

## 1.- Documentos XML.

Hasta ahora hemos trabajado con documentos básicos XML. Sólo declarábamos el tipo de documento que iba a ser (el ejemplar) pero no definíamos que cualidades tenía ese tipo de documento.

Ya vimos que un documento XML básico estaba formado por un **prólogo** y por un **ejemplar**.

# Prólogo

Informa al intérprete encargado de procesar el documento de todos aquellos datos que necesita para realizar su trabajo. Consta de dos partes:

- **Definición de XML.** Indicamos la versión de XML, la codificación y la autonomía. Hasta ahora todos los generados han sido independientes.

*<?xml version="1.0" encoding="utf-8" standalone="yes" ?>*

- **Declaración del tipo de documento.** Hasta ahora sólo indicábamos  
<!DOCTYPE nombre\_del\_ejemplar>

**<!DOCTYPE biblioteca>**

# Ejemplar

Contiene los datos del documento que se quiere procesar. **Es el elemento raíz y debe ser único.**

Está compuesto de elementos estructurados según una estructura de árbol. Los elementos pueden contener atributos.

```
<biblioteca>
  <libro>
    <titulo edición="2" pagina="540">La búsqueda</titulo>
    <autor>Sebastián Torres</autor>
    <editorial>Ediciones Plum</editorial>
    <isbn>978-2-7460-4344-1</isbn>
  </libro>
</biblioteca>
```

## **Documentos bien formados.**

Son sintácticamente correctos. Cumplen las reglas de sintaxis del lenguaje.

## **Documentos válidos.**

Además de bien formados cumplen los requisitos de la definición de estructura que se haya indicado en la definición del documento.

Es decir que el ejemplar debe ceñirse a un formato y estructura definido previamente. Necesitamos entonces especificar este formato de comunicación en XML.

## Lenguaje de marcas XML

Si cumple la sintaxis decimos **está bien formado**.

**Para definir la estructura de datos podemos usar DTD o XML Schema**

Si el documento XML cumple lo especificado en el DTD o XML Schema decimos que **es válido**.

## 2.- Declaración de tipos de documentos XML.

El DTD (*Document Type Definition*) establece que elementos son aceptados y en qué posición deben estar dentro de un documento XML.

La declaración de una DTD puede establecerse de dos formas.

1. **Interna.** La DTD se ubica dentro del propio documento XML.
2. **Externa.** La DTD se ubica en un archivo externo al documento XML.

Para validar más de un documento XML con la misma DTD debemos usar la *externa* para no tener que repetir la DTD dentro de cada documento XML. En el caso de que la DTD solo se utilice para validar un único documento XML, la DTD es habitual escribirla *internamente*.

**Interna.** Para referenciarlo como DTD interno, el atributo *standalone* en la declaración XML se debe marcar con un **Sí**. Esto significa que la declaración funciona al margen de la fuente externa.

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

```
<!DOCTYPE autor[
```

```
  <!ELEMENT autor (nombre, empresa, teléfono)>
```

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

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

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



```
<autor>
```

```
  <nombre>Tanmay Patil</nombre>
```

```
  <empresa>TutorialsPoint</empresa>
```

```
  <teléfono>(011) 123-4567</teléfono>
```

```
</autor>
```

```
<!DOCTYPE root-element [element-declarations]>
```

**01interna.xml**

Comprobación en XML Copy Editor

# Externa

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<!DOCTYPE autor SYSTEM "02autor.dtd">
<autor>
  <nombre>Tanmay Patil</nombre>
  <empresa>TutorialsPoint</empresa>
  <teléfono>(011) 123-4567</teléfono>
</autor>
```

*En este caso, sólo se admitirá en el artículo, un autor con su nombre, empresa y teléfono.*

```
<!ELEMENT autor (nombre, empresa,
teléfono)>
```

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

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

```
<!ELEMENT teléfono (#PCDATA)>
```

```
<!DOCTYPE root-element SYSTEM "file-name">
```

**02externa.xml**

**02autor.dtd**

Comprobación en XML Copy Editor



## Externa

Podemos referirnos a un DTD externo usando una de las dos posibilidades.

### Identificadores de sistema

Permite especificar la localización de un archivo externo que contiene declaraciones DTD

```
<!DOCTYPE name SYSTEM "address.dtd" [...]>
```



URI que identifica un recurso, es decir una **URL**

### Identificadores públicos

Los identificadores públicos aportan un mecanismo de localización de recursos DTD

