

# Tipos complexos

## Sumario

- 1 Tipos complexos
  - ◆ 1.1 Elementos XSD
    - ◇ 1.1.1 Que é un elemento complexo?
    - ◇ 1.1.2 Exemplos de elementos complexos
    - ◇ 1.1.3 Como definir un elemento complexo
  - ◆ 1.2 Elementos baleiros XSD
  - ◆ 1.3 Soamente elementos XSD
  - ◆ 1.4 Soamente texto XSD
  - ◆ 1.5 Contido misturado XSD
  - ◆ 1.6 Indicadores XSD
    - ◇ 1.6.1 Indicadores de orde
    - ◇ 1.6.2 Indicadores de ocorrencia
    - ◇ 1.6.3 Indicadores de grupo
  - ◆ 1.7 Elemento <any> XSD
  - ◆ 1.8 Elemento <anyAttribute>
  - ◆ 1.9 Elemento Substitution XSD
    - ◇ 1.9.1 Elemento de substitución block

## Tipos complexos

Os elementos complexos conteñen outros elementos ou atributos.

### Elementos XSD

#### Que é un elemento complexo?

Un elemento complexo é un elemento de XML que contén outros elementos e/ou atributos.

Hai catro clases de elementos complexos:

- elementos baleiros (con atributos)
- elementos que conteñen soamente outros elementos
- elementos que conteñen soamente texto (con atributos)
- elementos que conteñen tanto outros elementos como texto

NOTA: Cada un destes elementos pode conter atributos tamén.

#### Exemplos de elementos complexos

Un elemento de XML complexo, "produto", o cal está baleiro:

```
<produto pid="1345"/>
```

Un elemento de XML complexo, "empregado", o cal contén só outros elementos:

```
<empregado>
  <nome>Marta</nome>
  <apelidos>Pérez</apelidos>
</empregado>
```

Un elemento de XML complexo, "comida", o cal contén soamente texto:

```
<comida type="postre">Xeadro de vainilla</comida>
```

Un elemento de XML complexo, "descripcion" o cal contén ambos elementos e texto:

```
<descripcion>
Sucedu en <data lang="spanish">03.03.99</data> ....
</descripcion>
```

## Como definir un elemento complexo

Mira o seguinte elemento XML "empregado", que contén outros elementos:

```
<empregado>
  <nome>Marta</nome>
  <apelidos>Pérez</apelidos>
</empregado>
```

Podemos definir un elemento complexo nun XML Schema de dous xeitos diferentes:

1.- O elemento "empregado" pode ser declarado directamente nomeando o elemento, así:

```
<xs:element name="empregado">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="nome" type="xs:string"/>
      <xs:element name="apelidos" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Se empregamos este método soamente o elemento "empregado" poderá usar este tipo complexo. Fíxate que os elementos fillos nome e apelidos están dentro do indicador **<sequence>**. Isto quere dicir que os elementos fillos deben aparecer na mesma orde na que foron declarados.

2.- O elemento empregado pode ter un atributo que faga referencia ó nome do tipo complexo utilizado:

```
<xs:element name="empregado" type="infopersoal"/>

<xs:complexType name="infopersoal">
  <xs:sequence>
    <xs:element name="nome" type="xs:string"/>
    <xs:element name="apelidos" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

Si usamos este método, podemos permitir que outros elementos fagan referencia ó tipo complexo, por exemplo:

```
<xs:element name="empregado" type="infopersoal"/>
<xs:element name="estudiante" type="infopersoal"/>
<xs:element name="profesor" type="infopersoal"/>

<xs:complexType name="infopersoal">
  <xs:sequence>
    <xs:element name="nome" type="xs:string"/>
    <xs:element name="apelidos" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

Incluso podemos crear outro elemento complexo baseado no elemento complexo anterior para así engadir máis elementos, por exemplo:

```
<xs:element name="empregado" type="infopersoalcompleta"/>

<xs:complexType name="infopersoal">
  <xs:sequence>
    <xs:element name="nome" type="xs:string"/>
    <xs:element name="apelidos" type="xs:string"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="infopersoalcompleta">
  <xs:complexContent>
    <xs:extension base="infopersoal">
      <xs:sequence>
        <xs:element name="direccion" type="xs:string"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

```

        <xs:element name="cidade" type="xs:string"/>
        <xs:element name="pais" type="xs:string"/>
    </xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>

```

## Elementos baleiros XSD

Un elemento baleiro XSD non pode ter contido, soamente atributos.

