

XML Schema (parte 2)

LENGUAJES DE MARCAS Y SISTEMAS DE GESTIÓN DE INFORMACIÓN



Elementos en XML Schema

SIMPLES

[Sólo pueden contener texto.
No pueden contener otros
elementos ni atributos]

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

Se puede añadir

- `default="valor_por_defecto"`
- `fixed="valor_fijo"`

`xs:string`
`xs:decimal`
`xs:integer`
`xs:boolean`
`xs:date`
`xs:time`

COMPLEJOS

[Contienen otros elementos y/o
atributos. Aquí se incluyen
también los vacíos]



ATRIBUTOS



RESTRICCIONES

[Limitación del contenido de
elementos simples y atributos]

Dentro del elemento/atributo se introduce:

`<xs:simpleType>`

`<xs:restriction base=" " >`

(el type "baja" aquí)

`<xs: value=" " />`

`</xs:restriction>`

`</xs:simpleType>`

`enumeration`

`length`

`maxLength`

`minLength`

`maxInclusive`

`maxExclusive`

`minInclusive`

`minExclusive`

`pattern`

`totalDigits`

`fractionDigits`

`whiteSpace`

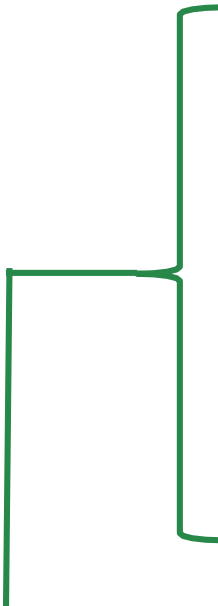
XML Schema → Atributos

Antes de nada, recuerda:

- Los elementos simples no tienen atributos.
- Por lo tanto, si un elemento tiene atributos, ya lo consideraremos de tipo complejo.

Los atributos se declaran igual que los elementos simples, pero cambiando *element* por *attribute*:

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



xs:string
xs:decimal
xs:integer
xs:boolean
xs:date
xs:time

(Esta declaración irá dentro de un elemento complejo. Luego veremos cómo)

XML Schema → Atributos

A los atributos, igual que a los elementos simples, se les podría añadir:

- **default="valor_por_defecto"** → Para darle un valor por defecto
- **fixed="valor_fijo"** → Para darle un valor fijo

Y, además, se les podría añadir:

- **use="required"** → Para que el atributo sea **obligatorio**
Si no se pone, los atributos por defecto son opcionales.

Ejemplos: `<xs:attribute name="unidades" type="xs:integer" default="10"/>`
`<xs:attribute name="vip" type="xs:boolean" use="required"/>`

} (Esto irá dentro de un elemento, insisto. No suelto)

XML Schema → Atributos

Y, como a los elementos simples, a los atributos también se les pueden añadir **restricciones**.

Se hace de la misma manera, y existen los mismos 12 tipos.

Por ejemplo:

enumeration	minInclusive
length	minExclusive
maxLength	pattern
minLength	totalDigits
maxInclusive	fractionDigits
maxExclusive	whiteSpace

```
<xs:attribute name="estrellas" use="required">
  <xs:simpleType>
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="1"/>
      <xs:maxInclusive value="5"/>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>
```

Si este atributo estuviera dentro del elemento hotel...



```
<hotel estrellas="5">Gran Sol</hotel>
```



```
<hotel estrellas="6">Hotel Paquirrín</hotel>
```



```
<hotel>Hostal Fuentetocino</hotel>
```

Elementos en XML Schema

SIMPLES

[Sólo pueden contener texto.
No pueden contener otros
elementos ni atributos]

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

Se puede añadir

- `default="valor_por_defecto"`
- `fixed="valor_fijo"`

`xs:string`
`xs:decimal`
`xs:integer`
`xs:boolean`
`xs:date`
`xs:time`

COMPLEJOS

[Contienen otros elementos y/o
atributos. Aquí se incluyen
también los vacíos]

ahora

ATRIBUTOS

Igual que los elementos simples, pero **attribute** en vez de *element*

Además de `default="valor_por_defecto"`
`fixed="valor_fijo"`, se puede añadir `use="required"`

RESTRICCIONES

[Limitación del contenido de
elementos simples y atributos]

Dentro del elemento/atributo se introduce:

`<xs:simpleType>`

`<xs:restriction base=" " >` (el type "baja" aquí)