```
<!DOCTYPE name PUBLIC "-//Beginning XML//DTD Address Example//EN">
```

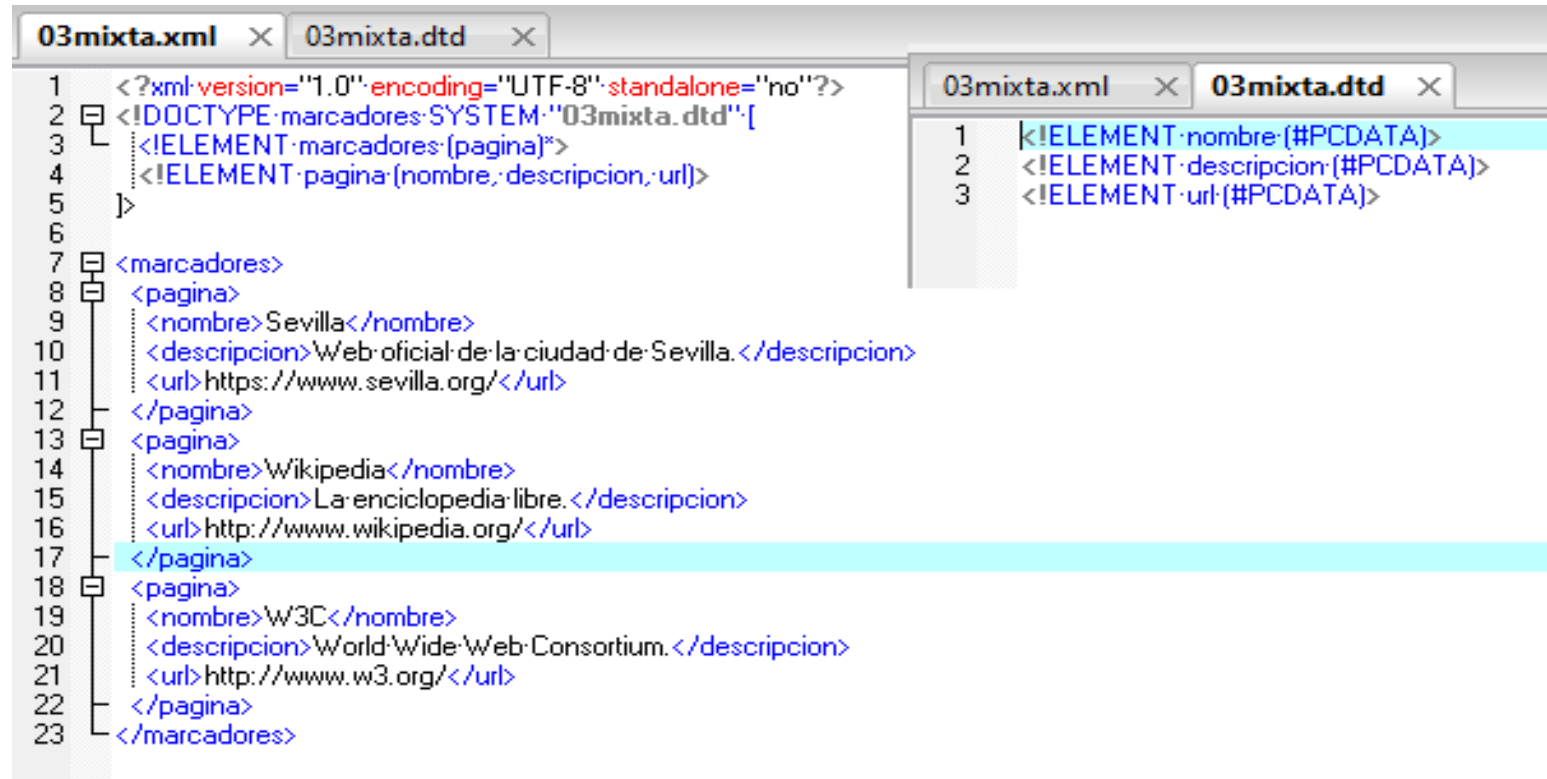


identificador público formal (*FPI*).

cadena de caracteres que se utiliza para identificar un producto, una especificación o un documento.

# Combinando Interna / Externa

Existe la posibilidad de combinar ambas propuestas. Se concatenarían las líneas de código. Si una regla aparece en ambos lugares, las internas tienen prioridad sobre las externa.



The screenshot shows two windows in XML Copy Editor. The left window, titled '03mixta.xml', contains an XML document with a DOCTYPE declaration pointing to '03mixta.dtd'. It lists three pages: Sevilla, Wikipedia, and W3C. The right window, titled '03mixta.dtd', shows the DTD definitions for the elements used in the XML: 'nombre', 'descripcion', and 'url'.

```
03mixta.xml
1 <?xml:version="1.0" encoding="UTF-8" standalone="no"?>
2 <!DOCTYPE marcadores SYSTEM "03mixta.dtd"[
3   <!ELEMENT marcadores (pagina)*>
4   <!ELEMENT pagina (nombre, descripcion, url)>
5 ]>
6
7 <marcadores>
8   <pagina>
9     <nombre>Sevilla</nombre>
10    <descripcion>Web oficial de la ciudad de Sevilla.</descripcion>
11    <url>https://www.sevilla.org/</url>
12  </pagina>
13  <pagina>
14    <nombre>Wikipedia</nombre>
15    <descripcion>La enciclopedia libre.</descripcion>
16    <url>http://www.wikipedia.org/</url>
17  </pagina>
18  <pagina>
19    <nombre>W3C</nombre>
20    <descripcion>World Wide Web Consortium.</descripcion>
21    <url>http://www.w3.org/</url>
22  </pagina>
23 </marcadores>
```

```
03mixta.dtd
1 <!ELEMENT nombre (#PCDATA)>
2 <!ELEMENT descripcion (#PCDATA)>
3 <!ELEMENT url (#PCDATA)>
```

**03mixta.xml**

**03mixta.dtd**

Comprobación en  
XML Copy Editor

Ejemplo. Documento XML que guarda *sms* ( cero o más de uno)

***04EjemploSMS.xml***

*<!ELEMENT BDsms (sms\*)>*

***04BSsms.dtd***

Ejemplo. Creación de un DTD y documento XML que recoja información (nombre y dirección) de diferentes alumnos de una clase.

***05Ejemplo.xml***

***05alumno.dtd***

Los bloques de un documento DTD:

- Entidades. Son caracteres especiales que permiten incluir datos que podrían confundirse con entidades propias del lenguaje de marcado. También puede usarse para predefinir valores (constantes). **!ENTITY**
- Elemento. Es el bloque principal con el que se desarrollan los documentos. **!ELEMENT**
- Atributos. Es la manera de añadir más información a un elemento. **!ATTLIST**

# Entidades (*!ENTITY*)

Espacio en blanco	&nbsp;
>	&gt;
<	&lt;
“	&quot;
‘	&apos;
&	&amp;

## Entidades generales internas analizables

<!ENTITY pi “3,141592”>	&pi;
<!ENTITY alf “Alien Life Form”>	&alf;

**06entidadesUTF.xml**

**06entidadesISO.xml**

Una **sección CDATA** es una etiqueta que comienza por <![CDATA[ y termina por ]]> y cuyo contenido el procesador XML no interpreta como marcas sino como texto. Es una alternativa en los textos con muchas entidades como &lt; y &amp; que dificultan la lectura del documento

## Entidades (***!ENTITY***)

Uso de una entidad dentro de otra. En la DTD se han declarado dos entidades generales internas analizables (color y frase). La primera aparece como parte del valor de la segunda.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!DOCTYPE frase [
  <!ELEMENT frase (#PCDATA)>
  <!ENTITY color "azul">
  <!ENTITY frase "El cielo es &color;.">
]>
<frase>&frase;
</frase>
```