Exemplo de elemento XML baleiro:

```
<produto prodid="1345" />
```

É posible declarar o elemento "produto" da seguinte maneira:

```

<xs:element name="produto">
  <xs:complexType>
    <xs:attribute name="prodid" type="xs:positiveInteger"/>
  </xs:complexType>
</xs:element>

```

Ou podemos darlle un nome ó tipo complexo e deixar que o elemento teña un atributo que faga referencia ó nome do tipo complexo (si usamos este método, diferentes elementos poderán referenciar ó mesmo tipo complexo):

```

<xs:element name="produto" type="tipopoproduto"/>

<xs:complexType name="tipopoproduto">
  <xs:attribute name="prodid" type="xs:positiveInteger"/>
</xs:complexType>

```

## Soamente elementos XSD

Un tipo complexo "elements-only" contén un elemento que contén soamente outros elementos.

Por exemplo temos un elemento "persoa" que contén soamente outros elementos:

```

<persoa>
  <nome>Emilio</nome>
  <apelidos>Martínez</apelidos>
</persoa>

```

Poderemos definir nun schema XML "persoa" do seguinte xeito:

```

<xs:element name="persoa">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="nome" type="xs:string"/>
      <xs:element name="apelidos" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

NOTA: O tag <sequence> indica que os elementos definidos ("nome" e "apelidos") deberán aparecer na orde indicada polo elemento "persoa".

Ou tamén poderíamos dar un nome a ese tipo complexo para poder emprega-lo noutros elementos. Por exemplo:

```

<xs:element name="persoa" type="tipopersoa"/>

<xs:complexType name="tipopersoa">
  <xs:sequence>
    <xs:element name="nome" type="xs:string"/>
    <xs:element name="apelidos" type="xs:string"/>
  </xs:sequence>
</xs:complexType>

```

## Soamente texto XSD

Un elemento complexo "text-only" pode conter texto e atributos.

Este tipo contén soamente contidos sinxelos (texto e atributos). Cando empregamos **simpleContent** deberemos definir unha extensión ou restrición dentro do elemento, por exemplo:

```
<xs:element name="algunnome">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="tipobase">
        ....
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

Ou

```
<xs:element name="algunnome">
  <xs:complexType>
    <xs:simpleContent>
      <xs:restriction base="tipobase">
        ....
      </xs:restriction>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

**NOTA:** Poderemos empregar a restrición/extensión para expandir ou limitar o tipo base para o elemento.

Exemplo de elemento XML "numeroszapatos" que contén soamente texto:

```
<numeroszapatos pais="france">35</numeroszapatos >
```

O seguinte exemplo declara un tipo complexo "numeroszapatos". O contido está definido como un integer e o elemento numeroszapatos tamén contén o atributo "pais":

```
<xs:element name="numeroszapatos ">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:integer">
        <xs:attribute name="pais" type="xs:string" />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

Incluso poderíamos sdarlle ao tipo complexo un nome e deixar que "numeroszapatos" teña un atributo que faga referencia a ese tipo complexo (deste xeito varios elementos poderían facer referencia ó mesmo tipo complexo):

```
<xs:element name="numeroszapatos" type="tipozapatos"/>

<xs:complexType name="tipozapatos">
  <xs:simpleContent>
    <xs:extension base="xs:integer">
      <xs:attribute name="pais" type="xs:string" />
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```

## Contido misturado XSD

Un elemento de tipo contido misturado pode ter atributos, elementos e texto.

Exemplo de elemento XML "carta" que contén tanto texto como outros elementos:

```
<carta>
  Estimado/a <nome>John Smith</nome>.
  O seu pedido <numpedido>1032</numpedido>
  será entregado <dataentrega>2011-01-19</dataentrega>.
</carta>
```

O seguinte schema declara o elemento "carta":

```
<xs:element name="carta">
  <xs:complexType mixed="true">
    <xs:sequence>
      <xs:element name="nome" type="xs:string"/>
      <xs:element name="numpedido" type="xs:positiveInteger"/>
      <xs:element name="dataentrega" type="xs:date"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

**NOTA:** Para permitir que poidamos introducir caracteres entre os elementos fillo, deberemos por atributo "mixed" a "true".

