



EFA
MORATALAZ

*1º CFGS Desarrollo de
Aplicaciones Web*

Lenguajes de Marcas y Sistemas de Gestión de la información

María Fátima Sánchez Fuentes

UT3 – Validación de datos.





EFA
MORATALAZ

*1º CFGS Desarrollo de
Aplicaciones Web*

Lenguajes de Marcas y Sistemas de Gestión de la información

INDICE

UT3 – Validación de datos.

1. Validación de datos.
 - 1.1 DTD.
 - 1.2 XML Schema.
2. Métodos de diseño XSD/XML Schema.

Validación de datos

1

La **diferencia** entre un documento **XML bien formado** y un **documento válido**, es que el documento bien formado respeta la norma y un documento válido, además de estar bien formado, **respeta la gramática definida en la DTD**.

Se pueden validar los documentos con **diversas tecnologías**:

- DTD.
- XML Schema.
- Schematron
- ...

A pesar de que existen varias tecnologías para la validación de documentos, **DTD es la más usada**.

DTD

1.1

DTD (*Document Type Definition*, Definición de Tipo de Documento) sirve para **definir la estructura** de un documento SGML o XML, permitiendo su **validación**.

La DTD se puede definir **dentro** del propio **documento XML** o puede ser un **fichero externo** vinculado al documento.

La forma más óptima de hacerlo es como un fichero externo ya que sirve como plantilla de gramática a muchos documentos, al contrario que si se realiza de forma explícita en un único documento (únicamente sirva para ese documento).

La declaración se pone justo después del prólogo y antes de los datos XML:

```
<?xml versión="1.0" encoding="ISO-8859-1"?>  
<!DOCTYPE agenda SYSTEM "agenda.dtd">  
<!--documento de ejemplo inspirado en vCard 3.0 -->
```

La declaración de una **DTD interna** es así:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE marcadores [
  <!ELEMENT marcadores (pagina)*>
  <!ELEMENT pagina (nombre, descripcion, url)>
  <!ELEMENT nombre (#PCDATA)>
  <!ELEMENT descripcion (#PCDATA)>
  <!ELEMENT url (#PCDATA)>
]>
<marcadores>
  <pagina>
    <nombre>IFP</nombre>
    <descripcion>Centro Estudios IFP.</descripcion>
    <url>http://www.ifp.org/</url>
  </pagina>
  <pagina>
    <nombre>Wikipedia</nombre>
    <descripcion>La enciclopedia libre.</descripcion>
    <url>http://www.wikipedia.org/</url>
  </pagina>
  <pagina>
    <nombre>W3C</nombre>
    <descripcion>WorldWideWeb Consortium.</descripcion>
    <url>http://www.w3.org/</url>
  </pagina>
</marcadores>
```


Existen dos tipos de DTD externa:

1. DTD externa privada.

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

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE marcadores SYSTEM "marcadores.dtd">
<marcadores>
  <pagina>
    <nombre>IFP</nombre>
    <descripcion>TutCentro estudios IFP.</descripcion>
    <url>http://www.ifp.org/</url>
  </pagina>
  <pagina>
    <nombre>Wikipedia</nombre>
    <descripcion>La enciclopedia libre.</descripcion>
    <url>http://www.wikipedia.org/</url>
  </pagina>
  <pagina>
    <nombre>W3C</nombre>
    <descripcion>WorldWideWeb Consortium.</descripcion>
    <url>http://www.w3.org/</url>
  </pagina>
</marcadores>
```

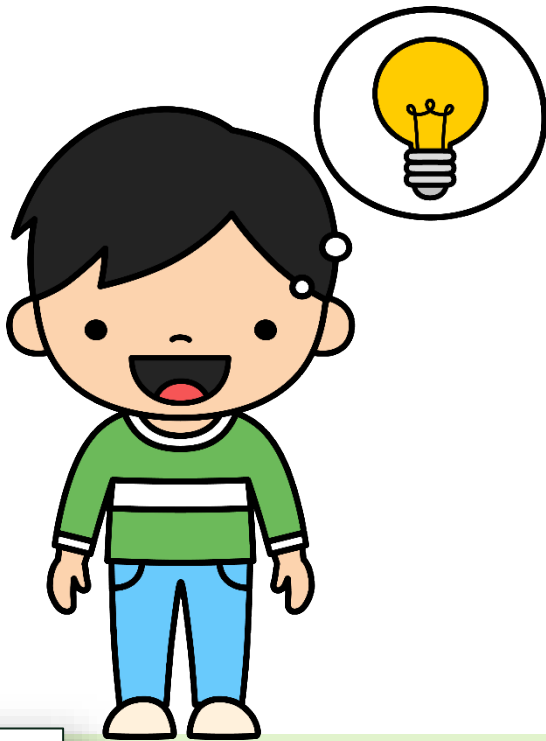
2. DTD externa pública.

```
<!DOCTYPE elemento-raíz PUBLIC"identificador-público" "URI">
```