***06entidadesanidadas.xml***

## Entidades (***!ENTITY***)

### *Entidades paramétricas internas analizables*

Nos permite dar nombres a partes de un DTD y hacer referencia a ellas a lo largo del mismo como una constante. Útiles cuando varios elementos del DTD comparten listas de atributos o especificaciones de contenidos. Se denotan por **%entidad;**

```
<!ENTITY % dirección "calle, número?, ciudad, cp">
```

```
<!ELEMENT beca (alumno, ies)>
```

```
<!ELEMENT alumno (dni, %dirección;)>
```

```
<!ELEMENT ies (nombre, %dirección;)>
```

***06Entidadesparametricas.xml***

***06Entidadesparametricas.dtd***

## Elementos (**!ELEMENT**)

Elementos NO Terminales. Definimos elementos que están compuestos por otros elementos hijos.

*<!ELEMENT A (B, C)>*, el elemento A está formado por un elemento B seguido de uno C

*<!ELEMENT clase (alumno)>*

*<!ELEMENT alumno (nombre, dirección)>*

Operadores.	Opcional ?	<i>&lt;!ELEMENT teléfono (trabajo?, casa)&gt;</i>
	Uno o más +	<i>&lt;!ELEMENT provincia (nombre, (cp, ciudad)+ )&gt;</i>
	Cero o más *	<i>&lt;!ELEMENT provincia (nombre, (cp, ciudad)* )&gt;</i> <i>&lt;!ELEMENT clase (alumno*)&gt;</i>
	Elección	<i>&lt;!ELEMENT alumno (nombre, (teléfono   correo) )&gt;</i>



## Ejemplo. Operadores.

<!ELEMENT clase (profesor+, alumno\*)>

<!ELEMENT profesor (#PCDATA)>

**<!ELEMENT alumno (nombre, (apellido1, apellido2)?, dirección, correo, teléfono?, (nif | nie), materias)>**

<!ELEMENT nombre (#PCDATA)>

<!ELEMENT apellido1 (#PCDATA)>

<!ELEMENT apellido2 (#PCDATA)>

<!ELEMENT dirección (#PCDATA)>

<!ELEMENT correo (#PCDATA)>

<!ELEMENT teléfono (#PCDATA)>

<!ELEMENT nif (#PCDATA)>

<!ELEMENT nie (#PCDATA)>

<!ELEMENT materias (asignatura+)>

<!ELEMENT asignatura (#PCDATA)>

***08operadores.xml***

***08operadores.dtd***

## Elementos (!ELEMENT)

Elementos Terminales. Se corresponden con hojas de la estructura del árbol formada por los datos del documento XML.

`<!ELEMENT nombre_del_elemento declaración_del_contenido>`

La declaración de contenido puede tener uno de los siguientes valores:

**EMPTY**. El elemento está vacío. Por ejemplo `<!ELEMENT br EMPTY>`    `<br/>`  
Añadimos al ejercicio 05

**ANY**. Admite como elemento cualquiera de los elementos declarados.  
No suele usarse salvo para pruebas

**(#PCDATA)**. Almacena texto entre la etiqueta de apertura y la de cierre

*Parser Carácter Data*

# Elementos (!ELEMENT)

(#PCDATA) en una lista de opciones permite contenido mixto

***09CDATAmixto.xml***

No se recomienda porque pierde rigidez la estructura.

Ediciones Informe regional del ventas		
Descripción: informe de ventas para las regiones Norte, Centro y Sur		
Fecha del informe: 30/12/2009		
Region	Trimestre	Libros Vendidos
Norte	1	24000
	2	38600
	3	NO_INFO
	4	NO_INFO
Centro	1	NO_INFO
	2	16080
	3	25000
	4	29000
Sur	1	27000
	2	31400
	3	40100
	4	30000

Se deben tener en cuenta las siguientes consideraciones:

- Es obligatorio que el informe lleve una fecha
- Se debe poder diferenciar la parte de la cabecera del informe de la parte con los datos
- Siempre deben aparecer las tres regiones en el informe, y ninguna más
- Para cada zona deben aparecer siempre los cuatro trimestres, aunque falte la información sobre los libros vendidos
- Si no se incluye el número de libros vendidos en los datos, en el informe aparecerá la cadena NO\_INFO
- El número de trimestre sólo puede tomar los valores 1, 2, 3 o 4

***07ediciones.xml***

Empleando sólo !ELEMENT



***07ediciones.dtd***

# Atributos (!ATTLIST)

`<!ATTLIST nombre-del-elemento nombre-del-atributo tipo-de-atributo valor-del-atributo>`

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

`<!DOCTYPE deportistas [`

`<!ELEMENT deportistas (futbol | F1 | tenis)*>`

`<!ELEMENT futbol (#PCDATA)>`

`<!ELEMENT F1 (#PCDATA)>`

`<!ATTLIST F1 país CDATA "España">`

`<!ELEMENT tenis (#PCDATA)>`

`]>`

`<deportistas>`

`<F1 país="Alemania">Sebastian Vettel</F1>`

`<F1>Fernando Alonso</F1>`

`<tenis>Rafael Nadal</tenis>`

`</deportistas>`

## 10Atributos1.xml

Para el elemento F1, país es un atributo definido de tipo CDATA (*Character DATA*), es decir, su valor será una cadena de caracteres.