`<xs: value=" " />`

`</xs:restriction>`

`</xs:simpleType>`

enumeration

length

maxLength

minLength

maxInclusive

maxExclusive

minInclusive

minExclusive

pattern

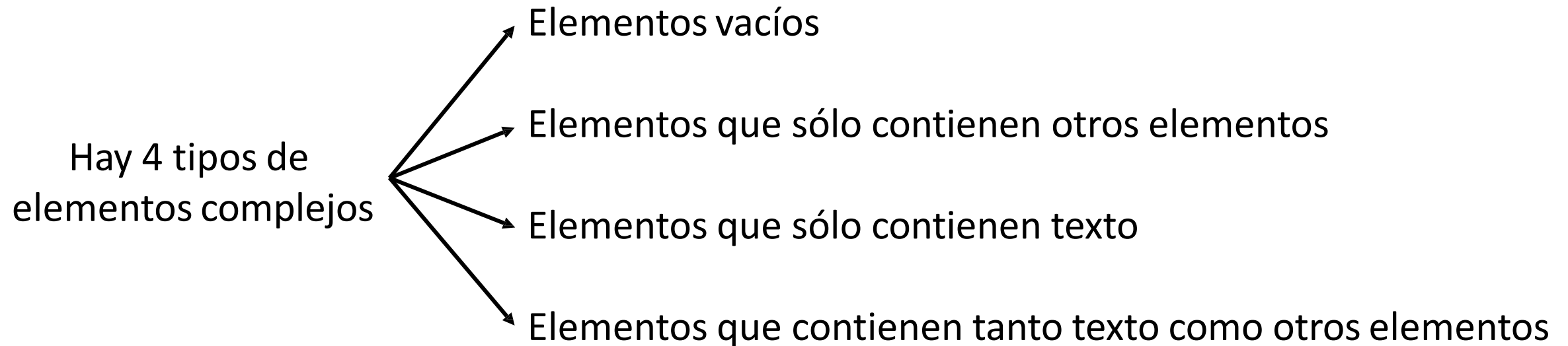
totalDigits

fractionDigits

whiteSpace

XML Schema → Elem. complejos

En XML Schema se consideran elementos complejos aquellos que contienen otros elementos y/o atributos. Aquí se incluyen también los vacíos.



XML Schema → Elem. complejos → Vacíos

Veremos directamente un ejemplo cómo declarar un elemento vacío...

XML

```
...  
<producto cod="1234"/>  
...
```



XML Schema

```
<xs:element name="producto">  
  <xs:complexType>  
    <xs:attribute name="cod" type="xs:integer"/>  
  </xs:complexType>  
</xs:element>
```

Para declarar un elemento vacío, dentro de él se incluye la etiqueta **<xs:complexType>** sin nada dentro.


En este caso, sin embargo, se incluye un atributo, para tener así un elemento vacío con atributo.

XML Schema → Elem. complejos → Otros elem.

Si un elemento contiene otros elementos...

XML

```
...  
<empleado>  
  <nombre>Pepe</nombre>  
  <apellido>Ruiz</apellido>  
</empleado>  
...
```



XML Schema

```
<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 él se incluye la etiqueta **<xs:complexType>**

Dentro de **<xs:complexType>**, se incluye una de estas 3 etiquetas

<xs:sequence>
<xs:choice>
<xs:all>

Y dentro de ella los elementos, que podrían tener al final estas 2 cosas

maxOccurs="■"
minOccurs="■"

XML Schema → Elem. complejos → Otros elem.

Indicadores de orden de aparición de los elementos

<xs:sequence> Los hijos deben aparecer en el orden en que se pongan

<xs:choice> Sólo uno de los elementos hijo puede aparecer

<xs:all> Los elementos hijo pueden aparecer en cualquier orden

Indicadores de ocurrencia (cuántas veces pueden aparecer)

maxOccurs="■" Número máximo de veces que puede aparecer

minOccurs="■" Número mínimo de veces que debe aparecer

A tener en cuenta

- Si se usa **all**, el indicador **maxOccurs** tiene que valer 1 y **minOccurs** puede valer 0 o 1.
- Sin embargo, si se usa **sequence**, podemos dar a **minOccurs** y **maxOccurs** los valores que queramos (5 y 8, por ejemplo).
- Si en un elemento ponemos **maxOccurs="unbounded"** significa que sus apariciones son ilimitadas.

XML Schema → Elem. complejos → Mixtos

XML

