# 1. Introducción

El lenguaje XML Schema o XSD (XML schema definition) surgió en 1998 y se usa como alternativa a documentos DTD, al no cubrir este las expectativas que se necesitan, ya que no permite especificar tipos de elementos o atributos más complejos, así como añadir otras restricciones más complejas.

Entre las ventajas que presenta, destacan:

* Más definición de tipos básicos.
* Mayor control sobre el número de ocurrencias de los elementos.
* Sintáxis similar a XML.

Los XML Schema son ficheros planos con extensión .xsd que permiten definir los elementos y atributos que componen un XML, así como sus tipos, el orden de aparición, el número de veces que puede repetirse cada elemento, valores por defecto, etc.

# 2. Creación y elementos de un XSD

La estructura de todo documento XSD es similar a la de un XML, pero presentando el elemento raiz <xs:schema>. Dicho elemento contiene una serie de atributos, entre los que destacan los siguientes:

* **Xmlns:alias**. Se especifica el espacio de nombres de donde provienen los elementos y tipos usados, cuyo valor tiene que ser <http://www.w3.org/2001/XMLSchema>. El alias suele ser **xs** o **xsd**, aunque, mientras se use la misma en todo el documento, no importa el nombre que se le asigne. Hacen referencia a las etiquetas del espacio de nombres.
* **elementFormDefault**. A la hora de declarar los elementos, hay que indicar si debe añadirse el espacio de nombres delante, cuyos posibles valores son “qualified” y “unqualified”.
* **attributeFormDefault**. A la hora de declara los atributos, hay que indicar si debe añadirse el espacio de nombres delante, cuyos posibles valores son “qualified” y “unqualified”, que indica que el primero de ellos debe ser obligatorio. Por defecto, se usa la segunda opción.
* **targetNamespace**. Se especifica el espacio de nombres de los elementos definidos.
* **version**. Se especifica la versión del documento de esquema.

Aunque no es obligatorio, la primera línea puede ser la declaración del XML. En la segunda, tal y como se muestra a continuación, se define el esquema, con la especificación de los distintos elementos. El alias en este caso es .xs:

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

## 2.1. Elementos

Puede realizarse una distinción entre los tipos de elementos de un XSD. Están los elementos simples y complejos. Elementos sin hijos ni atributos o simples se definen con la marca <xs:element>, conteniendo solamente valores de un determinado tipo.

<xs:element Nombre Tipo Valor/>

Donde:

* **Nombre**: se representa mediante el atributo **name**, asignándole el valor que se desee.
* **Tipo**: se usa el atributo **type**, que permite especificar el tipo de elemento (ver cuadro 4.6).
* **Valor**: puede ser un valor por defecto (**default**) que permite ser modificado o un valor fijo obligatorio (**fixed**) que no podrá cambiarse. Cada uno de ellos irá acompañado de su correspondiente asignación.

A continuación, se aprecia un elemento XSD llamado color de tipo cadena de caracteres o texto y valor por defecto “azul”.

|  |
| --- |
| <xs:element name=”color” type=”xs:string” default=”azul” /> |

### 2.1.1. Indicadores de ocurrencias

Indica las veces que un elemento debe aparecer o repetirse mediante dos atributos que informan del número máximo y mínimo de veces que puede aparecer.

|  |  |
| --- | --- |
| **Atributo** | **Significado** |
| **maxOccurs** | Se especifica un valor con el número máximo de veces que puede aparecer el elemento, cuyo valor por defecto es 1. Para especificar un número de veces ilimitado, el valor es “unbounded”. |
| **minOccurs** | Se especifica un valor con el número mínimo de veces que puede aparecer el elemento, cuyo valor por defecto es 1. Para especificar un número de veces ilimitado, el valor es “unbounded”. |

### 2.1.2. Elementos complejos

Contienen otros elementos y atributos. Dependiendo del contenido, pueden darse ejemplos de elementos con hijos, con contenido vacío y atributos, con contenido no vacío y atributos, entre otros.

Se definen con el tipo de datos <xs:complexType>, seguido de un indicador de orden y los elementos simples, que puede variar dependiendo del contenido que tenga.