El valor por defecto es "España" si no se especifica.

```
<deportistas>
  <F1 país="Alemania">Sebastian Vettel</F1>
  <F1 país="España">Fernando Alonso</F1>
  <tenis>Rafael Nadal</tenis>
</deportistas>
```

# Atributos (!ATTLIST)

## Declaración de varios atributos en un elemento

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

```
<!DOCTYPE deportistas [
```

```
  <!ELEMENT deportistas (futbol | F1 | tenis)*>
```

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

```
  <!ELEMENT F1 (#PCDATA)>
```

```
    <!ATTLIST F1 pais CDATA #FIXED "España">
```

```
    <!ATTLIST F1 fecha_de_nacimiento CDATA #IMPLIED>
```

```
    <!ATTLIST F1 equipo CDATA #REQUIRED>
```

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



```
<deportistas>
```

```
  <F1 fecha_de_nacimiento="03/07/1987" equipo="Ferrari">Sebastian
```

```
Vettel</F1>
```

```
  <F1 equipo="McLaren">Fernando Alonso</F1>
```

```
  <tenis>Rafael Nadal</tenis>
```

```
</deportistas>
```

## 10Atributos2.xml

El atributo equipo es obligatorio escribirlo, **#REQUIRED**.

El atributo fecha\_de\_nacimiento es opcional, **#IMPLIED**.

Por **#FIXED** todos los elementos F1 que aparezcan tendrán el atributo país con el valor "España"

```
<deportistas>
  <F1 fecha_de_nacimiento="03/07/1987" equipo="Ferrari" pais="España">Sebastian Vettel</F1>
  <F1 equipo="McLaren" pais="España">Fernando Alonso</F1>
  <tenis>Rafael Nadal</tenis>
</deportistas>
```

## Tipos de atributos en DTD

Tipo	Descripción
CDATA	( <i>Character DATA</i> ) El valor son datos de tipo carácter, es decir, texto.
Enumerado	El valor puede ser uno de los pertenecientes a una lista de valores escritos entre paréntesis "()" y separados por el carácter " ".
ID	El valor es un identificador único.
IDREF	El valor es un identificador que tiene que existir en otro atributo ID del documento XML.
IDREFS	El valor es una lista de valores que existan en otros atributos ID del documento XML, separados por espacios en blanco.
NMTOKEN	El valor es una cadena de caracteres, pudiendo contener letras minúsculas, letras mayúsculas, números, puntos ".", guiones medios "-", guiones bajos "_" o el carácter dos puntos ":".
NMTOKENS	El valor puede contener uno o varios valores de tipo NMTOKEN separados por espacios en blanco.
NOTATION	El valor es el nombre de una notación.
ENTITY	El valor es el nombre de una entidad.
ENTITIES	El valor puede contener uno o varios valores de tipo ENTITY separados por espacios en blanco.
Especiales	Existen dos atributos especiales: <i>xml:lang</i> y <i>xml:space</i> .

# Atributos (!ATTLIST)

## Declaración de varios atributos en un elemento

<!ATTLIST ciudad país **CDATA** #REQUIRED>

El valor del atributo país puede estar vacío.

***10Atributos3.xml***

<!ATTLIST F1 país (ESP | FRA | ITA | ALE) "ESP">

**Enumerados.** Se puede añadir uno por defecto y no sería necesario ponerlo.

***10Atributos4.xml***

<!ATTLIST F1 país (ESP | FRA | ITA | ALE) #REQUIRED>

**Enumerados.** Obligatorio. Hay que especificarlo.

<!ATTLIST F1 código **ID** #REQUIRED>

Cada elemento escrito en un documento XML sólo puede tener un atributo ID.  
En un documento XML, no pueden escribirse dos elementos que tengan el mismo valor en un atributo ID, aunque dicho atributo sea distinto  
Todo atributo declarado de tipo ID tiene que ser #IMPLIED (opcional) o #REQUIRED (obligatorio).

# Atributos (!ATTLIST)

<!ATTLIST director coddir ID #REQUIRED>

<!ATTLIST película dirección **IDREF**>

Los elementos película que se escriban deben incluir el atributo dirección cuyo valor estará asignado a un atributo ID de otro elemento del documento. En este caso, el valor estará asignado a un atributo coddir de un elemento director.

***10Atributos5.xml***

<!ATTLIST pelicula codpel ID>

<!ATTLIST director filmografia **IDREFS**>

Similar a IDREF pero puede incluirse una lista de valores de atributos ID

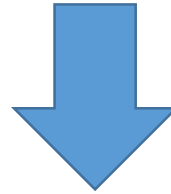
***10Atributos6.xml***



## Atributos (!ATTLIST)

Declaración de varios atributos en un elemento

```
<!ATTLIST película id_película ID #REQUIRED>  
<!ATTLIST película valoración CDATA "">  
<!ATTLIST película caratula ENTITY #IMPLIED>
```



```
<!ATTLIST película  
  id_película ID #REQUIRED  
  valoración CDATA ""  
  caratula ENTITY #IMPLIED>
```

# Notaciones (!NOTATION)

Las notaciones se pueden utilizar para especificar el formato de entidades externas (datos no XML), como por ejemplo un archivo que contenga una imagen. Dichas entidades externas no las analizará un procesador XML, sino que serán tratadas por el programa que procese el documento. También pueden emplearse para indicar que aplicación debe tratar ese archivo.

