etiquetas deben empezar por una letra o _

No pueden empezar por XML (o variantes)

Pueden contener letras, números, -, _ y .

No pueden contener espacios

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

```
<libro>
```

```
<autor>El ingenioso...</autor>
```

```
<titulo>Miguel de Cervantes</titulo>
```

```
<capitulo number="1">
```

```
<titulo>Que trata...</titulo>
```

```
<parrafo>En un lugar...</parrafo>
```

```
...
```

```
</capitulo>
```

```
<capitulo number="2">...</capitulo>
```

```
...
```

```
</libro>
```

Especificación de que es un documento XML y versión

Cada elemento XML tiene una marca de apertura y otra de cierre, la misma con /

Una etiqueta que no tenga contenido
<etq></etq>

Puede expresarse como:
<etq/>

5.2 Estructura y Sintaxis de XML

Las **reglas de creación de atributos**:

- Deben tener asignado un valor
- Los valores siempre van entrecomillados, admitiendo comillas simples y dobles(deben coincidir el tipo de comilla de cierre con el de apertura).
- Diferencian entre mayúsculas y minúsculas.
- Pueden comenzar con una letra o un guion bajo
- Pueden estar formados por caracteres alfanuméricos, guiones , guiones bajos o puntos.

Ejemplo de atributo válido:

```
<nombre familia="Lamnidae">Tiburón blanco</nombre>
```

5.2 Estructura y Sintaxis de XML

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

```
<libro>
```

```
  <autor>El ingenioso...</autor>
```

```
  <titulo>Miguel de Cervantes</titulo>
```

```
  <capitulo number="1">
```

```
    <titulo>Que trata...</titulo>
```

```
    <parrafo>En un lugar...</parrafo>
```

```
    ...
```

```
  </capitulo>
```

```
  <capitulo number="2">...</capitulo>
```

```
  ...
```

```
</libro>
```

Los elementos XML pueden tener atributos y valores asociados a los atributos. Siempre se entrecomillan

Los identificadores de las marcas, y los atributos de éstas y sus valores, son sensibles a mayúsculas y minúsculas

5.2 Estructura y Sintaxis de XML

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

```
<!--Ejemplo uso de comentarios.-->
```

```
<a>
```

```
<b>
```

```
<d cantidad="2">dd</d>
```

```
</b>
```

```
<e>
```

```
<f cantidad="8">ffffff</f>
```

```
<!--g puede aparecer varias veces.-->
```

```
<g cantidad="5">ggggg</g>
```

```
<g cantidad="2">gg</g>
```

```
</e>
```

```
</a>
```

```
<c cantidad="4">cccc</c>
```

**Se pueden incluir
comentarios**



5.2 Estructura y Sintaxis de XML

Ejemplo de un documento XML:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE document system "Elementos.dtd">
<!-- Aquí viene un comentario -->
<?xml-stylesheet type="text/css" href="Elementos.css"?>
<elemento>
  <nombre>Agua</nombre>
  <color>Azul</color>
</elemento>
<elemento>
  <nombre>Fuego</nombre>
  <color>Rojo</color>
</elemento>
</elementos>
```

Instrucción de procesamiento que indica que el documento es XML

Inclusión de un DTD externo

Un comentario

Instrucción de procesamiento que vincula al documento una hoja de estilo CSS

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE document system "Elementos.dtd">
<!-- Aquí viene un comentario -->
<?xml-stylesheet type="text/css" href="Elementos.css"?>
<elementos>
  <elemento>
    <nombre>Agua</nombre>
    <color>Azul</color>
  </elemento>
  <elemento>
    <nombre>Fuego</nombre>
    <color>Rojo</color>
  </elemento>
</elementos>
```

Elemento Raíz

Elementos

5.2.1 Elementos de un documento XML

En XML se puede caracterizar un elemento mediante atributos o elementos hijos.

No existe una regla para determinar en qué caso se deben utilizar uno u otro, pero sí casuísticas en las que los atributos no son suficientes para crear la estructura de datos que se desea.

Los elementos pueden contener texto u otros elementos.

Un elemento puede contener a otros elementos y estos, a su vez, ser contenedores de otros más . Esta estructura se denomina de árbol

5.2.1 Elementos de un documento XML

Criterios para decidir si un dato del documento que se pretende estructurar ha de representarse mediante un elemento o un atributo:

El dato **será un elemento si** cumple alguna de las siguientes condiciones:

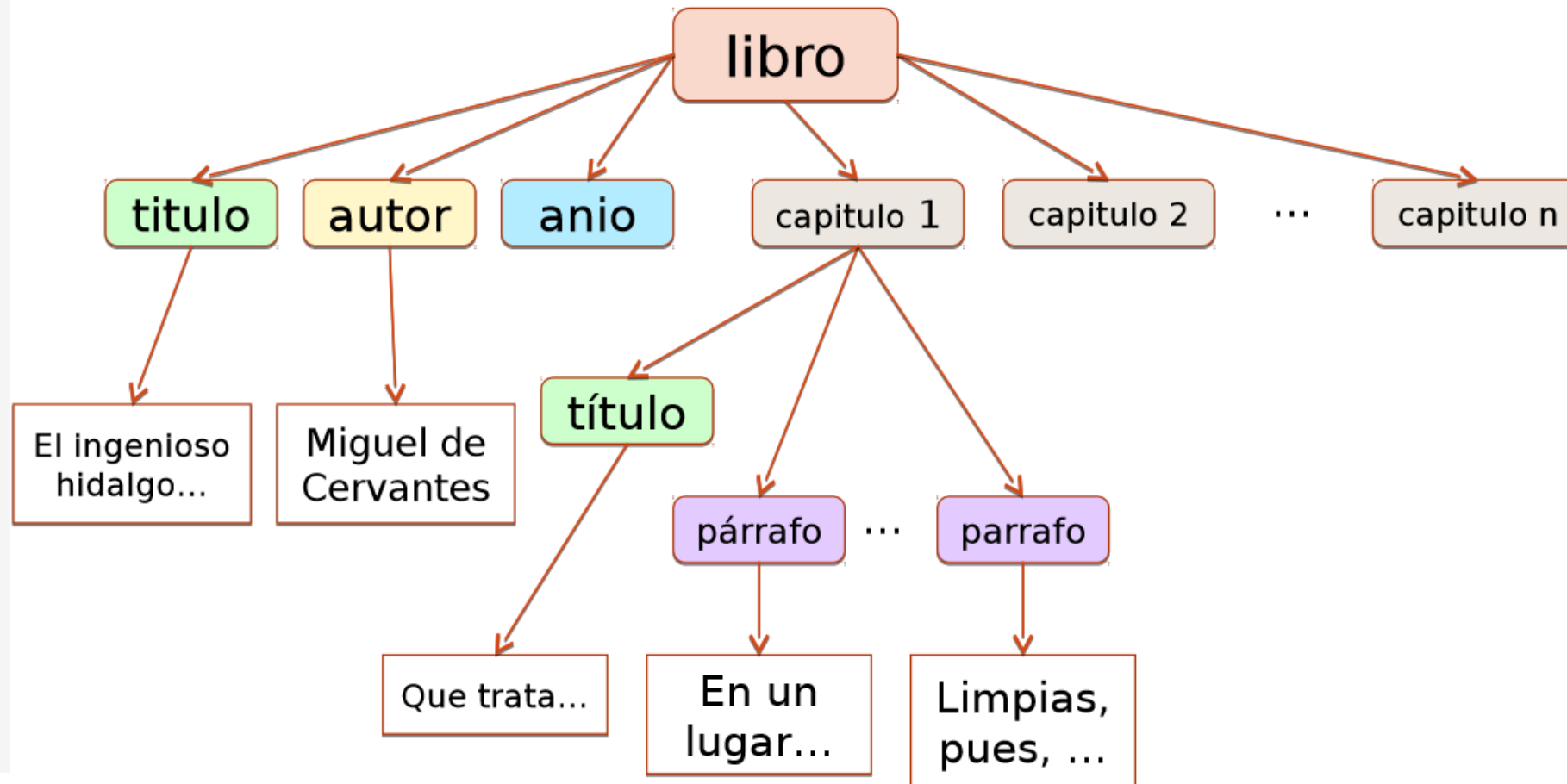
- Contiene subestructuras.
- Es de un tamaño considerable.
- Su valor cambia frecuentemente.
- Su valor va a ser mostrado a un usuario o aplicación.

Los casos en los que el dato **será un atributo** son:

- El dato es de pequeño tamaño y su valor raramente cambia, aunque hay situaciones en las que este caso puede ser un elemento.
- El dato solo puede tener unos cuantos valores fijos.
- El dato guía el procesamiento XML pero no se va a mostrar.

5.2.2 Estructura jerárquica de la información

Por su naturaleza, presentan una representación en árbol



5.2.2 Estructura jerárquica de la información

Las relaciones entre los elementos de un documento XML se denominan de la siguiente manera:

- Existe un único elemento raíz
- Cada uno de los elementos descendientes directos de un elemento se llama hijo
- El elemento ascendiente directo de un elemento se llama padre
- Los elementos que tienen un padre común se denominan hermanos

5.2.3 Comentarios

Igual que en HTML se ubican en cualquier lugar del documento, pero nunca dentro de una etiqueta de apertura o cierre, y por tanto tampoco dentro de un atributo

Se escriben para ser leídos por personas, por lo que el contenido no tiene que cumplir en ninguna sintaxis ni regla.

No son procesados, ni validados, ni participan de la estructura de datos que representa el documento.

5.2.4. Espacios de nombres

Los espacios de nombres son mecanismos que se usan para diferenciar etiquetas que se llaman igual cuando empleamos varios documentos XML.

Agrupan todos los elementos y atributos relacionados de una aplicación XML para que el software pueda reconocerlos con facilidad.

¿Cómo se declaran?

xmlns:"URI_namespace"

¿Y si se usa un prefijo que nos informe sobre cuál es el vocabulario al que está asociada esa definición?

xmlns:prefijo="URI_namespace"

En ambos casos URI_namespace es la localización del conjunto del vocabulario del espacio de nombres al que se hace referencia.

5.2.4. Espacios de nombres

- El ámbito de un espacio de nombres incluye el elemento donde se ha declarado y sus elementos descendientes.
- Dos atributos o elementos con el mismo nombre pero distinto espacio de nombres, son diferentes.
- Existe un espacio de nombres por defecto que es aquel al que no se le asigna un prefijo. El ámbito de ese espacio de nombres es el elemento en el que se ha declarado y sus descendientes, **pero no sus atributos**.

Ejemplo:

```
<bancos xmlns:empresa="http://myxml.org/empresas" xmlns:mobiliario="http://myxml.org/mobiliario">
  <empresa:banco>
    <nombre></nombre>
    <muebles>
      <silla cantidad="5"/>
      <mobiliario:banco cantidad="4" plazas="3"/>
    </muebles>
  </empresa:banco>
</bancos>
```

5.2.5 Entidades

Son un mecanismo para representar información dentro de un documento haciendo referencia a ella en lugar de incluirla directamente. La documentación puede ser desde un carácter predefinido hasta un fichero externo. Gracias a ellas se puede repartir el contenido de los documentos en varias fuentes.

Las entidades se definen en el DTD. Al declarar una entidad se asigna un alias a un bloque de texto. Ese alias se puede adjuntar a un documento XML para incrustar el bloque correspondiente.

