



UD5 - VALIDACIÓN DE XML



UD 5 - VALIDACIÓN DE XML

Índice de Contenidos

- ▶ 1 - Necesidad de validar
- ▶ 2 - Tecnologías de validación
- ▶ 3 - Herramientas específicas
- ▶ 4 - DTD
- ▶ 5 - XSD

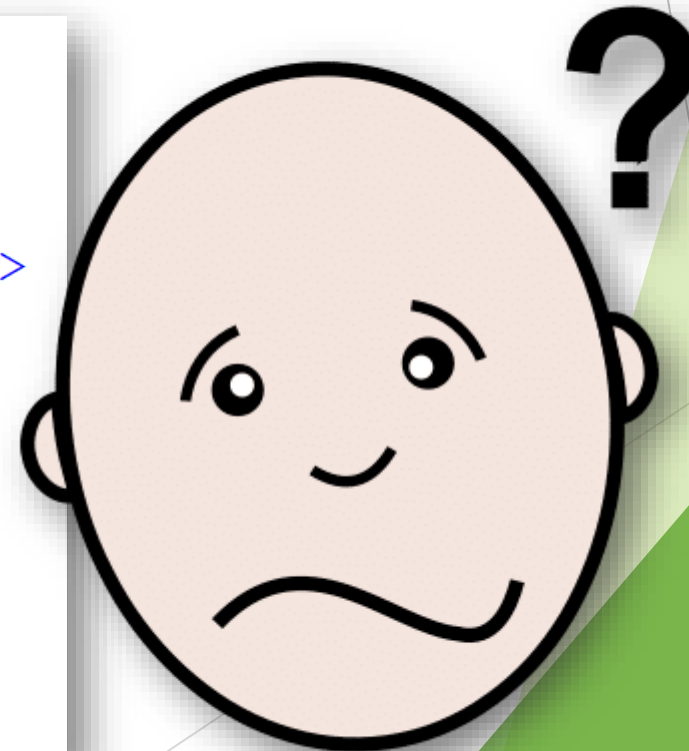
UD 5 - VALIDACIÓN DE XML

1 - Necesidad de validar

► 1 - Necesidad de validar

- Un documento XML bien formado es el que cumple las sintácticas propias del XML.
- Sin embargo esto no es suficiente, fíjate en el siguiente XML:

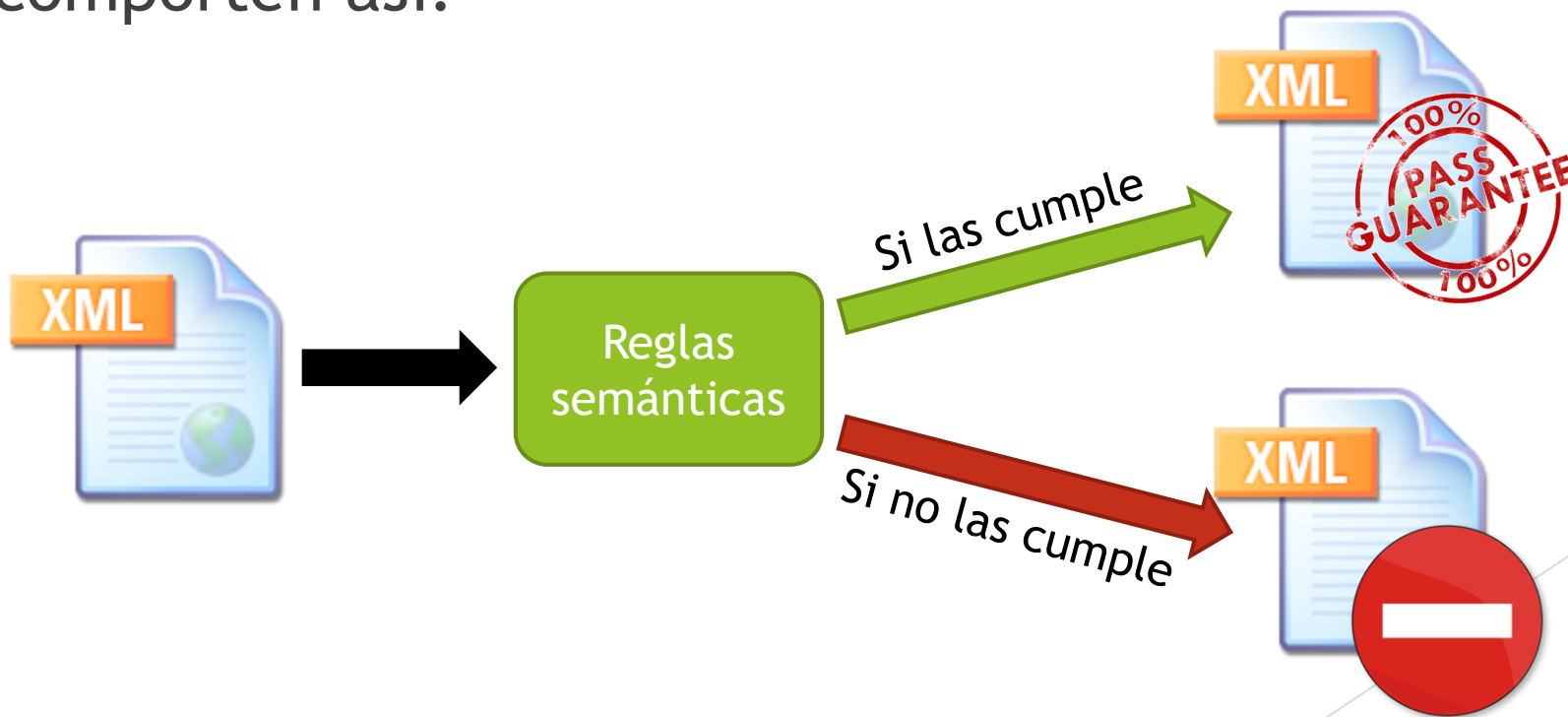
```
<zoo>
  <mamiferos>
    <animal>León</animal>
    <animal>Gacela</animal>
    <matricula>2365-FRT</matricula>
  </mamiferos>
  <reptiles>
    <animal>Cocodrilo</animal>
    <animal>Serpiente</animal>
    <animal>Serpiente</animal>
  </reptiles>
  <animal>Topo</animal>
</zoo>
```



UD 5 - VALIDACIÓN DE XML

1 - Necesidad de validar

- Necesitamos **validar documentos XML** para poder verificar que la información almacenada es **COHERENTE**, es decir, que **TIENE SENTIDO**.
- Para ello necesitamos establecer unas **reglas semánticas** que permitan describir qué información consideramos correcta y que se comporten así:



UD 5 - VALIDACIÓN DE XML

1 - Necesidad de validar

- ▶ Al documento XML bien formado que además cumple con las reglas semánticas se le llama **documento XML válido o validado**.
- ▶ Las reglas semánticas se pueden escribir dentro del propio documento XML, pero normalmente se escriben **un documento aparte**.
- ▶ Esto posibilita que dos grupos independientes de personas posean el mismo documento de reglas y puedan verificar si la información que envían o reciben es coherente.



UD 5 - VALIDACIÓN DE XML

2 - Tecnologías de validación

2 - Tecnologías de validación

Las reglas semánticas pueden escribirse en dos posibles notaciones:

1. **DTD** (Document Type Definition): es la más antigua pero sigue vigente. Se basa en un lenguaje propio para describir las reglas semánticas.
2. **XSD** (XML Schema Definition): es más moderna. Se basa en un dialecto de XML para describir las reglas semánticas.

Vamos a estudiar las dos notaciones ya que se usan ambas hoy en día.



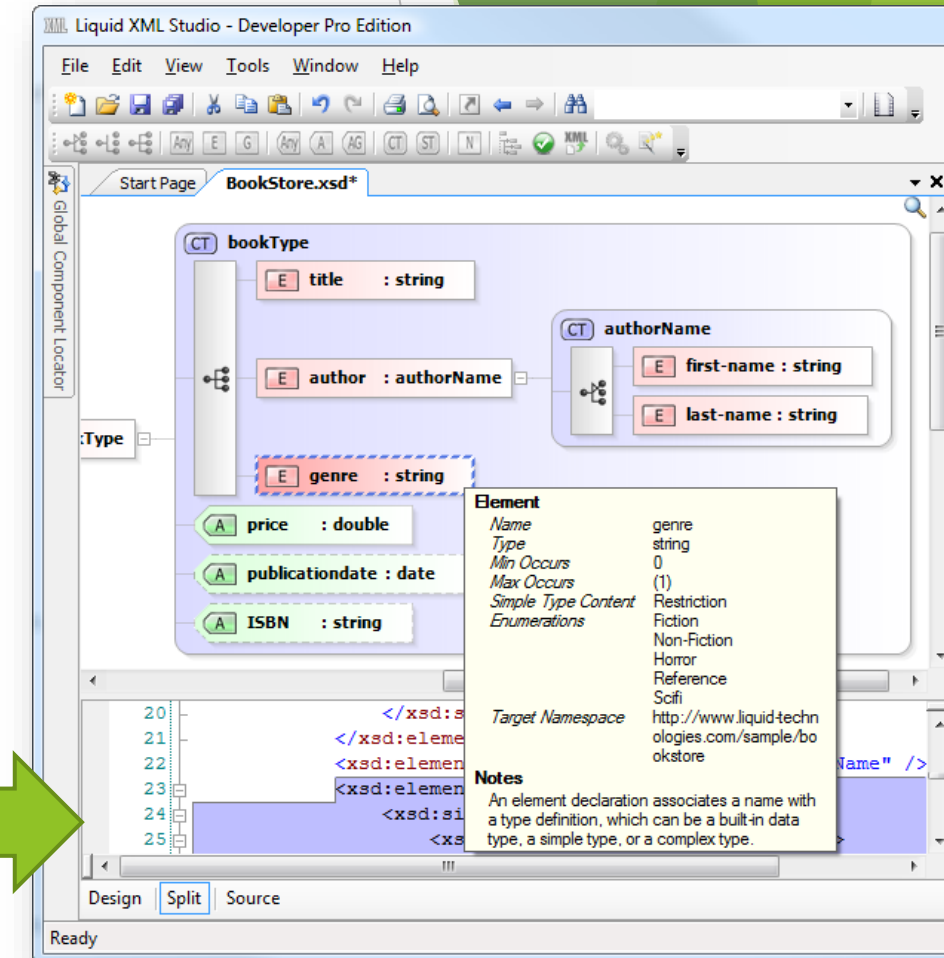
UD 5 - VALIDACIÓN DE XML

3 - Herramientas específicas

3 - Herramientas específicas

Hay muchas herramientas específicas para trabajar con XML y documentos validadores:

- ▶ <https://www.oxygenxml.com/> (Trial - 30 días)
- ▶ <https://www.xmlfox.com/> (versión básica freeware)
- ▶ <https://www.editix.com/> (Trial - 30 días)
- ▶ <https://www.liquid-technologies.com/xml-studio> (Trial - 30 días) permite un diseño de los documentos de validación en “modo gráfico”



UD 5 - VALIDACIÓN DE XML

3 - Herramientas específicas

En nuestro caso, seguiremos apostando por el software libre. Sin embargo, la versión actual del plugin “XML Tools” del Notepad++ ya no valida con documentos DTD, solo valida con XSD, así que vamos a dar el salto a otra herramienta específica:

- ▶ **XML Copy Editor** es un editor de documentos XML de software libre con capacidad de validación contra documentos DTD y XSD.
- ▶ Puedes descargarlo e instalarlo desde:
 - ▶ <https://sourceforge.net/projects/xml-copy-editor/>



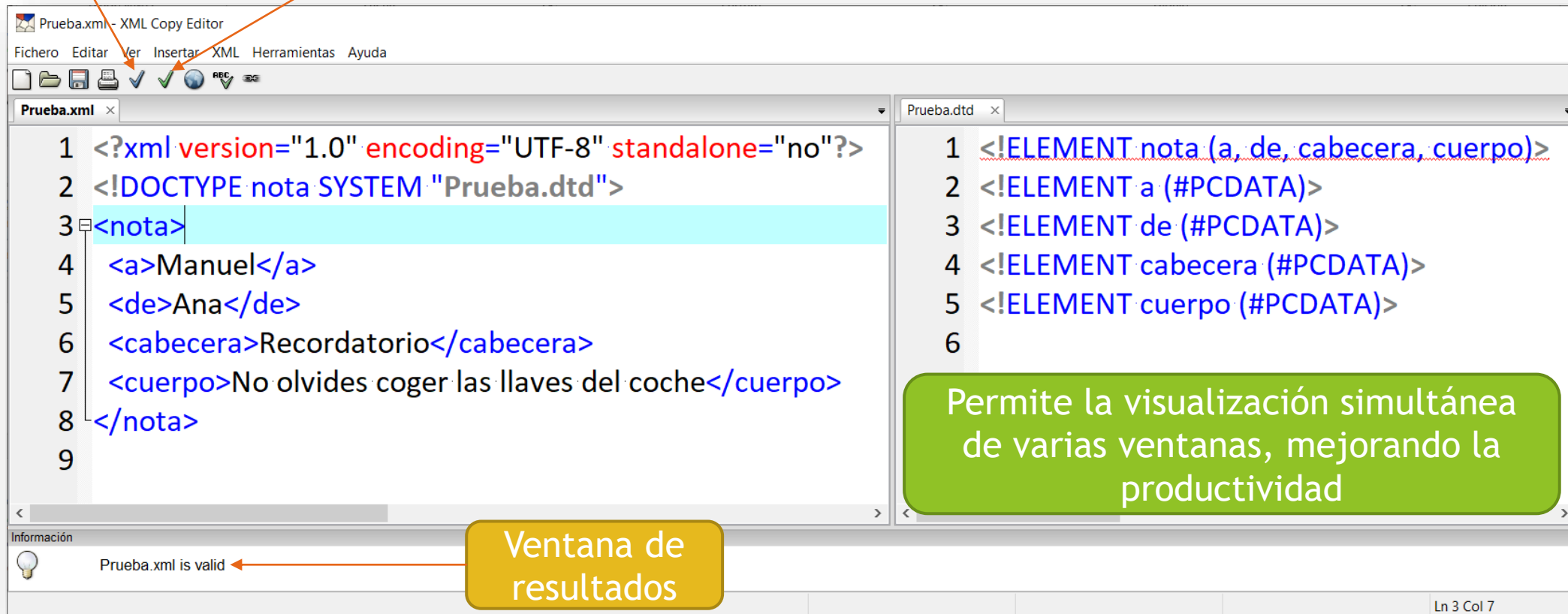
UD 5 - VALIDACIÓN DE XML

3 - Herramientas específicas

Hacemos una prueba de funcionamiento abriendo los ficheros Prueba.xml y Prueba.dtd

Comprobar
Bien-Formado

Validar



Ventana de
resultados

Ya sabemos más
o menos cómo
vamos a
trabajar, ahora
toca aprender
la notación del
DTD

Permite la visualización simultánea
de varias ventanas, mejorando la
productividad

UD 5 - VALIDACIÓN DE XML

4 - DTD

4 - DTD

- ▶ Los documentos DTD (Document Type Definition) define el tipo de documento con el que debe “encajar” un documento XML con una notación “propia”.
- ▶ A los ojos de un DTD un documento XML se compone de cuatro posibles elementos:
 - ▶ **Elementos:** formados por `<etiqueta>contenido</etiqueta>`
Ejemplo: `<animal>Perro</animal>`
 - ▶ **Contenidos o datos:** `<animal>Perro</animal>`
 - ▶ **Atributos:** `<persona dni="43837123E">...`
 - ▶ **Entidades de carácter:** `<` para el signo `<`



UD 5 - VALIDACIÓN DE XML

4 - DTD

NOTACIÓN PARA CONTENIDOS O DATOS

- ▶ DTD distingue dos tipos de datos:

- ▶ **PCDATA (Parsed Character DATA):** son datos en formato texto que serán analizados (*parsed*) por el programa (*parser*) que procese el documento XML en búsqueda de etiquetas o entidades de carácter. Esto haría que caracteres como & o < serían ilegales en este tipo de contenido.
- ▶ **CDATA (Character DATA):** son datos en formato texto que no serán analizados por el *parser* y que, por tanto, podrán contener cualquier combinación de caracteres. Esto haría que caracteres como & o < serían válidos en este tipo de contenido.



UD 5 - VALIDACIÓN DE XML

4 - DTD

► Nos interesa usar:

- **CDATA** cuando el contenido que vamos a representar es un texto libre que normalmente será escrito por un humano, dándole libertad para escribir cualquier combinación de caracteres. Ejemplos: la descripción de un artículo, la descripción de los síntomas de un paciente, una cita, una fórmula matemática...
- **PCDATA** para el resto de los casos. Ante la duda es más seguro usar PCDATA para detectar posibles ataques de scripting en el que un atacante pueda añadir código javascript entre dos etiquetas `<script></script>`



UD 5 - VALIDACIÓN DE XML

4 - DTD

NOTACIÓN PARA LOS ELEMENTOS

- ▶ DTD presenta dos formas de anotar los elementos:
 - ▶ Asignando una “categoría” a un elemento. Para ello se usa la siguiente la notación:

`<!ELEMENT nombre-elemento categoría>`

- ▶ Hay dos tipos de categorías:
 - ▶ **EMPTY**: se usa para definir elementos vacíos. Ejemplo:

`<!ELEMENT br EMPTY>`

Nos permite elementos sin contenido en XML como:

`
` o bien `
</br>`

**DEFINICIÓN
POR
CATEGORÍA:
EMPTY**

UD 5 - VALIDACIÓN DE XML

4 - DTD

- **ANY**: se usa en etapas tempranas del diseño de un DTD para indicar que se considera válido cualquier elemento analizable (parseable). También se usa cuando el elemento tiene un contenido que usa otro lenguaje de marcado (HTML). Ejemplo de uso de ANY:

<!ELEMENT **todo_vale** ANY>

Nos permitiría escribir elementos XML como:

<**todo_vale**>523453abcd</**todo_vale**>

o también

<**todo_vale**>

<otra_etiqueta>

Un contenido cualquiera

</otra_etiqueta>

</**todo_vale**>

**DEFINICIÓN
POR
CATEGORÍA:
ANY**

UD 5 - VALIDACIÓN DE XML

4 - DTD

- Definiendo el “modelo de contenido” de un elemento, utilizando la notación:

<!ELEMENT **nombre-elemento** (**modelo-contenido**)>

- El (**modelo-contenido**) puede ser de tres tipos:
 1. **Un tipo de dato concreto:** podemos usar los tipos PCDATA y CDATA del siguiente modo:

<!ELEMENT **nombre** (**#PCDATA**)>

<!ELEMENT **descripcion** (**#CDATA**)>

El # se usa por razones históricas

Esto permitiría escribir elementos XML como:

<**nombre**>**Juan**</**nombre**>

<**descripcion**>**Aquí vale cualquier texto o símbolos**</**descripcion**>

**DEFINICIÓN
DEL
MODELO DE
CONTENIDO:
#PCDATA Y
#CDATA**

UD 5 - VALIDACIÓN DE XML

4 - DTD

2. **Uno o varios elementos hijos:** se deben poner entre paréntesis los nombres de los elementos hijos pero existen distintos modificadores para indicar distintas situaciones (multiplicidad, secuencia, opcionalidad).

