UT 3: XML Schema

ÍNDICE

ESPACIOS DE NOMBRE

Actividad 1: Busca información sobre los espacios de nombre de XML.

Procura contestar al menos estas preguntas:

- **¿Por qué se usan?**
- **¿Cómo se usan?**
- **2 ejemplos de documentos con espacios de nombre**

ESPACIOS DE NOMBRE

- Los nombres de XML son definidos por el usuario → Conflictos al mezclar archivos de diferentes aplicaciones XML
    - Ejemplo: Etiqueta <table> de tabla de datos y <table> de mesa (en inglés)

```
<table>
  <tr>
    <td>Apples</td>
    <td>Bananas</td>
  </tr>
</table>
```

```
<table>
  <name>African Coffee Table</name>
  <width>80</width>
  <length>120</length>
</table>
```

- **Espacios de nombre** = Herramienta de XML para evitar los conflictos de nombres

ESPACIOS DE NOMBRE

- Para evitar estos conflictos, se usan prefijos:
  - **h**: HTML
  - **f**: furniture (muebles en inglés)

```
<h:table>
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>

<f:table>
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
```

ESPACIOS DE NOMBRE

- Para usar prefijos debemos definir un espacio de nombre para cada prefijo
- El espacio de nombre se puede definir con el atributo **xmlns** (**XML N**ame**S**pace) en la etiqueta de inicio de un elemento
- El espacio de nombre y el prefijo es válido para todos los elementos hijos
    - Se suelen definir en el elemento raíz

```
<h:table xmlns:h="http://www.w3.org/TR/html4/">
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>
```

```
<f:table xmlns:f="https://www.w3schools.com/furniture">
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
```

ESPACIOS DE NOMBRE

- La URL que se usa para definir el espacio de nombre se utiliza para darle un **nombre único** y **no se usa** → No hace falta que exista pero no se puede repetir
- **Espacios de nombre por defecto**: Definimos uno por defecto y nos ahorramos usa el prefijo para cada etiqueta hijo.
    - La notación es similar pero no se especifica prefijo

```
<h:table xmlns:h="http://www.w3.org/TR/html4/">
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>
```

```
<table xmlns="http://www.w3.org/TR/html4/">
  <tr>
    <td>Apples</td>
    <td>Bananas</td>
  </tr>
</table>
```

Espacio de nombre por defecto

ÍNDICE

INTRODUCCIÓN

- **XML Schema = XSD** (**X**ML **S**chema **D**efinition). Define la estructura del documento XML

¿Por qué usar XSD?

- Gran cantidad de formatos estandarizados. Muchos de ellos se definen con Schemas
- Alternativa basada en XML y más potente que los DTD

XSD tiene tipos de datos

- Más fácil describir el posible contenido del documento → restricciones, patrones…
- Más fácil comprobar si son válidos

INTRODUCCIÓN

Elemento <schema>

- Es la raíz de todos los archivos XSD
- Contiene atributos, normalmente estos:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="https://www.w3schools.com"
xmlns="https://www.w3schools.com"
elementFormDefault="qualified">
...
...
</xs:schema>
```

Especifica de donde vienen los elementos y tipos de datos usados y que usan el prefijo **xs**

Se usará siempre

INTRODUCCIÓN

Elemento <schema>

- Es la raíz de todos los archivos XSD
- Contiene atributos, normalmente estos:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="https://www.w3schools.com"
xmlns="https://www.w3schools.com"
elementFormDefault="qualified">
...
...
</xs:schema>
```

Especifica de donde tienen que venir los elementos que vamos a definir en el archivo