`<!NOTATION nombre-de-la-notación SYSTEM "identificador-del-sistema">`

```
<?xml version="1.0" encoding="UTF-8"
standalone="no"?>
```

```
<!DOCTYPE frutas [
  <!ELEMENT frutas (fruta)*>
  <!ELEMENT fruta EMPTY>
```

```
<!ATTLIST fruta foto ENTITY #REQUIRED>
<!ENTITY manzana SYSTEM "manzana.gif" NDATA gif>
<!ENTITY naranja SYSTEM "naranja.gif" NDATA gif>
<!NOTATION gif SYSTEM "image/gif">
```



```
<frutas>
  <fruta foto="manzana"/>
  <fruta foto="naranja"/>
</frutas>
```

Atributo tipo ENTITY, debe ser una de las entidades definidas.

## Notaciones (!NOTATION)

<!-- Notation que indica el programa para procesar jpg -->

<!NOTATION jpg SYSTEM "aplicaciones/visordeimagenes.exe">

<!-- Entidades con las imágenes . Importante indicar el camino relativo->

<!ENTITY foto\_p0360 SYSTEM "imagenes/p0360.jpg" NDATA jpg>

<!ENTITY foto\_p0437 SYSTEM "imagenes/p0437.jpg" NDATA jpg>

<!ENTITY foto\_p1201 SYSTEM "imagenes/p1201.jpg" NDATA jpg>

# Secciones condicionales

En DTD externas se pueden definir las secciones **IGNORE** e **INCLUDE**, para ignorar o incluir declaraciones. Tiene preferencia el IGNORE frente al INCLUDE

```
<!-- Mensaje largo -->
```

```
<![ INCLUDE [
```

```
<!ELEMENT mensaje (titulo, emisor, receptor, contenido, palabras)>
```

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

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

```
]]>
```

```
<!-- Mensaje corto -->
```

```
<![ IGNORE [
```

```
<!ELEMENT mensaje (emisor, receptor, contenido)>
```

```
]]>
```

```
<!-- Declaración de elementos comunes -->
```

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

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

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

***12seccionescondicionales.xml***

***12seccionescondicionales.dtd***

# Entidades (**!ENTITY**)

*Entidades generales externas no analizables*

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
```

```
<!DOCTYPE imagen [  
  <!ELEMENT imagen EMPTY>  
  <!ATTLIST imagen fuente ENTITY #REQUIRED>
```

Si es una URL pública se emplearía  
*PUBLIC "-//W3C//GIF logo//EN"*  
*"http://www.abrirllave.com/dtd/logo.gif"*

```
<!ENTITY logo SYSTEM "logo.gif" NDATA gif>
```

```
<!NOTATION gif SYSTEM "image/gif">  
>
```

```
<imagen fuente="logo"/>
```

Con NDATA (*Notation Data*) se indica que la entidad no es analizable. Caso de archivos con formatos binarios.

Indicamos que el elemento "imagen" que se incluya tiene que incluir obligatoriamente el atributo fuente cuyo valor será una entidad.

La notación gif es una declaración del tipo *MIME image/gif*

***13entidadNDATA.xml***

# COMENTARIOS

En una DTD se pueden escribir comentarios como en XML.

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

```
<!DOCTYPE ciudades [<!-- Ejemplo de documento XML con comentarios en su DTD
interna-->
```

```
<!ELEMENT ciudades (ciudad*)>
```

También en los DTD externos.

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