```
...  
<carta>  
  Estimado <nombre>David</nombre>:  
  Su orden <codigo>1032</codigo> se  
  enviará el <fecha>2018-05-21</fecha>  
</carta>  
...
```



XML Schema

```
<xs:element name="carta">  
  <xs:complexType mixed="true">  
    <xs:sequence>  
      <xs:element name="nombre" type="xs:string"/>  
      <xs:element name="codigo" type="xs:integer"/>  
      <xs:element name="fecha" type="xs:date"/>  
    </xs:sequence>  
  </xs:complexType>  
</xs:element>
```

Si un elemento contendrá tanto texto como otros elementos, en él se introduce **<xs:complexType mixed="true">** (ojo, los casos anteriores no tenían ese **mixed="true"**). Luego ya, dentro, se introducen los elementos hijos igual que hacíamos en los elementos que sólo contienen otros elementos (eligiendo **<xs:sequence>**, **<xs:choice>** o **<xs:all>** y dentro los *element*).

XML Schema → Elem. complejos → Resumen

Declarar elementos complejos en XML Schema

Vacíos	Contienen sólo otros elementos	Contienen sólo texto	Contienen texto y otros elementos (mixtos)
Dentro de él se introduce <xs:complexType> sin nada dentro. (Eso sí, podría tener algún atributo)	Dentro de él se introduce <xs:complexType> . Dentro de ella, una de estas 3: <xs:sequence> , <xs:all> o <xs:choice> . Y dentro de ella, se incluyen las etiquetas <xs:element...> correspondientes. Éstas, además, podrían tener maxOccurs="■" o minOccurs="■" .	Dentro de él se introduce <xs:complexType> . Dentro de ella, la etiqueta <xs:simpleContent> . En su interior, la etiqueta <xs:extension base="■"> con el tipo del contenido. Y, dentro de ella, por último, se incluye el atributo.	Igual que los que contienen otros elementos, pero a la etiqueta <xs:complexType> se le añade mixed="true" .



SIMPLES

[Sólo pueden contener texto.
No pueden contener otros
elementos ni atributos]

`<xs:element name="nombre_elemento" type=" " />`
Se puede añadir
 `default="valor_por_defecto"`
 `fixed="valor_fijo"`

`xs:string`
`xs:decimal`
`xs:integer`
`xs:boolean`
`xs:date`
`xs:time`

Elementos en XML Schema

COMPLEJOS

[Contienen otros elementos y/o
atributos. Aquí se incluyen
también los vacíos]

Declarar elementos complejos en XML Schema			
Vacíos	Contienen sólo otros elementos	Contienen sólo texto	Contienen texto y otros elementos (mixtos)
Dentro de él se introduce <code><xs:complexType></code> sin nada dentro. (Eso sí, podría tener algún atributo)	Dentro de él se introduce <code><xs:complexType></code> . Dentro de ella, una de estas 3: <code><xs:sequence></code> , <code><xs:all></code> o <code><xs:choice></code> . Y dentro de ella, se incluyen las etiquetas <code><xs:element...></code> correspondientes. Éstas, además, podrían tener <code>maxOccurs=" " </code> o <code>minOccurs=" " </code> .	Dentro de él se introduce <code><xs:complexType></code> . Dentro de ella, la etiqueta <code><xs:simpleContent></code> . En su interior, la etiqueta <code><xs:extension base=" " ></code> con el tipo del contenido. Y, dentro de ella, por último, se incluye el atributo.	Igual que los que contienen otros elementos, pero a la etiqueta <code><xs:complexType></code> se le añade <code>mixed="true"</code> .

ATRIBUTOS

Igual que los elementos simples, pero **attribute** en vez de *element*
Además de `default="valor_por_defecto"`, se puede añadir `use="required"`
`fixed="valor_fijo"`

RESTRICCIONES

[Limitación del contenido de
elementos simples y atributos]

Dentro del elemento/atributo se introduce:

```
<xs:simpleType>  
  <xs:restriction base=" " >  
    <xs:  value=" " />  
  </xs:restriction>  
</xs:simpleType>
```

(el type "baja" aquí)

enumeration	minInclusive
length	minExclusive
maxLength	pattern
minLength	totalDigits
maxInclusive	fractionDigits
maxExclusive	whiteSpace