



SCHEMA



XML Schema

XSD (XML Schema Definition) es un lenguaje, también llamado simplemente XML Schema, que sirve para **definir la estructura de un documento XML**, permitiendo su **validación**.

Es un **mecanismo para comprobar la validez de un documento XML**.

Es una forma alternativa y **más actual que los DTD**.

Los documentos XML que se validan contra un esquema se llaman **instancias del esquema**.

Los esquemas XML Schema **superan muchas de las limitaciones y debilidades de las DTDs**.

Enlazar documento XML con SCHEMA

Por lo tanto otra forma de validar un documento XML, es a través del SCHEMA, para ello tenemos que **enlazar el documento XML con el SCHEMA** añadiendo al elemento raíz:

`xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"`

- Hace referencia al espacio de nombres

`xsi:noNamespaceSchemaLocation="url al schema.xsd"`

- Permite referenciar al Schema


Un ejemplo de documento XML asociado a este XML Schema es el siguiente (libro.xml):



Un ejemplo de **documento XML asociado a este XML Schema** es el siguiente :

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<catalogo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="Schemas/SchemaElementos.xsd">
  <artista>
    <nombre>Marie</nombre>
    <apellido>Fredriksson</apellido>
    <grupo>Roxette</grupo>
    <nacionalidad>Suecia</nacionalidad>
  </artista>

  <artista>
    <nombre>Mark</nombre>
    <apellido>Knopfler</apellido>
    <grupo>Dire Straits</grupo>
    <nacionalidad>Escocia</nacionalidad>
  </artista>
</catalogo>
```



URL relativa del Schema

Documento SCHEMA (.xsd)

En todos los esquemas XML, **el elemento raíz es "schema"**. Ahora bien, para escribirlo, es muy común utilizar el prefijo xsd o xs.

Con `xmlns:xs="http://www.w3.org/2001/XMLSchema"` se ha indicado que:

- Los elementos y tipos de datos utilizados en el esquema pertenecen al espacio de nombres `http://www.w3.org/2001/XMLSchema`.
- Dichos elementos y tipos de datos deben llevar el **prefijo xs** (`xs:schema`, `xs:element`, `xs:complexType`, `xs:string...`).

Con `elementFormDefault="qualified"` indica que todos los elementos utilizados por el documento XML que se declararon en este esquema, deben estar calificados para espacios de nombres (deben llevar prefijo xs)

Por todo ello nuestro SCHEMA siempre comienza del siguiente modo:

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

Ejemplo 1: Tengo el siguiente documento XML

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

```
<catalogo>
```

```
  <artista>
```

```
    <nombre>Marie</nombre>
```

```
    <apellido>Fredriksson</apellido>
```

```
    <grupo>Roxette</grupo>
```

```
    <nacionalidad>Suecia</nacionalidad>
```

```
  </artista>
```

```
  <artista>
```

```
    <nombre>Mark</nombre>
```

```
    <apellido>Knopfler</apellido>
```

```
    <grupo>Dire Straits</grupo>
```

```
    <nacionalidad>Escocia</nacionalidad>
```

```
  </artista>
```

```
</catalogo>
```

Elementos simples

Los elementos simples **no pueden contener a otros elementos (hijos), ni atributos**,
Los elementos simples solamente pueden contener texto (caracteres).

<xs:element name="nombre_del_elemento" type="tipo_de_dato"/>

Tipos de datos: Los más comunes:

- xs:string caracteres.
- xs:decimal valor numérico.
- xs:integer valor numérico entero.
- xs:boolean verdadero o falso: "true" o "false"
- xs:date fecha YYYY-MM-DD.
- xs:time Tiempo hh:mm:ss

Defino los elementos simples:

```
<!--Defino los elementos simples-->
  <xs:element name="nombre" type="xs:string"/>
  <xs:element name="apellido" type="xs:string"/>
  <xs:element name="grupo" type="xs:string"/>
  <xs:element name="nacionalidad" type="xs:string"/>
```

Si se quiere indicar que un valor es fijo (fixed), si hay un valor por defecto (default):