- Para indicar que un elemento tiene 1 y solo 1 elemento hijo:

```
<!ELEMENT persona (dni)>
```

```
<!ELEMENT dni (#PCDATA)>
```

Esto permitiría escribir elementos XML como:

```
<persona><dni>28748343E</dni></persona>
```

- Usamos ? Para indicar 0 ó 1 ocurrencia (es opcional):

```
<!ELEMENT articulo (descripcion?)>
```

```
<!ELEMENT descripcion (#CDATA)>
```

Esto permitiría escribir elementos XML como:

```
<articulo><descripcion>Texto</descripcion></articulo>
```

Y también: `<articulo></articulo>`

**DEFINICIÓN
DEL
MODELO DE
CONTENIDO:
multiplicidad**

UD 5 - VALIDACIÓN DE XML

4 - DTD

- Usamos + para indicar que un elemento puede aparecer 1 o más veces:

```
<!ELEMENT mamiferos (animal+)>
```

```
<!ELEMENT animal (#PCDATA)>
```

Esto permitiría escribir elementos XML como:

```
<mamiferos>
```

```
  <animal>Perro</animal>
```

```
  <animal>Gato</animal>
```

```
</mamiferos>
```

- Usamos * Para indicar 0 o más ocurrencias:

```
<!ELEMENT bolsillo (moneda*)>
```

```
<!ELEMENT moneda (#PCDATA)>
```

Esto permitiría escribir elementos XML como:

```
<bolsillo><moneda>0,50</moneda><moneda>0,20</moneda>
```

```
</bolsillo>
```

**DEFINICIÓN
DEL
MODELO DE
CONTENIDO:
multiplicidad**

UD 5 - VALIDACIÓN DE XML

4 - DTD

- También podemos establecer una **secuencia ordenada** de elementos del siguiente modo:

<!ELEMENT **articulo** (**nombre**, **precio**, **descripcion**)>

<!ELEMENT **nombre** (#PCDATA)>

<!ELEMENT **precio** (#PCDATA)>

<!ELEMENT **descripcion** (#CDATA)>

Esto permitiría escribir elementos XML como:

<**articulo**>

<**nombre**>Reloj FR-234</**nombre**>

<**precio**>22,90</**precio**>

<**descripcion**>Reloj digital muy bueno</**descripcion**>

</**articulo**>

Si cambiamos el orden de los elementos o si falta alguno fallaría

**DEFINICIÓN
DEL
MODELO DE
CONTENIDO:
secuencia**

UD 5 - VALIDACIÓN DE XML

4 - DTD

- Para poder practicar todo esto necesitamos conocer la sintaxis que permite **asociar** un documento XML con su correspondiente DTD. Tenemos dos posibilidades:
 - Añadir el DTD dentro del documento XML:

Justo antes del contenido XML se pone `<!DOCTYPE nodo_raíz [reglas]>`

El nombre del nodo raíz debe coincidir con el que pone el DOCTYPE

```
*xml y dtd juntos.xml x
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE nota[
3   <!ELEMENT nota (a , de , cabecera , cuerpo)>
4   <!ELEMENT a (#PCDATA)>
5   <!ELEMENT de (#PCDATA)>
6   <!ELEMENT cabecera (#PCDATA)>
7   <!ELEMENT cuerpo (#PCDATA)>
8 ]>
9 <nota>
10  <a>Manuel</a>
11  <de>Ana</de>
12  <cabecera>Recordatorio</cabecera>
13  <cuerpo>No olvides coger las llaves del coche</cuerpo>
14 </nota>
```

**ASOCIACIÓN
ENTRE EL XML
Y EL DTD**

Realizamos los
ejercicios 1 al 3
del boletín

UD 5 - VALIDACIÓN DE XML

4 - DTD

- Poner el DTD en un documento externo de nuestro sistema (disco duro) con la notación siguiente:

**ASOCIACIÓN
ENTRE EL XML
Y EL DTD**

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE catalogo SYSTEM "catalogo.dtd">
<catalogo>
  <producto>
    <nombre>Teclado inalámbrico Zeus 23Q</nombre>
    <vendedor>Todo Teclados</vendedor>
    <precio>23.90</precio>
    <detalle>Opera en la frecuencia de 2.4 GHz</detalle>
    <detalle>Hasta 20 metros de distancia</detalle>
  </producto>
  <producto>
    <nombre>Smartwatch Pulserita Wear</nombre>
    <precio>120.90</precio>
    <opiniones>
      <opinion>Mi abuela me toma mejor el pulso</opinion>
      <opinion>No es resistente al agua</opinion>
    </opiniones>
  </producto>
</catalogo>
```

catalogo.dtd

```
<!ELEMENT catalogo (producto+)>
<!ELEMENT producto (nombre, vendedor?, precio, opiniones?, detalle*)>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT vendedor (#PCDATA)>
<!ELEMENT precio (#PCDATA)>
<!ELEMENT detalle (#PCDATA)>
<!ELEMENT opiniones (opinion+)>
<!ELEMENT opinion (#PCDATA)>
```

En este ejemplo tanto el XML y el DTD deben estar en la misma carpeta del disco duro del sistema

```
<!DOCTYPE catalogo SYSTEM "dtd/catalogo.dtd">
```

Con esta otra notación el DTD estaría en una carpeta llamada **dtd** situada en la misma carpeta que el documento XML. Mejor siempre rutas relativas.

UD 5 - VALIDACIÓN DE XML

4 - DTD

- También podemos establecer la **opcionalidad** entre 2 o más elementos del siguiente modo:

```
<!ELEMENT perfil (usuario | administrador| supervisor)>
```

```
<!ELEMENT usuario EMPTY>
```

```
<!ELEMENT administrador EMPTY>
```

```
<!ELEMENT supervisor EMPTY>
```

Esto permitiría escribir elementos XML como:

```
<perfil><administrador/></perfil>
```

 o también sería válido:

```
<perfil><supervisor/></perfil>
```

**DEFINICIÓN
DEL
MODELO DE
CONTENIDO:
opcionalidad**

- Solo nos queda un tipo de notación para acabar con “los elementos” de XML. Vamos a ella:

UD 5 - VALIDACIÓN DE XML

4 - DTD

3. **Contenido mixto:** permiten mezclar datos y elementos hijos. La notación utilizada es la siguiente:

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

```
<!ELEMENT negrita (cursiva | #PCDATA)>
```

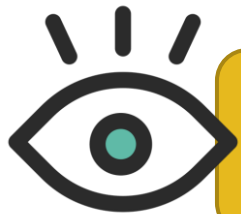
```
<!ELEMENT cursiva (negrita | #PCDATA)>
```

Esto permite escribir en un documento XML lo siguiente:

```
<parrafo>
```

```
Aquí un <cursiva>tema <negrita>importante</negrita></cursiva>
```

```
</parrafo>
```



OJO: No puede utilizarse +, el lenguaje solo permite * para indicar múltiples ocurrencias

**DEFINICIÓN
DEL
CONTENIDO
DEL
ELEMENTO:
mixto**

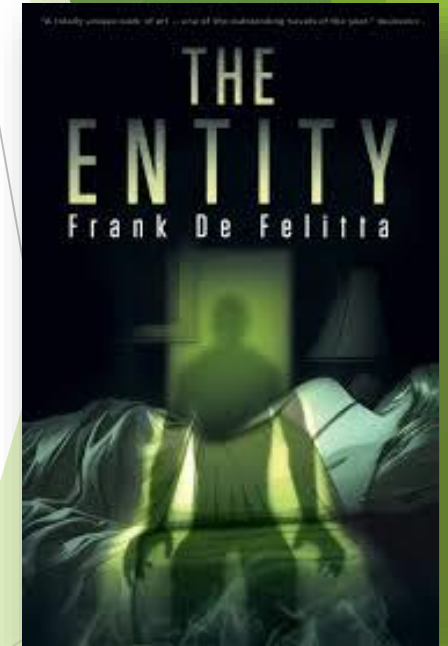
Realizamos los
ejercicios 4 al 6
del boletín

UD 5 - VALIDACIÓN DE XML

4 - DTD

NOTACIÓN PARA LAS ENTIDADES

- ▶ Recordemos que XML usaba entidades de carácter para poder escribir caracteres “conflictivos” para el parser o bien para codificar caracteres que no están disponibles en todos los teclados. Ejemplos:
 - ▶ < se sustituye por <
 - ▶ > se sustituye por >
 - ▶ & se sustituye por &
- ▶ Bueno, pues en un DTD podemos definir otras entidades para que estén disponibles en todos los documentos XML que usen dicho DTD. Veamos la notación:



UD 5 - VALIDACIÓN DE XML

4 - DTD

- ▶ `<!ENTITY instituto "IES Sotero Hernández">`
- ▶ `<!ENTITY copyleft "Copyleft. Some rights reserved">`

Esto permite escribir las nuevas entidades en nuestro documento XML siempre que estemos dentro de contenido tipo PCDATA

```
<parrafo>2020. &instituto; </parrafo>
```

```
<parrafo>&copyleft; </parrafo>
```

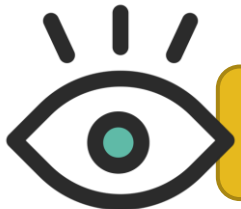
Y el parser procesaría las entidades y las sustituiría por:

2020. IES Sotero Hernández

Copyleft. Some rights reserved

**DEFINICIÓN
DE
ENTIDADES**

Realizamos el
ejercicio 7 del
boletín



OJO: Los navegadores tiene deshabilitado el procesado de entidades por razones de seguridad. Si lo pruebas no funciona.

UD 5 - VALIDACIÓN DE XML

4 - DTD

NOTACIÓN PARA LOS ATRIBUTOS

- Los atributos que pueda llevar un elemento se declaran del siguiente modo:

<!ATTLIST **empleado** **salario** CDATA "1000">



- Esto nos permite escribir:

<**empleado** **salario** = "2000">...</**empleado**> y también:

<**empleado**>...</**empleado**>

en este caso último se tomaría 1000 como valor del atributo salario

ATRIBUTO:
EL CIELO ES
AZUL

UD 5 - VALIDACIÓN DE XML

4 - DTD

- ▶ Veamos las distintas opciones del campo **Valor**:

- ▶ Valor “por defecto”, es la que ya hemos visto:

```
<!ATTLIST empleado salario CDATA “1000”>
```

- ▶ Valor requerido:

```
<!ATTLIST empleado dni CDATA #REQUIRED>
```

- ▶ Valor opcional (IMPLIED = “implícito”):

