XML SCHEMA XSD

LIMA UD3 – Tema 1

IES Plurilingüe Antón Losada Diéguez

Adrián Fernández González

Tabla de contenido

XML Schema XSD

1. Introducción

Con DTD se pueden establecer una serie de reglas básicas para definir la estructura del XML, pero se queda algo corto en varios aspectos.

Por ello, surge XML Schema (.xsd), una alternativa más potente y similar al propio XML.

2. Archivo XSD

La extensión de los archivos de XML Schema es XSD (.xsd) y su estructura es igual a un XML, con la salvedad que hay que estipular un *namespace* especial y seguir las reglas de XML Schema.

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="https://www.example.com">
    <xs:element name="note">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="to" type="xs:string"/>
                <xs:element name="from" type="xs:string"/>
                <xs:element name="heading" type="xs:string"/>
                <xs:element name="body" type="xs:string"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

El *namespace* con prefijo es la referencia a la norma *XMLSchema*.

El atributo *targetNamespace* define el namespace de los elementos a los que va a afectar. Este se estipula en el caso de incluir diversos XSD para un mismo documento o querer afectar a un conjunto de elementos en concreto.

Como puede verse, la estructura es similar a la de un XML normal, pero con elementos ya preestablecidos, como *schema*, la raíz del documento en el que se estipulan los datos anteriores.

2.1. Incluir el archivo

Una vez creado el archivo XSD, se incluye como un *namespace* con prefijo *xsi* de la etiqueta raíz y la ruta al archivo como valor del atributo *xsi:schemaLocation*.

```xml
<?xml version="1.0"?>

<note
xmlns="https://www.example.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="archivo.xsd">
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

3. Elementos simples

XML Schema define dos tipos de elementos, los simples, que carecen de atributos y cuyo contenido es textual; y los complejos, que admiten atributos y otros elementos en su interior.

Los elementos simples solo permiten contenido textual, pero con XML Schema se puede estipular el tipo concreto del contenido. Los más comunes son:

- **xs:string**: Texto en general.
- **xs:decimal**: Número decimal.
- **xs:integer**: Número entero.
- **xs:boolean**: Booleano.
- **xs:date**: Fecha.
- **xs:time**: Hora.

La declaración de los elementos simples es sencilla, una etiqueta *element* con dos atributos, *name* que define el nombre del elemento y *type*, que estipula es tipo.

```xml
<xs:element name="apellidos" type="xs:string"/>
<xs:element name="altura" type="xs:decimal"/>
<xs:element name="cumple" type="xs:date"/>
<xs:element name="hijos" type="xs:integer"/>
```

En este ejemplo pueden verse cuatro elementos con tipos distintos.

4. Elementos complejos

Los elementos complejos admiten otros elementos en su interior y atributos. Estos se declaran definiendo un tipo de datos complejo.

```xml
<xs:element name="empleado">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="nombre" type="xs:string"/>
            <xs:element name="apellido" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

En este ejemplo se declara un elemento empleado con otros dos en su interior, nombre y apellido, ambos con contenido de tipo texto.

4.1. Elementos vacíos

Para crear un elemento vacío, hay que definirle un tipo complejo sin nada dentro o con los atributos que se necesiten.

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

4.2. Contenido mixto

Si se necesita que un elemento pueda contener otros elementos y texto, es decir, tenga contenido mixto, basta con indicarlo en el tipo complejo estableciendo el atributo *mixed* a *true*.

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

4.3. Indicadores de orden

Los indicadores de orden permiten establecer, como su nombre indica, el orden en el que aparecen los elementos.

```
<xs:element name="empleado">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="nombre" type="xs:string"/>
            <xs:element name="apellido" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

Existen tres, *all*, *sequence* y *choice*.

4.3.1. All

El indicador *all* estipula que los elementos que contiene aparecen todos, pero en cualquier orden.

```
<xs:element name="persona">
    <xs:complexType>
        <xs:all>
            <xs:element name="altura" type="xs:integer"/>
            <xs:element name="peso" type="xs:integer"/>
        </xs:all>
    </xs:complexType>
</xs:element>
```

En este ejemplo, el elemento persona contiene dos elementos de tipo entero altura y peso que puede aparecer en cualquier orden.

4.3.2. Sequence

El indicador *sequence* indica que los elementos han de aparecer todos y en ese orden determinado.

```
<xs:element name="empleado">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="nombre" type="xs:string"/>
            <xs:element name="apellido" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

En este ejemplo, el elemento empleado contiene otros dos, nombre y apellido, que tienen que aparecer en ese mismo orden.

4.3.3. Choice

El indicador *choice* estipula que uno y solo uno de los elementos que contiene aparece.

```
<xs:element name="postre">
    <xs:complexType>
        <xs:choice>
            <xs:element name="fruta" type="xs:string"/>
            <xs:element name="tarta" type="xs:string"/>
        </xs:choice>
    </xs:complexType>
</xs:element>
```

En este ejemplo, el elemento postre contiene o un elemento fruta o un elemento tarta.