Por ejemplo, es posible crear una entidad que almacene la ficha de una empresa(nombre, dirección y CIF) e incrustarla en varios documentos sin necesidad de escribir el mismo texto una y otra vez. En caso de que la empresa cambie de dirección, solo habrá que cambiarla una vez. Las entidades nos permiten definir constantes en un documento XML.

Cuando se usan dentro del documento XML se limitan por "&" y ";", por ejemplo & entidad;

¿Cómo trabaja el intérprete con ellos?

Al procesar el documento XML, el intérprete sustituye la entidad por el valor que se le ha asociado en el DTD. No admiten recursividad, es decir, una entidad no puede hacer referencia a ella misma. Para definir una entidad en un DTD se usa el elemento

5.2.5 Entidades

Tipos de entidades:

- **Generales:** es un alias que se convierte en parte del documento XML. Se dividen a su vez en:
 - **Internas:** Mantiene sus valores en la propia declaración de la entidad.
Existen cinco entidades internas predefinidas de uso común que representan caracteres que provocarían ambigüedad en el procesamiento de XML:
 - `<`; símbolo de menor que (<)
 - `>`; símbolo de mayor que (>)
 - `&`; símbolo de ampersand (&)
 - `"`; comilla doble (")
 - `'`; comilla simple (')
 - **Externas:** Mantienen los valores en un archivo externo
- **De parámetros:** Permite dar nombres a partes de un DTD y hacer referencia a ellas a lo largo del mismo. Son especialmente útiles cuando varios elementos del DTD comparten listas de atributos o especificaciones de contenidos. Se denotan por % entidad;

5.2.6 Secciones CDATA

Son conjuntos de caracteres que el procesador no debe analizar.

Se utilizan para almacenar código XML o HTML que, de otra forma, impediría que el documento XML contenedor no estuviese bien formado. De esta forma podemos introducir caracteres especiales.

No pueden aparecer antes del elemento raíz, ni después de su cierre.

En una sección CDAT no es necesario sustituir los caracteres conflictivos por entidades, ya que son ignorados por el analizador. Si se incluyen etiquetas dentro de una sección CDATA, no serán tratadas como tal, sino como una sucesión de caracteres sin ningún sentido específico.

Ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<ficha>
  <nombre>Lenguaje C</nombre>
  <info.html>
    <![CDATA[
      <h1>LENGUAJES DE PROGRAMACIÓN</h1>
      <hr>
      <a href="enlace">C</a>
    ]]>
  </info.html>
</ficha>
```

5.3 Validación de XML

XML permite determinar las reglas que debe cumplir un determinado documento escrito en este lenguaje.

Estas reglas, junto con los correspondientes procesadores de XML, permiten garantizar que un documento y los datos almacenados en él están correctamente contruidos

5.3.1. XML bien formado y válido

- Un documento XML está bien formado cuando no tiene errores de sintaxis
- Un documento XML es válido cuando, además de no tener errores de sintaxis, no incumple ninguna de las normas establecidas en su estructura (DTD, XML Schema)

5.3.1. XML bien formado y válido

Para que esté **bien formado** tiene que cumplir las siguientes reglas:

- Tener uno y solo un elemento raíz
- Todos los elementos deben estar cerrados
- Los elementos tienen que estar anidados correctamente: no se pueden intercalar aperturas y cierres de elementos distintos.
- Todos los valores de los atributos están entrecomillados
- Los nombres de elementos y atributos han de cumplir con sus respectivas reglas.

Para que sea **válido** tiene que existir una definición que contenga las reglas semánticas que validar. Se definen mediante un DTD o un XML Schema.

5.3.2 DTD

DTD (*Document Type Definition*)

Es un lenguaje que permite especificar reglas que han de cumplir los documentos XML a los que se asocien.

Con el DTD podemos crear documentos de validación para archivos XML.

Ejemplo:

libro.dtd

```
<!DOCTYPE libro [  
  <!ELEMENT libro (titulo, autor, anio?, capitulo*)>  
  <!ELEMENT autor (#PCDATA)>  
  <!ELEMENT anio (#PCDATA)>  
  <!ELEMENT capitulo (tituloc, parrafo+)>  
  <!ATTLIST capitulo number CDATA "0">  
  <!ELEMENT titulo (#PCDATA)>  
  <!ELEMENT parrafo (#PCDATA)>  
>
```

5.3.2 DTD

Se puede insertar en el propio documento dentro de una etiqueta DOCTYPE

(menos habitual puesto que sólo se puede aplicar a un documento)

```
<!DOCTYPE nombre-elemento-raíz  
[  
    elementos-y-sus-relaciones  

```

Ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>  
<!DOCTYPE persona[  
    <!ELEMENT persona (nombre)>  
    <!ELEMENT nombre (#PCDATA)>  
<persona>  
    <nombre>Antonio</nombre>  
</persona>
```

O puede aparecer la ruta hacia el documento DTD en el prólogo del documento XML :

✓ **Un documento externo privado** (SYSTEM para indicarlo).

```
<!DOCTYPE nombre-elemento-raíz SYSTEM "URI">
```

La ruta puede ser:

- absoluta y entonces se indica su URL:

```
<!DOCTYPE raíz SYSTEM "http://www.company.com/docs.dtd">
```

```
<!DOCTYPE nota SYSTEM "http://www.empresa.com/nota.dtd">
```

- o relativa: <!DOCTYPE raíz SYSTEM "docs.dtd">

```
<!DOCTYPE nota SYSTEM "nota.dtd">
```

✓ **Un DTD externo de tipo PUBLIC:**

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

5.3.2 Entidades en los DTD

En un DTD, una entidad es el equivalente a una constante de programación. Cuando se define una entidad, se le asigna un nombre y un valor que sustituirá a las referencias que se hagan a dicho nombre.