```
<!ATTLIST empleado cochePropio CDATA #IMPLIED>
```

- ▶ Valor FIJO:

```
<!ATTLIST empleado horario CDATA #FIXED “L-V”>
```

esto hace que el atributo sea **opcional**,
pero si aparece debe tener el valor fijado.

**POSIBLES
VALORES**

UD 5 - VALIDACIÓN DE XML

4 - DTD

- Fíjate que la palabra que usamos para declarar atributos es ATTLIST (Attribute List), así que los ejemplos anteriores también se pueden condensar en la siguiente notación, ya que los aplicamos siempre al mismo elemento:

```
<!ATTLIST empleado salario CDATA "1000"  
dni CDATA #REQUIRED  
cochePropio CDATA #IMPLIED  
horario CDATA #FIXED "L-V">
```

**POSIBLES
VALORES**

Realizamos los
ejercicios 8 y 9
del boletín

UD 5 - VALIDACIÓN DE XML

4 - DTD

- ▶ Veamos las distintas opciones del campo Tipo de Atributo:

- ▶ CDATA (texto no analizable. OJO: No vale el tipo PCDATA):

<!ATTLIST empleado salario CDATA "1000">

- ▶ Lista de valores permitidos:

<!ATTLIST pago tipo (efectivo | tarjeta) #REQUIRED>

OJO: los posibles valores van SIN COMILLAS

- ▶ Identificador único (siempre debe ser #REQUIRED):

<!ATTLIST empleado dni ID #REQUIRED>

Para este tipo de atributo, el parser comprueba dos cosas:

- ▶ Que no se declaren dos atributos ID para un mismo elemento
- ▶ Y que no existan dos elementos con el mismo valor de ID (dni)

**TIPO DE
ATRIBUTO**

UD 5 - VALIDACIÓN DE XML

4 - DTD

- Referencia a un identificador único ya declarado:

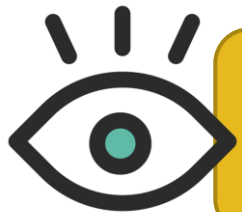
```
<!ATTLIST empleado id ID #REQUIRED  
idJefe IDREF #IMPLIED>
```

Esto nos permite escribir en XML:

```
<empleado id="E1">...</empleado>
```

```
<empleado id="E2" idJefe="E1">...</empleado>
```

Para este tipo de atributo, el parser comprueba que el valor del **idJefe** coincida con alguno de los **id** declarados en el XML.



OJO: Cuando usamos los tipos ID y/o IDREF, el parser EXIGE que los valores de los atributos sean “nombres válidos para etiquetas” en XML.

**TIPO DE
ATRIBUTO**

REGLAS NOMBRES XML:
Empiezan por una letra o _
Pueden contener a-z,
A-Z, . , - , _
No valen los espacios

UD 5 - VALIDACIÓN DE XML

4 - DTD

- Referencia a varios identificadores únicos:

```
<!ATTLIST empleado id ID #REQUIRED  
idJefes IDREFS #IMPLIED>
```

Esto nos permite escribir en XML:

```
<empleado id="E1">...</empleado>
```

```
<empleado id="E2">...</empleado>
```

```
<empleado id="E3" idJefes="E1 E2"> ...</empleado>
```

Abrimos el par de ficheros
XML-DTD "Empresa-
Empleados" del repositorio
y lo leemos

Se separan con
espacios, sin comas

**TIPO DE
ATRIBUTO**

UD 5 - VALIDACIÓN DE XML

4 - DTD

- Hay algunos tipos de valores más (NMTOKEN, NMTOKENS, ENTITY, y ENTITIES) que tienen poca utilidad. No los estudiaremos.

COMENTARIOS EN UN DTD

- Si en algún momento necesitamos hacer alguna aclaración o documentar alguna decisión, podemos incluir comentarios en un DTD del siguiente modo:

`<!-- Esto es un comentario`

`Y puede tener varias líneas -->`



Realizar los ejercicios
10 al 11 del boletín

UD 5 - VALIDACIÓN DE XML

4 - DTD

ASOCIACIÓN XML-DTD (otra vez)

- ▶ Hasta ahora hemos asociado los documentos XML con su DTD de dos formas:
 - ▶ **Interna:** el DTD está dentro del documento XML
 - ▶ **Externa:** el DTD está en otro documento que comparte el mismo sistema con el documento XML (notación SYSTEM).
- ▶ La notación SYSTEM, además se puede utilizar para indicar que el DTD está en otra máquina, aportando una URL localizar el recurso:

```
<!DOCTYPE empresa SYSTEM  
"http://www.dominio.com/dtd/Empresa-Empleados.dtd">
```



UD 5 - VALIDACIÓN DE XML

4 - DTD

- ▶ Además, cada vez que usamos la notación SYSTEM estamos indicando también que nuestro DTD es para uso **PRIVADO**. Es decir, para nuestra organización y colaboradores pero para nadie más.
- ▶ También podemos ser más generosos y hacer **PÚBLICO** nuestro DTD en Internet. Para ello podemos debemos usar la sintaxis:

<!DOCTYPE **nodo_raiz** **PUBLIC** "FPI" "URL">

- ▶ La **URL** nos localiza el DTD en Internet
- ▶ El **FPI** (Formal Public Identifier) identifica de forma única a nuestro DTD en el mundo. Esto ayuda a los parsers a no cargar en memoria dos veces el mismo DTD cuando se están procesando muchos documentos XML. Este FPI tiene una notación específica que es la siguiente:

UD 5 - VALIDACIÓN DE XML

4 - DTD

- **Notación del FPI:** Se usan 4 campos separados por //

Registro//Organización//Descripción//Idioma

- **Registro:** si nuestra organización ha registrado el DTD en la ISO (organización internacional de estandarización) entonces se pone un +, en caso contrario se pone un -
- **Organización:** es el nombre o dominio de la organización responsable del DTD
- **Descripción:** se pone el nombre del nodo raíz del documento XML que se describe en el DTD o bien una descripción breve.
- **Idioma:** identificador de 2 letras del idioma (EN para inglés, SP para español...)

Veamos un ejemplo real:

**NOTACIÓN
FPI**

UD 5 - VALIDACIÓN DE XML

4 - DTD

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"  
"http://www.w3.org/TR/html4/strict.dtd">
```

- ▶ **Registro:** - indica que la W3C no ha registrado el DTD en la ISO.
- ▶ **Organización:** W3C (World Wide Web Consortium)
- ▶ **Descripción:** DTD HTML 4.01
- ▶ **Idioma:** EN (inglés)

Si nosotros quisiéramos publicar uno de nuestro DTD podríamos hacer algo como:

```
<!DOCTYPE empleado PUBLIC "-//midominio.es//DTD empleado//SP"  
"http://www.midominio.es/dtd/empleado.dtd">
```

Las empresas que exponen sus datos en un Servicio Web (aseguradoras, aerolíneas...) publican su DTD para que las aplicaciones clientes conozcan el tipo de información que se considera válido

**EJEMPLOS
FPI**

UD 5 - VALIDACIÓN DE XML

4 - DTD

LIMITACIONES DE LOS DTD

- ▶ Un DTD no es un documento XML, luego no se puede verificar si está bien formado, así que podría tener errores.
- ▶ No soporta espacios de nombres (se necesitaban para poder resolver colisiones de nombrado al “combinar” documentos XML de distintas empresas).
- ▶ No existe el concepto de “tipo de dato” (String, Number...) que permita restringir los posibles valores de elementos y atributos.
- ▶ No se pueden especificar una lista de valores permitidos para los elementos, solo se puede hacer con atributos.
- ▶ No se puede dar un valor por defecto a los elementos, solo se puede hacer con atributos.



UD 5 - VALIDACIÓN DE XML

5 - XSD

5 - XSD

- ▶ Los documentos XSD (XML Schema Definition) describen la estructura de un documento XML utilizando el propio lenguaje XML para realizar la descripción.
- ▶ Los documentos XSD son mucho más versátiles y potentes que los DTD, permitiendo definir de un modo mucho más preciso el conjunto de reglas que valida un documento XML.
- ▶ También son más complejos de “digerir” que un DTD. Por ejemplo, la gestión de “namespaces” un grado de complejidad que se sale del ámbito de este módulo. Así que haremos una selección de los contenidos más relevantes.



UD 5 - VALIDACIÓN DE XML

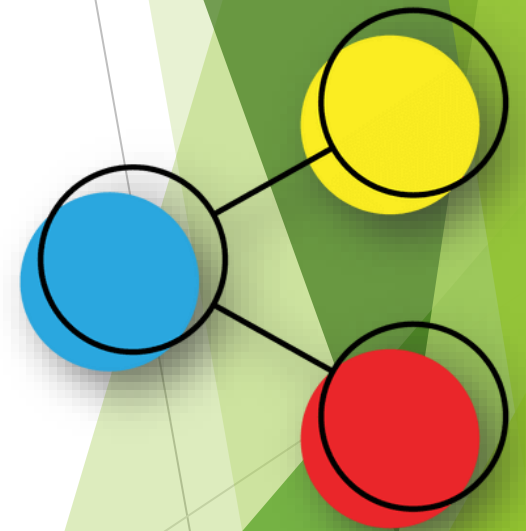
5 - XSD

TERMINOLOGÍA

Dado que tanto el XSD como el documento XML se escriben en lenguaje XML se adopta la siguiente terminología para diferenciar ambos documentos:

- ▶ **Esquema (schema):** es el documento XSD en el que se define la estructura o esquema de un documento XML.
- ▶ **Instancia (instance):** es el documento XML que se construye atendiendo a las reglas de un esquema.

Por otro lado, recordamos que un **namespace** es un conjunto de etiquetas que recibe un nombre único en el mundo y que sirve para resolver “colisiones” de nombres (igual que los paquetes de Java).



