

# Tema 4

## XML Schemas (2)

# Contidos

## Artigos

XML:Tipos simples	1
XML:Tipos complexos	5
XML:Grupos de elementos e atributos	10
XML:Anotaci�es	11
XML:Claves, unicidade e valores nulos	11

## Referencias

Fonte dos artigos e contribuintes	13
-----------------------------------	----

## Licenzas de artigos

Licenza	14
---------	----

# XML:Tipos simples

## Tipos simples

Como xa comentamos anteriormente, permítenos definir novos tipos baseados nos tipos simples predefinidos, aplicándolles certas restricións. O seu valor sempre é de tipo texto e non poden conter marcas (<>) doutros elementos, nin atributos.

Existen tres mecanismos para crear tipos simples baseados en tipos simples predefinidos ou, a vez, noutros tipos simples: restricións, listas e unións.

## Restricións

Ao definir un tipo simple podemos especificar, non só o tipo de dato que se espera no contido dos elementos e os atributos, senón unha gran variedade de restricións que nos permiten ler con tranquilidade o documento sabendo que todas esas condicións foron previamente validadas.

As restricións sempre se aplican sobre un tipo existente (denominado tipo base), e permiten restrinxir o conxunto dos seus posibles valores. A súa sintaxe é a seguinte.

```
<xs:simpleType name="nomeDoTipoSimple">
  <xs:restriction base="nomeDoTipoBase">
    ...
  </xs:restriction>
</xs:simpleType>
```

As posibles restricións sobre un tipo base son:

- ... **length**: lonxitude da cadea ou número de elementos da lista
- ... **minLength**: lonxitude mínima
- ... **maxLength**: lonxitude máxima
- ... **pattern**: encaixar cunha expresión regular
- ... **enumeration**: os valores do tipo só poden ser os especificados
- ... **whiteSpace**: controla a normalización dos espazos brancos (tabs, etc.)
- ... **maxInclusive**: valor superior incluído
- ... **maxExclusive**: valor superior excluído
- ... **minInclusive**: valor inferior incluído
- ... **minExclusive**: valor inferior excluído
- ... **totalDigits**: máximo número de díxitos (nos tipos numéricos)
- ... **fractionDigits**: máximo número de díxitos decimais

Dependendo do tipo base poderíase aplicar un posible conxunto de restricións. Por exemplo, as restricións sobre un tipo *string* son: *length*, *minLength*, *maxLength*, *pattern*, *enumeration* e *whiteSpace*. As restricións sobre un tipo de datos *integer* son: *totalDigits*, *pattern*, *whiteSpace*, *enumeration*, *maxInclusive*, *maxExclusive*, *minInclusive* e *minExclusive*.

Exemplos de restricións serían as seguintes: Exemplo 1: <toggleDisplay>

```
<xs:simpleType name="tipoNota">
  <xs:restriction base="xs:integer">
    <xs:minInclusive value="1"/>
    <xs:maxInclusive value="10"/>
  </xs:restriction>
```

```
</xs:simpleType>
<xs:element name="nota1Ev" type="tipoNota" />
```

ou ben

```
<xs:element name="nota1Ev">
  <xs:simpleType>
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="1"/>
      <xs:maxInclusive value="10"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

</toggedisplay>

Exemplo 2: <toggedisplay>

```
<xs:element name="nivel">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="baixo"/>
      <xs:enumeration value="medio"/>
      <xs:enumeration value="alto"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

ou ben

```
<xs:element name="nivel" type="tipoDeNivel"/>
<xs:simpleType name="tipoDeNivel">
  <xs:restriction base="xs:string">
    <xs:enumeration value="baixo"/>
    <xs:enumeration value="medio"/>
    <xs:enumeration value="alto"/>
  </xs:restriction>
</xs:simpleType>
```

</toggedisplay>

Exemplo 3: <toggedisplay>

```
<xs:element name="matricula">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern
value="[0-9][0-9][0-9][0-9][A-Z][A-Z][A-Z]" />
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

</toggedisplay>

Exemplo 4: <toggledisplay>