4.3.4. Combinación

Los indicadores de orden pueden combinarse, incluyéndose uno dentro de otro para estipular distintas combinaciones de elementos.

```
<xs:element name="persona">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="altura" type="xs:integer"/>
            <xs:element name="peso" type="xs:integer"/>
            <xs:choice>
                <xs:element name="a" type="xs:integer"/>
                <xs:element name="b" type="xs:integer"/>
            </xs:choice>
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

En este ejemplo, el elemento persona contiene dos elementos, altura, peso y luego a o b, en ese orden.

4.4. Indicadores de ocurrencia

Los indicadores de ocurrencia, como su nombre indican permiten estipular la cantidad de veces que aparece un elemento o un grupo. Estos son *minOccurs* y *maxOccurs* y, a diferencia

de los anteriores, se estipulan como atributos de los elementos, indicadores de orden o grupos.

Por defecto, el mínimo y máximo está establecido en 1, siendo todo elemento obligatorio y como mucho aparecer una vez.

```
<xs:element name="empleado">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="nombre" type="xs:string"/>
            <xs:element name="apellido" type="xs:string" maxOccurs="2"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

En este ejemplo, el elemento empleado contiene un elemento nombre y uno o dos apellidos.

```
<xs:element name="postre">
    <xs:complexType>
        <xs:choice minOccurs="0">
            <xs:element name="fruta" type="xs:string"/>
            <xs:element name="tarta" type="xs:string"/>
        </xs:choice>
    </xs:complexType>
</xs:element>
```

En este ejemplo, el elemento postre puede contener, o no, uno de esos dos elementos.

Hay que tener en cuenta que **el indicador *choice* solo admite modificar el mínimo a 0**, ni admite un mayor mínimo ni otro máximo.

Para establecer múltiples elementos de un choice, es necesario englobarlo en otro indicador de orden.

```
<xs:element name="postre">
    <xs:complexType>
        <xs:sequence maxOccurs="5">
            <xs:choice minOccurs="0">
                <xs:element name="fruta" type="xs:string"/>
                <xs:element name="tarta" type="xs:string"/>
            </xs:choice>
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

Como puede verse, se pueden combinar para establecer, en este ejemplo, que el elemento postre puede tener de 1 a 5 ocurrencias de 0 o 1 de esos dos elementos. Por tanto, con esta combinación, el elemento postre tiene dentro de 0 a 5 combinaciones de elementos fruta y tarta.

Para establecer que no hay máximo, se establece el valor a *unbounded*.

```
<xs:element name="persona">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="hijo" type="xs:string" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

En este ejemplo, el elemento persona tiene un número ilimitado de elementos hijo.

4.5. Reutilizar tipos complejos

Los tipos complejos pueden definirse dentro de los elementos, como hasta ahora, o de forma independiente para ser reutilizados por varios elementos o aumentar la legibilidad.

Para ello, el tipo complejo (*complexType*) se crea de forma independiente y se le estipula un nombre identificador con el atributo *name*. Este identificador se estipulará como tipo de aquellos elementos que lo requieran.

```
<xs:element name="menu_a" type="menu_dia"/>
<xs:element name="menu_b" type="menu_dia"/>

<xs:complexType name="menu_dia">
    <xs:sequence>
        <xs:element name="primero" type="xs:string"/>
        <xs:element name="segundo" type="xs:string"/>
        <xs:element name="postre" type="xs:string"/>
        <xs:element name="bebida" type="xs:string"/>
    </xs:sequence>
</xs:complexType>
```

En este ejemplo, los elementos *menu_a* y *menu_b* son de tipo *menu_dia*, por lo que ambos contienen esos cuatro elementos en ese orden concreto.

Esto puede combinarse, de tal forma que un elemento dentro de un tipo complejo puede, a su vez, ser de tipo complejo.

4.6. Grupos de elementos

Los grupos permiten agrupar la definición de un conjunto de elementos para su reutilización posterior.

```
<xs:group name="persona">
    <xs:sequence>
        <xs:element name="nombre" type="xs:string"/>
        <xs:element name="apellido1" type="xs:string"/>
        <xs:element name="apellido2" type="xs:date"/>
    </xs:sequence>
</xs:group>
```

Este grupo, podrá ser utilizado posteriormente en el lugar en el que iría un elemento mediante la siguiente etiqueta:

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

4.7. Cualquiera. Any

Si se quiere estipular que dentro de un elemento hay un elemento cualquiera, se utiliza la etiqueta *any*.

```
<xs:any minOccurs="0"/>
```

En este ejemplo, con el indicador de ocurrencia mínima a 0, estipula que puede haber un elemento cualquiera o no.