Tamén poderíamos darlle ó tipo complexo un nome e deixar o elemento "carta" cun atributo que faga referencia a ese tipo complexo.

```
<xs:element name="carta" type="tipocarta"/>

<xs:complexType name="tipocarta" mixed="true">
  <xs:sequence>
    <xs:element name="nome" type="xs:string"/>
    <xs:element name="numpedido" type="xs:positiveInteger"/>
    <xs:element name="dataentrega" type="xs:date"/>
  </xs:sequence>
</xs:complexType>
```

## Indicadores XSD

Podemos controlar COMO van ser empregados os elementos nos documentos, empregando os indicadores.

Hai 7 indicadores: Indicadores de orde:

- All
- Choice
- Sequence

Indicadores de ocorrencias:

- maxOccurs
- minOccurs

Indicadores de Grupo:

- Group name
- attributeGroup name

### Indicadores de orde

**Indicador All:** O indicador "All" especifica que os elementos fillos poden aparecer en calquera orde e que cada elemento fillo debe aparecer soamente unha vez:

```

<xs:element name="persoa">
  <xs:complexType>
    <xs:all>
      <xs:element name="nome" type="xs:string"/>
      <xs:element name="apelidos" type="xs:string"/>
    </xs:all>
  </xs:complexType>
</xs:element>

```

**NOTA:** Cando empregamos o indicador <all> podemos establecer <minOccurs> a 0 ou 1 e <maxOccurs> soamente a 1.

**Indicador Choice:** O indicador <choice> especifica que pode aparecer tanto un elemento fillo coma o outro:

```

<xs:element name="persoa">
  <xs:complexType>
    <xs:choice>
      <xs:element name="empregado" type="empregado"/>
      <xs:element name="membro" type="membro"/>
    </xs:choice>
  </xs:complexType>
</xs:element>

```

**Indicador Sequence:** O indicador <sequence> especifica que os elementos fillo deben aparecer na orde especificada:

```

<xs:element name="persoa">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="nome" type="xs:string"/>
      <xs:element name="apelidos" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

## Indicadores de ocorrencia

Os indicadores de ocorrencia úsanse para definir cantas veces pode aparecer un elemento.

**NOTA:** Para os indicadores "Order" e "Group" (any, all, choice, sequence, group name e group reference) o valor por defecto para maxOccurs e minOccurs é 1.

**Indicador maxOccurs:** O indicador <maxOccurs> especifica o número máximo de veces que pode aparecer un elemento:

```

<xs:element name="persoa">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="nome_completo" type="xs:string"/>
      <xs:element name="alcumes" type="xs:string" maxOccurs="10"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

No exemplo anterior indícase que o elemento "alcumes" pode aparecer un mínimo de unha vez (o valor por defecto para minOccurs é 1) é un máximo de 10 veces no elemento "persoa".

**Indicador minOccurs:** O indicador <minOccurs> especifica o número mínimo de veces que pode aparecer un elemento:

```

<xs:element name="persoa">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="nome_completo" type="xs:string"/>
      <xs:element name="alcumes" type="xs:string"
        maxOccurs="10" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

O exemplo indica que o elemento <alcumes> pode aparecer un mínimo de 0 veces e un máximo de 10 veces no elemento "persoa".

**NOTA:** Para permitir que un elemento apareza un número ilimitado de veces, use `maxOccurs="unbounded"`.

## Indicadores de grupo

Os indicadores de grupo úsanse para definir conxuntos relacionados de elementos.

**Grupos de elementos:** Os "Element groups" defínense coa declaración:

```
<xs:group name="nomedegrupo">
...
</xs:group>
```

Teremos que definir dentro de declaración `group` un elemento tipo `all`, `choice` ou `sequence`. O seguinte exemplo define un grupo chamado "grupopersoas", cun grupo de elementos que deben aparecer na orde exacta:

```
<xs:group name="grupopersoas">
  <xs:sequence>
    <xs:element name="nome" type="xs:string"/>
    <xs:element name="apelidos" type="xs:string"/>
    <xs:element name="nacemento" type="xs:date"/>
  </xs:sequence>
</xs:group>
```

Cando xa temos definido un grupo podemos referencia-lo noutra definición, por exemplo:

```
<xs:group name="grupopersoas">
  <xs:sequence>
    <xs:element name="nome" type="xs:string"/>
    <xs:element name="apelidos" type="xs:string"/>
    <xs:element name="nacemento" type="xs:date"/>
  </xs:sequence>
</xs:group>

<xs:element name="persoa" type="infopersoa"/>

<xs:complexType name="infopersoa">
  <xs:sequence>
    <xs:group ref="grupopersoas"/>
    <xs:element name="pais" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

**Grupos de Atributos:** Os "Attribute groups" defínense coa declaración:

```
<xs:attributeGroup name="nomegrupo">
...
</xs:attributeGroup>
```

O seguinte exemplo define un attribute group chamado "grupoatributospersoas":

```
<xs:attributeGroup name="grupoatributospersoas">
  <xs:attribute name="nome" type="xs:string"/>
  <xs:attribute name="apelidos" type="xs:string"/>
  <xs:attribute name="nacemento" type="xs:date"/>
</xs:attributeGroup>
```

Unha vez que xa temos definido o grupo de atributos, podemos referencialo noutra definición como:

```
<xs:attributeGroup name="grupoatributospersoas">
  <xs:attribute name="nome" type="xs:string"/>
  <xs:attribute name="apelidos" type="xs:string"/>
  <xs:attribute name="nacemento" type="xs:date"/>
</xs:attributeGroup>

<xs:element name="persoa">
  <xs:complexType>
    <xs:attributeGroup ref="grupoatributospersoas"/>
  </xs:complexType>
</xs:element>
```

## Elemento <any> XSD

O elemento <any> permite estender o documento XML con elementos non especificados no Schema.

O seguinte exemplo é un fragmento dun esquema chamado "familia.xsd" que amosa a declaración para o elemento "persoa". Empregando o elemento <any> podemos estender (despois dos <apelidos>) o contido de "persoa" con calquera elemento:

```
<xs:element name="persoa">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="nome" type="xs:string"/>
      <xs:element name="apelidos" type="xs:string"/>
      <xs:any minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Agora queremos estender o elemento "persoa" un elemento "fillos". Neste caso poderemos facelo aínda que o autor do schema nunca declarara un elemento "fillos".

Mira o exemplo "fillos.xsd":

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.iessanclemente.net"
xmlns="http://www.iessanclemente.net"
elementFormDefault="qualified">

  <xs:element name="fillos">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="nomefillo" type="xs:string"
          maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

O seguinte arquivo "familia.xml" emprega compoñentes dos dous diferentes esquemas; "familia.xsd" e "fillos.xsd":

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<persoas xmlns="http://www.microsoft.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:SchemaLocation="http://www.microsoft.com familia.xsd
http://www.iessanclemente.net fillos.xsd">

  <persoa>
    <nome> Patricia</nome>
    <apelidos>Conde</apelidos>
    <fillos>
      <nomefillo>Lucia</nomefillo>
    </fillos>
  </persoa>

  <persoa>
    <nome>Irene</nome>
    <apelidos>Sainz</apelidos>
  </persoa>

</persoas>
```

O arquivo xml anterior é válido por que o esquema "familia.xsd" permítenos estender o elemento "persoa" con elementos opcionais despois do elemento "apelidos".

Os elementos <any> e <anyAttribute> empréganse para facer EXTENSIBLES os documentos. Permiten conter ós documentos elementos adicionais que non están declarados no Schema XML principal.