```
<xs:element nombre="nacionalidad" type="xs:string" default="España"/>
<xs:element nombre="nacionalidad" type="xs:string" fixed="España"/>
```


Indicadores de ocurrencia (maxOccurs, minOccurs):

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. (Unbounded = ilimitado)





Ejemplo 1 Elementos: El documento XML y el SCHEMA de nuestro ejemplo quedaría

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<catalogo
xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
```

```
xsi:noNamespaceSchemaLocation="SchemaElementos.
xsd">
```

```
  <artista>
    <nombre>Marie</nombre>
    <apellido>Fredriksson</apellido>
    <grupo>Roxette</grupo>
    <nacionalidad>Suecia</nacionalidad>
  </artista>
```

```
  <artista>
    <nombre>Mark</nombre>
    <apellido>Knopfler</apellido>
    <grupo>Dire Straits</grupo>
    <nacionalidad>Escocia</nacionalidad>
  </artista>
```

```
</catalogo>
```

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

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

```
  <!-- Defino el elemento raíz -->
```

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

```
    <xs:complexType>
```

```
      <xs:sequence>
```

```
        <!--Referencia al elemento artista definido más adelante-->
```

```
        <xs:element ref="artista" minOccurs="0" maxOccurs="unbounded"/>
```

```
        <!-- minOccurs, maxOccurs: el n de veces que puede aparecer el elemento artista -->
```

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

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

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

```
  <!--Defino el elemento artista -->
```

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

```
    <xs:complexType>
```

```
      <xs:sequence>
```

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

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

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

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

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

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

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

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

Ejercicio 1:

Diseña un documento XML bien formado y validado utilizando SCHEMA, que permita almacenar la información sobre **librería**, donde para cada **libro** debe aparecer: **nombre**, **autor**, **editorial y género** al que pertenece, **fecha** de publicación, **valoración** de los lectores en forma numérica entera (positiva) y **comentarios** donde aparecen los últimos comentarios realizados por los lectores, pueden aparecer **entre 0 y 3** comentarios.



Ejercicio 2:

Diseña un documento XML bien formado y validado utilizando SCHEMA, que permita almacenar la información sobre **un cine**, donde para cada **película** debe aparecer: **título, director, actor y género** al que pertenece, **fecha** de estreno, **duracion en minutos, productora e idioma original**.

El elemento actor se puede repetir entre 1 y 3 veces.

La productora tiene como valor fijo "Netflix"

Idioma tiene como valor por defecto "Esp"

Atributos

La sintaxis para definir un atributo es:

`<xs:attribute name="nombre_atributo" type="tipo_dato"/>`

Los tipos más comunes son:

- xs:string caracteres.
- xs:decimal valor numérico.
- xs:integer valor numérico entero.
- xs:boolean verdadero o falso: "true" o "false"
- xs:date fecha YYYY-MM-DD.
- xs:time Tiempo hh:mm:ss

Ejemplo 2: Atributos

En el catálogo de artistas anterior, queremos añadir al elemento artista los siguientes atributos:

- número de identificación que será obligatorio
- Instrumento que usa el artista.

Documento XML:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<catalogo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="SchemaAtributos.xsd">

  <artista id="1" instrumento="guitarra">
    <nombre>Marie</nombre>
    <apellido>Fredriksson</apellido>
    <grupo>Roxette</grupo>
    <nacionalidad>Suecia</nacionalidad>
  </artista>

  <artista id="2" instrumento="guitarra">
    <nombre>Mark</nombre>
    <apellido>Knopfler</apellido>
    <grupo>Dire Straits</grupo>
    <nacionalidad>Escocia</nacionalidad>
  </artista>

</catalogo>
```

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

  <!-- Defino el elemento raíz -->
  <xs:element name="catalogo">
    <xs:complexType>
      <xs:sequence>
        <!--Referencia al elemento artista definido más adelante-->
        <xs:element ref="artista" minOccurs="0" maxOccurs="unbounded"/>
        <!-- minOccurs, maxOccurs: el n de veces que puede aparecer el elemento artista -->
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <!--Defino el elemento artista -->
  <xs:element name="artista">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="nombre" type="xs:string"/>
        <xs:element name="apellido" type="xs:string"/>
        <xs:element name="grupo" type="xs:string"/>
        <xs:element name="nacionalidad" type="xs:string"/>
      </xs:sequence>
      <!-- Defino los atributos -->
      <xs:attribute name="id" type="xs:integer" use="required"/>
      <xs:attribute name="instrumento" type="xs:string" use="optional"/>
      <!-- Por defecto si no especificamos use será opcional -->

    </xs:complexType>
  </xs:element>

</xs:schema>
```

