

ÍNDICE

1. INTRODUCCIÓN XSD	3
1.1 ¿QUÉ ES UN XSD (XML SCHEMA DEFINITION)?	3
1.2 COMPARACIÓN DTD-XSD	3
1.3 INSTANCIAS	4
2. VALIDACIÓN DE UN ESQUEMA	5
2.1 DOCUMENTO BIEN FORMADO	5
2.2 DOCUMENTO VÁLIDO	6
3. ASOCIACIÓN DE UN XML (INSTANCIA) A UN XSD (ESQUEMA).....	7
3.1 ESPACIO DE NOMBRES	7
3.2 XML SIN ESPACIO DE NOMBRES	9
3.3 XML CON ESPACIO DE NOMBRES	12
3.4 ELEMENTOS CALIFICADOS Y NO CALIFICADOS.....	14
ACTIVIDAD 12. VINCULAR XML A XSD	16
ACTIVIDAD 13. VINCULAR XML A XSD CON VARIOS ESPACIOS DE NOMBRES	16
ACTIVIDAD 14. VINCULAR XML A XSD CON VARIOS ESPACIOS DE NOMBRES Y SIN ESPACIOS DE NOMBRES	16
4. ELEMENTOS.....	17
4.1 ELEMENTOS SIMPLES	17
ACTIVIDAD 15. DEFINIR UN ELEMENTO SIMPLE	17
4.2 ELEMENTOS COMPLEJOS	18
▪ INDICADORES DE ORDEN.....	18
ACTIVIDAD 16. DEFINIR ELEMENTOS COMPLEJOS CON INDICADOR DE ORDEN	20
▪ INDICADORES DE OCURRENCIA (FRECUENCIA)	21
ACTIVIDAD 17. DEFINIR ELEMENTOS COMPLEJOS CON INDICADOR DE OCURRENCIA.....	21
▪ ELEMENTO <ANY>.....	22
ACTIVIDAD 18. ELEMENTO ANY	22
5. ATRIBUTOS.....	23
5.1 DECLARACIÓN DE ATRIBUTO	23
EJEMPLO COMPLETO: ACTIVIDAD 19. INSERTAR ATRIBUTOS.....	23
5.2 ELEMENTO EMPTY CON ATRIBUTOS.....	24
ACTIVIDAD 20. ELEMENTOS VACÍOS CON ATRIBUTOS	24
5.3 ATRIBUTO <ANYATTRIBUTE>	25
ACTIVIDAD 21. UTILIZACIÓN DE ANYATTRIBUTE	25
6. TIPOS DE DATOS	26
6.1 TIPOS PREDEFINIDOS	26

6.2	RESTRICCIONES EN LOS DATOS (FACETAS)	27
	ACTIVIDAD 22. RESTRICCIONES EN LOS TIPOS DE DATOS	28
6.3	TIPOS CONSTRUIDOS POR EL USUARIO	29
	ACTIVIDAD 23. TIPOS DE DATOS PERSONALIZADOS	30
6.4	EXPRESIONES REGULARES	31
	ACTIVIDAD 24. TIPOS DE DATOS CON PATTERN	32

Documentación:

https://www.w3schools.com/xml/schema_intro.asp

1. Introducción XSD

1.1 ¿Qué es un XSD (XML Schema Definition)?

Es una **gramática** que permite definir la estructura de documentos XML (elementos y atributos posibles, anidación, orden, valores permitidos, obligatoriedad...), permitiendo su validación.

Es una alternativa a DTD, de hecho actualmente es más utilizado, ya que presenta algunas ventajas, aunque por otro lado es menos legible para las personas.

Un esquema es un fichero **con extensión .xsd**, que es a su vez un documento XML, que establece la forma de crear documentos XML. **Cada esquema define un vocabulario o lenguaje distinto, que será siempre un lenguaje XML válido.**

Un esquema es un documento XML con extensión .xsd

En el documento XML definido con un esquema se incluye una referencia al .xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema">
  <element name="capitulos">
    <complexType>
      <sequence>
        <element name="capitulo" maxOccurs="unbounded">
          <complexType>
            <sequence>
              <element name="titulo" type="string"/>
              <element name="pagina-inicial" type="integer"/>
              <element name="pagina-final" type="integer"/>
            </sequence>
          </complexType>
        </element>
      </sequence>
    </complexType>
  </element>
</schema>
```

fichero.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<capitulos xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="Libro.xsd">
  <capitulo>
    <titulo>Presentación</titulo>
    <pagina-inicial>7</pagina-inicial>
    <pagina-final>10</pagina-final>
  </capitulo>
  <capitulo>
    <titulo>Madrid antes de ser Madrid</titulo>
    <pagina-inicial>11</pagina-inicial>
    <pagina-final>20</pagina-final>
  </capitulo>
  <capitulo>
    <titulo>Madrid medieval</titulo>
    <pagina-inicial>21</pagina-inicial>
    <pagina-final>68</pagina-final>
  </capitulo>
  <capitulo>
    <titulo>Madrid de los Austrias</titulo>
    <pagina-inicial>69</pagina-inicial>
    <pagina-final>114</pagina-final>
  </capitulo>
</capitulos>
```

fichero.xml

1.2 Comparación DTD-XSD

Una definición DTD tiene algunas limitaciones:

- ▶ La definición DTD no es un documento XML válido, por lo que no se puede comprobar si está bien formado
- ▶ No se puede restringir el tipo de datos (solo texto #PCDATA)
- ▶ No permite emplear espacios de nombres
- ▶ Otras limitaciones: no se puede dar un valor por defecto a un elemento, no se puede concretar el número de repeticiones en la cardinalidad de los elementos...

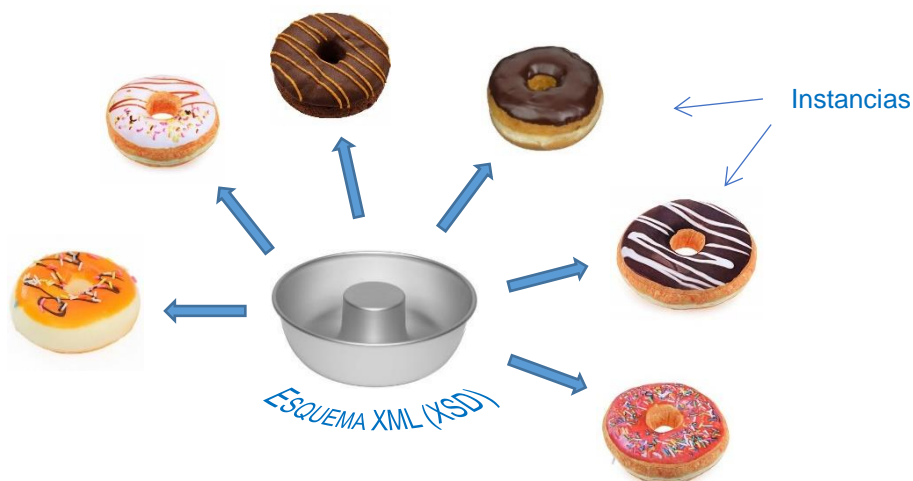
Características XSD:

- ▶ La definición de un esquema XSD es un documento XML bien formado
- ▶ Se distinguen distintos tipos de datos
- ▶ Es más potente que DTD: existe mayores posibilidades de definición (restricciones y detalles)
- ▶ Permite emplear espacios de nombres
- ▶ No admiten la definición de entidades (para ello se emplea DTD)