Sintaxis de una **entidad interna**: **<! ENTITY nombre-entidad "texto de reemplazo">**

Sintaxis para referenciar a una **entidad interna**: **&nombre-entidad**

Ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fichas [
|   <!ENTITY poblacion "Almuñecar, Granada">
| ]>
<fichas>
|   <ficha>
|       <empresa nombre="Tractores Galiano"></empresa>
|       <poblacion>&poblacion;</poblacion>
|   </ficha>
|   <ficha>
|       <empresa nombre="Congelados Parra"></empresa>
|       <poblacion>&poblacion;</poblacion>
|   </ficha>
</fichas>
```

5.3.2 Entidades en los DTD

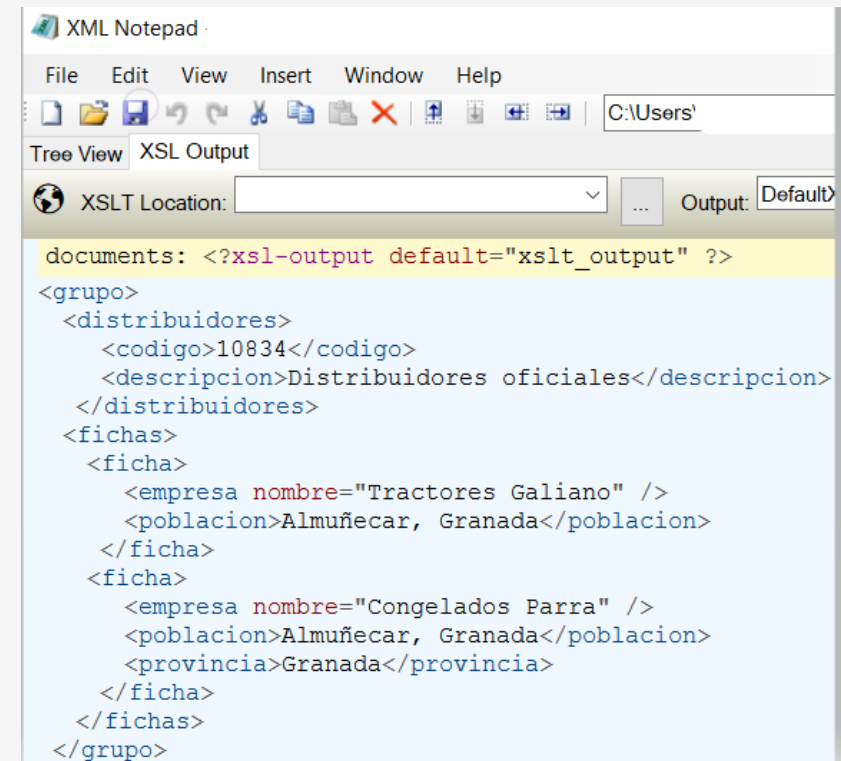
Sintaxis de una **entidad externa pública**:

<! ENTITY nombre-entidad PUBLIC "id-publico" "URI/URL">

Sintaxis de una **entidad externa privada**:

<! ENTITY nombre-entidad SYSTEM "URI/URL">

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fichas [
    <!ENTITY poblacion "Almuñecar, Granada">
    <!ENTITY provincia SYSTEM "provincia.txt">
    <!ENTITY detalle SYSTEM "ficha.xml">
]>
<grupo>
    &detalle;
    <fichas>
        <ficha>
            <empresa nombre="Tractores Galiano"></empresa>
            <poblacion>&poblacion;</poblacion>
        </ficha>
        <ficha>
            <empresa nombre="Congelados Parra"></empresa>
            <poblacion>&poblacion;</poblacion>
            <provincia>&provincia;</provincia>
        </ficha>
    </fichas>
</grupo>
```



The screenshot shows the XML Notepad application window. The menu bar includes File, Edit, View, Insert, Window, and Help. The toolbar contains various icons for file operations and editing. The address bar shows the path C:\Users\... The Tree View tab is active, displaying the XSLT transformation output. The output is shown in a light blue background with the following XML structure:

```
documents: <?xsl-output default="xslt_output" ?>
<grupo>
  <distribuidores>
    <codigo>10834</codigo>
    <descripcion>Distribuidores oficiales</descripcion>
  </distribuidores>
  <fichas>
    <ficha>
      <empresa nombre="Tractores Galiano" />
      <poblacion>Almuñecar, Granada</poblacion>
    </ficha>
    <ficha>
      <empresa nombre="Congelados Parra" />
      <poblacion>Almuñecar, Granada</poblacion>
      <provincia>Granada</provincia>
    </ficha>
  </fichas>
</grupo>
```

5.3.2 Anotaciones en los DTD

Se utilizan para identificar el formato de entidades que no son XML y que, por tanto, no se van a procesar.

Permite definir y utilizar otros tipos NO XML, por ejemplo tipo MIME (image/jpeg, etc...)

<! NOTATION nombre PUBLIC >

<! NOTATION nombre PUBLIC "id-publico" >

<! NOTATION nombre SYSTEM "URI">

5.3.2 Elementos DTD

```
<!ELEMENT nombreElemento (elementoHijo1, elementoHijo2, ...)>
```

```
<!ELEMENT nombreElemento Tipo>
```

El tipo puede ser:

ANY - Cualquier valor

EMPTY - Elemento vacío

(#PCDATA) - *Parsed Character Data.*

Texto que se va a ser analizado por el procesador

(#CDATA) - *Character Data.*

Texto que no va a ser analizado y por tanto no se podrán detectar entidades

5.3.2 Elementos DTD

```
<!ELEMENT nombreElemento (elementoHijo1, elementoHijo2, ...)>
```

Podemos indicar la **cantidad** de elementos hijos:

ElementoHijo - Aparece una sola vez y es obligatorio

ElementoHijo? - Aparece 0 o 1 vez

ElementoHijo* - Aparece 0 o más veces

ElementoHijo+ - Aparece 1 o más veces

(Elemento1|Elemento2) - Aparece el Elemento1 o el Elemento2

() - Permite agrupar expresiones

5.3.2 Atributos DTD

```
<!ATTLIST nombreElemento nombreAtributo tipo valor>
```

Si un elemento tiene varios atributos se indica como:

```
<!ATTLIST nombreElemento nombreAtributo1 tipo1 valor1>
```

```
<!ATTLIST nombreElemento nombreAtributo2 tipo2 valor2>
```

O bien:

```
<!ATTLIST nombreElemento  
    nombreAtributo1 tipo1 valor1  
    nombreAtributo2 tipo2 valor2  
>
```

Ambas formas son equivalentes

5.3.2 Atributos DTD

Tipos de atributos

Tipo	Descripción
CDATA	Cadena de caracteres
(valor1 valor2 ...)	Una lista de valores posibles
ID	Es el único identificador del atributo. No debe aparecer más de una vez.
IDREF	Se usa para hacer referencia a la identidad de otro elemento. Se utiliza para establecer conexiones entre los elementos.
IDREFS	Lista de referencias separadas por espacios a identificadores de otros elementos
ENTITY	Representa una entidad externa en el documento.
ENTITIES	Referencia a un conjunto de entidades
NMTOKEN	Es similar al CDATA . El valor del atributo consiste en un nombre XML válido.
NMTOKENS	Lista de nombres XML válidos.
NOTATION	Un nombre de notación
xml:lang	Indica el idioma del contenido
xml:space	Señala si se han de respetar los espacios, las tabulaciones y los retorno de carro múltiples (valor <<preserve>>) o eliminarlos (valor <<default>>)

5.3.2 Atributos DTD

Lista de valores posibles para los atributos:

#REQUIRED: el atributo es obligatorio, aunque no se especifica ningún valor predeterminado.

#IMPLIED: el atributo no es obligatorio y no se especifica ningún valor predeterminado.

#FIXED “valor”: el atributo tiene un valor fijo.

“valor”: el atributo tiene un valor predeterminado

5.3.3 XML Schema

Son un conjunto de reglas predefinidas para validar ficheros XML y especificar su formato correcto.

Un esquema define los elementos que pueden aparecer en un documento XML, así como los atributos que pueden asociarse a éstos.

También define información *estructural* tal como enumerar los descendientes de un elemento, la secuencia en que pueden aparecer, sus tipos, etc.