UD 5 - VALIDACIÓN DE XML

5 - XSD

ESQUELETO DE UN XSD

Como en cualquier XML tenemos dos elementos:

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

El típico
“Prólogo”

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

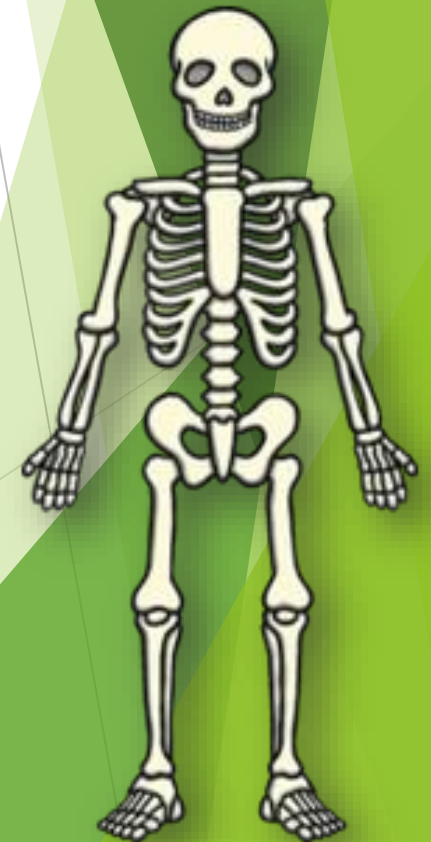
...nodos hijos...

Un nodo raíz llamado
“schema”

```
</xs:schema>
```

El “schema” contendrá otros “nodos hijos” con los que se definen los elementos y las reglas que permitan validar la instancia XML.

Además, tiene, al menos, un atributo (xmlns) que declara el espacio de nombres en el que se definen la etiqueta “schema” y otras más. También, crea un prefijo (xs) para referirnos a los elementos del espacio de nombres declarado de una forma más concisa.



UD 5 - VALIDACIÓN DE XML

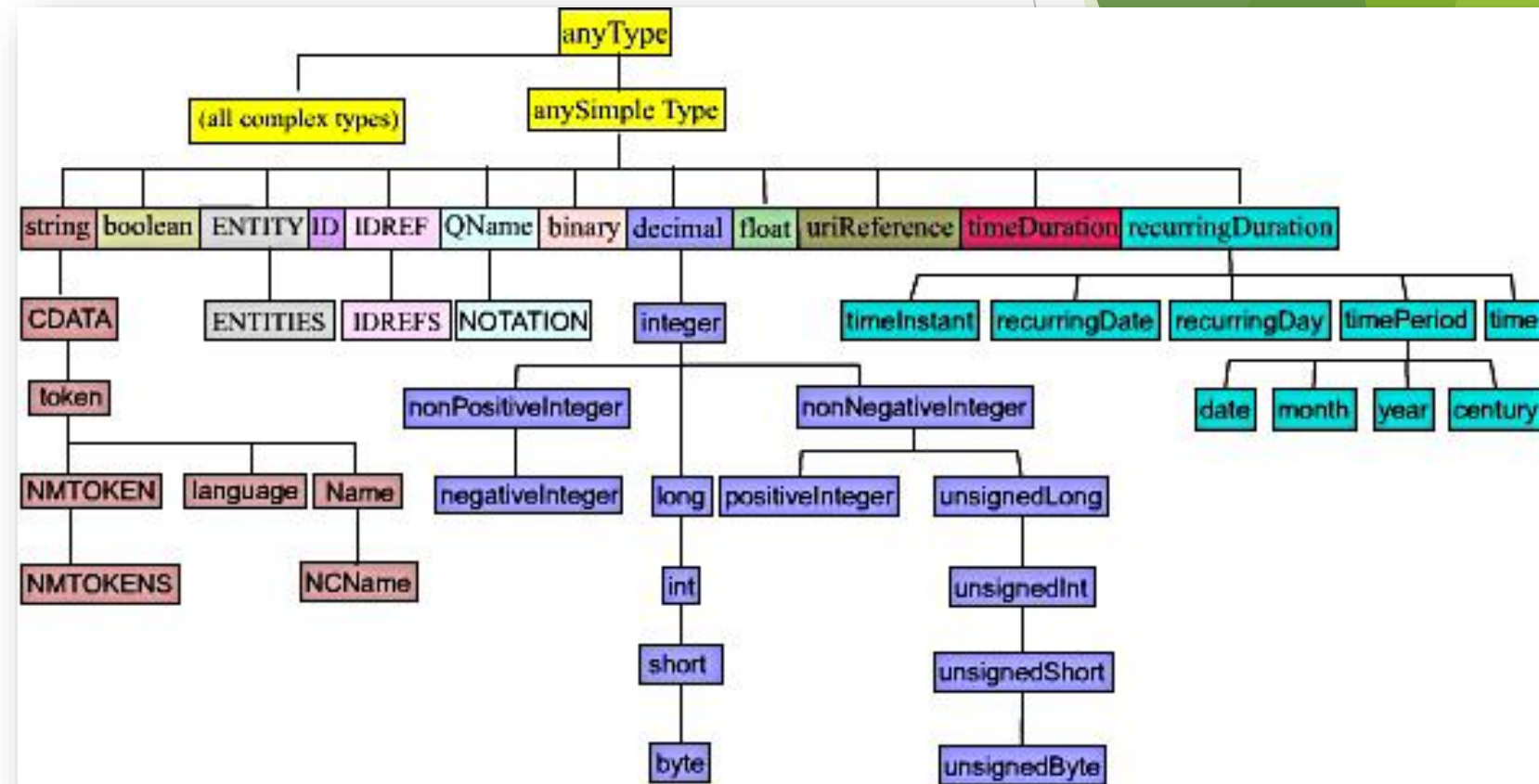
5 - XSD

TIPOS DE DATOS PREDEFINIDOS

En DTD solo teníamos dos tipos de datos incluidos en el lenguaje (PCDATA y CDATA).

En XSD tenemos un montón de tipos y, además, se les pueden añadir restricciones y también crear tipos de datos nuevos.

Es mucho más versátil.



Tranquilidad... Solo estudiaremos los más usuales

UD 5 - VALIDACIÓN DE XML

5 - XSD

Los tipos más usuales son:

- **xs:string** → puede contener cualquier texto o número, incluyendo tabuladores, retornos de carro (INTRO)... No puede contener <etiquetas> porque darían un error en el parser.
- **xs:decimal** → puede contener cualquier cantidad numérica usando el punto como separador decimal. Ej: 23, +45.23, -93.4498, 0...
- **xs:integer** → igual que antes pero solo con números enteros + ó -
- **xs:boolean** → solo puede almacenar *true* o *false*, produciría un error si se encuentra otra cosa.
- **xs:date** → guarda fechas en el formato YYYY-MM-DD. Ej: 2021-02-23
- **xs:time** → guarda instantes de tiempo en el formato HH:MM:SS.
Ejemplo: 09:14:56

UD 5 - VALIDACIÓN DE XML

5 - XSD

ELEMENTOS SIMPLES

Un elemento simple es aquel que no contiene nodos hijos. Veamos la sintaxis de su declaración:

```
<xs:element name="nombre_elem" type="tipo_datos"/>
```

Ejemplos:

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



```
<opinion>  
La película era una m!3rd4  
</opinion>
```

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



```
<precio>  
35.99  
</precio>
```



UD 5 - VALIDACIÓN DE XML

5 - XSD

ELEMENTOS COMPLEJOS

Un elemento complejo es aquel que contiene nodos hijos o atributos.

Es decir queremos definir cosas como:

```
<producto>
```

```
  <nombre>Ratón Genius 23R</nombre>
```

```
  <precio>12.00</precio>
```

```
</producto>
```

Y también cosas como:

```
<persona dni="12345678Q"></persona>
```



UD 5 - VALIDACIÓN DE XML

5 - XSD

Veamos la sintaxis de su declaración:

```
<xs:element name="nombre_elem">  
  <xs:complexType>  
    <xs:modelo de contenido>  
      Declaración de nodos hijos o de  
      atributos asociados a "nombre_elem"  
    </xs:modelo de contenido>  
  </xs:complexType>  
</xs:element>
```

El modelo de contenido podrá ser de tres tipos: “sin orden”, secuencial y alternativo/opcional.



UD 5 - VALIDACIÓN DE XML

5 - XSD

Veamos ejemplos de cada uno:

► “Sin orden”:

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

```
  <xs:complexType>
```

```
    <xs:all>
```

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

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

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

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

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

```
</xs:element>
```

XML

```
<ordenador>
  <procesador>Intel i5</procesador>
  <placa-base>NiSu 3000</placa-base>
  <memoria>Tracktor 23RDF</memoria>
</ordenador>
```

All exige que aparezcan los elementos una vez cada uno pero en cualquier orden.

UD 5 - VALIDACIÓN DE XML

5 - XSD

► Secuencial

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

XML

```
<producto>
```

```
  <nombre>Ratón Genius 23R</nombre>
```

```
  <precio>12.00</precio>
```

```
</producto>
```

```
  <xs:complexType>
```

```
    <xs:sequence>
```

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

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

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

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

```
</xs:element>
```

Sequence exige que aparezcan los elementos una vez cada uno y en el orden descrito.