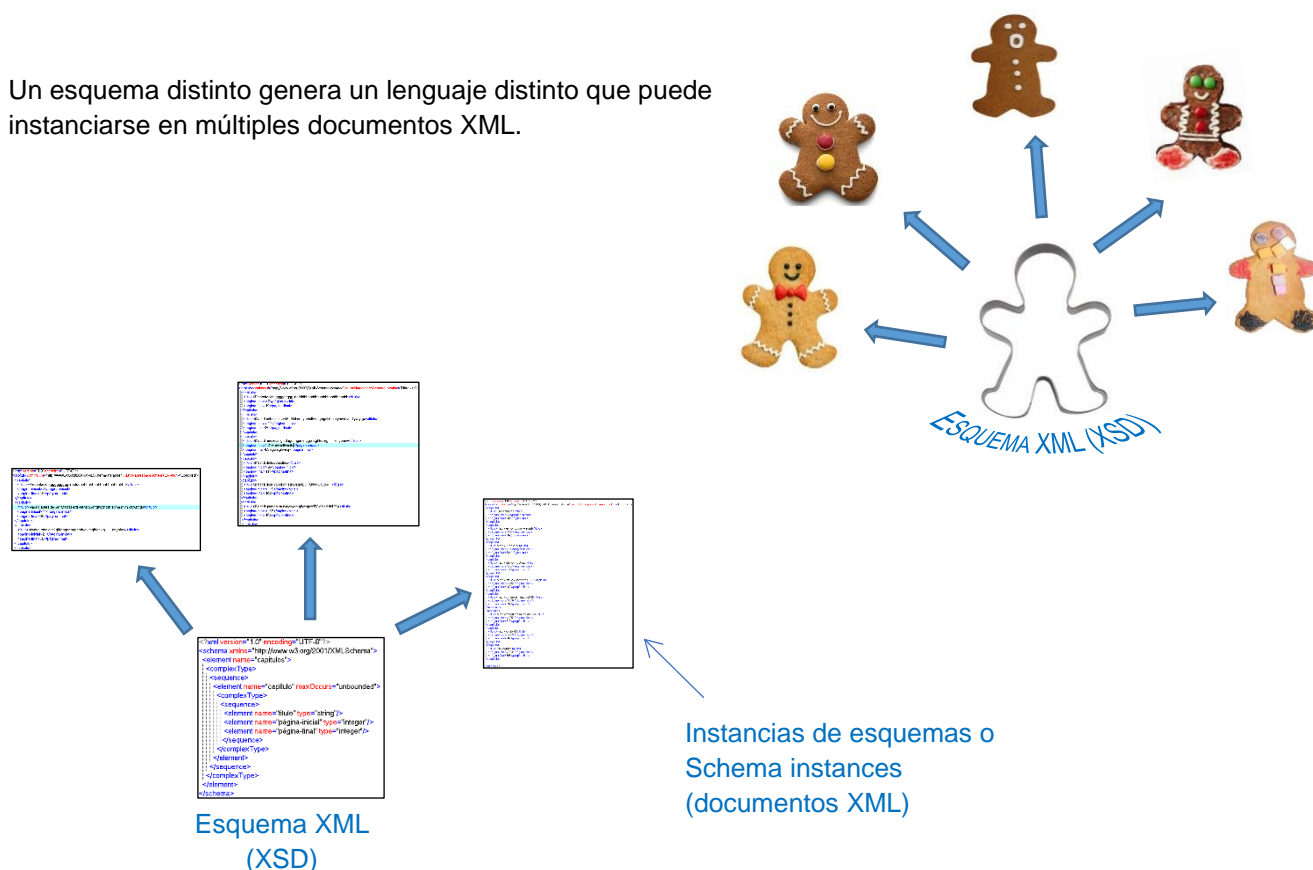
1.3 Instancias

Los documentos XML validados por un esquema XSD concreto se denominan instancias del esquema.

El concepto es similar a tener un molde (esquema o schema), del cual se generan varias copias o instancias (documentos XML validados). Esta idea es análoga a las clases y objetos de la programación orientada a objetos.



Un esquema distinto genera un lenguaje distinto que puede instanciarse en múltiples documentos XML.



Documento de instancia con esquema válido: documento XML que se describe con un esquema XSD y cumple todas sus restricciones.

2. Validación de un esquema

2.1 Documento bien formado

Un esquema XSD sirve para generar o validar documentos XML, pero a su vez **es un documento XML**. Por ello sigue las mismas reglas para estar bien formado.

A modo de recordatorio, para que un documento esté bien formado, al menos debe cumplir los siguientes puntos:

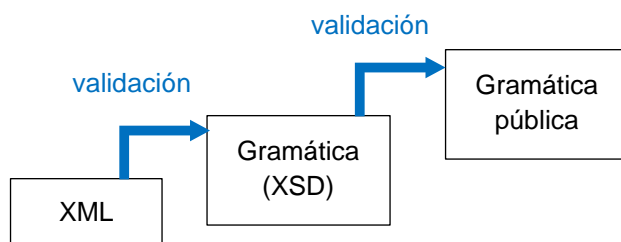
- ➔ El documento contiene únicamente caracteres Unicode válidos.
- ➔ Se diferencia entre mayúsculas y minúsculas.
- ➔ Hay un elemento raíz que contiene al resto de elementos.
- ➔ Los nombres de los elementos y de sus atributos no contienen espacios.
- ➔ El primer carácter de un nombre de elemento o de atributo puede ser una letra, dos puntos (:) o guion bajo (_).
- ➔ El resto de caracteres pueden ser también números, guiones (-), guiones bajos (_) o puntos (.).
- ➔ Los caracteres "<" y "&" sólo se utilizan como comienzo de marcas.
- ➔ Las letras no inglesas (á, Á, ñ, Ñ...) están permitidas. Sin embargo, es recomendable no utilizarlas para reducir posibles incompatibilidades con programas que puedan no reconocerlas.
- ➔ Las etiquetas de apertura, de cierre y vacías están correctamente anidadas (no se solapan) y no falta ni sobra ninguna etiqueta de apertura o cierre.
- ➔ Las etiquetas de cierre coinciden con las de apertura (incluso en el uso de mayúsculas y minúsculas).
- ➔ Las etiquetas de cierre no contienen atributos.
- ➔ Ninguna etiqueta tiene dos atributos con el mismo nombre.
- ➔ Todos los atributos tienen algún valor.
- ➔ Los valores de los atributos están entre comillas (simples o dobles).
- ➔ No existen referencias en los valores de los atributos.
- ➔ Puede haber espacio o salto de línea al final de una etiqueta, pero no al principio.

Correcto: <elemento > </elemento >

Incorrecto: < elemento> </ elemento>

2.2 Documento válido

Hay que recordar que para que un documento sea válido, **debe estar bien formado y además debe respetar las reglas de su gramática.**



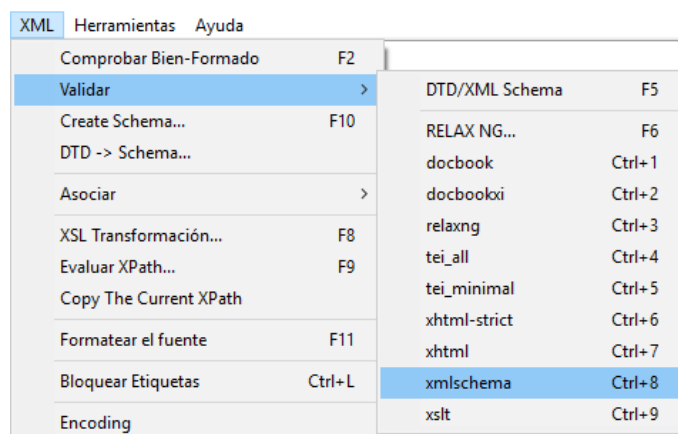
Existen múltiples validadores online que realizan la validación del fichero XSD (por ejemplo <http://www.utilities-online.info/xsdvalidation>).