```
<xs:element name="codCliente">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:length value="8"/><!-- exactamente 8
car cteres -->
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

</toggledisplay>

Exemplo 5: <toggledisplay>

```
<xs:element name="codCliente">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[a-zA-Z0-9]{8}"/> <!--
exactamente 8 car cteres cumprindo o patr n -->
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

</toggledisplay>

Exemplo 6: <toggledisplay>

```
<xs:element name="nomeDeUsuario">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:minLength value="6"/>
      <xs:maxLength value="15"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

</toggledisplay>

## Listas

Unha lista consiste nunha secuencia de valores dun certo tipo, separada por espazos. A s a sintaxe   a seguinte.

```
<xs:simpleType name="nomeDoTipoLista">
  <xs:list itemType="tipoDosValores"/>
</xs:simpleType>
```

Por exemplo: <toggledisplay>

```
<xs:simpleType name="tipoTemperaturas">
  <xs:list itemType="xs:integer"/>
</xs:simpleType>
<xs:element name="temperaturas" type="tipoTemperaturas"/>
```

permitir, a o seguinte contido no documento XML:

```
<temperaturas>9 13 15 21 17 12</temperaturas>
```

```
</toggledisplay>
```

As listas non sempre son un bo recurso debido a que se perden as vantaxes do formato XML. Adoita ser mellor idea incrementar o número de marcas do seguinte xeito.

```
<temperaturas>
  <temperatura>9</temperatura>
  <temperatura>13</temperatura>
  <temperatura>15</temperatura>
  <temperatura>21</temperatura>
  <temperatura>17</temperatura>
  <temperatura>12</temperatura>
</temperaturas>
```

## Unións

Unha unión *f* unha estrutura que nos permite unir nun único tipo valores con distintos tipos de datos. A súa sintaxe *f*.

```
<xs:simpleType name="nomeTipoUnión">
  <xs:union>
    <xs:simpleType> ... </xs:simpleType>
    <xs:simpleType> ... </xs:simpleType>
    ...
  </xs:union>
</xs:simpleType>
```

ou ben

```
<xs:simpleType name="nomeTipoUnión">
  <xs:union memberTypes="tipo1 tipo2 ..." />
</xs:simpleType>
```

Por exemplo: <toggledisplay>

```
<xs:simpleType name="{tipoNota}">
  <xs:union>
    <xs:simpleType>
      <xs:restriction base="{xs:integer}">
        <xs:maxLength value="1"/>
        <xs:minLength value="10"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType>
      <xs:restriction base="{xs:string}">
        <xs:enumeration value="{Non presentado}" />
      <xs:restriction>
    </xs:simpleType>
  </xs:union>
</xs:simpleType>
```

```
</toggledisplay>
```

# XML:Tipos complexos

## Tipos complexos

Son aqueles elementos XML que, ademais de poder ter atributos, o seu contido pode ter outros elementos e/ou contido textual.

Se definen as,:

```
<xs:complexType name="nomeDoTipoCompleto">
    ...
</xs:complexType>
```

## Declaración dos atributos dun elemento

No caso de que un elemento teña atributos, a declaración destes realizarase no interior da definición do tipo complexo.

Debido a que os atributos se poden igualarse a valores de tipo simple, a súa definición poderase facer dos seguintes xeitos.

```
<xs:attribute name="nomeAtributo" type="tipoDoAtributo"
[default="valorPorDefecto" | fixed="valorFijo"] [use="required"] />
```

ou ben

```
<xs:attribute name="nomeAtributo" [default="valorPorDefecto" |
fixed="valorFijo"] [use="required"] >
    <xs:simpleType>
        <xs:restriction type="tipoSimple">
            ...
        </xs:restriction>
    </xs:simpleType>
</xs:attribute>
```

O atributo *use="required"* obriga a introducir o atributo co seu elemento, pois por defecto son opcionais (*use="optional"*).