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
  <head>
    <title>Título</title>
  </head>
  <body>
    <p>Párrafo</p>
  </body>
</html>
```


Visto desde un punto de vista DTD, todos los documentos XML están formados por los **siguientes bloques de construcción**:

- Elementos.
- Atributos.



Por tanto, un **documento XML** será **válido** si además de no tener **errores de sintaxis** cumple lo indicado en las declaraciones de elementos, atributos, entidades y notaciones, de la **DTD a la que esté asociado**.

ELEMENTOS

Los **elementos** son los **bloques de construcción** principales de los documentos XML. Ejemplos de elementos XML podrían ser "nota" y "mensaje". Los elementos pueden contener texto, otros elementos o estar vacíos.

La **sintaxis** de los elementos es la siguiente:

<!ELEMENT nombre-del-elemento tipo-de-contenido**>**

Los **sub-elementos** pueden especificarse como:

- **Nombres de elementos.**
- **#PCDATA** (Parsed Character DATA): cadenas de caracteres o texto.
- **EMPTY** (elemento vacío, no puede tener contenido) o **ANY** (puede contener cualquier cosa, no hay restricción de los subelementos).

```
<?xml version="1.0" encoding="UTF-8"?>  
<!DOCTYPE ciudad [<!ELEMENT ciudad (#PCDATA)>]>  
<ciudad>Madrid</ciudad>
```

En el ejemplo XML, el elemento “*ciudad*” puede contener cualquier texto (cadena de caracteres). Escribiendo #PCDATA(Parsed Character Data) entre paréntesis “()”, se ha indicado que el **elemento “ciudad” puede contener una cadena de caracteres analizable.**

Sub-elemento EMPTY

EMPTY → elemento vacío, no puede tener contenido, pero si puede tener atributos.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE persona [
  <!ELEMENT persona (nombre, mayor_de_edad, ciudad)>
  <!ELEMENT nombre (#PCDATA)>
  <!ELEMENT mayor_de_edad EMPTY>
  <!ELEMENT ciudad (#PCDATA)>
]>
<persona>
  <nombre>Elsa</nombre>
  <mayor_de_edad/>
  <ciudad>Pamplona</ciudad>
</persona>
```

El elemento “*mayor_de_edad*” declarado como vacío.

Sub-elemento ANY

ANY → puede contener cualquier cosa, no hay restricción de los subelementos

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE persona [
  <!ELEMENT persona ANY>
  <!ELEMENT nombre (#PCDATA)>
  <!ELEMENT ciudad (#PCDATA)>
]>
<persona>
<nombre>Antonio</nombre> vive en <ciudad>Madrid</ciudad>.
</persona>
```

El elemento “*persona*” puede contener texto y otros elementos (no hay restricción).

Sub-elemento NOMBRES DE ELEMENTOS

Nombres de elementos → en una DTD pueden haber elementos PADRE y elementos HIJOS. Los hijos (también llamados sucesores) tienen que escribirse entre “()” y separados por “,”.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE persona [
<!ELEMENT persona (nombre, fecha_de_nacimiento, ciudad)>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT fecha_de_nacimiento (#PCDATA)>
<!ELEMENT ciudad (#PCDATA)>
]>
<persona>
  <nombre>Ana</nombre>
  <fecha_de_nacimiento>26-12-2000</fecha_de_nacimiento>
  <ciudad>Valladolid</ciudad>
</persona>
```

El elemento PADRE “*persona*” contiene a los elementos “*nombre*”, “*fecha_de_nacimiento*” y “*ciudad*”