Es frecuente que los editores XML permitan realizar la validación de los esquemas.

Validación de esquemas con XML Copy Editor

Abre el esquema Libro.xsd.

- 1) Comprueba si está bien formado
- 2) Comprueba si es válido empleando la opción de menú XML -> Validar -> Xmlschema



- Observa que la validación del esquema no se hace con la misma opción que la validación XML, ya que el elemento <schema> es específico de los esquemas y no se valida con una gramática de la forma habitual.

3. Asociación de un XML (instancia) a un XSD (esquema)

3.1 Espacio de nombres

Ya que existe libertad para elegir los nombres de los elementos XML, siempre que sigan las normas sintácticas básicas, es posible que si se van a emplear varios esquemas se den conflictos en nombres repetidos.

Por ejemplo, si se tienen los siguientes documentos XML:

Fichero1.xml: describe un mueble
(concretamente una mesa)

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

Fichero2.xml: describe una tabla HTML

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

FicheroTotal.xml

```
<table>
  <name>African Coffee Table</name>
  <width>80</width>
  <length>120</length>
</table>
<table>
  <tr>
    <td>Apples</td>
    <td>Bananas</td>
  </tr>
</table>
```



Existe un conflicto con el elemento `<table>`, que aparece dos veces con distinta estructura

Solución: aplicar espacios de nombres que identifiquen distintos esquemas

```
<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>
```

Un **espacio de nombres** es un conjunto de nombres que no se repiten


Prefijo: hace referencia a un espacio de nombres concreto, en el que se declaran todos los elementos y atributos

Un **espacio de nombres XML** es una recomendación W3C para proporcionar elementos y atributos con nombre único en un archivo XML. **Un archivo XML puede contener nombres de elementos o atributos procedentes de varios vocabularios XML.** Si a cada uno de estos vocabularios se le da un espacio de nombres, un ámbito semántico propio, referenciado a una URI donde se listen los términos que incluye, se resuelve la ambigüedad existente entre elementos o atributos que se llamen igual.

Cada espacio de nombres se identifica con un **URI** (uniform resource identifier), que es un identificador único. Por ejemplo: <http://www.w3.org/2001/XMLSchema> e <http://www.w3.org/2001/XMLSchema-instance>.

Un URI identifica a un espacio de nombres, pero no contiene directamente la especificación del esquema (fichero xsd).

Como se verá más adelante, dentro de un fichero XML, para asociar un prefijo determinado a un espacio de nombres se emplea el formato:

`xmlns:prefijo="URI"`

 (xml name space)

Por ejemplo:

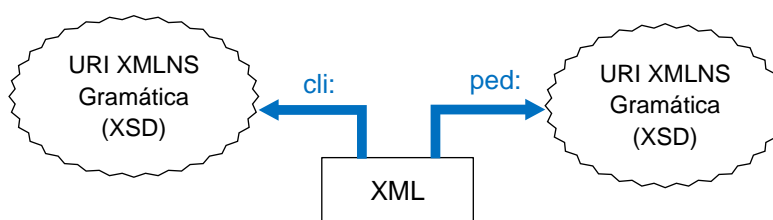
```
xmlns:f="https://empresa.com/muebles"
xmlns:h="https://codigoHTML.org/tablas"
```

Otro ejemplo:

Fichero xml con datos de clientes y productos solicitados por éste. Tanto cliente como producto tienen un identificador "numero_ID". Para que no haya conflicto se pueden usar dos namespaces diferentes:

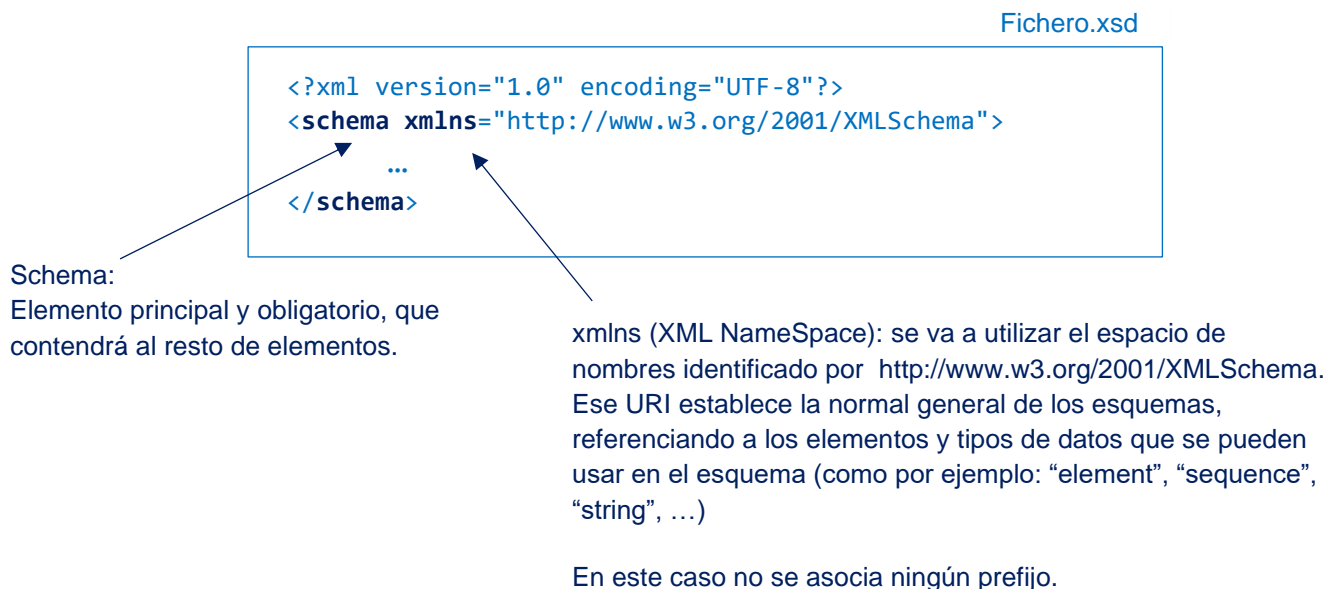
```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<cli:cliente ... xmlns:cli="https://www.espacioNombresXML.org/cliente"
                xmlns:ped="https://www.espacioNombresXML.org/pedido"
  <cli:numero_ID>123456</cli:numero_ID>
  <cli:nombre>Nananana</cli:nombre>
  <cli:telefono>99999999</cli:telefono>
  <ped:pedido>
    <ped:numero_ID>65656565</ped:numero_ID>
    <ped:articulo>Caja herramientas</ped:articulo>
    <ped:precio>187.90</ped:precio>
  </ped:pedido>
</cli:cliente>
```



3.2 XML sin espacio de nombres

En el fichero XSD es necesario utilizar el elemento reservado `<schema>`, que contendrá al resto de elementos. El formato básico de un esquema XSD es:



Ejemplo de un esquema XSD completo:

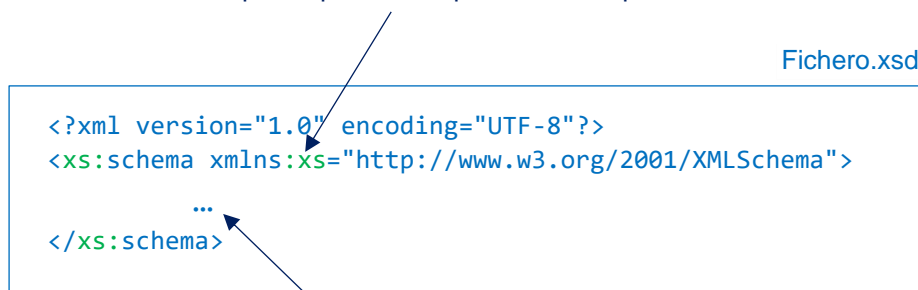
```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema">
  <element name="carta">
    <complexType>
      <sequence>
        <element name="palo" type="string"/>
        <element name="valor" type="positiveInteger"/>
      </sequence>
    </complexType>
  </element>
</schema>
```

Los elementos (`<element>`, `<sequence>`, `<complexType>`...), atributos (name, type,...) y tipos de datos (string, positiveInteger,...) que se utilizan, **no** se pueden elegir sino que deben estar en el espacio de nombres `http://www.w3.org/2001/XMLSchema` (especificación general de esquemas)

Opcionalmente puede declararse un prefijo asociado a la norma de esquemas, para indicar que todos los nombres de elementos, atributos y tipos que se usen estarán allí declarados. Es habitual usar **xs:** o **xsd:** (*schema*), aunque puede emplearse otro cualquiera.

Si se emplea, todas las definiciones dentro del esquema deben ir precedidas de ese prefijo. **Es recomendable emplearlo para evitar problemas con las definiciones de tipos de datos construidos por el usuario.**

Se asocia el prefijo **xs:** a los elementos y tipos de datos declarados en el espacio de nombres `http://www.w3.org/2001/XMLSchema`, es decir, los que se pueden emplear en el esquema



Todos los nombres de elementos y tipos deberán estar precedidos por el prefijo **xs:**

Ejemplo de un esquema XSD completo:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="carta">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="palo" type="xs:string"/>
        <xs:element name="valor" type="xs:positiveInteger"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

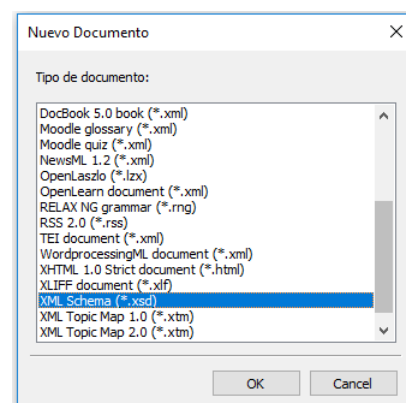
Creación de esquemas con XML Copy Editor

Abre un elemento nuevo desde el editor



Elige el tipo de documento XML Schema

Observa la estructura básica del esquema que se genera.



En el fichero XML:

xmlns (XML NameSpace): se va a utilizar el espacio de nombres identificado por <http://www.w3.org/2001/XMLSchema-instance>. Ese URI establece la norma general de las instancias de esquemas, permitiendo utilizar atributos de instancias como por ejemplo "noNamespaceSchemaLocation".

Se asocia al prefijo **xsi** (**xml instance**), aunque podría ser cualquier otro. **Ese prefijo xsi: solo se emplea para utilizar atributos relacionados con los esquemas, no para los propios elementos de datos en el xml.**

Fichero.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<elemento_raíz xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="fichero.xsd">
  ...
</elemento_raíz>
```

noNamespaceSchemaLocation: atributo que indica dónde estará el esquema para validar los elementos XML que aparezcan sin espacio de nombres ni prefijo

Ejemplo completo:

carta.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="carta">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="palo" type="xs:string"/>
        <xs:element name="valor" type="xs:positiveInteger"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Documento
XML

```
<?xml version="1.0" encoding="UTF-8"?>
<carta xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="carta.xsd">
  <palo>trébol</palo>
  <valor>4</valor>
</carta>
```

Los elementos XML que no tienen espacio de nombre ni prefijo (carta, palo y valor) se validarán con el esquema indicado en noNamespaceSchemaLocation, es decir, en el fichero baraja.xsd

Vincular un XML a un esquema de forma automática con XML Copy Editor

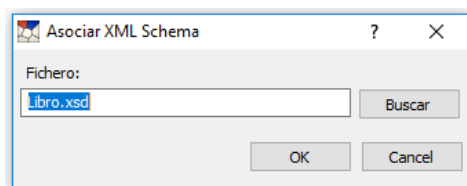
Los editores XML suelen incluir utilidades para trabajar con esquemas.

- 1) Abre el fichero HistoriaDeMadrid.xml

Comprueba si está bien formado. ✓

Comprueba si es válido. ✓

- 2) Para asociar el xml al esquema Libro.xsd, puede hacerse manualmente o utilizando la opción de menú "XML → Asociar → XML Schema". Selecciona el esquema y comprueba cómo el editor incluye un atributo noNamespaceSchemaLocation.



- 3) Comprueba de nuevo si es válido. ✓

- 4) Modifica la ruta indicada en el atributo noNamespaceSchemaLocation para que sea una ruta relativa, no absoluta.

3.3 XML con espacio de nombres

En el fichero XSD es necesario indicar el espacio de nombres asociado con el esquema:

Se asocia el prefijo **xs:** a los elementos y tipos de datos que se pueden usar dentro del esquema (están en el espacio de nombres "http://www.w3.org/2001/XMLSchema")

Fichero.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="https://www.dominio.com"
  elementFormDefault="qualified">
  ...
</xs:schema>
```

targetNamespace: espacio de nombres asociado a este esquema (URI)

elementFormDefault: por defecto, todos los elementos del documento XML son "calificados"

Para asociar el fichero XML a un esquema empleando un prefijo, es necesario declarar el prefijo que se va a utilizar (xmlns:prefijo="espacio_nombres") e indicar dónde está el esquema de ese espacio de nombres (schemaLocation="espacio_nombres ruta_fichero.xsd"):

xmlns (XML NameSpace): se asocia el prefijo **xsi** a los elementos que están en el espacio de nombres <http://www.w3.org/2001/XMLSchema-instance> (como por ejemplo "SchemaLocation")

Fichero.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<elemento_raíz xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:pref="https://www.dominio.com"
  xsi:schemaLocation="https://www.dominio.com fichero.xsd"
  ...
</elemento_raíz>
```

SchemaLocation: atributo que asocia un esquema a un espacio de nombres. Está compuesto por parejas de valores separados por un espacio:

Xmlns (xml namespace): se asocia el prefijo **pref**: (puede ser cualquiera, a elección del programador) a un espacio de nombres

- URI del espacio de nombres (por ej. <https://www.dominio.com>)
- Ruta del fichero XSD correspondiente a ese espacio de nombres.

Ejemplo completo:

carta.xsd


```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="https://www.dominio.com" elementFormDefault="qualified" >
  <xs:element name="carta">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="palo" type="xs:string"/>
        <xs:element name="valor" type="xs:positiveInteger"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Documento
XML

```
<?xml version="1.0" encoding="UTF-8"?>
<brj:carta xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="https://www.dominio.com carta.xsd"
  xmlns:brj="https://www.dominio.com">
  <brj:palo>trébol</brj:palo>
  <brj:valor>4</brj:valor>
</brj:carta>
```

Los elementos XML con el prefijo **brj** se validan con el espacio de nombres <https://www.dominio.com> y el esquema carta.xsd