Exemplo 1: <toggleDisplay>

```
<xs:complexType name="tipoPexina">
    ...
    <xs:attribute name="numDeCopias" type="xs:positi vel nteger"
use="required" />
</xs:complexType>
<xs:element name="pexina" type="tipoPexina" />
```

ou ben

```
<xs:element name="pexina">
    <xs:complexType>
        ...
        <xs:attribute name="numDeCopias"
type="xs:positi vel nteger" use="required" />
```

```

    </xs: complexType>
<xs: element>

```

que permitir, a um documento XML o seguinte elemento:

```
<p€xi na numDeCopias="3"> . . . </p€xi na>
```

```
</toggleDisplay>
```

Exemplo 2: <toggleDisplay>

```

<xs: element name="cliente">
  <xs: complexType>
    <xs: attribute name="idCliente" type="xs: integer"/>
  </xs: complexType>
</xs: element>

```

permitir, a um documento XML o seguinte elemento:

```
<cliente idCliente="255"/>
```

```
</toggleDisplay>
```

## Elementos con sub-elementos

O contido dun elemento pode conter • s, a vez outros elementos. Para especificar as s, as caracter, sticas podemos utilizar algunha das seguintes estruturas.

### Secuencia ordenada: a estrutura <xs:sequence>

Todos os elementos que conteña a estrutura ter•n que ir no documento na orde indicada.

Exemplo: <toggleDisplay>

```

<xs: element name="pessoa">
  <xs: complexType>
    <xs: sequence>
      <xs: element name="nome" type="xs: string"/>
      <xs: element name="apelido1" type="xs: string"/>
      <xs: element name="apelido2" type="xs: string"/>
      <xs: element name="endereçoElectr•nico"
type="xs: string" minOccurs="0" maxOccurs="unbounded"/>
    </xs: sequence>
  </xs: complexType>
</xs: element>

```

```
</toggleDisplay>
```



**Secuencia desordenada: a estrutura <xs:all>**

Similar a anterior pero onde todos os elementos que conteña a estrutura teñen que aparecer, aínda que non importa a súa orde. A diferenza da estrutura anterior, os elementos deben aparecer como máximo unha vez, é dicir, o atributo *minOccurs* dos elementos só pode estar a 0 ou a 1 e *maxOccurs* só pode estar a 1 (si se especifica).

**Elección: a estrutura <xs:choice>**

Elíxese un só dos elementos indicados dentro da estrutura.

Exemplo: <toggleDisplay>

```
<xs:element name="persoa">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="nome" type="xs:string"/>
      <xs:element name="apelido1" type="xs:string"/>
      <xs:element name="apelido2" type="xs:string"/>
      <xs:choice>
        <element name="NIF" type="xs:string"/>
        <element name="NumTarxetaResidencia"
type="xs:string"/>
      </xs:choice>
    </xs:sequence>
  </xs:complexType>
</xs:element>

</toggleDisplay>
```

**Calquera elemento: a estrutura <xs:any>**

A estrutura <xs:any> permite que o esquema acepte nese lugar calquera elemento e o seu contido. Ten atributos para configurar por exemplo, o aceptar unicamente elementos de certos espazos de nomes, ou ben que a validación sexa máis ou menos rixida en devanditos contidos.

**Elementos con sub-elementos e texto**

Son aqueles que, no seu contido, permiten mesturar texto libre xunto con outros elementos. Decláranse especificando o atributo *mixed="true"* da etiqueta <xs:complexType>.

Exemplo: <toggleDisplay> Mostramos a continuación parte dun documento XML e o seu correspondente esquema:

```
<mensaxeErro>
  A páxina < sitioWeb>www.larompiente.com</ sitioWeb>
  xerou o erro n,mero <numErro>404</numErro>:
  <descErro>Páxina ' estadoDoMar.html ' non
atopada</descErro>
  És <hora>23: 12: 45</hora>
</mensaxeErro>

<xs:element name="mensaxeErro">
  <xs:complexType mixed="true">
    <xs:sequence>
      <xs:element name="sitioWeb" type="xs:string"/>
```

```

        <xs:element name="numErro" type="xs:integer"/>
        <xs:element name="descErro" type="xs:string"/>
        <xs:element name="hora" type="xs:time"/>
    </xs:sequence>
</xs:complexType>
</xs:element>

</toggleDisplay>

```

## Contido simple e contido complexo

Dependendo do contido dun elemento de tipo complexo, podemos describilos usando dous tipos de contidos: simple e complexo. Ademais nos serve para crear novos tipos complexos derivando de tipos existentes, facilitando a reusabilidade.