Sub-elemento NOMBRES DE ELEMENTOS

Nombres de elementos → A su vez, los elementos HIJOS pueden tener sus propios HIJOS.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE persona [
  <!ELEMENT persona (nombre, fecha_de_nacimiento, ciudad)>
  <!ELEMENT nombre (#PCDATA)>
  <!ELEMENT fecha_de_nacimiento (dia,mes,año)>
  <!ELEMENT dia (#PCDATA)>
  <!ELEMENT mes (#PCDATA)>
  <!ELEMENT año (#PCDATA)>
  <!ELEMENT ciudad (#PCDATA)>
]>
<persona>
  <nombre>Esterban</nombre>
  <fecha_de_nacimiento>
    <dia>26</dia>
    <mes>12</mes>
    <año>1985</año>
  </fecha_de_nacimiento>
  <ciudad>Sevilla</ciudad>
</persona>
```

El elemento HIJO *“fecha_de_nacimiento”*, que ahora es PADRE también, puede contener a los elementos *“día”*, *“mes”* y *“año”*.

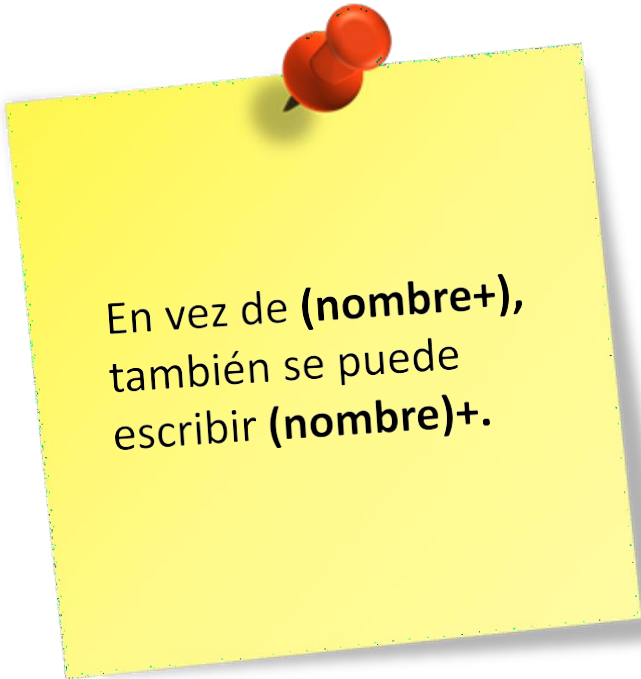
CARDINALIDAD DE LOS ELEMENTOS

En una DTD, para definir **el número de veces que pueden aparecer los elementos** de un documento XML, utilizando los operadores adecuados.

OPERADOR	CARDINALIDAD	SIGNIFICADO
?	0-1	El elemento es opcional, pudiendo aparecer una sola vez o ninguna
*	0-N	El elemento poder aparecer cero, una o más veces.
+	1-N	El elemento tiene que aparecer, obligatoriamente, una o más veces.
	1	El elemento tendrá que aparecer obligatoriamente una única vez

CARDINALIDAD “+”

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



En vez de **(nombre+)**,
también se puede
escribir **(nombre)+**.

El elemento “*nombre*” tiene que aparecer una o más veces (“+”).
Si después de “*nombre*” no estuviera el operador “+” el documento no sería válido, ya que significaría que “*persona*” solo debería contener “*nombre*” una vez. Sin embargo, lo contiene 3 veces.

CARDINALIDAD “*”

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE receta_de_cocina[
<!ELEMENT receta_de_cocina (nombre, ingrediente*)>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT ingrediente (#PCDATA)>
]>
<receta_de_cocina>
  <nombre>Tortilla de patatas</nombre>
  <ingrediente>Huevo</ingrediente>
  <ingrediente>Patata</ingrediente>
  <ingrediente>Aceite</ingrediente>
  <ingrediente>Sal</ingrediente>
</receta_de_cocina>
```

El elemento “*ingrediente*” puede que aparecer cero, una o más veces (“*”).
Por eso “*receta_de_cocina*” contiene varios “*ingredientes*”.

CARDINALIDAD “?”

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

El elemento “*mayor_de_edad*” puede aparecer una o ninguna vez(“?”).

OPCIONALIDAD DE ELEMENTOS

En la DTD asociada a un documento XML, se pueden declarar elementos que contengan elementos opcionales. Para ello, se utiliza el operador de elección, representado por una barra vertical (|).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE articulo [
  <!ELEMENT articulo (codigo| id)>
  <!ELEMENT codigo (#PCDATA)>
  <!ELEMENT id (#PCDATA)>
]>
<articulo>
  | <codigo>AF-33</codigo>
</articulo>
```

Ejemplo

¿QUÉ NOS INDICA ESTE CÓDIGO DTD?

- Pueden aparecer 0 o más elementos “*artículo*”.
- El elemento “*artículo*” contiene un elemento “*código*” o un elemento “*id*”.
- El elemento “*artículo*” contiene obligatoriamente un elemento “*nombre*”.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE articulos[
  <!ELEMENT articulos (articulo)*>
  <!ELEMENT articulo ((codigo| id), nombre)>
  <!ELEMENT codigo (#PCDATA)>
  <!ELEMENT id (#PCDATA)>
  <!ELEMENT nombre (#PCDATA)>
]>
<articulos>
  <articulo>
    <codigo>AF-47</codigo>
    <nombre>Martillo</nombre>
  </articulo>
  <articulo>
    <id>2056</id>
    <nombre>Destornillador</nombre>
  </articulo>
</articulos>
```

ATRIBUTOS

Los **atributos** definen una **propiedad** de un **elemento**.