Su valor dependerá del archivo

INTRODUCCIÓN

Elemento <schema>

- Es la raíz de todos los archivos XSD
- Contiene atributos, normalmente estos:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="https://www.w3schools.com"
xmlns="https://www.w3schools.com"
elementFormDefault="qualified">
...
...
</xs:schema>
```

Indica el espacio de nombre por defecto

INTRODUCCIÓN

Enlazando un XML con un Schema

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

<note xmlns="https://www.w3schools.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="https://www.w3schools.com note.xsd">

<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

Espacio de nombre por defecto. Si no ponemos el prefijo, pertenecerán a este

INTRODUCCIÓN

Enlazando un XML con un Schema

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

<note xmlns="https://www.w3schools.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="https://www.w3schools.com note.xsd">

<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

Especifica de donde vienen los elementos y tipos de datos usados y que usan el prefijo **xsi**

Se usará siempre

INTRODUCCIÓN

Enlazando un XML con un Schema

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

<note xmlns="https://www.w3schools.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="https://www.w3schools.com note.xsd">

<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

Especifica el espacio de nombres que vamos a validar + archivo usado

INTRODUCCIÓN

Crea un archivo XML y un Schema en NetBeans

- **¿Qué pasa si no usas “targetNamespace” en el .xsd?**

INTRODUCCIÓN

- Dependiendo de dónde coloquemos la definición de los elementos del esquema, varía su ámbito de aplicación:
    - **Ámbito global**: se colocan dentro de la etiqueta raíz schema y no están dentro de ninguna otra. Estos elementos se pueden utilizar en cualquier parte del esquema.
    - **Ámbito local**: están definidos dentro de otros elementos. Se pueden utilizar sólo dentro del elemento en el que están definidos y no en todo el documento

INTRODUCCIÓN

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:doc="http://www. educacion.navarra.es "
           targetNamespace="http://www. educacion.navarra.es">

  <xs:element…> <!--Definición global -->
      <xs:simpleType…> <!--Definición local -->
         …
      </xs:simpleType>
         ……
   </xs:element >

   <xs:simpleType…> <!--Definición global  -->
      …
   </xs:simpleType>
</xs:schema>
```

ÍNDICE

ELEMENTOS SIMPLES Y COMPUESTOS

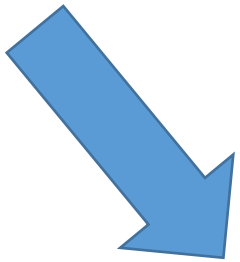
Elementos simples

- Son elementos que solo pueden contener texto. No pueden contener otros elementos ni atributos
- Solo texto = datos en alguno de los tipos de XSD con restricciones o patrones
- Sintaxis `<xs:element name="xxx" type="yyy"/>`
- Tipos básicos: xs:string, xs:decimal, xs:integer, xs:boolean, xs:date, xs:time
- Valores por defecto y fixed: atributos *default* y *fixed*

ELEMENTOS SIMPLES Y COMPUESTOS

Elementos simples `<xs:element name="xxx" type="yyy"/>`

```
<lastname>Refsnes</lastname>
<age>36</age>
<dateborn>1970-03-27</dateborn>
```



```
<xs:element name="lastname" type="xs:string"/>
<xs:element name="age" type="xs:integer"/>
<xs:element name="dateborn" type="xs:date"/>
```

ELEMENTOS SIMPLES Y COMPUESTOS

Elementos compuestos

- Es un elemento que contiene otros elementos y/o atributos
- Hay 4 tipos:
    1. Elementos que contienen otros elementos
    2. Elementos que contienen solo texto
    3. Elementos que contienen elementos y texto (mixtos)
    4. Elementos vacíos
- Cada uno de estos tipos puede tener atributos

ELEMENTOS SIMPLES Y COMPUESTOS

Elementos compuestos: elementos que contienen otros elementos

- **Opción 1**: definición local

```
<employee>
  <firstname>John</firstname>
  <lastname>Smith</lastname>
</employee>
```

Los elementos complejos se definen como un tipo complejo

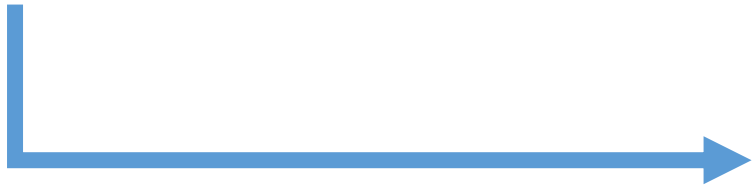
```
<xs:element name="employee">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

ELEMENTOS SIMPLES Y COMPUESTOS

Elementos compuestos: <u>elementos que contienen otros elementos</u>

- **Opción 1**: definición local

```
<employee>
  <firstname>John</firstname>
  <lastname>Smith</lastname>
</employee>
```



Con xs:sequence marcamos los elementos que componen a otro