Defino los atributos de mi schema:

```
<!--Defino los elementos simples-->
<xs:element nombre="nombre" type="xs:string"/>
<xs:element nombre="apellido" type="xs:string"/>
<xs:element nombre="grupo" type="xs:string"/>
<xs:element nombre="nacionalidad" type="xs:string"/>

<!--Defino el atributo-->
<xs:attribute name="instrumento" type="xs:string" />
```

Si se quiere indicar que un valor es fijo (fixed), si hay un valor por defecto (default):

```
<xs:attribute name="componentes" type="xs:integer" default="2"/>
<xs:attribute name="componentes" type="xs:integer" fixed="2"/>
```

Los atributos son opcionales por defecto. Para especificar que el **atributo es obligatorio**, utilice el atributo "utilizar": **use="required"**

Ejercicio 3

Diseña un documento XML bien formado y validado utilizando SCHEMA, que permita almacenar la información de la **carta** de un restaurante, donde para cada **menú** debe aparecer: **nombre**, **los platos** que lo componen (entre 1 y 4) **y postre**.

El elemento menú tiene como **atributos**:

- El nombre del cocinero que elaboró dicho menú.
- El número de calorías
- El precio que de aparecer obligatoriamente.

Ejercicio 4

Diseña un documento XML bien formado y validado utilizando SCHEMA, que permita almacenar la información sobre el **ticket** de una tienda de ropa, donde para cada ticket debe aparecer: nombre del **dependiente** y **articulos** comprados.

- El elemento artículo se puede repetir indefinidamente.

El elemento ticket tiene como **atributos**:

- Nombre de la **tienda**. Debe aparecer obligatoriamente.
- **Fecha** de la compra.
- **Hora** de la compra ,

Elementos complejos

Un elemento complejo contiene otros elementos y/o atributos.

Definir un elemento complejo:

```
<employee>  
  <firstname>John</firstname>  
  <lastname>Smith</lastname>  
</employee>
```

Forma1: Declararlo directamente

```
<xs:element name="employee">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Forma2: Hacer referencia al tipo de elemento. De este modo varios elementos pueden hacer referencia al mismo tipo.

```
<xs:element name="employee" type="personinfo"/>

<xs:complexType name="personinfo">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

Elementos complejos

Un elemento complejo es un elemento XML que contiene otros elementos y / o atributos.

```
<xs:element name="nombre_elemento">
  <xs:complexType>
    <xs:sequence>
      <!-- Defino los elementos simples -->
      <xs:element name="elemento_hijo_1" type="tipo_hijo_1"/>
      <xs:element name="elemento_hijo_1" type="tipo_hijo_2"/>
    </xs:sequence>
    <!-- Defino los atributos -->
    <xs:attribute name="nombre_atributo" type="tipo-atributo"/>
  </xs:complexType>
</xs:element>
```

Restricciones o facetas:

Al definir el elemento o atributo en el SCHEMA, puedo darle un nombre al type y definirlo más adelante utilizando restricciones (facetas):

```
<xs:complexType>
  <xs:sequence>
    <xs:element name="identificacion" type="Tipodni" />
    .....
    .....
  </xs:sequence>
</xs:complexType>

<xs:simpleType name="Tipodni">
  [Restricciones]
</xs:simpleType>
```

Restricciones en cadenas de caracteres:

En el siguiente ejemplo el único valor aceptable es UNA de las letras LOWERCASE de la a a la z:

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

En el siguiente ejemplo el único valor aceptable es TRES de las letras MAYÚSCULAS de la a a la z:

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

Debe haber exactamente ocho caracteres seguidos y esos caracteres deben ser letras minúsculas o mayúsculas de la a a la z, o un número del 0 al 9:

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

Restricciones de longitud:

Para limitar la longitud de un valor en un elemento, usamos las restricciones `length`, `maxLength` y `minLength`.