Modifica el fichero HistoriaDeMadrid.xml para que tenga asociado el esquema Libro.xsd con el prefijo "lb" siguiendo los pasos:

- 1) En el esquema Libro.xsd:
 - Añade el espacio de nombres con el atributo `targetNamespace` (inventa el URI).
 - Añade el atributo `elementFormDefault="qualified"`
 - Comprueba que el esquema es válido.
- 2) En el fichero XML, en el elemento raíz "capítulos", debes:
 - Añadir el prefijo lb y asociarlo al espacio de nombres con `xmlns:lb`. En este atributo se indica el URI.
 - Sustituir el atributo `noNamespaceSchemaLocation` por `schemaLocation`. Ten en cuenta que en este atributo hay que indicar el URI y la ruta del fichero xsd.
- 3) Además, es necesario que en el XML todos los elementos vayan precedidos por el prefijo. Por ejemplo `<título></título>` pasa a ser `<lb:título></lb:título>`
- 4) Comprueba que el documento XML es válido 

3.4 Elementos calificados y no calificados

En un esquema, los elementos se dividen en globales y locales:

➔ **Elementos globales:** son hijos directos del elemento raíz, es decir, cuelgan directamente del elemento `<schema>`

➔ **Elementos locales:** resto de elementos

Se dice que un elemento está calificado si debe incluir el prefijo de su espacio de nombres.

Los elementos globales deben estar calificados obligatoriamente.

En el esquema XSD puede indicarse si los elementos están o no calificados empleando el atributo:

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

Si no se incluye el atributo `elementFormDefault`, por defecto se considera el valor "unqualified"

Ejemplo completo:

indice.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.lm.com/indice"
  elementFormDefault="XXX">
  <xs:element name="índice" >
    <xs:complexType>
      <xs:sequence>
        <xs:element name="entrada" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="nombre" type="xs:string"/>
              <xs:element name="página" type="xs:positiveInteger"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Documento
XML

XXX =
qualified

```
<?xml version="1.0" encoding="UTF-8"?>
<ind:índice xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.lm.com/indice marcadores.xsd"
  xmlns:ind="http://www.lm.com/indice">
  <ind:entrada>
    <ind:nombre>Prólogo</ind:nombre>
    <ind:página>2</ind:página>
  </ind:entrada>
  <ind:entrada>
    <ind:nombre>Capítulo 1</ind:nombre>
    <ind:página>13</ind:página>
  </ind:entrada>
</ind:índice>
```

XXX =
unqualified

```
<?xml version="1.0" encoding="UTF-8"?>
<ind:índice xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.lm.com/indice marcadores.xsd"
  xmlns:ind="http://www.lm.com/indice">
  <entrada>
    <nombre>Prólogo</nombre>
    <página>2</página>
  </entrada>
  <entrada>
    <nombre>Capítulo 1</nombre>
    <página>13</página>
  </entrada>
</ind:índice>
```

Modifica el esquema Libro.xsd de forma que el atributo elementFormDefault tenga el valor "unqualified".

Modifica el documento HistoriaDeMadrid.xml para que sus elementos no estén calificados. Para ello:

- El elemento <capítulos>, al ser global, debe seguir calificado, es decir, debe mantener el prefijo.
- El resto de elementos, al ser locales, deben estar no calificados (sin prefijo).

Comprueba que el documento XML es válido



Actividad 12. Vincular XML a XSD

- 12.1 Sin espacio de nombres
- 12.2 Con espacio de nombres sin calificar
- 12.3 Con espacio de nombres calificado

Actividad 13. Vincular XML a XSD con varios espacios de nombres

- 13.1 Validación con dos esquemas calificados
- 13.2 Validación con dos esquemas no calificados

Actividad 14. Vincular XML a XSD con varios espacios de nombres y sin espacios de nombres

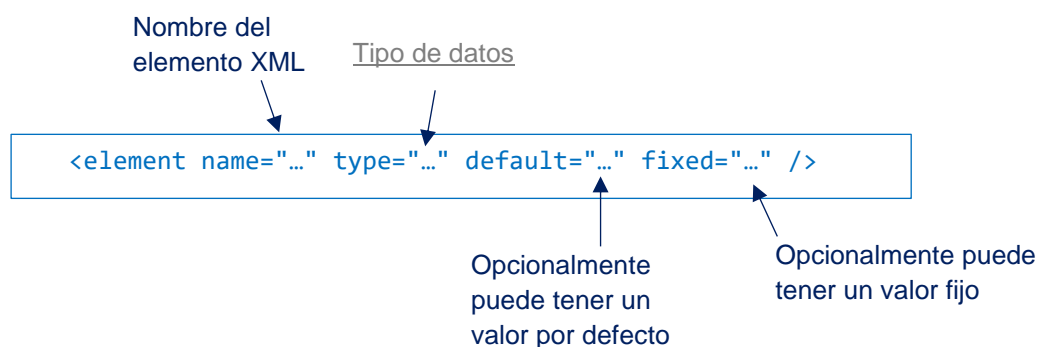
4. Elementos

4.1 Elementos simples

Un elemento se identifica por una etiqueta en xml. Los elementos simples:

- ▶ Solo contienen texto, aunque puede ser cualquier tipo de dato (string, integer, date...)
- ▶ Pueden tener un valor por defecto o fijo
- ▶ NO pueden contener atributos
- ▶ NO pueden contener otros elementos hijos

Formato:



Ejemplos:

```
<element name="color" type="string" />
<element name="color" type="string" default="rojo"/>
<element name="color" type="string" fixed="rojo"/>
```

Ejemplo completo:

miSchema.xsd

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="película" type="xs:string" />
</xs:schema>
```

Documento
XML

```
<?xml version="1.0" encoding="UTF-8"?>
<película xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="miSchema.xsd">
  El Padrino
</película>
```

Actividad 15. Definir un elemento simple

4.2 Elementos complejos

Los elementos complejos:

- ▶ Pueden ser elementos vacíos
- ▶ Pueden contener otros elementos
- ▶ Pueden contener solo texto
- ▶ Pueden contener otros elementos y texto (se denominan mixtos)
- ▶ Pueden tener atributos

Formato básico:

Indica que es un elemento complejo mixto (contiene otros elementos y texto). Opcional

```
<element name="..." >
  <complexType mixed="true" >
    ...
  </complexType>
</element>
```

▪ Indicadores de orden

Establecen el orden que deben tener los elementos hijos.

Formato:

```
<element name="..." >
  <complexType>
    <sequence | all | choice >
      Elementos-hijo
    </sequence | all | choice >
  </complexType>
</element>
```

Indicador de orden:

- **<sequence>** se deben incluir todos los elementos en orden (cada uno solo una vez)
- **<all>** se deben incluir todos los elementos, sin importar el orden (cada uno solo una vez)
- **<choice>** solo se incluye uno de los elementos (una única vez)

Ejemplos (solo se incluye la definición y uso el elemento):

```
<xs:element name="película">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="título" type="xs:string" />
      <xs:element name="duración" type="xs:integer" />
      <xs:element name="fechaEstreno" type="xs:date" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Los elementos título, duración y fechaEstreno deben aparecer y estar en orden

DOCUMENTO
XML

```
<película>
  <título>El Padrino</título>
  <duración>175</duración>
  <fechaEstreno>1972-03-15</fechaEstreno>
</película>
```

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

El elemento carta es mixto: contiene a otros elementos (fecha y despedida) y texto

DOCUMENTO
XML

```
<carta>
  <fecha>2001-01-01</fecha>
  Espero que estés bien.
  <despedida>Hasta pronto</despedida>
</carta>
```

```

<xs:element name="película">
  <xs:complexType>
    <xs:all>
      <xs:element name="título" type="xs:string" />
      <xs:element name="duración" type="xs:integer" />
      <xs:element name="fechaEstreno" type="xs:date" />
    </xs:all>
  </xs:complexType>
</xs:element>

```

Los elementos título, duración y fechaEstreno deben aparecer pero pueden no estar en orden

DOCUMENTO
XML

```

<película>
  <título>El Padrino</título>
  <fechaEstreno>1972-03-15</fechaEstreno>
  <duración>175</duración>
</película>

```

```

<xs:element name="ciudadano">
  <xs:complexType>
    <xs:choice>
      <xs:element name="DNI" type="xs:string" />
      <xs:element name="NIF" type="xs:string" />
    </xs:choice>
  </xs:complexType>
</xs:element>

```

Debe aparecer uno de los elementos, DNI o NIF.