1	_____
2	_____
3	_____

UD 5 - VALIDACIÓN DE XML

5 - XSD

► Opcional:

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

```
  <xs:complexType>
```

```
    <xs:choice>
```

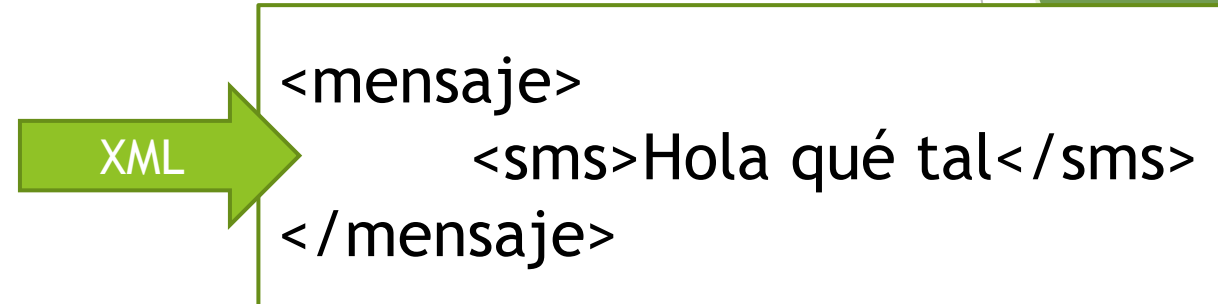
```
      <xs:element name="sms" type="xs:string"/>
```

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

```
    </xs:choice>
```

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

```
</xs:element>
```



Choice te obliga a escoger uno y solo uno de los elementos hijos que se definen

THE
CHOICE

UD 5 - VALIDACIÓN DE XML

5 - XSD

NUESTRO PRIMER XSD

```
<?xml version="1.0" encoding="UTF-8"?>  
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">  
  <xs:element name="nota">  
    <xs:complexType>  
      <xs:sequence>  
        <xs:element name="a" type="xs:string"/>  
        <xs:element name="de" type="xs:string"/>  
        <xs:element name="cabecera" type="xs:string"/>  
        <xs:element name="cuerpo" type="xs:string"/>  
      </xs:sequence>  
    </xs:complexType>  
  </xs:element>  
</xs:schema>
```

Tecleamos este documento en el XML Copy Editor y lo salvamos como nota.xsd

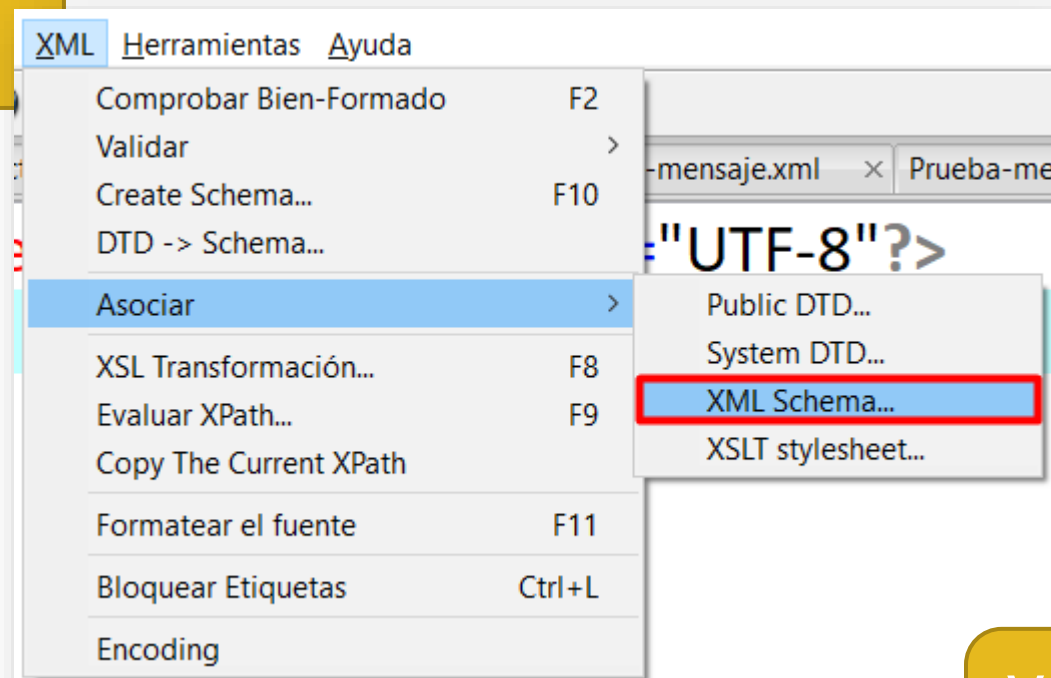
UD 5 - VALIDACIÓN DE XML

5 - XSD

CREAMOS UN XML Y LO ASOCIAMOS CON SU XSD

A continuación, crea un fichero `nota.xml` y escribe un contenido válido

Asocia el fichero XML al XSD mediante los siguientes elementos del menú:



Valida el documento XML y corrige los fallos, si los hay



UD 5 - VALIDACIÓN DE XML

5 - XSD

- Al asociar el XML con el XSD se insertan los siguientes atributos en el nodo raíz:

Establece el prefijo `xsi` y lo asocia al namespace de la URL.

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

Se usa la etiqueta `noNamespaceSchemaLocation` definida dentro del namespace identificado por `xsi` y, a continuación, pone la ruta (relativa o absoluta) al fichero XSD que servirá de validador.

Realizamos los ejercicios 1 y 2 del boletín de XSD



UD 5 - VALIDACIÓN DE XML

5 - XSD

NÚMERO DE OCURRENCIAS

Ahora queremos indicar cuántas veces se puede repetir un elemento. Para ello se usa la siguiente notación:

```
<xs:element name="articulo" type="xs:string" minOccurs="0" maxOccurs="10"/>
```

Si no especificamos alguno, se asume el valor por defecto 1

Indicamos que el artículo podrá aparecer de 0 a 10 veces

Si queremos que un elemento se pueda repetir un número ilimitado de veces entonces se usa:

`maxOccurs="unbounded"`

multiplicity

UD 5 - VALIDACIÓN DE XML

5 - XSD

Ejemplo de uso:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="familia">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="persona" minOccurs="2" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="nombre" type="xs:string"/>
              <xs:element name="edad" type="xs:integer"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<familia xmlns:xsi="http://www.w3.org/2001/XMLSchema"
  xsi:noNamespaceSchemaLocation="Familia.xsd">
  <persona>
    <nombre>Pablo Casado</nombre>
    <edad>40</edad>
  </persona>
  <persona>
    <nombre>Maria Cansada</nombre>
    <edad>39</edad>
  </persona>
  <persona>
    <nombre>Pedrito Casado Cansada</nombre>
    <edad>4</edad>
  </persona>
</familia>
```

UD 5 - VALIDACIÓN DE XML

5 - XSD

COMENTARIOS Y ANOTACIONES


Dado que nuestros XSD van ganando complejidad, podemos querer comentar alguna línea. Para ello, se usa la misma notación que en XML (coincide con la de DTD y HTML):

`<!-- Esto es un comentario`

`Y puede tener varias líneas -->`

También tenemos la posibilidad de **documentar** nuestro XSD con **anotaciones** formales del siguiente modo:

```
<xs:annotation>  
  <xs:documentation>Esquema XML de e-commerce</xs:documentation>  
  <xs:documentation>Author: Borja Mon De York</xs:documentation>  
  <xs:documentation>Revision: 16</xs:documentation>  
</xs:annotation>
```



Es información para los desarrolladores. Normalmente, el parser ignora esta información

UD 5 - VALIDACIÓN DE XML

5 - XSD

ELEMENTOS DE CONTENIDO MIXTO

Permiten mezclar datos y elementos hijos. Ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="raiz">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="elem-mixto">
          <xs:complexType mixed="true">
            <xs:choice minOccurs="0" maxOccurs="unbounded">
              <xs:element name="elem1" type="xs:string"/>
              <xs:element name="elem2" type="xs:string"/>
            </xs:choice>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Se usa el atributo `mixed="true"` en combinación con algún modelo de contenidos (all, sequence, choice)

XML

`<elem-mixto>` Ahora puedo alternar entre texto y etiquetas `<elem1>` de las que hemos definido `</elem1>` en nuestro `<elem2>` documento XSD `</elem2>` y también puedo `<elem1>` repetir etiquetas `</elem1>`
`</elem-mixto>`

UD 5 - VALIDACIÓN DE XML

5 - XSD

ELEMENTOS VACÍOS Y ELEMENTOS “ANY”

- Los **elementos vacíos** son los que no pueden tener contenido. Veamos la sintaxis:

```
<xs:element name="elem-vacio">  
  <xs:complexType/>  
</xs:element>
```

XML

```
<elem-vacio/>
```

Son elementos de poca utilidad o usados en etapas tempranas del diseño del XSD

- Los **elementos “any”** permiten que aparezca cualquiera de los elementos ya definidos en el documento:

```
<xs:element name="articulo">  
  <xs:complexType>  
    <xs:sequence>  
      <xs:element name="nombre" type="xs:string"/>  
      <xs:element name="precio" type="xs:decimal"/>  
      <xs:any/>  
    </xs:sequence>  
  </xs:complexType>  
</xs:element>
```

XML

```
<articulo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
  | | | | | xsi:noNamespaceSchemaLocation="Elementos-any.xsd">  
  <nombre>Tijeras</nombre>  
  <precio>11.95</precio>  
  <precio>9.95</precio>  
</articulo>
```

Realizamos el ejercicio 7 del boletín de XSD

UD 5 - VALIDACIÓN DE XML

5 - XSD

ATRIBUTOS

- La declaración de un atributo se realiza con la sintaxis:

```
<xs:attribute name="nombre-atrib" type="tipo"/>
```

- Ejemplo:

```
<xs:element name="persona" maxOccurs="unbounded">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="nombre" type="xs:string"/>
    </xs:sequence>
    <xs:attribute name="edad" type="xs:integer"/>
    <xs:attribute name="dni" type="xs:string"/>
  </xs:complexType>
</xs:element>
```

XML

```
<persona edad="40" dni="12345678A">
  <nombre>Daniel Casado</nombre>
</persona>
```

```
<persona edad="39">
  <nombre>María Cansada</nombre>
</persona>
```

OJO: Los atributos se definen a la misma altura que el modelo de contenido de los elementos

ATRIBUTO:
LA NIEVE ES
BLANCA

UD 5 - VALIDACIÓN DE XML

5 - XSD

- ▶ **Atributos obligatorios:** por defecto, el uso de los atributos en un elemento es opcional. Si queremos que sean obligatorios hay que especificarlo así:

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

- ▶ **Valor por defecto de un atributo:** es el valor que se aplica en el caso de que no se especifique un valor en el atributo.

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

- ▶ **Valor fijo de un atributo:** en este caso queremos especificar que un atributo es opcional pero que, en el caso de que aparezca, solo se permitirá el valor fijado.

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

UD 5 - VALIDACIÓN DE XML

5 - XSD

► Ejemplo de uso:

```
<xs:element name="personas">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="persona" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="nombre" type="xs:string"/>
          </xs:sequence>
          <xs:attribute name="edad" type="xs:integer"/>
          <xs:attribute name="dni" type="xs:string" use="required"/>
          <xs:attribute name="idioma" type="xs:string" default="español"/>
          <xs:attribute name="pais-residencia" type="xs:string" fixed="España"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

XML

```
<persona edad="40" dni="12345678A" pais-residencia="España">
  <nombre>Daniel Casado</nombre>
</persona>
<persona dni="55345678A" idioma="Portugués">
  <nombre>María Cansada</nombre>
</persona>
```

Realizamos los ejercicios 8 al 9 del boletín de XSD

UD 5 - VALIDACIÓN DE XML

5 - XSD

CREACIÓN DE TIPOS

- ▶ Hasta ahora hemos trabajado con los tipos básicos que incorpora XSD.
- ▶ Sin embargo, en ocasiones nos puede interesar crear nuestros propios tipos y utilizarlos tantas veces como queramos. Por ejemplo, podríamos crear un tipo llamado “**tipo-teléfono**” que sea capaz de almacenar números telefónicos con una determinada notación.
- ▶ Al igual que los elementos, los tipos pueden ser **simples** (si contienen un contenido o dato) o **complejos** (si contienen uno o más nodos hijos).
- ▶ Veamos cómo creamos cada uno de ellos:

**CREACIÓN
DE TIPOS**

UD 5 - VALIDACIÓN DE XML

5 - XSD

► Tipos simples:

- Se crean siempre tomando como base un tipo ya existente y modificándolo de algún modo. Ejemplo:

```
<xs:element name="persona" minOccurs="2" maxOccurs="unbounded">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="nombre" type="xs:string"/>
      <xs:element name="edad">
        <xs:simpleType>
          <xs:restriction base="xs:integer">
            <xs:minInclusive value="0"/>
            <xs:maxInclusive value="120"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Fíjate que en el elemento omitimos el atributo `type` porque lo declaramos justo después

Estamos definiendo el tipo “en-línea” y no lo “bautizamos” poniéndole un nombre. De esta forma si quisiéramos usarlo en otro nodo tendríamos que definirlo de nuevo.



XML

```
<persona>
  <nombre>Pablo Casado</nombre>
  <edad>40</edad>
</persona>
<persona>
  <nombre>Maria Cansada</nombre>
  <edad>39</edad>
</persona>
```

UD 5 - VALIDACIÓN DE XML

5 - XSD

- También podemos darle un nombre al tipo y usarlo cada vez que nos interese. Ejemplo:

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

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

Definimos el tipo después de la etiqueta `xs:schema` y le ponemos un nombre

```
<xs:element name="familia">  
  <xs:complexType>  
    <xs:sequence>  
      <xs:element name="persona" minOccurs="2" maxOccurs="unbounded">  
        <xs:complexType>  
          <xs:sequence>  
            <xs:element name="nombre" type="xs:string"/>  
            <xs:element name="edad" type="tipo-edad"/>  
          </xs:sequence>
```

Usamos el tipo creado tantas veces como queramos



UD 5 - VALIDACIÓN DE XML

5 - XSD

► Tipos complejos:

► En este caso podremos declarar elementos y atributos. Ejemplo:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:simpleType name="tipo-edad">
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="0"/>
      <xs:maxInclusive value="120"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="tipo-persona">
    <xs:sequence>
      <xs:element name="nombre" type="xs:string"/>
      <xs:element name="edad" type="tipo-edad"/>
    </xs:sequence>
  </xs:complexType>
```

Definimos el tipo después de la etiqueta `xs:schema` y le ponemos un nombre

```
<xs:element name="familia">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="persona" type="tipo-persona" minOccurs="2" maxOccurs="unbounded"/>
    </xs:sequence>
```

Usamos el tipo creado tantas veces como queramos

UD 5 - VALIDACIÓN DE XML

5 - XSD

► Aplicación de tipos sobre elementos y atributos:

- Hemos visto como crear un **tipo simple y complejo**, darle un nombre y aplicarlo sobre un elemento. Además, los **tipos simples pueden ser aplicados sobre atributos**. Ejemplo:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:simpleType name="tipo-edad">
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="0"/>
      <xs:maxInclusive value="120"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="tipo-persona">
    <xs:sequence>
      <xs:element name="nombre" type="xs:string"/>
    </xs:sequence>
    <xs:attribute name="edad" type="tipo-edad"/>
  </xs:complexType>
```

XML

```
<persona edad="4">
  <nombre>Pedrito Cansado</nombre>
</persona>
```

Realizamos los ejercicios
10 al 11 del boletín de XSD

UD 5 - VALIDACIÓN DE XML

5 - XSD

- **Restricciones sobre tipos simples:** XSD permite definir restricciones de un tipo simple para fabricar tipos de datos más precisos. Veamos los principales:
 - **Sobre valores numéricos:**

```
<xs:simpleType name="tipo-aleatorio">  
  <xs:restriction base="xs:decimal">  
    <!-- Valor >= 0.0 -->  
    <xs:minInclusive value="0.0"/>  
    <!-- Valor < 1.0 -->  
    <xs:maxExclusive value="1.0"/>  
  </xs:restriction>  
</xs:simpleType>
```

```
<xs:simpleType name="tipo-precio">  
  <xs:restriction base="xs:decimal">  
    <!-- Como máximo 2 decimales, pero pueden ser menos -->  
    <xs:fractionDigits value="2"/>  
    <!-- Como máximo 8 dígitos, pero pueden ser menos -->  
    <xs:totalDigits value="8"/>  
  </xs:restriction>  
</xs:simpleType>
```

```
<xs:simpleType name="tipo-num-dni">  
  <xs:restriction base="xs:integer">  
    <!-- Como máximo 8 dígitos, pero pueden ser menos -->  
    <xs:totalDigits value="8"/>  
  </xs:restriction>  
</xs:simpleType>
```

UD 5 - VALIDACIÓN DE XML

5 - XSD

► Sobre textos:

```
<xs:simpleType name="tipo-dni">  
  <xs:restriction base="xs:string">  
    <!-- Exactamente 9 caracteres-->  
    <xs:length value="9"/>  
  </xs:restriction>  
</xs:simpleType>
```

```
<xs:simpleType name="tipo-password">  
  <xs:restriction base="xs:string">  
    <xs:minLength value="6"/>  
    <xs:maxLength value="20"/>  
  </xs:restriction>  
</xs:simpleType>
```

```
<xs:simpleType name="tipo-estado-civil">  
  <xs:restriction base="xs:string">  
    <xs:enumeration value="Soltero/a"/>  
    <xs:enumeration value="Casado/a"/>  
  </xs:restriction>  
</xs:simpleType>
```

UD 5 - VALIDACIÓN DE XML

5 - XSD

- **Y PATRONES sobre cualquier tipo:** los patrones se construyen mediante “expresiones regulares” (regex). Estas expresiones siguen una notación ESTÁNDAR y permiten describir de forma precisa una combinación válida de caracteres en un texto.
- Veamos algunos ejemplos:

```
<xs:simpleType name="tipo-letra-dni">  
  <xs:restriction base="xs:string">  
    <!-- Solo vale una letra mayúscula -->  
    <xs:pattern value="[A-Z]"/>  
  </xs:restriction>  
</xs:simpleType>
```

```
<xs:simpleType name="tipo-palabra">  
  <xs:restriction base="xs:string">  
    <xs:pattern value="[A-Za-z]{1,120}"/>  
  </xs:restriction>  
</xs:simpleType>
```

```
<xs:simpleType name="tipo-iniciales-para-carta">  
  <xs:restriction base="xs:string">  
    <!-- Solo valen tres letras mayúsculas -->  
    <xs:pattern value="[A-Z][A-Z][A-Z]"/>  
  </xs:restriction>  
</xs:simpleType>
```

También se puede escribir así:
`<xs:pattern value="[A-Z]{3}"/>`

```
<xs:simpleType name="tipo-dni">  
  <xs:restriction base="xs:string">  
    <xs:pattern value="[0-9]{8}[A-Z]"/>  
  </xs:restriction>  
</xs:simpleType>
```

UD 5 - VALIDACIÓN DE XML

5 - XSD

► Y para rizar el rizo...

```
<xs:simpleType name="tipo-dominio">  
  <xs:restriction base="xs:string">  
    <!-- Ejemplo: mi.dominio.com -->  
    <xs:pattern value="([A-Za-z]+\.)+([A-Za-z]+)"/>  
  </xs:restriction>  
</xs:simpleType>
```

Los () actúan agrupando una expresión y el + indica que la expresión anterior puede repetirse 1 o más veces

El . es un símbolo del lenguaje de las expresiones regulares. Para poder referirnos al carácter '.' debemos "escaparlo del lenguaje", eso se hace poniéndole delante una \