Este ejemplo define una restricción. El valor del elemento o atributo debe tener exactamente ocho caracteres:

```
<xs:restriction base="xs:string">  
  <xs:length value="8"/>  
</xs:restriction>
```

Restricciones de valores

El siguiente ejemplo define una restricción. El valor del elemento o atributo no puede ser menor que 0 ni mayor que 120:

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


Restricciones sobre un conjunto de valores

Para limitar el contenido de un elemento XML a un conjunto de valores aceptables, usaríamos la restricción de enumeración.

El siguiente ejemplo define una restricción. Los únicos valores aceptables son: Audi, Golf, BMW:

```
<xs:restriction base="xs:string">
  <xs:enumeration value="Audi" />
  <xs:enumeration value="Golf" />
  <xs:enumeration value="BMW" />
</xs:restriction>
```

Más información sobre restricciones

Ejemplo 3

En el catálogo de artistas habrá las siguientes restricciones o facetas:

- El nombre y apellidos del artista y el grupo estará formado por una cadena de caracteres que comienzan con una letra mayúscula y las demás en minúsculas.
- La edad de los artistas debe estar entre los 10 y los 100 años
- El atributo instrumento del elemento artista solo puede tomar los valores: guitarra, batería o bajo.



```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<catalogo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="Schemas/ej1C.xsd">
```

```
  <artista instrumento="guitarra">
    <nombre>Marie</nombre>
    <apellido>Fredriksson</apellido>
    <edad>68</edad>
    <grupo>Roxette</grupo>
    <nacionalidad>Suecia</nacionalidad>
  </artista>
```

```
  <artista instrumento="bajo">
    <nombre>Julian</nombre>
    <apellido>Casablanca</apellido>
    <edad>72</edad>
    <grupo>Strokes</grupo>
    <nacionalidad>Estados Unidos</nacionalidad>
  </artista>
```

```
</catalogo>
```

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

```
<!-- Defino el elemento raíz -->
<xs:element name="catalogo">
  <xs:complexType>
    <xs:sequence>
      <!--Referencia al elemento artista definido más adelante-->
      <xs:element ref="artista" minOccurs="0" maxOccurs="unbounded"/>
      <!-- minOccurs, maxOccurs: el n de veces que puede aparecer el elemento artista -->
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

```
<!--Defino el elemento artista -->
<xs:element name="artista">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="nombre" type="TipoNombre"/>

      <xs:element name="apellido" type="TipoNombre"/>
      <xs:element name="edad" type="TipoEdad"/>
      <xs:element name="grupo" type="TipoNombre"/>
      <xs:element name="nacionalidad" type="xs:string"/>
    </xs:sequence>

    <!-- Defino los atributos -->
    <xs:attribute name="instrumento" type="TipoInstrumento" use="optional"/>
    <!-- Por defecto si no especificamos use será opcional -->

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

```
<!--Defino El TipoNombre con las restricciones/facetadas pedidas-->
<xs:simpleType name="TipoNombre">
  <xs:restriction base="xs:string">
    <!-- Primer caracter 1 mayúscula, seguido de minúsculas -->
    <xs:pattern value="[A-Z][a-z]"]*" />
  </xs:restriction>
</xs:simpleType>

<!--Defino El TipoEdad con las restricciones/facetadas pedidas-->
<xs:simpleType name="TipoEdad">
  <xs:restriction base="xs:integer">
    <xs:minInclusive value="10"/>
    <xs:maxInclusive value="100"/>
  </xs:restriction>
</xs:simpleType>

<!--Defino El TipoInstrumento con las restricciones/facetadas pedidas-->
<xs:simpleType name="TipoInstrumento">
  <xs:restriction base="xs:string">
    <xs:enumeration value="guitarra"/>
    <xs:enumeration value="bateria"/>
    <xs:enumeration value="bajo"/>
  </xs:restriction>
</xs:simpleType>
</xs:schema>
```

Ejercicio 5

Diseña un documento XML bien formado y validado utilizando SCHEMA, que permita almacenar la información de un **centro** educativo, teniendo en cuenta las siguientes características para definir los elementos, los atributos y las restricciones:



Desc.	Ocurrencias	Elemento	Etiqueta	Tipo de dato
	[1..1]	+ Elemento raíz	<centro>	
	[1..n]	+ +Cursos que imparte el centro	<curso>	
		+ + Atributo identificador del curso	id	TipoID
a)	[1]	+ + + nombre del curso	<nombre>	TipoNombre
b)	[1...3]	+ + + nombre de los profesores que lo imparten	<profesor>	TipoNombre
c)	[1]	+ + + Numero de alumnos inscritos	<n_alumnos>	TipoNAlumnos

Tipos de datos - descripción:

Formato: TipoID: **Identificativo único y obligatorio**. Empieza por C y le siguen 2 números.

a) Formato: TipoNombre: El valor debe tener como máximo 15 caracteres.

b) Formato: TipoNAlumnos: Número entero que no puede ser inferior a 10 ni superior a 50

Ejercicio 6

Diseña un documento XML bien formado y validado utilizando SCHEMA, que permita almacenar la información de una **floristería**, teniendo en cuenta las siguientes características para definir los elementos, los atributos y las restricciones:



Desc.	Ocurrencias	Elemento	Etiqueta	Tipo de dato
	[1..1]	+ Elemento raíz	<catalogo>	
	[1..n]	+ +Flores que venden	<flor>	
		+ + Atributo de flor con el precio	precio	TipoPrecio
a)	[1]	+ + + nombre de la flor	<nombre>	xs:string
b)	[1]	+ + + temporada en la que florecen	<temporada>	TipoTemporada
c)	[1...4]	+ + + Color de la flor	<color>	xs:string

Tipos de datos - descripción:

Formato: TipoPrecio: **obligatorio**. Un numero de cualquier cifra seguido del símbolo €

b) Formato: TipoTemporada puede tomar el valor: Primavera, verano, otoño o invierno.

Ejercicio 7

Diseña un documento XML bien formado y validado utilizando SCHEMA, que permita almacenar la información sobre tienda de ropa, teniendo en cuenta las siguientes características para definir los elementos, los atributos y las restricciones:



Desc.	Ocurrencias	Elemento	Etiqueta	Tipo de dato
	[1..1]	+ Elemento raíz	<catalogo>	
	[1..n]	++ articulo	<articulo>	
		++ Atributo código identificador del articulo	id	TipoID
a)	[1]	+++ tipo de articulo	<tipo>	TipoArticulo
b)	[1...6]	+++ color del articulo	<color>	xs:string
c)	[1]	+++ Talla del articulo	<talla>	TipoTalla
d)	[1]	+++ material del articulo	<material>	xs:string

Tipos de datos - descripción:

Formato: TipoID: **Identificativo único y obligatorio**. Empieza por A y le siguen 3 números.

- a) Formato: TipoArticulo puede tomar el valor: jersey, chaqueta, camisa, vestido, falda o pantalón.
- b) Formato: TipoTalla: Un número entero entre 36 y 48.

Ejercicio 8

Diseña un documento XML bien formado y validado utilizando SCHEMA, que permita almacenar la información sobre los ticket de compra de un bar, teniendo en cuenta las siguientes características para definir los elementos, los atributos y las restricciones:



Desc.	Ocurrencias	Elemento	Etiqueta	Tipo de dato
	[1..1]	+ Elemento raíz	<ordenes>	
	[1..n]	+ +ticket	<ticket>	
		+ +Atributo código identificador del artículo	id	TipoID
		+ +Atributo con la fecha de la compra	fecha	xs:date
a)	[1...6]	+ + + Tipo consumición	<consumicion>	TipoConsumicion
b)	[1]	+ + + Precio total	<precio>	TipoPrecio
c)	[1]	+ + + nombre del camarero	<nombre>	xs:string

Tipos de datos - descripción:

Formato: TipoID: **Identificativo único y obligatorio**. Empieza por T y le siguen 3 números.

a) Formato: TipoConsumición puede tomar el valor: refresco, copa, vino, cafe o tapa.

b) Formato: TipoPrecio puede estar formado: Un número de cualquier cifra seguido del símbolo € o un número de cualquier cifra seguido del símbolo \$