5.3.3 XML Schema

Algo más completo (más tipos de datos) que DTD y en lenguaje XML

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="libro" type="librostruct"/>
  <xsd:complexType name="librostruct">
    <xsd:sequence>
      <xsd:element name="titulo" type="xsd:string"/>
      <xsd:element name="autor" type="xsd:string"/>
      <xsd:element name="anio" type="xsd:string" minOccurs="0" />
      <xsd:element name="capitulo" type="capitulostruct" maxOccurs="unbounded" />
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="capitulostruct">
    ....
  </xsd:complexType>
</xsd:schema>
```

5.3.3 XML Schema

Los **DTD** permiten diseñar un **vocabulario para ficheros XML**, pero, ¿qué sucede cuando los valores de los elementos y atributos de esos ficheros han de corresponder a datos de un tipo determinado, o cumplir determinadas restricciones que no pueden reflejarse en los DTD?

Para ello se definen **XML Schemas**.

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

Los elementos XML que se utilizan para generar un esquema han de pertenecer al espacio de nombre XML Schema, que es:

<http://www.w3.org/2001/XMLSchema>.

5.3.3 XML Schema: Estructura básica

Los documentos XML Schema se suelen almacenar con la extensión xsd. En el documento XSD se define la estructura y se indican los elementos que debe contener el documento XML y de qué tipo son cada uno.

La estructura básica se compone de:

- La declaración del documento XML

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

- La declaración del elemento raíz y del espacio de nombres

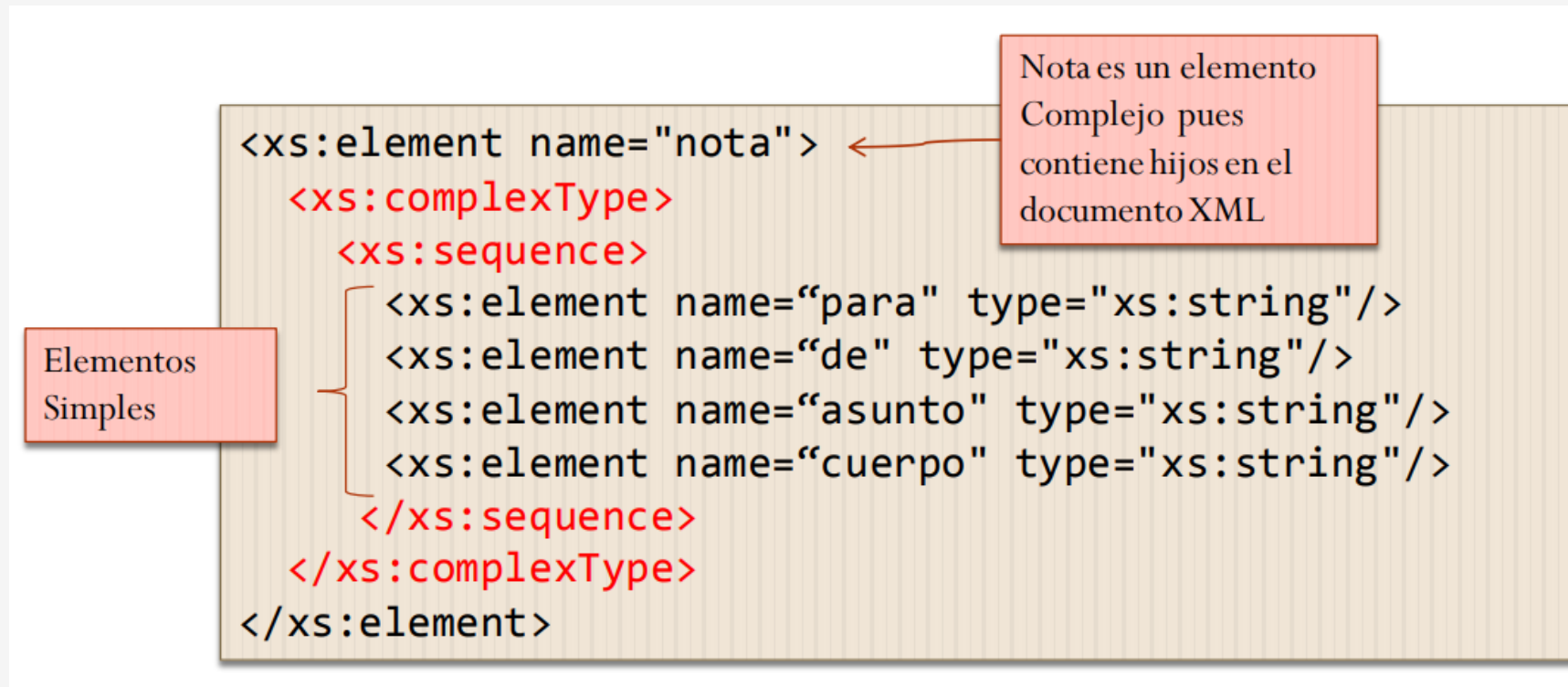
```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
```

- La declaración de los elementos (su jerarquía, relación, atributos y demás restricciones)
- La declaración de los atributos.

5.3.3 XML Schema: Elementos

Un esquema contiene elementos que pueden ser:

- Simples: No pueden tener ni elementos ni atributos
- Complejos: Pueden contener otros elementos y atributos



5.3.3 XML Schema: Elementos

Elementos

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

Ejemplo

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

Con valores por defecto

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

Con valores fijos

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

5.3.3 XML Schema: Elementos simples

- Un elemento simple es un elemento XML que sólo contiene texto. *(No puede contener otros elementos o atributos)* .

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

- Los tipos Pueden ser
xs: string, xs: decimal, xs: integer, xs: boolean, xs: date, xs: time

XML	XSD
<pre><nombre>Juan</nombre> <edad>18</edad> <fecha>15-03-2012</fecha></pre>	<pre><xs:element name="nombre" type="xs:string"/> <xs:element name="edad" type="xs:integer"/> <xs:element name="fecha" type="xs:date"/></pre>

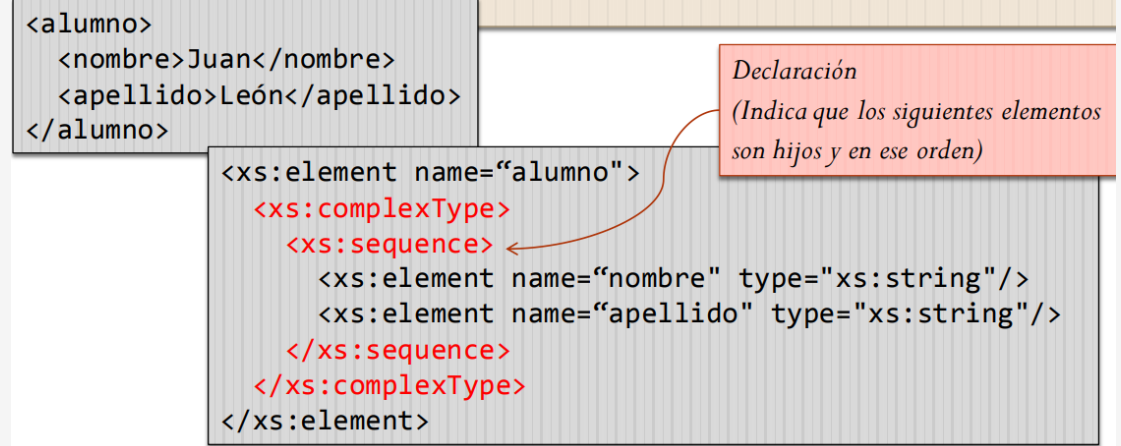
5.3.3 XML Schema: Elementos complejos

Se declaran utilizando la etiqueta `complexType` y pueden:

- Contener a otros elementos, pudiendo contener opcionalmente atributos
- Estar vacíos, pudiendo contener opcionalmente atributos
- Tener contenido mixto: otros elementos y texto, pudiendo contener opcionalmente atributos.

Sintaxis básica:

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



5.3.3 XML Schema: Elementos complejos tipos personalizados

XSD

```
<xsd:element name="empleado" type="tipoPersona"/>
<xsd:element name="estudiante" type="tipoPersona"/>

<xsd:complexType name="tipoPersona">
  <xsd:sequence>
    <xsd:element name="nombre" type="xsd:string"/>
    <xsd:element name="apellidos" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
```

Importante: al tratarse de un tipo, debe colocarse al final del documento xsd antes de la etiqueta de cierre `</xs:schema>`

5.3.3 XML Schema: Elementos complejos referencias

```
• <!-- Definición de elementos-->
<xs:element name="nombre" type="xsd:string" >
<xs:element name="apellidos" type="xsd:string" >

<!-- Definición de atributos -->
<xs:attribute name="repetidor" type="xs:string" />

<!-- Definición de elementos complejos -->
<xsd:element name="alumno"
  <xsd:complexType >
    <xsd:sequence>
      <xsd:element ref="nombre"/>
      <xsd:element ref="apellidos"/>
      <xsd:attribute ref="repetidor" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

5.3.3 XML Schema: Subelementos

Los elementos pueden contener subelementos.