<xs:element name=”nombreElementoCompuesto”>

<xs:complexType>

<Indicador\_orden>

Elementos/Atributos

</Indicador\_orden>

</xs:complexType>

</xs:element>

Donde:

* **<xs:complexType>**: se usa para definir elementos de tipo complejo.
* Indicador\_orden: en este apartado, hay que especificar un indicador de orden de aparición de los distintos elementos, que puede ser tres distintos:
  + **<xs:sequence>**: se especifican los hijos del elemento principal, siguiendo la secuencia u orden de aparición indicada.
  + **<xs:choice>**: se especifica el elemento hijo que puede aparecer de entre todos. Solo uno de ellos.
  + **<xs:all>**: los hijos pueden aparecer en cualquier orden una sola vez, la secuencia aparecerá en el orden que se desee.

### Ejemplo <xs:sequence>

Declara un XSD con elementos complejos mediante referencias. La secuencia de elementos que debe aparecer es nombre, email y fechaNac, en ese orden. La referencia se realiza con el atributo ref y cuyo valor es el nombre del elemento situado en la parte inferior, donde se especifica el tipo.

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

<xs:element name="usuario">

<xs:complexType>

**<xs:sequence>**

<xs:element ref=”nombre”/>

<xs:element ref=”email”/>

<xs:element ref=”fechaNac”/>

**</xs:sequence>**

</xs:complexType>

</xs:element>

<xs:element name=”nombre” type=”xs:string” />

<xs:element name=”email” type=”xs:string” />

<xs:element name=”fechaNac” type=”xs:date” />

</xs:schema>

El ejemplo anterior puede realizarse sin referencias (mejor opción):

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

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

<xs:element name="usuario">

<xs:complexType>

<xs:sequence>

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

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

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

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:schema>

Un XML correcto a la hora de validarlo sería:

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

<usuario xmlns:xs="http://www.w3.org/2001/XMLSchema-instance" xs:noNamespaceSchemaLocation="pr1.xsd">

<nombre>Lorena J. Abascal</nombre>

<email>a@b.com</email>

<fechaNac>2018-10-25</fechaNac>

</usuario>

Prueba a cambiar el orden de los elementos. ¿Qué ocurre?

### Ejemplo <xs:choice>

Realiza un XSD en el que el usuario se identifique con su nombre de usuario o email, con uno de los dos.

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

<xs:element name="usuario">

<xs:complexType>

<xs:choice>

<xs:element name=”nombre” type=”xs:string”/>

<xs:element name=”email” type=”xs:string”/>

</xs:choice>

</xs:complexType>

</xs:element>

</xs:schema>

Posible XML válido:

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

<usuario xmlns:xs="http://www.w3.org/2001/XMLSchema-instance" xs:noNamespaceSchemaLocation="pr2.xsd">

<email>a@b.com</email>

</usuario>

Prueba a insertar también el nombre. ¿Qué ocurre?

### Ejemplo <xs:all>

Recordamos que para ser válido este indicador, basta con que estén todos los elementos sin importar el orden.

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

<xs:element name="usuario">

<xs:complexType>

<xs:all>

<xs:element name=”nombre” type=”xs:string”/>

<xs:element name=”email” type=”xs:string”/>

<xs:element name=”fechaNac” type=”xs:date”/>

</xs:all>

</xs:complexType>

</xs:element>

</xs:schema>

Pruebalo con un xml haciendo que falle.

### Ejercicio resuelto

A partir de este XML, genera un XML asociado válido.

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

<nombre xmlns:xs="http://www.w3.org/2001/XMLSchema-instance" xs:noNamespaceSchemaLocation="pr3.xsd" dni="11111111G">María</nombre>

**Solución**:

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

<xs:element name="nombre">

<xs:complexType>

<xs:simpleContent>

<xs:extension base="xs:string">

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

</xs:extension>

</xs:simpleContent>

</xs:complexType>

</xs:element>

</xs:schema>

## Ejercicio 1

Crea un XSD para un XML de un elemento vacío con atributo como el siguiente:

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

<nombre dni=”11111111G”></nombre>

**Solución:**

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

<xs:element name="nombre">

<xs:complexType>

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

</xs:complexType>

</xs:element>

</xs:schema>

## 2.2. Tipos de datos XSD

Los más comunes son:

|  |  |
| --- | --- |
| **Tipo** | **Descripción** |
| **decimal** | Número decimal. |
| **boolean** | Un único valor lógico (1,0). |
| **dateTime** | Fecha y hora en formato (AAAA-MM-DD T HH:MM:SS). |
| **string** | Texto o cadena de caracteres. |
| **date** | Fecha: año-mes-día (aaaa-mm-dd). |
| **time** | Hora: hh:mm:ss. |
| **integer** | Número entero positivo o negativo. |
| **positiveInteger** | Número entero positivo. |
| **long** | Entero de 64 bits. |
| **short** | Entero de 16 bits. |

*Cuadro 4.6*

### 2.2.1. Tipos de datos simples personalizados

En ocasiones, es interesante crear tipos de datos nuevos, dotándolos de mayor precisión y eficiencia. Estos pueden ser asignados a elementos o atributos, siendo reutilizados en el esquema XSD.

Estructura:

<xs:simpleType name=”Tcat1”>

…

</xs:simpleType>

### 2.2.2. Tipos de datos complejos

Ya vistos, vemos un ejemplo:

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

<usuario xmlns:xs="http://www.w3.org/2001/XMLSchema-instance" xs:noNamespaceSchemaLocation="pr1.xsd">

<nombre>María Gómez</nombre>

<email>a@b.com</email>

<fechaNac>1983-11-10</fechaNac>

</usuario>

Y su XSD asociado:

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

<xs:element name="usuario" type="Tuser" />

<xs:complexType name="Tuser">

<xs:sequence>

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

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

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

</xs:sequence>

</xs:complexType>

</xs:schema>

## 2.3. Atributos

Para definir un atributo:

<xs:attribute NombreA TipoA USO ValorA />

Siendo:

NombreA: atributo **name**, asignandole el valor que se desee.

TipoA: atributo **type** para especificar el tipo de elemento.

USO: se especifica la opcionalidad o no del atributo mediante **use**, que puede tener los valores:

* prohibited
* required
* optional

ValorA: puede ser 

* default: valor por defecto.
* fixed: valor fijo.

Ej:

<xs:attribute **name**=”dni” **type**=”xs:integer” **use**=”required” />

## 2.4. Restricciones

|  |  |
| --- | --- |
| **Restricción o faceta** | **Descripción** |
| **Length** | Contiene un atributo value donde se especifica un valor de longitud fija. |
| **minLength** | Contiene un atributo value donde se especifica el valor de longitud mínima. |
| **maxLength** | Contiene un atributo value donde se especifica el valor de longitud máxiam. |
| **whiteSpace** | Contiene un atributo value donde se especifica un valor que puede ser (“preserve”,”replace” y “collapse”). Trata el uso de los espacios en blanco, retornos de carro, etc., que puedan aparecer. |
| **maxExclusive** | Contiene un atributo value donde se especifica el valor, que es menor al especificado. |
| **minExclusive** | Contiene un atributo value donde se especifica el valor, que es mayor al especificado. |
| **minInclusive** | Contiene un atributo value donde se especifica el valor, que es mayor o igual al especificado. |
| **maxInclusive** | Contiene un atributo value donde se especifica el valor, que es menor o igual al especificado. |
| **totalDigits** | Contiene un atributo value donde se especifica el valor, que es el número máximo de dígitos de un número teniendo en cuenta los decimales. |
| **fractionDigits** | Contiene un atributo value donde se especifica el valor, que es el número máximo de dígitos de decimales de un número. |
| **Enumeration** | Contiene un atributo value donde se especifica uno de los valores admitidos de la lista. |
| **pattern** | Contiene un atributo value donde se especifica un rango de caracteres admitidos o expresión regular. |

Ejemplo de uso de restricciones

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

<bdproductos>

<producto>

<nombre>teclado</nombre>

<codigo>C001</codigo>

<categoria>A</categoria>

<iva>21</iva>

<precio>15</precio>

<descuento>10</descuento>

</producto>

<producto>

<nombre>monitor</nombre>

<codigo>C002</codigo>

<categoria>B</categoria>

<iva>21</iva>

<precio>147.2</precio>

<descuento>9</descuento>

</producto>

</bdproductos>

El esquema asociado es el que aparece a continuación:

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

<xs:element name="bdproductos">

<xs:complexType>

<xs:sequence>

</xs:element maxOccurs=”unbounded” ref=”producto” />

</xs:sequence>

</xs:complexType>

</xs:element>

<xs:element name="producto">

<xs:complexType>

<xs:sequence>

</xs:element name=”nombre” type=”xs:string” />

</xs:element name=”codigo” type=”Tlongitudentre” />

</xs:element name=”categoria” type=”Tcat1” />

</xs:element name=”iva” type=”xs:integer” />

</xs:element name=”precio” type=”Tdecimal” />

</xs:element name=”descuento” type=”Tdescuento” />

</xs:sequence>

</xs:complexType>

</xs:element>

<xs:simpletype name=”Tdescuento”>

<xs:restriction base=”xs:integer”>

<xs:minInclusive value=”1” />

<xs:maxInclusive value=”10” />

</xs:restriction>

</xs:simpletype>

<xs:simpletype name=”Tdecimal”>

<xs:restriction base=”xs:decimal”>

<xs:totalDigits value=”4” />

<xs:fractionDigits value=”2” />

</xs:restriction>

</xs:simpletype>

<xs:simpletype name=”Tcat”>

<xs:restriction base=”xs:string”>

<xs:enumeration value=”A” />

<xs:enumeration value=”B” />

<xs:enumeration value=”C” />

<xs:enumeration value=”D” />

<xs:enumeration value=”E” />

</xs:restriction>

</xs:simpletype>

<xs:simpletype name=”Tcat1”>

<xs:restriction base=”xs:string”>

<xs:pattern value=”[A-E]” />

</xs:restriction>

</xs:simpletype>

<xs:simpletype name=”Tlongitud”>

<xs:restriction base=”xs:string”>

<xs:length value=”4” />

</xs:restriction>

</xs:simpletype>

<xs:simpletype name=”Tlongitudentre”>

<xs:restriction base=”xs:string”>

<xs:minLength value=”1” />

<xs:maxLength value=”4” />

</xs:restriction>

</xs:simpletype>

</xs:schema>

### 2.4.1. Restricciones mediante expresiones regulares

Como se puede apreciar en las restricciones creadas anteriormente, no hay diferencia entre Tcat y Tcat1 salvo la evidente de ahorro de código y eficacia. Pasamos a definir las expresiones que pueden usarse:

* [0-9]: número del 0 al 9.
* [a-z]: letra minúscula perteneciente a ese intervalo.
* [A-Z]: letra mayúscula perteneciente a ese intervalo.
* [aeiou]: un carácter de los que aparecen entre los corchetes.
* [^aeiou]: un carácter que no aparezca entre corchetes.
* {X}: las llaves indican que tiene que repetirse X veces el contenido que haya delante.

### Ejercicio

Elabora un patrón para DNI.

## 2.5. Asociar un XSD a un documento XML

La asociación de un documento XSD a un XML se realiza mediante un espacio de nombres con una serie de atributos. Lo vemos con un ejemplo, probadlo.

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

<bdalumnos xmlns:xs=”<http://www.w3.org/2001/XMLSchema-instance>” xs:noNamespaceSchemaLocation=”**xsdalumnos.xsd**”>

<alumno>

<nombre>Luís Fonsi</nombre>

<padre>Luís</padre>

<madre>Carlota</madre>

<edad>37</edad>

<ciudad>Málaga</ciudad>

</alumno>

<alumno>

<nombre>Paquirrín</nombre>

<padre>Desconocido</padre>

<madre>La Pantoja</madre>

<edad>37</edad>

<ciudad>Chipiona</ciudad>

</alumno>

</bdalumnos>

El esquema asociado es el que aparece a continuación:

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"a attributeFormDefault=”unqualified”>

<xs:element name="bdalumnos">

<xs:complexType>

<xs:sequence>

</xs:element maxOccurs=”unbounded” name=”alumno”>

</xs:complexType>

<xs:sequence>

</xs:element name=”nombre” type=”xs:string” />

</xs:element name=”padre” type=”xs:string” />

</xs:element name=”madre” type=”xs:string” />

</xs:element name=”edad” type=”xs:unsignedByte” />

</xs:element name=”ciudad” type=”xs:string” />

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:schema>