```
<!-- país es atributo del elemento ciudad -->
```

```
<!ATTLIST ciudad país CDATA #REQUIRED>
```



```
<ciudades>
```

```
<ciudad país="Italia">Roma</ciudad> <!-- comentario -->
```

```
<ciudad país="Francia">París</ciudad>
```

```
<ciudad país="">Viena</ciudad>
```

```
</ciudades>
```

***14Comentarios.xml***

## 2.- XML Schema. Esquemas.

Hemos estado generando documentos XML con la definición de reglas que lo componen (DTD). Otra manera de formalizar estas reglas es usando los XML Schema (llamados XSD, XML Schema Definition).

Surgió en 1998 y se recomendó su uso en 2001 por el W3C.

La diferencia con los DTD es que los esquemas están basados en XML y permiten definir de manera muy clara los tipos de datos.



Si un elemento se sabe que es de tipo *double* pero luego se escribe como *string* o *integer* no se admitirá la validez de un documento XML hasta que se corrija ese error.

Las características principales de un esquema son las siguientes:

- Define que elementos pueden aparecer en un documento XML.
- Define que atributos pueden aparecer en un documento XML.
- Define que elementos son compuestos, cuales son sus hijos y en el orden en que aparecen.
- Define que elementos pueden estar vacíos.
- **Define los tipos que pueden emplearse en cada elemento o atributo.**
- Define la obligatoriedad, opcionalidad de elementos y atributos.



**Diferencia básica con los documentos DTD.**



Los esquemas son documentos de texto plano con la extensión .xsd que se vincularán a un documento .xml

## DOCUMENTO XML

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

<curso xmlns:xsi="<http://www.w3.org/2001/XMLSchema-instance>"  
xsi:noNamespaceSchemaLocation="fichero.xsd">

... árbol de elementos ...

</curso>

PRÓLOGO

RAÍZ

Si el fichero .xsd es local

01EjemploXSD.xml

Hay que **indicar que vamos a emplear instancias del espacio de nombres mediante el atributo "xmlns:"**. Normalmente a este espacio de nombres **se le asigna el prefijo "xsi"** (aunque se puede usar cualquier prefijo)

Si el archivo está en un servidor externo

xsi:SchemaLocation="http://www.gmr.v.es fichero.xsd">

## DOCUMENTO XSD

The diagram shows an XSD document structure with the following XML code and annotations:

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

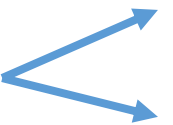
Annotations:

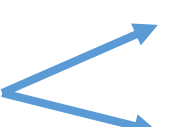
- PRÓLOGO**: Points to the XML declaration `<?xml version="1.0" encoding="UTF-8"?>`.
- RAÍZ <xs:schema>**: A bracket groups the `<xs:schema>` element and its children.
- 01EjemploXSD.xsd**: The filename of the document.

Especificamos el espacio de nombres mediante el atributo “xmlns:” y el prefijo “xs:”

Especificamos que tendrá un elemento sólo llamado “curso”.

Podemos indicar si en el documento es necesario especificar el espacio de nombres en los elementos y/o atributos mediante un prefijo.

**elementFormDefault**  *qualified*  
*unqualified (por defecto)*

**attributeFormDefault**  *qualified*  
*unqualified (por defecto)*

## Elementos simples en documento XSD

Llamamos elemento simple a todo aquel que puede contener información y que no tiene elementos hijos asociados ni atributos. Las hojas del árbol.

`<xs:element name="nombre_elemento_XML" type="tipo_elemento">`

`<nombre>Casimiro</nombre>`

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



Cadena de caracteres

`<fechaNac>2001-11-23</fechaNac>`

`<fechaNac>2001-11-23+01:00</fechaNac>`

`<xs:element name="fechaNac" type="xs:date"/>`



Fecha y/u hora

## Elementos simples en documento XSD

<hora>22:43:23</hora>

<hora>22:43:23+02:00</hora>

<hora>22:43:23.4</hora>

<xs:element name="hora" type="xs:date"/>



Fecha y/u hora

<fechayhora>2001-11-23T22:43:23+01:00</fechayhora>

<fechayhora>2001-11-23T22:43:23.8</fechayhora>

<xs:element name="fechayhora" type="xs:date"/>



Fecha y/u hora

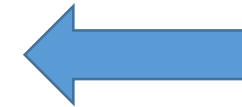
## Elementos simples en documento XSD

<precio>340.43</precio>

<precio>-340.43</precio>



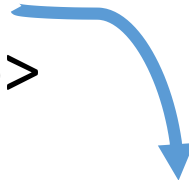
<xs:element name="precio" type="xs:decimal"/>



decimal

<precio>340</precio>

<precio>-340</precio>



<xs:element name="precio" type="xs:integer"/>



Entero

**decimal:** número decimal de precisión (dígitos significativos) arbitraria

**float:** número de punto flotante de 32 bits de precisión simple

**double:** número de punto flotante de 64 bits de doble precisión

**xs:positiveInteger**

**xs:negativeInteger**

## Elementos simples en documento XSD

<pagado>true</pagado>

<pagado>false</pagado>

<xs:element name="pagado" type="xs:boolean"/>

Booleano

---

<xs:element name="foto" type="xs:hexBinary"/>

Documentos o  
formatos binarios

## Elementos simples en documento XSD

Podemos incluir dos opciones más en `xs:element`, *default* y *fixed*.

```
<xs:element name="cuotaPagada" type="xs:boolean" default="false"/>
```

Valor por defecto si no se incluye.

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

02ElementosXSD.xml

Contenido fijo para ese elemento

```
<nombre>Sevilla</nombre>
```

02ElementosXSD.xsd

El elemento puede estar vacío y si no lo está, su contenido debe coincidir con el especificado en el atributo "fixed"



## Elementos complejos en documento XSD

Estos elementos pueden estar compuestos por otros elementos o/y tener atributos propios.

- Un elemento complejo puede estar vacío, es decir, no contener elementos ni texto, pero sí tener al menos un atributo.
- Un elemento complejo puede contener contenido mixto, es decir, contener uno o más elementos, además de texto. Por otra parte, podría tener atributos, o no.

*<seleccionador nombre="Luis Enrique Martínez"/>*

*<seleccionador país="España" deporte="fútbol" genero="masc">*

*<nombre>Luis Enrique Martínez</nombre>*

*</seleccionador>*

**XML**

*<seleccionador país="España">Luis Enrique Martínez</seleccionador>*

## Elementos complejos en documento XSD

Para definir elementos complejos hay que indicarlo añadiendo un elemento llamado **<xs:complexType>**.

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