```
<xs:element name="employee">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

ELEMENTOS SIMPLES Y COMPUESTOS

Elementos compuestos: <u>elementos que contienen otros elementos</u>

- **Opción 1**: definición local

```
<employee>
  <firstname>John</firstname>
  <lastname>Smith</lastname>
</employee>
```

Definición de los elementos internos de <employee>

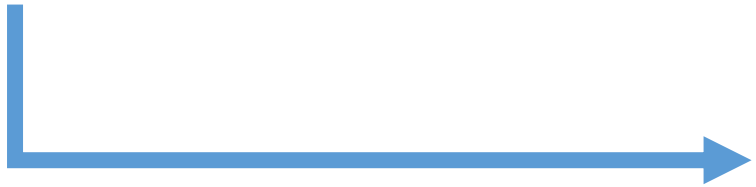
```
<xs:element name="employee">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

ELEMENTOS SIMPLES Y COMPUESTOS

Elementos compuestos: <u>elementos que contienen otros elementos</u>

- **Opción 2**: definición global

```
<employee>
  <firstname>John</firstname>
  <lastname>Smith</lastname>
</employee>
```



```
<xs:element name="employee" type="personinfo"/>

<xs:complexType name="personinfo">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

ELEMENTOS SIMPLES Y COMPUESTOS

Elementos compuestos: elementos que contienen otros elementos

- **Opción 2**: definición global:
  - Reutilizar tipo

```
<xs:element name="employee" type="personinfo"/>
<xs:element name="student" type="personinfo"/>
<xs:element name="member" type="personinfo"/>
```

Definición (**global**) del tipo de datos
Le hemos dado el nombre personinfo

```
<xs:complexType name="personinfo">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

ELEMENTOS SIMPLES Y COMPUESTOS

Elementos compuestos: elementos que contienen otros elementos

- **Opción 2**: definición global:
  - Reutilizar tipo

```
<xs:element name="employee" type="personinfo"/>
<xs:element name="student" type="personinfo"/>
<xs:element name="member" type="personinfo"/>
```

Elementos del tipo definido

```
<xs:complexType name="personinfo">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

ELEMENTOS SIMPLES Y COMPUESTOS

Elementos compuestos

- **Opción 2**: definición global:
  - Reutilizar tipo
  - Usar como base

Nuevo tipo fullpersoninfo

```
<xs:element name="employee" type="fullpersoninfo"/>

<xs:complexType name="personinfo">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

```
<xs:complexType name="fullpersoninfo">
  <xs:complexContent>
    <xs:extension base="personinfo">
      <xs:sequence>
        <xs:element name="address" type="xs:string"/>
        <xs:element name="city" type="xs:string"/>
        <xs:element name="country" type="xs:string"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

ELEMENTOS SIMPLES Y COMPUESTOS

Elementos compuestos

- **Opción 2**: definición global:
  - Reutilizar tipo
  - Usar como base

Toma como base personinfo…

```
<xs:element name="employee" type="fullpersoninfo"/>

<xs:complexType name="personinfo">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="fullpersoninfo">
  <xs:complexContent>
    <xs:extension base="personinfo">
      <xs:sequence>
        <xs:element name="address" type="xs:string"/>
        <xs:element name="city" type="xs:string"/>
        <xs:element name="country" type="xs:string"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

ELEMENTOS SIMPLES Y COMPUESTOS

Elementos compuestos

- **Opción 2**: definición global:
  - Reutilizar tipo
  - Usar como base

…y AÑADE estos elementos

```
<xs:element name="employee" type="fullpersoninfo"/>

<xs:complexType name="personinfo">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="fullpersoninfo">
  <xs:complexContent>
    <xs:extension base="personinfo">
      <xs:sequence>
        <xs:element name="address" type="xs:string"/>
        <xs:element name="city" type="xs:string"/>
        <xs:element name="country" type="xs:string"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

ELEMENTOS SIMPLES Y COMPUESTOS

Elementos compuestos: texto (+ atributos)