La **sintaxis** de los atributos es la siguiente:

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

DECLARACIÓN DE UN ATRIBUTO INDICANDO UN VALOR POR DEFECTO

Aquí se muestra un ejemplo...

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE deportistas [
<!ELEMENT deportistas (futbol | f1 | tenis)*>
<!ELEMENT futbol (#PCDATA)>
<!ELEMENT f1 (#PCDATA)>
<!ATTLIST f1 pais CDATA "España">
<!ELEMENT tenis (#PCDATA)>
]>

<deportistas>
  <f1 pais="Alemania">SebastianVettel</f1>
  <f1>Fernando Alonso</f1>
  <tenis>Rafael Nadal</tenis>
</deportistas>
```

Se indica que el elemento “f1” puede tener un atributo “pais”, cuyo **valor por defecto** es “España”.



¡OJO! El valor por defecto del atributo es “España”, pero este puede ser modificado. Si queremos que no sea modificado, se debe poner *#FIXED* “España”.

Aquí se muestra un ejemplo...

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE deportistas [
<!ELEMENT deportistas (futbol | f1 | tenis)*>
<!ELEMENT futbol (#PCDATA)>
<!ELEMENT f1 (#PCDATA)>
<!ATTLIST f1 pais CDATA "España">
<!ELEMENT tenis (#PCDATA)>
]>

<deportistas>
  <f1 pais="Alemania">SebastianVettel</f1>
  <f1>Fernando Alonso</f1>
  <tenis>Rafael Nadal</tenis>
</deportistas>
```

Para el elemento “*f1*”, el atributo “*país*” es de tipo CDATA (CharacterDATA), es decir, su valor es una cadena de caracteres.

En el elemento “*f1*” de **Fernando Alonso** no se ha puesto el valor del atributo “*país*”.

¿QUÉ OCURRIRÁ?

Que aparecerá por **defecto** el **valor del atributo** “España”.

```
▼ <deportistas>
  <f1 pais="Alemania">Sebastian Vettel</f1>
  <f1 pais="España">Fernando Alonso</f1>
  <tenis>Rafael Nadal</tenis>
</deportistas>
```

DECLARACIÓN DE VARIOS ATRIBUTOS EN UN ELEMENTO

Hasta ahora hemos visto la declaración de un atributo para un elemento, pero se pueden declarar **varios atributos para un elemento concreto**.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE deportistas [
<!ELEMENT deportistas (futbol | f1 | tenis)*>
<!ELEMENT futbol (#PCDATA)>
<!ELEMENT f1 (#PCDATA)>
<!-- Atributos para el elemento f1 -->
<!-- Atributo obligatorio -->
<!-- Atributo opcional -->
<!ELEMENT tenis (#PCDATA)>
]>
<deportistas>
  <f1 pais="Alemania" fecha_de_nacimiento="03/07/1987" equipo="Ferrari">SebastianVettel</f1>
  <f1 equipo="McLaren">Fernando Alonso</f1>
  <tenis>Rafael Nadal</tenis>
</deportistas>
```

En la DTD del siguiente documento XML se ha indicado que el elemento “f1” puede tener **tres atributos**:

- *país.*
- *fecha_de_nacimiento*
- *equipo.*

Donde:

- *equipo* → es obligatorio (#REQUIRED).
- *fecha_de_nacimiento* → es opcional (#IMPLIED).

Cuando se declara más de un atributo para un elemento no es necesario escribir varias veces `<!ATTLIST>`.

Se puede escribir de la siguiente manera:

```
<!ATTLIST f1 pais CDATA "España"  
              fecha_de_nacimiento CDATA #IMPLIED  
              equipo CDATA #REQUIRED>
```

TIPOS DE DECLARACIÓN DE ATRIBUTOS EN UNA DTD

Valor	Significado
valor entre comillas dobles (") o simples (').	El atributo tiene un valor por defecto.
#REQUIRED	El atributo es obligatorio escribirlo.
#IMPLIED	El atributo es opcional escribirlo.
#FIXED valor entre comillas dobles (") o simples (').	El valor del atributo es fijo.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE deportistas [
  <!ELEMENT deportistas (futbol | f1 | tenis)*>
  <!ELEMENT futbol (#PCDATA)>
  <!ELEMENT f1 (#PCDATA)>
  <!-- <!ATTLIST f1 pais CDATA #REQUIRED -->
  <!ELEMENT tenis (#PCDATA)>
]>

<deportistas>
  <f1 pais="Alemania">Sebastian Vettel</f1>
  <f1>Fernando Alonso</f1>
  <tenis>Rafael Nadal</tenis>
</deportistas>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE deportistas [
  <!ELEMENT deportistas (futbol | f1 | tenis)*>
  <!ELEMENT futbol (#PCDATA)>
  <!ELEMENT f1 (#PCDATA)>
  <!-- <!ATTLIST f1 pais CDATA #IMPLIED -->
  <!ELEMENT tenis (#PCDATA)>
]>

<deportistas>
  <f1 pais="Alemania">Sebastian Vettel</f1>
  <f1>Fernando Alonso</f1>
  <tenis>Rafael Nadal</tenis>
</deportistas>
```



```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE deportistas [
    <!ELEMENT deportistas (futbol | f1 | tenis)*>
    <!ELEMENT futbol (#PCDATA)>
    <!ELEMENT f1 (#PCDATA)>
    <!-- <!ATTLIST f1 pais CDATA #FIXED "España">
    <!ELEMENT tenis (#PCDATA)>
]>

<deportistas>
    <f1 pais="España">Carlos Sainz</f1>
    <f1>Fernando Alonso</f1>
    <tenis>Rafael Nadal</tenis>
</deportistas>
```

TIPOS DE ATRIBUTOS MÁS RELEVANTES EN UNA DTD

Un atributo de tipo **CDATA** (Character DATA), es aquel cuyo valor puede ser una **cadena de caracteres** (texto).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ciudades [
    <!ELEMENT ciudades (ciudad)*>
    <!ELEMENT ciudad (#PCDATA)>
    <!-- <!ATTLIST ciudad pais CDATA #REQUIRED -->
]>

<ciudades>
    <ciudad pais="Italia">Roma</ciudad>
    <ciudad pais="Francia">París</ciudad>
    <ciudad pais="Alemania">Berlín</ciudad>
    <ciudad pais="">Viena</ciudad>
</ciudades>
```

Un atributo de tipo **enumerado** indica que su valor puede ser uno de los pertenecientes a una **lista de valores** escritos entre paréntesis "(" y separados por el carácter "|".

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE deportistas [
  <!ELEMENT deportistas (futbol | f1 | tenis)*>
  <!ELEMENT futbol (#PCDATA)>
  <!ELEMENT f1 (#PCDATA)>
  <!-- <!ATTLIST f1 pais (ESP | FRA | ITA | ALE) "ESP"> -->
  <!ELEMENT tenis (#PCDATA)>
]>

<deportistas>
  <f1 pais="ALE">Sebastian Vettel</f1>
  <f1>Fernando Alonso</f1>
  <f1 pais="ESP">Carlos Sainz</f1>
  <tenis>Rafael Nadal</tenis>
</deportistas>
```

¡OJO! En este caso, se ha especificado "ESP" como valor por defecto. De modo que es obligatorio que aparezca en la declaración entre los valores de la enumeración "()".

Los atributos declarados **NMTOKEN** son aquellos cuyo valor será una **cadena de caracteres**, pudiendo contener letras minúsculas, letras mayúsculas, números, puntos ".", guiones medios "-", guiones bajos "_" o el carácter dos puntos ":".

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE usuarios [
  <!ELEMENT usuarios (usuario)*>
  <!ELEMENT usuario (#PCDATA)>
  <!ATTLIST usuario clave NMTOKEN #REQUIRED>
]>

<usuarios>
  <usuario clave="123456789">Ana</usuario>
  <usuario clave="ab-c-d-fg">Iker</usuario>
  <usuario clave="A1_B2..C3">Elsa</usuario>
</usuarios>
```

En el valor de un atributo NMTOKEN no se pueden escribir espacios en blanco ni caracteres especiales, tales como: *, \$, %, &, ?, @...

XSD

1.2

XML schema permite definir la estructura y el contenido de un documento XML, a través de un lenguaje de etiquetado basado en XML, por lo tanto es una tecnología similar a DTD.

XML Schema son definidos en documentos con extensión “.xsd” y en los documentos que se basen en ese esquema se incluirá una referencia al archivo con la extensión mencionada.

En la primera etiqueta, correspondiente al elemento raíz y denominado ***schema***, hace referencia al **espacio de nombre XMLSchema publicado por W3C**.

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

TIPOS DE ELEMENTOS

- **ELEMENTOS SIMPLES:**

Es un elemento que no contiene atributos o sub-elementos, únicamente puede contener texto.