Existen tres tipos:

- `xs:sequence`: secuencia de elementos obligatorios y en el mismo orden
- `xs:choice`: secuencia de elementos alternativos
Solo debe aparecer uno
- `xs:all`: secuencia de elementos opcionales.
No es obligatorio que aparezcan todos ni en el mismo orden.

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

```
<xs:element name="persona">
  <xs:complexType>
    <xs:choice>
      <xs:element name="empleado" type="empleado"/>
      <xs:element name="miembro" type="miembro"/>
    </xs:choice>
  </xs:complexType>
</xs:element>
```

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

5.3.3 XML Schema: Atributos

Los elementos con atributos son elementos complejos.

Los atributos se declaran como tipos simples.

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

XML	XSD
<code><curso letra="A">1</curso></code>	<code><xs:element name="curso" type="xs:integer"/> <xs:attribute name="letra" type="xs:string"/></code>

Valores por defecto, fijos u opcionales

```
<xs:attribute name="letra" type="xs:string" default="A"/>  
<xs:attribute name="letra" type="xs:string" fixed="A"/>  
<xs:attribute name="letra" type="xs:string" use="required"/>
```


5.3.3 XML Schema: Atributos

Los atributos son opcionales por defecto, para indicar que son obligatorios

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

Con valores por defecto

```
<xs:attribute name="lang" type="xs:string" default="EN"/>
```

Con valores fijos

```
<xs:attribute name="lang" type="xs:string" fixed="EN"/>
```

Elementos complejos vacíos y con atributos

XML

```
<producto id="1345" />
```

XSD

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

5.3.3 XML Schema: Restricciones

Podemos restringir los valores que pueden tomar los elementos y los atributos . A estas restricciones se les denomina **facetas**.

Restricción de valor

```
<xs:element name="age">
  <xs:simpleType>
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="0"/>
      <xs:maxInclusive value="120"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

O bien:

```
<xs:element name="age" type="ageLimit"/>

<xs:simpleType name="ageLimit">
  <xs:restriction base="xs:integer">
    <xs:minInclusive value="0"/>
    <xs:maxInclusive value="120"/>
  </xs:restriction>
</xs:simpleType>
```

Conjunto de valores

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

5.3.3 XML Schema: Restricciones

¿Cuáles son las restricciones que podemos aplicar sobre los valores de los datos de un elemento o atributo?

Están definidos por las facetas, que solo pueden aplicarse sobre tipos simples utilizando el elemento `xs:restriction`. Se expresan como un elemento dentro de una restricción y se pueden combinar para lograr restringir más el valor del elemento. Son, entre otros:

length, minlength, maxlenghtg: Longitud del tipo de datos.

enumeration: Restringe a un determinado conjunto de valores.

whitespace: Define el tratamiento de espacios (preserve/replace, collapse).

(max/min)(In/Ex)clusive: Límites superiores/inferiores del tipo de datos. Cuando son Inclusive el valor que se determine es parte del conjunto de valores válidos para el dato, mientras que cuando se utiliza Exclusive, el valor dado no pertenece al conjunto de valores válidos.

totalDigits, fractionDigits: número de dígitos totales y decimales de un número decimal.

pattern: Permite construir máscaras que han de cumplir los datos de un elemento. La siguiente tabla muestra algunos de los caracteres que tienen un significado especial para la generación de las máscaras

5.3.3 XML Schema: Restricciones

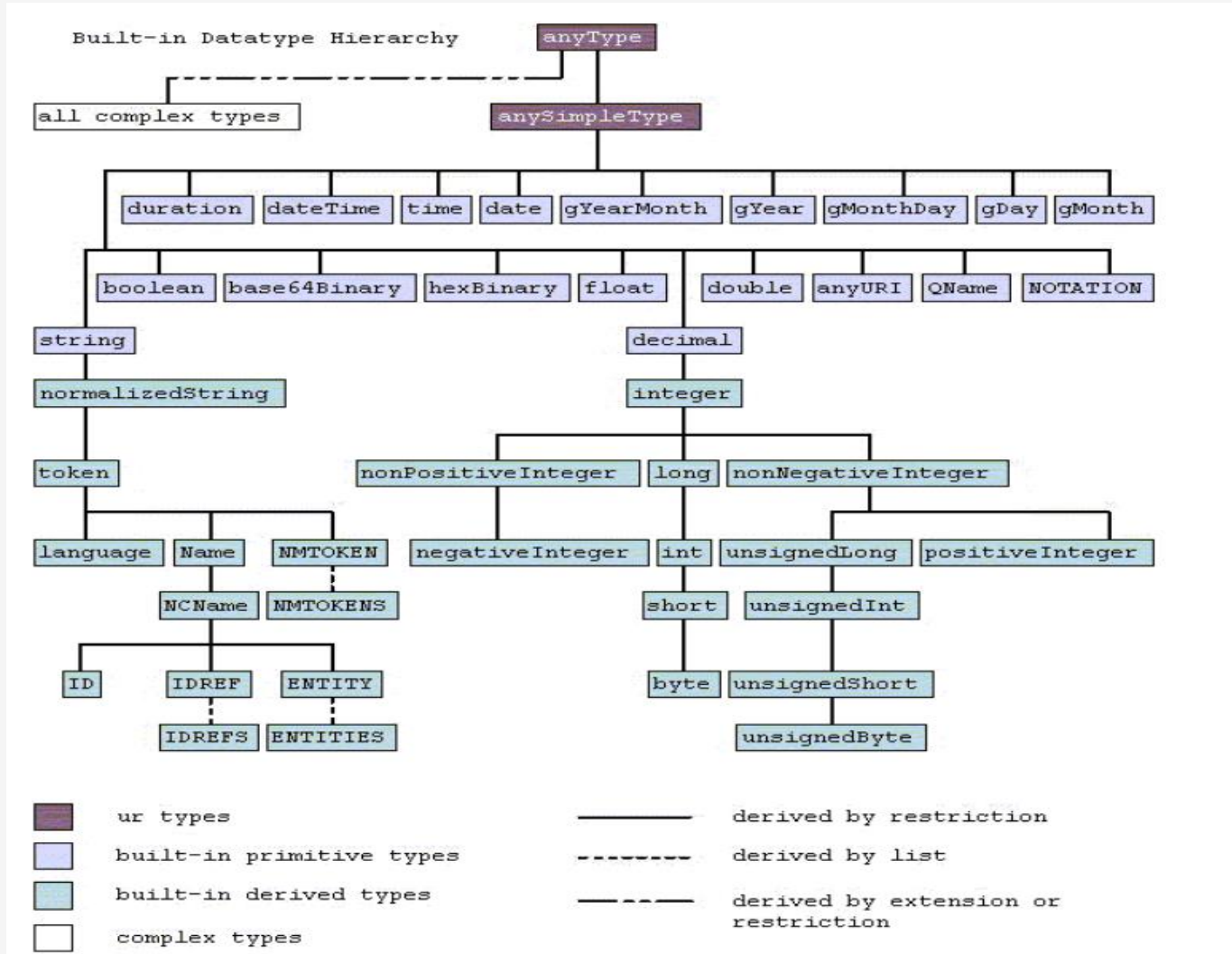
Patrón	Significado
[A-Z a-z]	Letra.
[A-Z]	Letra mayúscula.
[a-z]	Letra minúscula.
[0-9]	Dígitos decimales.
\D	Cualquier carácter excepto un dígito decimal.
(A)	Cadena que coincide con A.
A B	Cadena que es igual a la cadena A o a la B.

Patrón	Significado
AB	Cadena que es la concatenación de las cadenas A y B.
A?	Cero o una vez la cadena A.
A+	Una o más veces la cadena A.
A*	Cero o más veces la cadena A.
[abcd]	Alguno de los caracteres que están entre corchetes.
[^abcd]	Cualquier carácter que no esté entre corchetes.

5.3.3 XML Schema: tipos de datos

En este enlace encontrarás la relación de tipos de datos con su jerarquía y relaciones:

<https://www.w3.org/TR/xmlschema-2/>



5.3.3 XML Schema: tipos de datos

Algunos de estos valores predefinidos son:

- string, se corresponde con una cadena de caracteres UNICODE.
- boolean, representa valores lógicos, es decir que solo pueden tomar dos valores, true o false.
- integer, número entero positivo o negativo.
- positiveInteger, número entero positivo.
- negativeInteger, número entero negativo.
- decimal, número decimal, por ejemplo, 8,97.
- dateTime, representa una fecha y hora absolutas.
- duration, representa una duración de tiempo expresado en años, meses, días, horas, minutos segundos. El formato utilizado es: PnYnMnDTnHnMnS. Por ejemplo para representar una duración de 2 años, 4 meses, 3 días, 5 horas, 6 minutos y 10 segundos habría que poner:P2Y4M3DT5H6M7S. Se pueden omitir los valores nulos, luego una duración de 2 años será P2Y. Para indicar una duración negativa se pone un signo – precediendo a la P.
- time, hora en el formato hh:mm:ss.
- date, fecha en formato CCYY-MM-DD.
- gYearMonth, representa un mes de un año determinado mediante el formato CCYY-MM.
- gYear, indica un año gregoriano, el formato usado es CCYY.
- gMonthDay, representa un día de un mes mediante el formato –MM-DD.
- gDay, indica el ordinal del día del mes mediante el formato –DD, es decir el 4º día del mes será – 04.
- gMonth, representa el mes mediante el formato –MM. Por ejemplo, febrero es –02.
- anyURI, representa una URI.
- language, representa los identificadores de lenguaje, sus valores están definidos en RFC 1766.
- ID, IDREF, ENTITY, NOTATION, NMTOKEN. Representan lo mismo que en los DTD's

En este enlace encontrarás los tipos de datos admitidos por el estándar.

[XML Schema Part 2: Datatypes Second Edition \(w3.org\)](https://www.w3.org/XML/Schema)

5.3.3 XML Schema: comentarios

Existe un conjunto de elementos para incluir documentación que proporcione información legible por las personas y las máquinas.

Para agregar comentarios usaremos **xs:annotation**

Este elemento puede ser contenedor de los elementos **xs:documentation** y **xs:appinfo**, además de cualquier otro tipo de contenido, incluido HTML.

```
<xs:element name="mensaje">
  <xs:annotation>
    <xs:appinfo>Validador de mensajes</xs:appinfo>
    <xs:documentation xml:lang="es">
      Este esquema valida mensajes privados seguros
    </xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="de" type="xs:string"/>
      <xs:element name="para" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

5.3.4 Otras formas de validación

- RELAX NG (Regular Language for XML Next Generation)
- Schematron

5.4 XML aplicado

- En aplicaciones : para datos de configuración de una aplicación en los que e almacenan colores del texto o idioma; perfiles de seguridad de un sistema; preferencias de usuario de un videojuego; intercambio de datos entre aplicaciones; etc.
- En la web: Para diseñar páginas web con XHTML; transformaciones XSLT para convertir un documento XML en una pagina web
- En servicios web: intercambio de datos entre sistemas informáticos.

Existen dos tipos de WS: SOAP y REST.

5.5 Herramientas de visualización y edición XML

- Editores de texto
- Navegadores web
- XML Notepad
- Liquid Studio
- XMLPad
- Eclipse
- Editores web