... contido simple (<xs:simpleContent>): texto e atributos, sen elementos.

... contido complexo (<xs:complexContent [mixed=true]>): texto, elementos e atributos

Por defecto, se non se especifica o tipo do contido nun tipo complexo (*complexType*), as„mese contido complexo (*complexContent*). ‡ o caso do exemplo do ep,grafe anterior.

No interior do contido (simple ou complexo) indicaremos se queremos estender ou restrinxir o tipo base. Para iso utilizaremos as seguintes etiquetas:

... <xs:restriction base="tipoBaseRestrxido">

... <xs:extension base="tipoBaseExtendido">

Exemplo 1: <toggleDisplay>

```
<di stanci a uni dade="cm">75</di stanci a>
```

necesitar,a un esquema como o seguinte:

```

<xs:element name="di stanci a" type="di stanci aConUni dades"/>
    <xs:complexType name="di stanci aConUni dades">
        <xs:simpleContent>
            <xs:extension base="xs:integer">
                <xs:attribute name="uni dade"
type="xs:string" />
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>

```

</toggleDisplay>

Exemplo 2: <toggleDisplay>

```

<xs:simpleType name="pei xes">
    <xs:restriction base="xs:string">
        <xs:enumeration value="robal i za"/>
        <xs:enumeration value="sargo"/>
        <xs:enumeration value="troi ta"/>
    </xs:restriction>
</xs:simpleType>

<xs:complexType name="pei xeCapturado">

```

```

    <xs:simpleContent>
      <xs:extension base="peixes">
        <xs:attribute name="peso" type="xs:decimal" />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:complexType name="capturasDeRfo">
    <xs:simpleContent>
      <xs:restriction base="peixeCapturado">
        <xs:enumeration value="troita"/>
      </xs:restriction>
    </xs:simpleContent>
  </xs:complexType>

```

</toggedisplay>

Exemplo 3: <toggedisplay>

```

  <xs:complexType name="endereco">
    <xs:sequence>
      <xs:element name="r,a" type="xs:string"/>
      <xs:element name="poboaci•n" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>

```

```

  <xs:complexType name="enderecoEspa„a">
    <xs:complexContent>
      <xs:extension base="endereco">
        <xs:sequence>
          <xs:element name="c•di goPostal "
type="xs:string"/>
          <xs:element name="provinci a"
type="xs:string"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

```

</toggedisplay>

# XML: Grupos de elementos e atributos

## Grupos de elementos e atributos

Permitem agrupar elementos com `<xs:group>` ou atributos com `<xs:attributeGroup>`, puidendoos utilizar posteriormente para aplicalos a outros elementos.

Exemplo 1: `<toggledisplay>`

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

<xs:group name="endereco">
  <xs:sequence>
    <xs:element name="rua" type="xs:string"/>
    <xs:element name="numero" type="xs:integer"/>
    <xs:element name="bairro" type="xs:string"/>
  </xs:sequence>
</xs:group>

<xs:element name="pessoa" type="tipoPessoa"/>
<xs:complexType name="tipoPessoa">
  <xs:all>
    <xs:group ref="nomeCompleto"/>
    <xs:group ref="endereco"/>
    <xs:element name="provincia" type="xs:string"/>
  </xs:all>
</xs:complexType>

</toggledisplay>
```

Exemplo 2: `<toggledisplay>`

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

<xs:element name="pessoa">
  <xs:complexType>
    <xs:attributeGroup ref="atributosPessoa"/>
  </xs:complexType>
</xs:element>

</toggledisplay>
```

# XML:Anotaci ns

## Anotaci ns

Permiten facer comentarios estruturados no esquema. Creanse coa etiqueta `<xs:annotation>` e permiten d as estruturas no seu interior: `<documentation>` e `<appinfo>`. A primeira serve para introducir comentarios que lean as persoas, e a segunda cont n informaci n acesible  s aplicaci ns.

  moi normal que o texto introducido conte a a s a vez etiquetas para estruturar o texto: por exemplo, que te a formato XHTML para poder ser visualizado nun navegador e poder introducir formato, enlaces, etc.