```
  <xs:complexType>
```

```
    <xs:attribute name="numero" type="xs:positiveInteger"/>
```

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

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

**ATRIBUTOS**

*<xs:attribute name="nombre\_atributo" type="tipo\_dato"/>*

Si no se indica nada, el atributo será opcional.

Puede ser obligatorio con ***use="required"***.

Podemos indicar valor por defecto, ***default="X"***.

Para indicar tiene un valor fijo, ***fixed="X"***.

## Elementos complejos en documento XSD

Ir  acompa ado de `<xs:sequence>` para hacer saber el orden de los elementos hijos. Todo anidado con los cierres correspondientes.

<code>&lt;xs:element name="bingo"&gt;</code>	<code>&lt;bingo&gt;</code>	
<code>&lt;xs:complexType&gt;</code>	<code>&lt;bola numero="22"/&gt;</code>	
<code>&lt;xs:sequence&gt;</code>	<code>&lt;nombrePopular&gt;Los dos patitos&lt;/nombrePopular&gt;</code>	
<code>&lt;xs:element name="bola"&gt;</code>	<code>&lt;/bingo&gt;</code>	
<code>&lt;xs:complexType&gt;</code>		
<code>&lt;xs:attribute name="numero" type="xs:positiveInteger"/&gt;</code>		03ElementosXSD.xml
<code>&lt;/xs:complexType&gt;</code>		
<code>&lt;/xs:element&gt;</code>		03ElementosXSD.xsd
<code>&lt;xs:element name="nombrePopular" type="xs:string"/&gt;</code>		
<code>&lt;/xs:sequence&gt;</code>		03ElementosXSDconREF.xsd
<code>&lt;/xs:complexType&gt;</code>		
<code>&lt;/xs:element&gt;</code>		

## Valores únicos y referenciados (xs:ID y xs:IDREF)

ID. El valor debe ser único en el documento XML.

IDREF. Debe tener un valor de un elemento o atributo de tipo ID dentro del documento XML.

11ElementosXSD.xml

11ElementosXSD.xsd

## Restricciones o facetas en documento XSD

XML Schema permite definir restricciones a los posibles valores de los tipos de datos. Dichas restricciones se pueden establecer en diferentes aspectos, llamados facetas que permiten definir **restricciones sobre los posibles valores de atributos o elementos**.

- **xs:length**. Especifica una longitud fija.
- **xs:minLength**. Especifica una longitud mínima.
- **xs:maxLength**. Especifica una longitud máxima.
- **xs:pattern**. Especifica un patrón de caracteres admitidos.
- **xs:enumeration**. Especifica una lista de valores admitidos.
- **xs:whiteSpace**. Especifica cómo se debe tratar a los posibles espacios en blanco, las tabulaciones, los saltos de línea y los retornos de carro que puedan aparecer.

## Restricciones o facetas en documento XSD

- **xs:maxInclusive**. Especifica que el valor debe ser menor o igual que el indicado.
- **xs:maxExclusive**. Especifica que el valor debe ser menor que el indicado.
- **xs:minInclusive**. Especifica que el valor debe ser mayor o igual que el indicado.
- **xs:minExclusive**. Especifica que el valor debe ser mayor que el indicado.
- **xs:totalDigits**. Especifica el número máximo de dígitos que puede tener un número.
- **xs:fractionDigits**. Especifica el número máximo de decimales que puede tener un número.

Podemos aplicar las facetas tanto a elementos simples (usando en este caso **<xs:simpleType>** o las complejas que ya llevan incorporado **<xs:complexType>**

# Restricciones o facetas en documento XSD

```
<xs:element name="mes">
  <xs:simpleType>
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="1"/>
      <xs:maxInclusive value="12"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Hay que indicar el tipo de datos sobre el que se pondrán restricciones.

Pueden incorporarse varias facetas indicando el valor. Ejemplo con **minInclusive** y **maxInclusive**

04ElementosXSD.xml

En este ejemplo, mes es un elemento que podrá tomar un valor entero entre 1 y 12.

04ElementosXSD.xsd

# Restricciones o facetas en documento XSD

Ejemplo con **xs:enumeration**. Sirve para definir una lista de valores admitidos

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

```
  <xs:simpleType>
```

```
    <xs:restriction base="xs:string">
```

```
      <xs:enumeration value="verde"/>
```

```
      <xs:enumeration value="amarillo"/>
```

```
      <xs:enumeration value="rojo"/>
```

```
    </xs:restriction>
```

```
  </xs:simpleType>
```

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

El valor de color será una cadena con valor  
"verde", "amarillo" o "rojo"

04ElementosXSD.xml

Ejemplo con **xs:length**.  
Sirve para limitar el  
número de caracteres

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

```
  <xs:simpleType>
```

```
    <xs:restriction base="xs:string">
```

```
      <xs:length value="12"/>
```

```
    </xs:restriction>
```

```
  </xs:simpleType>
```

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

04ElementosXSD.xsd



## Restricciones o facetas en documento XSD

**<xs:whiteSpace value="preserve"/>** respeta todas las tabulaciones, los saltos de línea y los retornos de carro pero puede tomar también los valores **replace** (los sustituye por espacios en blanco) y **collapse** (como *replace* pero los espacios en blanco consecutivos los deja en uno).

04ElementosXSD.xml

**xs:pattern** sirve para definir un patrón de caracteres admitidos.  
El valor del patrón tiene que ser una expresión regular.

04ElementosXSD.xsd

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

**Restricciones en  
atributos**

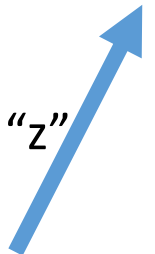


04bElementosXSD.xml

04bElementosXSD.xsd



Una letra minúscula de la "a" a la "z"



Crearemos un tipo de dato personalizado

## Indicadores de secuencia en XSD

Los indicadores permiten establecer cómo se van a escribir –o utilizar– los elementos en un documento XML. Hay siete tipos de indicadores que se pueden clasificar en:

- **Indicadores de orden:** secuencia (*sequence*), todo (*all*) y elección (*choice*).
- **Indicadores de ocurrencia:** *maxOccurs* y *minOccurs*.
- **Indicadores de grupo:** de elementos (*group*) y de atributos (*attributeGroup*).

## Indicadores de secuencia en XSD

### Indicadores de orden

**xs:sequence** sirve para especificar el orden en el que obligatoriamente deben aparecer los elementos hijo de un elemento.

**xs:all** sirve para indicar que dichos elementos pueden aparecer en cualquier orden.

**xs:choice** sirve para especificar que solamente se permite escribir uno de los elementos hijo.

05ElementosXSD.xml

05bElementosXSD.xml

05ElementosXSD.xsd

05bElementosXSD.xsd

# Indicadores de secuencia en XSD

## Indicadores de ocurrencia

**maxOccurs** y **minOccurs** permiten establecer, respectivamente, el número máximo y mínimo de veces que puede aparecer un determinado elemento. El valor por defecto para *maxOccurs* y *minOccurs* es 1.