```
<xs:simpleType name="tipo-email">  
  <xs:restriction base="xs:string">  
    <!--Y esta es la versión simple-->  
    <xs:pattern value="[a-zA-Z0-9\.]+"@([a-zA-Z]+\.)+[a-zA-Z]{2,4}"/>  
  </xs:restriction>  
</xs:simpleType>
```

Las expresiones regulares se usan en muchos ámbitos. Si necesitáis aprender más, esta web es estupenda para practicar:
<https://regexr.com/>

Realizamos los ejercicios 12 y 13 del boletín de XSD

UD 5 - VALIDACIÓN DE XML

Anexo - Estrategias de diseño de un XSD

ANEXO - ESTRATEGIAS DE DISEÑO DE UN XSD

- ▶ El diseño de un XSD puede ser complicado de escribir y de leer. Por ese motivo vamos a ver tres modelos de diseño para facilitarnos la vida.
- ▶ **Modelo de elementos anidados:** se basa en declarar “sobre la marcha” los elementos tal y como se vayan necesitando produciéndose un código con muchos elementos contenidos dentro de otros (también se le conoce como “modelo de muñecas rusas”). Se caracteriza por:
 - ▶ No se suelen definir tipos con nombre, de modo que no hay reutilización de elementos ya definidos.
 - ▶ Genera un código difícil de leer y mantener.
 - ▶ Sólo es apropiado para modelos de datos de pequeño tamaño o en etapas tempranas de un proyecto.



Es el que hemos
usados al comienzo
del apartado de XSD

UD 5 - VALIDACIÓN DE XML

Anexo - Estrategias de diseño de un XSD

- **Modelo basado en tipos con nombre:** se basa en declarar todos los tipos de datos, tanto simples como complejos, que se vayan a utilizar al principio del documento y después definir los elementos haciendo uso de los tipos declarados.

Se caracteriza por:

- Se posibilita la reutilización de tipos ya definidos en el documento.
- Genera un código más fácil de leer y mantener.
- Apto para modelos de datos de tamaño de cualquier tamaño.
- Se pueden “aislar” los tipos definidos en un XSD de forma que otro documento XSD pueda usarlos como punto de partida.

Este modelo lo hemos usado al resolver el ejercicio 13 del boletín



UD 5 - VALIDACIÓN DE XML

Anexo - Estrategias de diseño de un XSD

- **Modelo basado en referencias a elementos:** se basa en declarar todos los elementos que se vayan a utilizar en el documento y después se hace referencia a ellos desde el elemento raíz. Usa el atributo *ref* para hacer la referencia a un elemento ya definido:

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

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

Primero define
un elemento

```
<xs:element name="familia">  
  <xs:complexType>  
    <xs:sequence>  
      <xs:element ref="persona" minOccurs="2" maxOccurs="unbounded"/>  
    </xs:sequence>  
  </xs:complexType>  
</xs:element>  
</xs:schema>
```

Después “refiere” al
elemento ya declarado

Dado que hay
varios elementos
“dispersos” por
el esquema, se
considera como
ELEMENTO RAÍZ
aquel que no es
referido por
ningún otro
elemento

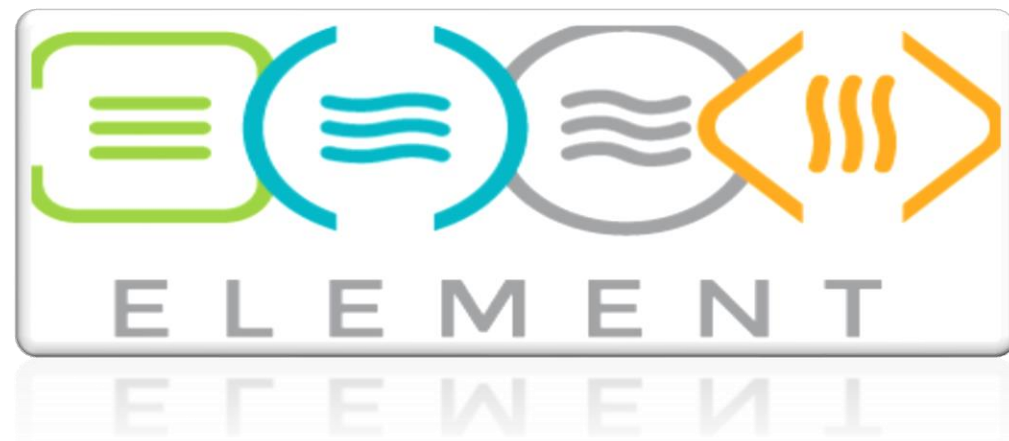


UD 5 - VALIDACIÓN DE XML

Anexo - Estrategias de diseño de un XSD

Este modelo presenta unas características similares al “modelo de tipos con nombre”:

- Posibilita la reutilización de elementos ya definidos en el documento.
- Genera un código más fácil de leer y mantener.
- Apto para modelos de datos de tamaño de cualquier tamaño.
- Se pueden “aislar” los elementos definidos en un XSD de forma que otro documento XSD pueda usarlos como punto de partida (lo estudiamos en el siguiente apartado).



Realizamos el ejercicio 14
del boletín de XSD

UD 5 - VALIDACIÓN DE XML

Anexo - Estrategias de diseño de un XSD

MODELO DE TIPOS VS MODELO DE REFERENCIAS

- La diferencia entre ambos es sutil:

Modelo de tipos con nombre

Se aísla el tipo y se le da un nombre. Esto permite crear varios elementos con el mismo tipo pero con nombres diferentes.

```
<xs:complexType name="tipo-importe">
  <xs:sequence>
    <xs:element name="importe" type="xs:decimal"/>
    <xs:element name="moneda" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```



```
<xs:element name="pedido">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="gastos-compra" type="tipo-importe"/>
      <xs:element name="gastos-envio" type="tipo-importe"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Modelo de referencias a elementos

Al definir un elemento se le pone un nombre, obligando a utilizar ese mismo nombre para todos los elementos cada vez que se use la referencia. Esto impide que dos nodos que refieran al mismo elemento puedan tener el mismo “padre”.

```
<xs:element name="gasto">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="importe"
        type="xs:decimal"/>
      <xs:element name="moneda"
        type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```



```
<xs:element name="pedido">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="gastos-compra">
        <xs:complexType>
          <xs:sequence>
            <xs:element ref="gasto"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="gastos-envio">
        <xs:complexType>
          <xs:sequence>
            <xs:element ref="gasto"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

UD 5 - VALIDACIÓN DE XML

Anexo - Estrategias de diseño de un XSD

CONCLUSIONES SOBRE LOS DISTINTOS MODELOS DE DISEÑO:

- ▶ El menos recomendable es el modelo de “elementos anidados”.
- ▶ El modelo basado en tipos con nombre es el más versátil.
- ▶ La diferencia entre los modelos de “referencias” y de “tipos” es bastante sutil. La elección de uno u otro casi que se podría reducir a una cuestión de “estilo” o de “gustos personales”.
- ▶ El lenguaje permite hacer también modelos “híbridos” entre cualesquiera de los tres modelos estudiados.

CONCLUSION



UD 5 - VALIDACIÓN DE XML

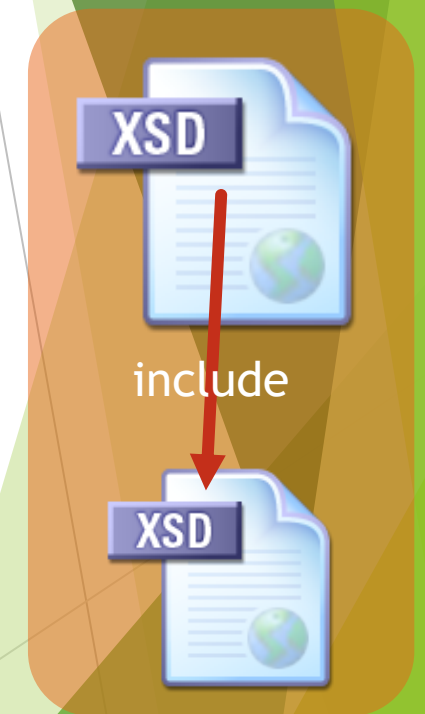
Anexo - Estrategias de diseño de un XSD

INCLUSIÓN DE UN XSD EN OTRO XSD

- ▶ Tanto el “modelo de referencias” como el “modelo de tipos” permiten realizar la separación de elementos o tipos que se repiten con frecuencia en un archivo XSD separado. De este modo, se podrán compartir estas definiciones entre varios archivos XSD.
- ▶ Este efecto se consigue mediante una etiqueta que permite incluir un XSD en otro. Su sintaxis es:

```
<xs:include schemaLocation="definiciones-comunes.xsd"/>
```

- ▶ Veamos un ejemplo para que se entienda mejor:



UD 5 - VALIDACIÓN DE XML

Anexo - Estrategias de diseño de

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
3
4 <xs:include schemaLocation="tipos-comunes.xsd"/>
5
6 <xs:element name="persona">
7   <xs:complexType>
8     <xs:sequence>
9       <xs:element name="nombre" type="xs:string"/>
10      <xs:element name="apellidos" type="xs:string"/>
11      <xs:element name="direccion" type="tipo-direccion"/>
12    </xs:sequence>
13  </xs:complexType>
14 </xs:element>
15 </xs:schema>
```

```
tipos-comunes.xsd
1 <?xml version="1.0" encoding="UTF-8"?>
2 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
3   <xs:complexType name="tipo-direccion">
4     <xs:sequence>
5       <xs:element name="calle" type="xs:string"/>
6       <xs:element name="numero" type="xs:string"/>
7       <xs:element name="localidad" type="xs:string"/>
8       <xs:element name="municipio" type="xs:string"/>
9     </xs:sequence>
10  </xs:complexType>
11 </xs:schema>
```

XML

```
persona.xml
1 <?xml version="1.0" encoding="UTF-8"?>
2 <persona xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:noNamespaceSchemaLocation="persona.xsd">
4   <nombre>Yola</nombre>
5   <apellidos>Prieto Bastante</apellidos>
6   <direccion>
7     <calle>Sol</calle>
8     <numero>17</numero>
9     <localidad>Sevilla</localidad>
10    <municipio>Sevilla</municipio>
11  </direccion>
12 </persona>
```

Realizamos el ejercicio 15
del boletín de XSD

UD 5 - VALIDACIÓN DE XML

Anexo - Estrategias de diseño de un XSD

TRABAJO CON NAMESPACES

- ▶ Durante toda la unidad hemos ignorado deliberadamente el uso de *namespaces*. Recordamos que éstos se usan **para evitar colisiones de nombres** cuando se fusionan documentos XML realizados por distintos equipos humanos.
- ▶ El uso de los *namespaces* complica la sintaxis y la comprensión de los documentos XSD y XML, quedando fuera del ámbito de alcance de este módulo profesional.
- ▶ No obstante, si se quiere profundizar a este respecto se recomienda la lectura de este artículo:

<https://www.liquid-technologies.com/xml-schema-tutorial/xsd-namespaces>

Namespace A

<nombre>
<dirección>
<edad>
<precio>

Namespace B

<nombre>
<pedido>
<direccion>
<precio>

UD 5 - VALIDACIÓN DE XML