- Se declaran con un tipo de datos complejo **(<xs:complexType>)** con contenido simple **(<xs:simpleContent>)** ya que es texto y atributos
- Se toma un tipo de datos y se extiende (con **<xs:extension>**) se le añaden los atributos, convirtiendo al tipo en complejo
- Ejemplo: El elemento <textarea> de HTML acepta texto y atributos (name, cols, rows,…)

ELEMENTOS SIMPLES Y COMPUESTOS

Elementos compuestos: texto (+ atributos)

- Ejemplo: El elemento <textarea> de HTML acepta texto y atributos (name, cols, rows,…)

```
<textarea name="comentarios"  cols="10" rows="5">
  Introduzca aquí sus comentarios
</textarea>
```

```
<xs:element name="textarea">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs.string">
        <xs:attribute name="name" type="xs:string"/>
        <xs:attribute name="cols" type="xs:positiveInteger"/>
        <xs:attribute name="rows" type="xs:positiveInteger"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

- <xs:simpleContent> especifica que el elemento complejo no tiene elementos descendientes

ELEMENTOS SIMPLES Y COMPUESTOS

Elementos compuestos: texto (+ atributos)

```
<zapatossize country="france">35</zapatossize>
```

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

Definición LOCAL

```
<xs:element name="zapatosize" type="zapatostype"/>

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

Definición GLOBAL

ELEMENTOS SIMPLES Y COMPUESTOS

Elementos compuestos: <u>contenido mixto</u>

- Contienen texto y otros elementos (y puede que atributos)
- Se usa el atributo **mixed="true"** en <xs:complexType>
- El valor por defecto de mixed es false, por lo que no hace falta ponerlo en el resto de elementos

ELEMENTOS SIMPLES Y COMPUESTOS

Elementos compuestos: <u>contenido mixto</u>

```
<letter>
  Dear Mr.<name>John Smith</name>.
  Your order <orderid>1032</orderid>
  will be shipped on <shipdate>2001-07-13</shipdate>.
</letter>
```

```
<xs:element name="letter">
  <xs:complexType mixed="true">
    <xs:sequence>
      <xs:element name="name" type="xs:string"/>
      <xs:element name="orderid" type="xs:positiveInteger"/>
      <xs:element name="shipdate" type="xs:date"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Definición LOCAL

```
<xs:element name="letter" type="lettertype"/>

<xs:complexType name="lettertype" mixed="true">
  <xs:sequence>
    <xs:element name="name" type="xs:string"/>
    <xs:element name="orderid" type="xs:positiveInteger"/>
    <xs:element name="shipdate" type="xs:date"/>
  </xs:sequence>
</xs:complexType>
```

Definición GLOBAL

ELEMENTOS SIMPLES Y COMPUESTOS

Elementos compuestos: elementos vacíos

- No puede tener contenido, solo atributos
- Definimos un tipo que acepte elementos dentro, pero no definimos estos elementos

ELEMENTOS SIMPLES Y COMPUESTOS

Elementos compuestos: <u>elementos vacíos</u>

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

```
<xs:element name="product">
  <xs:complexType>
    <xs:attribute name="prodid" type="xs:integer"/>
  </xs:complexType>
</xs:element>
```

Definición LOCAL