06ElementosXSD.xml

```
<xs:element name="pais" maxOccurs="unbounded">
```

“país” puede aparecer 1 o ilimitadas veces

06ElementosXSD.xsd

```
<xs:element name="pais" minOccurs="0" maxOccurs="unbounded">
```

“país” puede aparecer 0 o ilimitadas veces

```
<xs:element name="ciudad" type="xs:string" minOccurs="0" maxOccurs="3"/>
```

“ciudad” puede aparecer entre 0 y 3 veces

# Indicadores de secuencia en XSD

## Indicadores de grupo

**xs:group** sirve para agrupar un conjunto de declaraciones de elementos relacionados.

```
<xs:group name="datosBasicos">  ← Definimos el grupo de atributos
  <xs:sequence>
    <xs:element name="nombre" type="xs:string"/>
    <xs:element name="edad" type="xs:positiveInteger"/>
    <xs:element name="pais" type="xs:string"/>
  </xs:sequence>
</xs:group>
```

07ElementosXSD.xml

07ElementosXSD.xsd

Donde haga falta hacemos referencia al grupo usando **<xs:group ref="datosBasicos"/>**

**<xs:group ref="datosBasicos" maxOccurs="unbounded"/>** permitiría repetir los tres elementos cuantas veces queramos.

# Indicadores de secuencia en XSD

## Indicadores de grupo

**xs:attributeGroup** sirve para definir un grupo de atributos.

```
<xs:attributeGroup name="datosBasicos">  
  <xs:attribute name="nombre" type="xs:string"/>  
  <xs:attribute name="edad" type="xs:positiveInteger"/>  
  <xs:attribute name="pais" type="xs:string"/>  
</xs:attributeGroup>
```



Definimos el grupo de elementos

07bElementosXSD.xml

07bElementosXSD.xsd

Donde haga falta hacemos referencia al grupo usando **<xs:attributeGroup ref="datosBasicos"/>**

## Personalizar estructuras nuevas

Podemos crear estructura con elementos y grupo de elementos para emplearlos en el documento para no repetir definiciones.

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

08ElementosXSD.xml

```
<xs:sequence>
```

```
<xs:group ref="datosBasicos"/>
```

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

08ElementosXSD.xsd

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

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

```
<xs:element name="cuidador" type="datosDePersona" />
```

```
<xs:group name="datosBasicos">
```

```
<xs:sequence>
```

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

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

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

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

```
</xs:group>
```

04cElementosXSD.xml

04cElementosXSD.xsd



Al incluir el elemento.

## Referencias para reutilizar elementos (*ref*)

Para estructurar los documentos XSD podemos reutilizar elementos que sólo tendremos que definir una vez y referenciarlos con el atributo *ref* en lugar de *name*

09ElementosXSD.xml

```
<xs:element name="hotel">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="nombre" type="xs:string"/>
      <xs:element name="categoria" type="xs:string"/>
      <xs:element ref="observaciones"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="observaciones" type="xs:string"/>
```

09ElementosXSD.xsd

```
<xs:element name="ciudad">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="pais" type="xs:string"/>
      <xs:element name="nombre" type="xs:string"/>
      <xs:element ref="observaciones"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```



## Elemento con contenido mixto

Cuando tenemos un elemento complejo con contenido y además lleva incorporado algún atributo u otros elementos hijos debemos tratarlo de forma especial.

```
<xs:element name="persona" maxOccurs="unbounded">
```

10ElementosXSD.xml

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

10ElementosXSD.xsd

```
<xs:sequence>
```

10bElementosXSD.xml

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

10bElementosXSD.xsd

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

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

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

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

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

```
<persona>
```

```
<nombre>Eva</nombre> vive en <ciudad>París</ciudad>  
y tiene <edad>25</edad> años.
```

```
</persona>
```

### 3.- Herramientas.

Estas son algunas de las herramientas que facilitan el trabajar con XML, DTD y Schema XML.

- **Editor, generador y validador XML COPY EDITOR**
- Editor <https://www.oxygenxml.com/> OXYGEN. Es de pago con período de 30 días de prueba.
- Extensión XML (Red Hat) para Visual Studio Code
- Validador DTD <https://es.rakko.tools/tools/51/>

## XML COPY EDITOR.

Algunas de las utilidades interesantes que aporta XML COPY como editor son:

- Al crear un documento nuevo nos facilita las líneas básicas (sintaxis) del documento dependiendo de la extensión: .xml, .dtd, .xsd, .xsl (transformación).
- Ver -> Wrap Words (para ajustar las líneas al tamaño de la pantalla, el ALT-Z de VSCode).
- Ver -> Esquema de color / Ver -> Tamaño de texto
- Ver -> Ocultar Atributos solo / Ver -> Etiquetas y Atributos
- XML -> Bloquear Etiquetas
- XML -> Formatear el fuente. Estructura el documento con tabulaciones.

## XML COPY EDITOR.

Utilidades interesantes de XML COPY vinculadas:

- Editar -> Toggle Comment para des/convertir una línea en comentario.
- XML -> Create Schema, nos ofrecerá la posibilidad de **crear una estructura de datos DTD o Schema XML del documento XML activo.**
- XML -> DTD → Schema **crea el Schema XML equivalente a un archivo DTD dado.**
- XML -> Asociar (DTD, Schema XML, estilos). Incluye la línea correspondiente asociando el fichero externo elegido.
- XML -> Encoding