4.8. Referenciar elementos

Los elementos pueden ser creados dentro de los tipos complejos en los que van a ser usados o de forma independiente y luego referenciados.

Para ello, en aquellos lugares en los que se quiera añadir el elemento, se estipula una etiqueta de elemento pero solo con un atributo *ref*, cuyo valor es el nombre del elemento a referenciar.

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

<xs:complexType name="info">
    <xs:sequence>
        <xs:element ref="nombre"/>
    </xs:sequence>
</xs:complexType>
```

En este ejemplo, primero se declara el elemento nombre y luego se le referencia dentro de un tipo complejo mediante el atributo *ref* de la etiqueta *element*.

Las referencias pueden usarse en elementos simples o complejos.

5. Atributos

Los atributos se estipulan mediante la etiqueta *attribute* siguiendo la misma estructura que la de un elemento simple.

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

Como puede verse en el ejemplo, *name* es el nombre del atributo y *type* es el tipo de valor admitido. Los posibles tipos son los mismos que en los elementos simples.

Solo los elementos de tipo complejo admiten atributos, ya que el atributo se le añade dentro del tipo complejo, como cualquier otro contenido. La diferencia es que van al final y fuera de los indicadores de orden.

```
<xs:element name="menu">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="primero" type="xs:string"/>
            <xs:element name="segundo" type="xs:string"/>
        </xs:sequence>
        <xs:attribute name="dia" type="xs:date"/>
    </xs:complexType>
</xs:element>
```

En este ejemplo se puede ver un elemento menú que tiene dos elementos, primero y segundo y un atributo día de tipo fecha. Como puede observarse, el atributo está al final, fuera de la etiqueta *sequence*.

5.1. Valores por defecto y fijos

A los atributos se les puede establecer un valor por defecto mediante el atributo *default* o un valor fijo mediante el atributo *fixed*.

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

En estos dos ejemplos, idioma se estipula con un valor por defecto o un valor fijo.

5.2. Obligatorio

Si se necesita que un atributo sea obligatorio, se estipula poniendo como *required* su atributo *use*.

```
<xs:attribute name="fecha" type="xs:date" use="required"/>
```

Por defecto, este atributo está puesto como *optional*, indicando la opcionalidad del mismo.

5.3. Grupos de atributos

Los grupos de atributos, al igual que los de elementos, permiten aunar un conjunto de atributos bajo un mismo nombre para su reutilización dentro de diversos tipos complejos.

Para ello, se definen los atributos de forma normal bajo una etiqueta *attributeGroup* con un nombre identificativo.

```
<xs:attributeGroup name="nutricion">
    <xs:attribute name="calorias" type="xs:integer"/>
    <xs:attribute name="grasas" type="xs:decimal"/>
</xs:attributeGroup>
```

Luego, se incluye en el lugar de un atributo mediante una referencia:

```
<xs:attributeGroup ref="nutricion"/>
```

5.4. Cualquiera. anyAttribute

Si se quiere indicar que un elemento admite un atributo cualquiera, esté definido en el XSD o no, se estipula con la etiqueta *anyAttribute* de igual forma que cualquier otro atributo.

```
<xs:anyAttribute/>
```

6. Restricciones

Las restricciones permiten establecer límites a los valores de elementos y atributos.

Estas pueden ser desde un valor mínimo y máximo hasta un conjunto de valores permitidos o que cumplan una expresión regular determinada.

Para ello, se define un tipo simple con la etiqueta *simpleType* y dentro las restricciones, usando como base un tipo básico ya existente.

```
<xs:simpleType name="max3">
    <xs:restriction base="xs:decimal">
        <xs:totalDigits value="3"/>
        <xs:fractionDigits value="1"/>
    </xs:restriction>
</xs:simpleType>
```

En este ejemplo se puede ver un tipo *max3* que extiende del tipo *decimal* al que se le estipula un número máximo de 3 dígitos, de los cuales 1 decimal.

En este caso se definió el tipo de forma independiente para su posterior referenciado, pero se puede estipular dentro de un elemento.

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

En este ejemplo se han definido las restricciones dentro del propio elemento, estipulando los tres posibles valores posibles de tipo *string*.

7. Extensión de tipos

La extensión de tipos permite ampliar un tipo ya existente añadiéndole nuevas características.

Para ello, se define un nuevo tipo complejo y se le añade lo que se quiera extender mediante una etiqueta *simpleContent* si no se van a añadir nuevos elementos (etiqueta *element*) o *complexContent* si se añaden nuevos elementos.

```
<xs:complexType name="datos">
    <xs:simpleContent>
        <xs:extension base="max3">
            <xs:attribute name="fecha" type="xs:date"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
```

En este ejemplo se extiende el tipo *max3* del ejemplo anterior añadiéndole un atributo.