```
<xs:element name="nombre_elemento" type="tipo_elemento"/>
```

Los tipos de datos más utilizados son *xs:string*, *xs:decimal*, *xs:integer*, *xs:boolean*, *xs:date* y *xs:time*.

- **ELEMENTOS COMPLEJOS:**

Son elementos que contienen sub-elementos o atributos. Estos se definen mediante la etiqueta “complexType”.

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

Un elemento complejo puede estar **vacío** (no contiene elementos ni texto, pero sí tiene al menos un atributo).

EJEMPLO DE ELEMENTOS SIMPLES:

```
<nombre>María</nombre>  
<edad>23</edad>
```

Se ha definido un elemento simple “nombre”

- Es de tipo *string*.

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

Se ha definido un elemento simple “edad”

- Es de tipo *integer*.

EJEMPLO DE ELEMENTOS COMPLEJOS:

```
<xs:element name="persona">
  <xs:complexType mixed="true">
    <xs:sequence>
      <xs:element name="nombre" type="xs:string"/>
      <xs:element name="ciudad" type="xs:string"/>
      <xs:element name="edad" type="xs:positiveInteger"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Se ha definido un elemento “persona”.

- Es de tipo complejo.
- Contiene otros elementos.

INDICADORES EN XSD

Los indicadores permiten establecer **cómo se van a escribir/utilizar** los elementos en un documento XML.

Se clasifican en **tres grandes grupos**:

- **INDICADORES DE ORDEN.**
- **INDICADORES DE OCURRENCIA.**
- **INDICADORES DE GRUPO.**

En este tema nos centraremos en los dos primeros grupos, ya que son básicos.

INDICADORES DE ORDEN

(`xs:sequence`, `xs:all`, `xs:choice`)

`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.

INDICADORES DE OCURRENCIA

maxOccurs, 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.

```
<?xml version="1.0" encoding="UTF-8"?>
<paises xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="paises.xsd">
  <pais>
    <nombre>España</nombre>
    <ciudad>Madrid</ciudad>
  </pais>
  <pais>
    <nombre>México</nombre>
    <ciudad>Guadalajara</ciudad>
    <ciudad>Monterrey</ciudad>
  </pais>
  <pais>
    <nombre>Francia</nombre>
  </pais>
</paises>
```

Queremos lo siguiente:

- “pais” aparezca una o muchas veces.
- “nombre” solamente pueda escribirse una vez en cada país.
- De cada “país” se pueden especificar entre 0 y 2 ciudades.

```

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="paises">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="pais" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="nombre" type="xs:string"/>
              <xs:element name="ciudad" type="xs:string"
                minOccurs="0" maxOccurs="2"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>

```

Queremos lo siguiente:

- “pais” aparezca una o muchas veces.
- “nombre” solamente pueda escribirse una vez en cada país.
- De cada “país” se pueden especificar entre 0 y 2 ciudades.

ATRIBUTOS

Para definir una atributo de un elemento se hace uso de la siguiente expresión:

```
<xs:attribute name="nombre_del_atributo" type="tipo_de_dato" />
```

Es importante destacar que cuando un elemento tiene al menos un atributo, este elemento ya se convierte en un elemento **complejo** (ver diapositivas de elementos complejos).

```
<curso grupo="A">1</curso>  
  
<xs:element name="curso" type="xs:integer"/>  
<xs:attribute name="grupo" type="xs:string" />
```

En este caso tenemos lo siguiente:

- Un elemento “curso” de tipo *integer*.
- “curso” tiene un atributo “grupo” de tipo *string*.

TIPOS DE ATRIBUTOS

Existen varios tipos de declaración de atributos.

FIXED

- Sirve para indicar el valor fijo del atributo.
- El atributo puede aparecer o no, pero si aparece, debe contener sólo ese valor especificado de forma obligatoria.

```
<xs:attribute name="grupo" type="xs:string" fixed="B"/>
```

DEFAULT

- Sirve para indicar un valor por defecto del atributo.
- Si el atributo no está, el procesador de XML hace uso de este valor por defecto.

```
<xs:attribute name="grupo" type="xs:string" default="B"/>
```

Especificar el nombre y el tipo de atributo es obligatorio. Sin embargo, el resto de cosas es **opcional**.

REQUIRED

- Sirve para indicar que el atributo es obligatorio, es decir, debe aparecer SI o SI.

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

OPTIONAL

- Sirve para indicar que el atributo puede o no aparecer.