DOCUMENTOS
XML

```

<ciudadano>
  <DNI>123456789</DNI>
</ciudadano>

```

```

<ciudadano>
  <NIF>123456789-N</NIF>
</ciudadano>

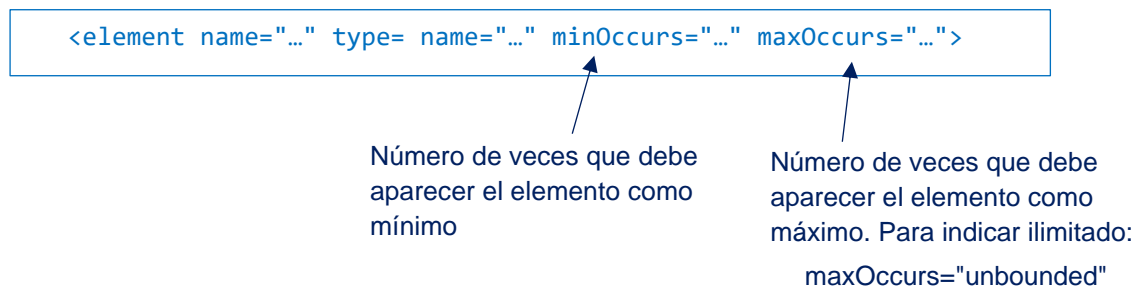
```

Actividad 16. Definir elementos complejos con indicador de orden

- 16.1 Definir un único elemento complejo
- 16.2 Anidar elementos complejos

■ Indicadores de ocurrencia (frecuencia)

Establecen el número de veces que puede aparecer un elemento hijo.



Ejemplo (solo se incluye la definición y uso el elemento):

```
<xs:element name="contenedor">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="descripción" type="xs:string" />
      <xs:element name="caja" type="xs:string" minOccurs="0" maxOccurs="50"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

DOCUMENTO
XML

```
<contenedor>
  <descripción>Losetas grandes</descripción>
  <caja>modelo-100</caja>
  <caja>modelo-101</caja>
</contenedor>
```

Actividad 17. Definir elementos complejos con indicador de ocurrencia

▪ Elemento <any>

El elemento <any> es cualquier elemento.

<any/>

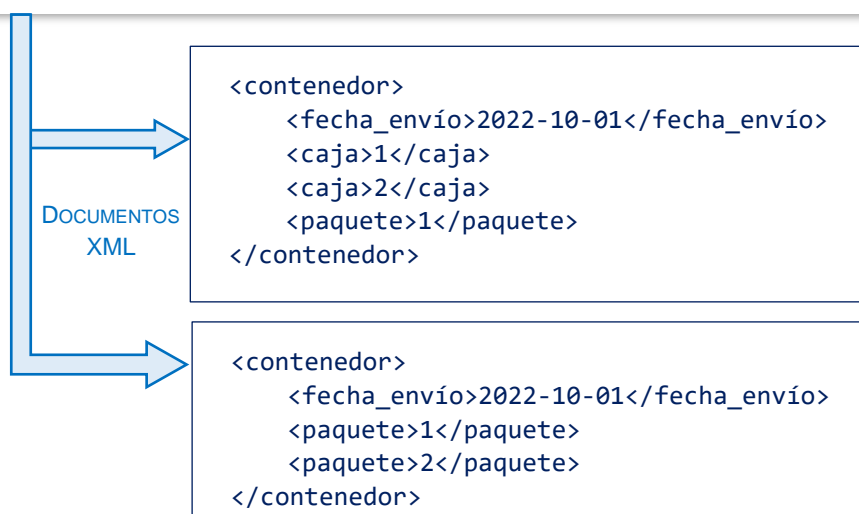
Puede usarse para permitir que un elemento tenga **cualquier otro elemento** como hijo. Hay que destacar que "cualquier otro elemento" permite elegir un elemento sin restricciones, pero para que el documento sea válido, **ese elemento debe estar declarado en algún sitio** (en caso contrario no podría validarse).

Ejemplo (solo se incluye la definición y uso el elemento):

```
<xs:element name="contenedor">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="fecha_envío" type="xs:date" />
      <xs:any minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="caja" type="xs:positiveInteger" />
<xs:element name="paquete" type="xs:positiveInteger" />
```

Observa que hay dos tres elementos globales en este esquema (contenedor, caja y paquete).

Los elementos caja y paquete podrían estar declarados en otro esquema distinto.



Actividad 18. Elemento any

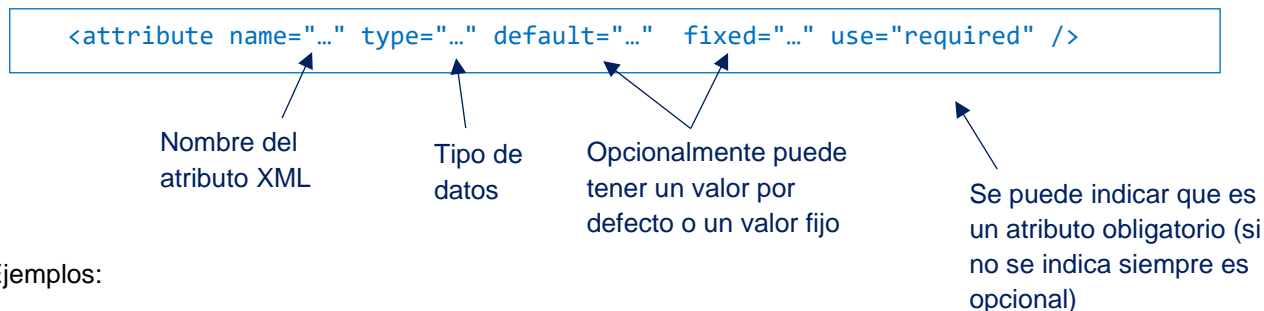
- 18.1 Validación con <any>
- 18.2 Elementos de varios esquemas

5. Atributos

5.1 Declaración de atributo

Un elemento complejo puede contener atributos.

Formato:



Ejemplos:

```
<attribute name="edad" type="integer" />
<attribute name="edad" type="integer" default="18" />
<attribute name="edad" type="integer" fixed="18" />
<attribute name="edad" type="integer" use="required" />
```

miXSD.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="juego">
    <xs:complexType>
      <xs:all>
        <xs:element name="título" type="xs:string" />
        <xs:element name="duración" type="xs:integer" />
      </xs:all>
      <xs:attribute name="dificultad" type="xs:string"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Documento
XML