## Elemento <anyAttribute>

O elemento <anyAttribute> permítenos estender o documento XML con atributos non especificados no Schema.

O seguinte exemplo é un fragmento dun Schema XML chamado "familia.xsd". Avisa a declaración do elemento <persoa>. Empregando o elemento <anyAttribute> podemos engadir calquera número de atributos ó elemento "persoa":

```
<xs:element name="persoa">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="nome" type="xs:string"/>
      <xs:element name="apelidos" type="xs:string"/>
    </xs:sequence>
    <xs:anyAttribute/>
  </xs:complexType>
</xs:element>
```

Agora queremos estender o elemento "persoa" cun atributo "xenero". Mira o código do schema "atributos.xsd":

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.iessanclemente.net"
xmlns="http://www.iessanclemente.net"
elementFormDefault="qualified">

  <xs:attribute name="xenero">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:pattern value="home|muller"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>

</xs:schema>
```

O seguinte arquivo XML (chamado "familiares.xsd") emprega compoñentes dos dous diferentes schemas; "familia.xsd" e "atributos.xsd":

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<persoas xmlns="http://www.microsoft.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:SchemaLocation="http://www.microsoft.com familia.xsd
http://www.iessanclemente.net atributos.xsd">

  <persoa xenero="muller">
    <nome>Raquel</nome>
    <apelidos>López</apelidos>
  </persoa>

  <persoa xenero="home">
    <nome>Martín</nome>
    <apelidos>García</apelidos>
  </persoa>

</persoas>
```

O arquivo anterior é válido por que emprega o schema "familia.xsd" que nos permite engadir un atributo ó elemento "persoa".

Os elementos <any> e <anyAttribute> empréganse para facer EXTENSIBLES os documentos. Permiten conter ós documentos elementos adicionais que non están declarados no Schema XML principal.

## Elemento Substitution XSD

Nos schemas XML podemos substituír un elemento por outro.

Por exemplo digamos que temos dous países diferentes: Francia e Italia. Poderíamos ter a posibilidade de deixar que o usuario escolla cando quere empregar nomes de elementos pertencentes a Francia ou a Italia no documento XML.

Para solucionar este problema, poderíamos definir un substitutionGroup no schema XML. Primeiro teríamos que declarar o elemento cabeceira e logo declarar outros elementos que poderían ser substituídos polo elemento cabeceira.

```
<xs:element name="nome" type="xs:string"/>
<xs:element name="nomen" substitutionGroup="nome"/>

<xs:element name="nome" type="xs:string"/>
<xs:element name="nomen" substitutionGroup="nome"/>

<xs:complexType name="infocomprador">
  <xs:sequence>
    <xs:element ref="nome"/>
  </xs:sequence>
</xs:complexType>

<xs:element name="acheteur" type="infocomprador"/>
<xs:element name="acquirente" substitutionGroup="acheteur"/>
```

Un documento XML válido en base ó schema anterior podería ser:

```
<acheteur>
  <nome>Pedro Garcia</nome>
</acheteur>
```

ou ben:

```
<acquirente>
  <nomen>Pedro Garcia</nomen>
</acquirente>
```

## Elemento de substitución block

Para evitar que outros elementos substitúan a outro elemento especificado, empregaremos o atributo "block":

```
<xs:element name="nome" type="xs:string" block="substitution"/>
```

Mira o seguinte fragmento XML:

```
<xs:element name="nome" type="xs:string"/>
<xs:element name="nomen" substitutionGroup="nome"/>

<xs:complexType name="infocomprador">
  <xs:sequence>
    <xs:element ref="nome"/>
  </xs:sequence>
</xs:complexType>

<xs:element name="acheteur" type="infocomprador" block="substitution"/>
<xs:element name="acquirente" substitutionGroup="acheteur"/>
```

Un documento válido podería ser:

```
<acheteur>
  <nome>Pedro Garcia</nome>
</acheteur>
```

Pero o seguinte documento xa NON sería válido:

```
<acquirente>
  <nomen>Pedro Garcia</nomen>
</acquirente>
```