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

PROHIBITED

- Sirve para indicar que el atributo no puede aparecer en el elemento.

```
<xs:attribute name="A1" use="prohibited"/>
```

VISIÓN GENERAL DE ELEMENTOS Y ATRIBUTOS

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

```
  <xs:complexType>
```

```
    <xs:sequence>
```

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

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

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

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

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

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

Secuencia ordenada de elementos hijos del elemento "cliente".

Atributo del elemento "cliente".

DECLARACIONES GLOBALES Y LOCALES

En un **schema** es posible declarar elementos de **forma global** y luego hacer **referencias** a ellos **desde otros elementos**.

Para **referenciar** a un elemento se utiliza el **atributo "ref"** cuyo **valor es el nombre del elemento referenciado**, en lugar del atributo "name" seguido de la definición del elemento.

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

  <xs:element name="Titulo" type="xs:string"/>
  <xs:element name="Autores" type="xs:string" maxOccurs="10"/>
  <xs:element name="Editorial" type="xs:string"/>

  <xs:element name="Libro">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Titulo"/>
        <xs:element ref="Autores"/>
        <xs:element ref="Editorial"/>
      </xs:sequence>
      <xs:attribute name="precio" type="xs:double" />
    </xs:complexType>
  </xs:element>
</xs:schema>
```

ATRIBUTOS EN ELEMENTOS SIMPLES

```
<idiomas>
  <idioma>Japones</idioma>
  <idioma nivel="A2">Español</idioma>
  <idioma nivel="B2">Alemán</idioma>
  <idioma nivel="C2">Ingles</idioma>
</idiomas>
```

```
<xs:element name="idiomas">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="idioma" maxOccurs="unbounded">
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="xs:string">
              <xs:attribute name="nivel" type="xs:string" use="optional" default="A2">
            </xs:attribute>
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

RESTRICCIONES (FACETAS)

Las facetas permiten definir **restricciones** sobre los **valores** de los **elementos y atributos**.

Faceta	Descripción
<i>xs:length</i>	Especifica una longitud fija.
<i>xs:minLength</i>	Especifica una longitud mínima.
<i>xs:maxLength</i>	Especifica una longitud máxima.
<i>xs:pattern</i>	Especifica un patrón de caracteres admitidos.
<i>xs:enumeration</i>	Especifica una lista de valores admitidos.
<i>xs:whiteSpace</i>	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.
<i>xs:maxInclusive</i>	Especifica que el valor debe ser menor o igual que el indicado.
<i>xs:maxExclusive</i>	Especifica que el valor debe ser menor que el indicado.
<i>xs:minExclusive</i>	Especifica que el valor debe ser mayor que el indicado.
<i>xs:minInclusive</i>	Especifica que el valor debe ser mayor o igual que el indicado.
<i>xs:totalDigits</i>	Especifica el número máximo de dígitos que puede tener un número.
<i>xs:fractionDigits</i>	Especifica el número máximo de decimales que puede tener un número.

<code>xs:maxInclusive</code>	Especifica que el valor debe ser menor o igual que el indicado.
<code>xs:maxExclusive</code>	Especifica que el valor debe ser menor que el indicado.
<code>xs:minExclusive</code>	Especifica que el valor debe ser mayor que el indicado.
<code>xs:minInclusive</code>	Especifica que el valor debe ser mayor o igual que el indicado.

```
<xs:element name="dias">
  <xs:simpleType>
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="1"/>
      <xs:maxInclusive value="31"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Se ha definido un elemento "días" con la restricción:

- El valor que tome debe estar comprendido entre el 1 y el 31.

"SimpleType" se pone cuando se especifican **restricciones** para un elemento simple

<code>xs:enumeration</code>	Especifica 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>
```

Se ha definido un elemento “color” con la restricción:

- Los únicos valores admitidos son verde, amarillo y rojo.

`xs:pattern`Especifica un patrón de **caracteres admitidos**.

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

Se ha definido un elemento “letra” con la restricción:

- El único valor admitido es una letra minúscula entre la “a” y la “z”.

`xs:pattern`Especifica un patrón de **caracteres admitidos**.

```
<xs:element name="initials">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[A-Z] [A-Z] [A-Z] "/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Se ha definido un elemento “initials” con la restricción:

- Acepta tres letras mayúsculas entre la “A” y la “Z”.

xs:pattern

Especifica un patrón de **caracteres admitidos**.

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

Se ha definido un elemento “initials” con la restricción:

- Acepta tres letras mayúsculas o minúsculas entre la “A” y la “Z”.

*xs:pattern*Especifica un patrón de **caracteres admitidos**.

```
<xs:element name="initials">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[xyz]"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Se ha definido un elemento “initials” con la restricción:

- Acepta una de las tres letras especificadas.