```
<xs:element name="product" type="prodtype"/>

<xs:complexType name="prodtype">
  <xs:attribute name="prodid" type="xs:integer"/>
</xs:complexType>
```

Definición GLOBAL

ÍNDICE

ATRIBUTOS

Atributos

- Solo aparecen en los elementos compuestos
- Se definen con un tipo simple `<xs:attribute name="xxx" type="yyy"/>`
- Por defecto, son opcionales → Atributo **use** para obligatorio: **use="required"**
    - Para especificar que es opcional (aunque es el valor por defecto): **use="optional"**
- También admiten los atributos **fixed** y **default** para valores fijos y por defecto

ÍNDICE

RESTRICCIONES

Restricciones o facetas (facets)

- Sirven para determinar los valores aceptados para los elementos y atributos
- Se escriben dentro del tipo (<xs:SimpleType> o <xs:ComplexType>)
- Algunos ejemplos:
    - Longitud de un atributo (máxima, mínima o exacta)
    - Enumeración de posibles valores
    - Valores que encajen en un determinado patrón (expresiones regulares)
    - ……

Faceta	Descripción
`xs:maxInclusive`	El valor debe ser menor o igual que el indicado.
`xs:maxExclusive`	El valor debe ser menor que el indicado.
`xs:minExclusive`	El valor debe ser mayor que el indicado.
`xs:minInclusive`	El valor debe ser mayor o igual que el indicado.
`xs:enumeration`	Lista de valores admitidos.
`xs:pattern`	Patrón de caracteres admitidos.
`xs:length`	Longitud fija.
`xs:minLength`	Longitud mínima.
`xs:maxLength`	Longitud máxima.
`xs:totalDigits`	El número máximo de dígitos que puede tener un número.
`xs:fractionDigits`	El número máximo de decimales que puede tener un número.
`xs:whiteSpace`	Cómo se debe tratar a los posibles espacios en blanco, las tabulaciones, los saltos de línea y los retornos de carro que puedan aparecer.

RESTRICCIONES

Restricciones o facetas (facets) – min/max Inclusive/Exclusive

- 4 combinaciones: **minInclusive**, **minExclusive**, **maxInclusive**, **maxExclusive**
  - Inclusive: incluye el igual // Exclusive: no lo incluye

```
<xs:element name="mes">
   <xs:simpleType>
      <xs:restriction base="xs:integer">
         <xs:minInclusive value="1"/>
         <xs:maxInclusive value="12"/>
      </xs:restriction>
   </xs:simpleType>
</xs:element>
```

Definición de un elemento **mes**
Su valor es un entero entre 1 y 12
(ambos incluidos)

RESTRICCIONES

Restricciones o facetas (facets) – enumeration

- Sirve para expresar los únicos valores posibles del elemento

```
<xs:element name="color">
   <xs:simpleType>
      <xs:restriction base="xs:string">
         <xs:enumeration value="verde"/>
         <xs:enumeration value="amarillo"/>
         <xs:enumeration value="rojo"/>
      </xs:restriction>
   </xs:simpleType>
</xs:element>
```

Definición de un elemento **color**
Su contenido es de tipo string y solo puede vale verde, amarillo o rojo

RESTRICCIONES

Restricciones o facetas (facets) – pattern

- Sirve para limitar el contenido de un elemento a aquellos que tengan una determinada forma
- Ejemplos:
    - Usa solo letras minúsculas
    - Usa solo 3 letras, mayúsculas o minúsculas, seguidas de un número del 0 al 3
    - ….
- Expresiones regulares

RESTRICCIONES

Restricciones o facetas (facets) – pattern

- Los corchetes [ ] definen rangos o conjuntos → [A-Z] [a-z] [0-9]
- Símbolos *, +, | se usan igual que en DTD
- [0-9]{10} → el elemento [0-9] se repite 10 veces → 10 cifras del 0 al 9

```
<xs:element name="letra">
   <xs:simpleType>
      <xs:restriction base="xs:string">
         <xs:pattern value="[a-z]"/>
      </xs:restriction>
   </xs:simpleType>
</xs:element>
```

Definición de un elemento **letra**
Su contenido es una única letra minúscula

RESTRICCIONES

Restricciones o facetas (facets) – length, minLength, maxLength

- Sirven para restringir los valores de un string a aquellos de una determinada longitud (o fijar un mínimo o máximo a estos)

```
<xs:element name="clave">
   <xs:simpleType>
      <xs:restriction base="xs:string">
         <xs:length value="12"/>
      </xs:restriction>
   </xs:simpleType>
</xs:element>
```

Definición de un elemento **clave**
Su contenido es de tipo string y tiene una longitud de 12 caracteres

RESTRICCIONES

Restricciones o facetas (facets) – totalDigits, fractionDigits

- Parecidos a los length
- Se usan en números
- **totalDigits**: número máximo de cifras totales
- **fractionDigits**: número máximo de cifras decimales

RESTRICCIONES

Restricciones o facetas (facets) – whiteSpace

- Sirven para definir qué hacer con los espacios en blanco, saltos de línea y tabulaciones que haya dentro de un valor.
- Posibles valores:
    - **preserve**: mantiene los espacios en blanco (valor por defecto)
    - **replace**: sustituye todos los saltos de línea y tabulaciones por espacios en blanco
    - **collapse**: elimina los espacios al principio y final, elimina si hay varios (como en HTML)

ÍNDICE

EXTENSIONES

- Sirven para agregar características a un elemento simple o compuesto

- Ya las hemos visto 2 veces:

    1. **Para añadir atributos a un elemento con solo texto** → Dentro de un simpleContent

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

EXTENSIONES

- Sirven para agregar características a un elemento simple o compuesto
- Ya las hemos visto 2 veces. Nos basta con conocer esos dos usos:

1. **Para añadir atributos a un elemento con solo texto** → Dentro de un simpleContent
2. **Para añadir elementos a un tipo declarado como global** → Dentro de un complexContent