As anotaci ns se poden incluír no contido de pr cticamente todas as estruturas vistas dos esquemas.

Exemplo: `<toggledisplay>`

```
<xs:annotation>
  <xs:appinfo>
    <app num= 5 />
  </xs:appinfo>
  <xs:documentation xml:lang="es">
    Este elemento representa unha aplicaci n da empresa.
    Os atributos deste elemento configuran...
  </xs:documentation>
</xs:annotation>
```

`</toggledisplay>`

# XML:Claves, unicidade e valores nulos

## Claves, unicidade e valores nulos

Igual que os DTDs permit n a trav s dos atributos ID e IDREF, establecer a unicidade dun elemento e as s as referencias, os XSD te en mecanismos m is potentes para proporcionar este tipo de control dos elementos e atributos do documento.

Para establecer estas regras no esquema, utilizaremos as etiquetas `<xs:unique>`, `<xs:key>` e `<xs:keyref>`:

... `<xs:key name= nome >`: permite definir una clave para o elemento, que debe existir e ser  nica para a combinaci n desexada de atributos e elementos.

... `<xs:unique name= nome >`: permite definir que a combinaci n especificada de valores sexa  nica, pero pode non existir.

... `<xs:keyref name= nome  refer= referenciaDaClave >`: permite establecer referencias entre elementos.

Todas estas construcci ns aceptan como contido unha  nica etiqueta `<xs:selector>` e unha ou m is etiquetas `<xs:field>`.

... A etiqueta selector cont n unha expresi n XPath que especifica o conxunto de elementos nos que os valores especificados pola etiqueta field deben ser  nicos.

... As etiquetas field conte en unha expresi n XPath que especifica os valores de atributos ou elementos que deben ser  nicos para os elementos especificados por selector.

Exemplo: `<toggledisplay>`

```
<xs:complexType name="{cli entes}">
  <xs:sequence>
    <xs:element name="{cli ente}" type="{ti poCli ente}" />
  </xs:sequence>
  <xs:key name="{NI F}">
    <xs:selector xpath="{cli i : cli ente}" />
    <xs:field xpath="{cli i : nome}" />
    <xs:field xpath="{cli i : apel i dos}" />
  </xs:key>
</xs:complexType>

<!-- e, noutro lugar do documento, unha referencia a un cliente:
-->

<xs:keyref name="{cli enteDeZoaf}" refer="{NI F}">
  <xs:selector xpath="{cli i : cli enteDeZoaf}" />
  <xs:field xpath="{cli i : ni ff}" />
</xs:keyref>

</toggledisplay>
```

# Fonte dos artigos e contribuintes

**XML:Tipos simples** `Fonte: <http://www.plategaxml.es/index.php?oldid=684> `Contribuintes: David, Victor

**XML:Tipos complexos** `Fonte: <http://www.plategaxml.es/index.php?oldid=718> `Contribuintes: David, Victor

**XML:Grupos de elementos e atributos** `Fonte: <http://www.plategaxml.es/index.php?oldid=688> `Contribuintes: David, Victor

**XML:Anotações** `Fonte: <http://www.plategaxml.es/index.php?oldid=634> `Contribuintes: Victor

**XML:Claves, unicidade e valores nulos** `Fonte: <http://www.plategaxml.es/index.php?oldid=635> `Contribuintes: Victor

# Licenza

---

Attribution-Noncommercial-Share Alike 3.0 Unported  
<http://creativecommons.org/licenses/by-nc-sa/3.0/>