`xs:pattern`Especifica un patrón de **caracteres admitidos**.

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

Se ha definido un elemento “prodid” con la restricción:

- Acepta 0 o más ocurrencias de letras en minúscula.

Pej: a

hola

numerodetelefono

`xs:pattern`Especifica un patrón de **caracteres admitidos**.

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

Se ha definido un elemento “letra” con la restricción:

- Altera una letra minúscula y otra mayúscula.

Pej: hOIA

*xs:pattern*Especifica un patrón de **caracteres admitidos**.

```
<xs:element name="genero">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="hombre|mujer"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Se ha definido un elemento “genero” con la restricción:

- Acepta uno de los dos valores.

`xs:pattern`Especifica un patrón de **caracteres admitidos**.

```
<xs:element name="contrasenia">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[a-zA-Z0-9]{8}"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Se ha definido un elemento “contrasenia” con la restricción:

- Acepta caracteres en minúscula, mayúscula y números.
- Tiene que ser exactamente 8 caracteres.

Pej: alumno01

ALumNo9

xs:lengthEspecifica una **longitud fija**.

```
<xs:element name="clave">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:length value="12"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Se ha definido un elemento “clave” con la restricción:

- Su valor tiene que ser una cadena de 12 caracteres.

Pej: alumno876-4


```
<xs:element name="clave">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[a-zA-Z0-9]*/>
      <xs:length value="10"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Se ha definido un elemento “clave” con la restricción:

- El valor puede ser mayúscula o minúscula de la “A” a la “Z”.
- Puede tener dígitos entre el 0 y el 9.

Pej: abcde12345

```
<xs:element name="clave">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[a-zA-Z0-9]{10}"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

xs:lengthEspecifica una **longitud fija**.

```
<xs:element name="password">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:minLength value="4"/>
      <xs:maxLength value="8"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Se ha definido un elemento “password” con la restricción:

- Tiene que tener como mínimo 4 caracteres y como máximo 8 caracteres.

xs:lengthEspecifica una **longitud fija**.

```
<xs:element name="password">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:minLength value="4"/>
      <xs:maxLength value="8"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Se ha definido un elemento “password” con la restricción:

- Tiene que tener como mínimo 4 caracteres y como máximo 8 caracteres.

Métodos de diseño XSD/XML Schema

2

DISEÑO ANIDADO o MUÑECAS RUSAS

```

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="libro">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="titulo" type="xs:string"/>
        <xs:element name="autor" type="xs:string"/>
        <xs:element name="personaje" minOccurs="0"
          maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="nombre" type="xs:string"/>
              <xs:element name="amigoDe" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
              <xs:element name="desde" type="xs:date"/>
              <xs:element name="calificación" type="xs:string"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="isbn" type="xs:string"/>
    </xs:complexType>
  </xs:element>
</xs:schema>

```

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  <xs:element name="root">
    <xs:complexType base="xs:string">
      <xs:sequence>
        <xs:element name="child" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

- Anidan** las declaraciones de elementos unas dentro de otras.
- Se describe cada elemento y atributo en el mismo lugar donde se declaran.

¿CÓMO SE LOGRA?

Partiendo de un documento XML, se van definiendo de forma completa los elementos en el mismo orden que van apareciendo en el documento instancia, de forma secuencial.

```
curs="unbounded"/>
```

MÉTODOS DE DISEÑO o de USO DE REFERENCIAS A ELEMENTOS Y ATRIBUTOS

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="titulo" type="xs:string"/>
  <xs:element name="autor" type="xs:string"/>
  <xs:element name="nombre" type="xs:string"/>
  <xs:element name="amigoDe" type="xs:string"/>
  <xs:element name="desde" type="xs:date"/>
  <xs:element name="calificacion" type="xs:string"/>
  <xs:attribute name="isbn" type="xs:string"/>

  <xs:element name="personaje">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="nombre"/>
        <xs:element ref="amigoDe" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="desde"/>
        <xs:element ref="calificación"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="libro">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="titulo"/>
        <xs:element ref="autor"/>
        <xs:element ref="personaje" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute ref="isbn"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="titulo" type="xs:string"/>
  <xs:element name="autor" type="xs:string"/>
  <xs:element name="nombre" type="xs:string"/>
  <xs:element name="amigoDe" type="xs:string"/>
  <xs:element name="..." type="xs:string"/>
  <xs:element name="..." type="xs:string"/>
  <xs:element name="..." type="xs:string"/>
```

Se **declaran** primero los
elementos y atributos para
después **referenciarlos**.

```
<xs:element name="..." type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
</xs:element>
</xs:schema>
```