```
<xs:complexType name="fullpersoninfo">
  <xs:complexContent>
    <xs:extension base="personinfo">
      <xs:sequence>
        <xs:element name="address" type="xs:string"/>
        <xs:element name="city" type="xs:string"/>
        <xs:element name="country" type="xs:string"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

ÍNDICE

INDICADORES

Indicadores

- Establecen cómo se van a utilizar los elementos en un documento XML
- Corresponden a lo que en DTD era *****, **+**, **|** , etc
- Son 7, que se dividen en 3 tipos:
    - De orden: todo (**all**), secuencia (**sequence**) y elección (**choice**)
    - De ocurrencia: **maxOccurs** y **minOccurs**
    - De grupo: de elementos (**group**) y de atributos (**attributeGroup**)

INDICADORES

Indicadores de <u>orden</u>

- Secuencia (**sequence**) → Indica los elementos que van a aparecer y su orden
- **All** → Indica los elementos que van a aparecer pero no el orden (pueden hacerlo en cualquiera)
    - **Intentaremos evitar usarlo**
- Elección (**choice**) → Indica que solo se puede escribir uno de los elementos hijos

```
<xs:element name="person">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
   </xs:sequence>
  </xs:complexType>
</xs:element>
```

```
<xs:element name="person">
 <xs:complexType>
  <xs:choice>
   <xs:element name="employee" type="employee"/>
   <xs:element name="member" type="member"/>
  </xs:choice>
 </xs:complexType>
</xs:element>
```

INDICADORES

Indicadores de <u>ocurrencia</u>

- **maxOccurs** → Indica el número máximo de ocurrencias de un elemento
  - Usaremos **maxOccurs = "unbounded"** para indicar que se puede usar un número ilimitado de veces
- **minOccurs** → Indica el número mínimo de ocurrencias de un elemento
  - El valor por defecto de minOccurs es 1

```
<xs:element name="person">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="full_name" type="xs:string"/>
   <xs:element name="child_name" type="xs:string"  maxOccurs="10" minOccurs="0"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

INDICADORES

Indicadores de grupo

- Son dos: **<xs:group>** y **<xs:attributeGroup>**
- Ambos sirven para agrupar un conjunto de declaraciones relacionadas, pudiendo referirnos a ellas con el nombre asignado
    - **<xs:group>** agrupa elementos
    - **<xs:attributeGroup>** agrupa atributos
- Facilitan la escritura de archivos XSD complejos

INDICADORES

Indicadores de grupo

- **<xs:group>**

```
<xs:complexType name="datosDePersona">
  <xs:sequence>
    <xs:group ref="datosBasicos"/>
    <xs:element name="telefono" type="xs:string"/>
  </xs:sequence>
</xs:complexType>

<xs:group name="datosBasicos">
  <xs:sequence>
    <xs:element name="nombre" type="xs:string"/>
    <xs:element name="edad" type="xs:positiveInteger"/>
    <xs:element name="pais" type="xs:string"/>
  </xs:sequence>
</xs:group>
```

INDICADORES

Indicadores de grupo

- **< xs:attributeGroup >**

```
<xs:element name="personas">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="persona" maxOccurs="unbounded">
        <xs:complexType>
          <xs:attributeGroup ref="datosDePersona"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:attributeGroup name="datosDePersona">
  <xs:attribute name="nombre" type="xs:string"/>
  <xs:attribute name="edad" type="xs:positiveInteger"/>
  <xs:attribute name="pais" type="xs:string"/>
</xs:attributeGroup>
```