```
<juego xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="miXSD.xsd"
  dificultad="fácil">
  <título>Parchís</título>
  <duración>35</duración>
</juego>
```

Ejemplo completo: Actividad 19. Insertar atributos

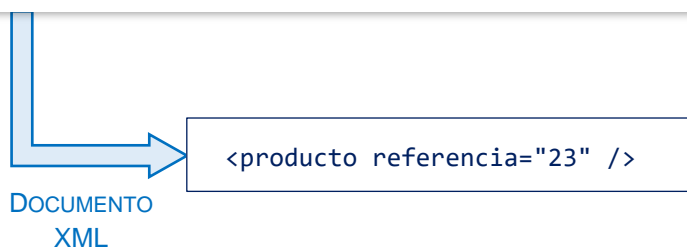
5.2 Elemento EMPTY con atributos

En ocasiones se desea que un elemento no tenga ningún contenido, es decir, que sea un elemento vacío o EMPTY. En este caso, es habitual configurar el elemento a través del valor de sus atributos.

Ejemplo (solo se incluye la definición y uso el elemento):

```
<xs:element name="producto">
  <xs:complexType>
    <xs:attribute name="referencia" type="xs:positiveInteger"/>
  </xs:complexType>
</xs:element>
```

El elemento producto es un elemento vacío (EMPTY), es decir, sin contenido



Actividad 20. Elementos vacíos con atributos

5.3 Atributo <anyAttribute>

El elemento <anyAttribute> es cualquier atributo.

```
<anyAttribute />
```

Puede usarse para permitir que un elemento tenga **cualquier atributo**. Hay que destacar que “cualquier atributo” permite elegir un atributo sin restricciones, pero para que el documento sea válido, **ese atributo debe estar declarado en algún sitio** (en caso contrario no podría validarse).

Ejemplo completo:

miXSD.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="grupo">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="persona" minOccurs="1" >
          <xs:complexType>
            <xs:all>
              <xs:element name="nombre" type="xs:string" />
              <xs:element name="edad" type="xs:integer" />
            </xs:all>
              <xs:attribute name="ocupación" type="xs:string" use="required"/>
              <xs:anyAttribute/>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:attribute name="nacionalidad" type="xs:string"/>
  </xs:schema>
```

El atributo “nacionalidad”
podría estar declarado en
un esquema distinto

DOCUMENTO
XML

```
<grupo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="miXSD.xsd" >
  <persona ocupación="profesora" nacionalidad="española">
    <nombre>Mar</nombre>
    <edad>35</edad>
  </persona>
</grupo>
```

Actividad 21. Utilización de anyAttribute

- 21.1 Atributos en el mismo esquema
- 21.2 Atributos en otro esquema

6. Tipos de datos

Indica cómo debe ser el contenido de los elementos o atributos. Se distinguen tipos predefinidos y tipos contruidos por el usuario.

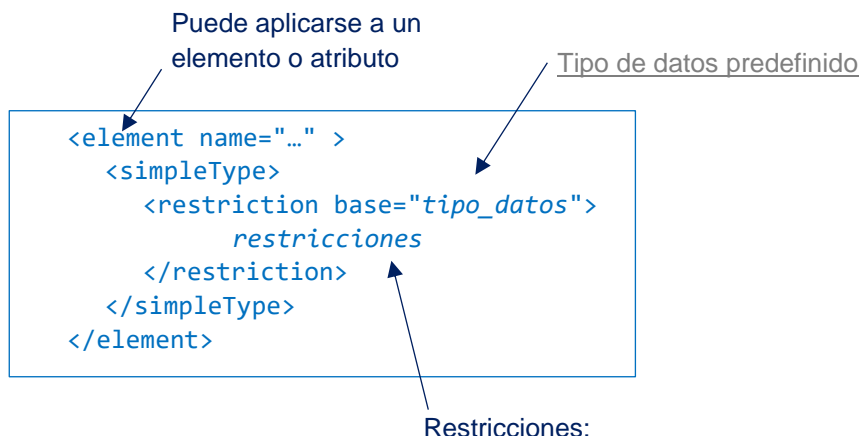
6.1 Tipos predefinidos

TIPOS DE DATOS PREDEFINIDOS			
Tipo	Descripción	Ejemplo Schema	Ejemplo XML
string	Cadena de texto	<code><element name="nombre" type="string" /></code>	<code><nombre>Alba</nombre></code>
boolean	true o false	<code><element name="hecho" type="boolean" /></code>	<code><hecho>true</hecho></code> <code><hecho>false</hecho></code>
Integer	Núm. entero	<code><element name="grados" type="integer" /></code>	<code><grados>-45</grados></code>
positiveInteger	Entero positivo	<code><element name="gr" type="positiveInteger" /></code>	<code><gr>90</gr></code>
decimal	Número decimal	<code><element name="grados" type="decimal" /></code>	<code><grados>90</grados></code>
float		<code><element name="grados" type="float" /></code>	<code><grados>90.5</grados></code>
date	fecha	<code><element name="alta" type="date" /></code>	<code><alta>2022-05-26</alta></code>
time	hora	<code><element name="cita" type="time" /></code>	<code><cita>17:55:00</cita></code>
dateTime	Fecha y hora	<code><element name="cita" type="dateTime" /></code>	<code><cita>2022-05-26T17:55:00</cita></code>

Nota: existen más tipos de datos predefinidos. Ver [built-in-datatypes](#)

6.2 Restricciones en los datos (facetas)

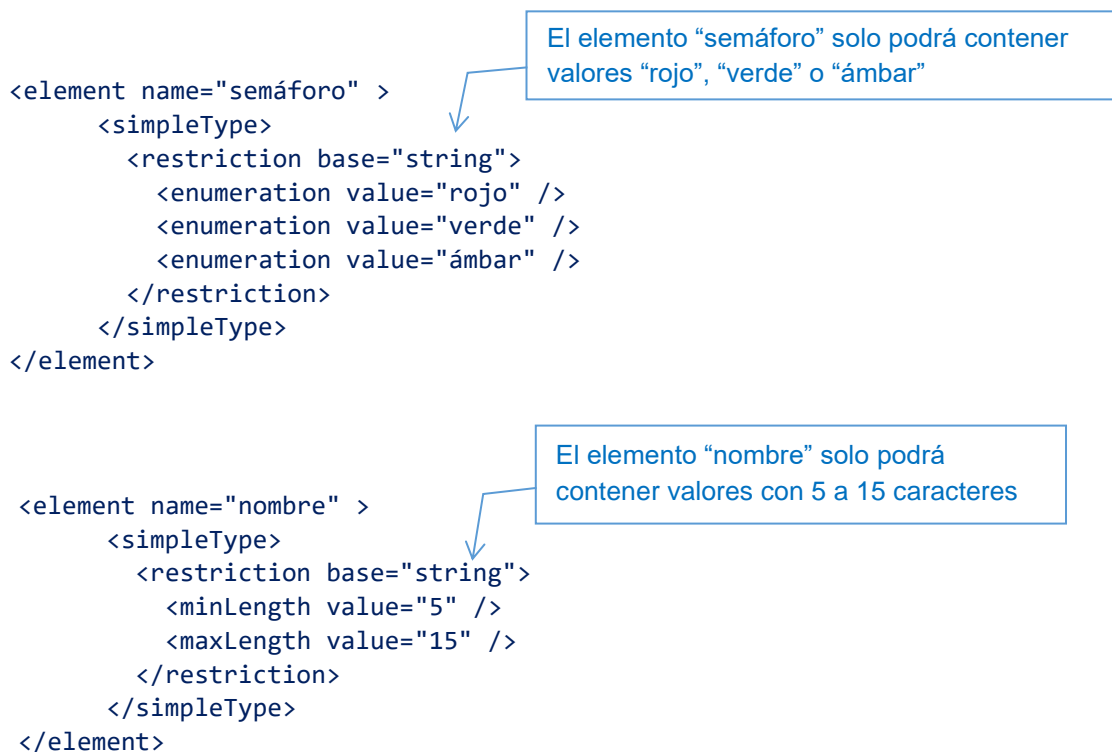
En ocasiones es interesante limitar el posible valor que pueda tener un dato. Formato básico:



- Restricciones:
- <enumeration value="..."/> valor posible
 - <length value="..."/> longitud exacta del string
 - <minLength value="..."/> longitud mínima del string
 - <maxLength value="..."/> longitud máxima del string
 - <minInclusive value="..."/> valor mínimo del número
 - <maxInclusive value="..."/> valor máximo del número
 - <totalDigits value="..."/> Dígitos máx. totales del integer
 - <pattern value="..."/> expresión regular

Nota: existen más tipos de restricciones. Ver facets

Ejemplos:



```
<element name="nombre" >  
  <simpleType>  
    <restriction base="string">  
      <length value="10" />  
    </restriction>  
  </simpleType>  
</element>
```

El elemento "nombre" solo podrá
contener valores de 10 caracteres

```
<element name="edad" >  
  <simpleType>  
    <restriction base="integer">  
      <minInclusive value="18" />  
      <maxInclusive value="120" />  
    </restriction>  
  </simpleType>  
</element>
```

El elemento "edad" solo podrá contener
valores numéricos de 18 a 120

```
<attribute name="IBAN" >  
  <simpleType>  
    <restriction base="integer">  
      <totalDigits value="24" />  
    </restriction>  
  </simpleType>  
</attribute>
```

El atributo "IBAN" solo podrá contener
valores con 24 dígitos

```
<element name="dominio" >  
  <simpleType>  
    <restriction base="string">  
      <pattern value="[a-z]*[0-3]" />  
    </restriction>  
  </simpleType>  
</element>
```

El elemento "dominio" debe cumplir el
patrón especificado en la expresión regular

Actividad 22. Restricciones en los tipos de datos

6.3 Tipos construidos por el usuario

Empleando restricciones, es posible crear tipos de datos personalizados. La ventaja es que, una vez declarado el tipo de datos, es posible utilizarlo por varios elementos o atributos.

Existen dos posibilidades:

→ simpleType

Crear tipos de datos empleando restricciones de datos. Se da un nombre a una serie de restricciones para poder reutilizarlas fácilmente.

```
<simpleType name="..." >
  <restriction base="tipo_datos">
    Restricciones
  </restriction>
</simpleType>
```

Nombre del tipo de datos propio

→ complexType

Crear tipos de datos empleando elementos complejos. Se da un nombre a una agrupación de elementos y atributos para poder reutilizarla fácilmente.

```
<complexType name="..." >
  <sequence | all | choice >
    Elementos-hijo
  </sequence | all | choice >
</complexType>
```

Nombre del tipo de datos propio

Cuando se usan tipos propios, es obligatorio utilizar un prefijo para los tipos de datos predefinidos (por ejemplo `xs:` o `xsd:`).

Ejemplos (solo se incluye la definición y uso el elemento):

Prefijo obligatorio al existir un tipo de datos propio

```
<xs:element name="estudiante" type="tipoPersona"/>
<xs:element name="paciente" type="tipoPersona"/>

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

El tipo de datos "tipoPersona" es utilizado en dos elementos

DOCUMENTO
XML

```
<estudiante>
  <nombre>Maeva</nombre>
  <apellido>Bousquet</apellido>
</estudiante>
<paciente>
  <nombre>María</nombre>
  <apellido>Sobrino</apellido>
</paciente>
```

Prefijo obligatorio al existir un tipo de datos propio

```
<xs:element name="evento">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="título" type="xs:string" />
      <xs:element name="realización" type="estación" />
    </xs:sequence>
    <xs:attribute name="previsión" type="estación" />
  </xs:complexType>
</xs:element>
<xs:simpleType name="estación">
  <xs:restriction base="xs:string">
    <xs:enumeration value="primavera"/>
    <xs:enumeration value="verano"/>
    <xs:enumeration value="otoño"/>
    <xs:enumeration value="invierno"/>
  </xs:restriction>
</xs:simpleType>
```

El tipo de datos "estación" es utilizado en un elemento y en un atributo

DOCUMENTO
XML

```
<evento previsión="invierno">
  <título>Concierto cuarteto Adam</título>
  <realización>primavera</realización>
</evento>
```

Actividad 23. Tipos de datos personalizados

- 23.1 Tipos de datos personalizados con restricciones (simpleType)
- 23.2 Tipos de datos personalizados con complexType

Nota: si se quiere vincular un esquema a un XML con espacio de nombres, y existen tipos de datos propios, es necesario incluir en el esquema el atributo resaltado:

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

6.4 Expresiones regulares

Las expresiones regulares se utilizan en las restricciones de tipos de datos para imponer un patrón en el contenido de un elemento o atributo. Ejemplo:

```
<xs:element name="letra" type="vocal"/>
<xs:simpleType name="vocal">
  <xs:restriction base="xs:string">
    <xs:pattern value="[aeiou]"/>
  </xs:restriction>
</xs:simpleType>
```

El elemento "letra" solo podrá contener los valores "a", "e", "i", "o", "u"

PATRÓN	SIGNIFICADO	EJEMPLOS	
		PATRÓN	VALIDACIONES CORRECTAS
<i>caracteres</i>	Esos caracteres tal y como están	pattern="mu"	mu
.	Cualquier carácter	pattern="b.a"	bca
	OR	pattern="a b"	a, b (una sola)
[...]	Un carácter contenido en los corchetes (OR entre varios elementos)	pattern="[abc]"	a, b, c (una sola)
[inicio-fin]	Rango	pattern="[a-d]"	Cualquier letra minúscula (una sola)
[^...]	Negación	pattern="^[abc]"	Cualquier carácter que no sea a, b ni c
?	Opcional	pattern="a?b"	Dos posibilidades: b, ab
*	Repetición de 0 a n veces	pattern="a*b"	Varias posibilidades: b, ab, aab, aaaab, aaaaaaaaaaab, ...
+	Repetición de 1 a n veces	pattern="a+b"	Varias posibilidades: ab, aab, aaaab, aaaaaaaaaaab, ...
{n}	Repetición n veces	pattern="ba{3}"	baaa
{n,m}	Repetición de n a m veces. Puede omitirse la m para indicar que no hay límite superior en las repeticiones	pattern="a{3,5}"	Tres posibilidades: aaa, aaaa, aaaaa
		pattern="[a-z]{3}"	abc, ttt, gbt, abb, aaa, ...
		pattern="a{3,} "	aaa, aaaa, aaaaa, aaaaaa, ...
\d	Dígito, equivale a [0-9]	pattern="\d"	0, 1, 2, 3, 4, ...
\D	Cualquier carácter que no sea un dígito	pattern="\D"	A, f, g, G, b, ...
\w	Cualquier carácter alfanumérico	pattern="\w"	a, b, A, u, 5, ...
\s	Espacio	pattern="a\s b"	a b
(...)	Agrupaciones	pattern="(w\s z)*"	a z, f z g z, 5 zy zi zo z, ...

Actividad 24. Tipos de datos con pattern

24.1 Patrones sencillos

24.2 Patrones aplicados