

UT04. 04_XML. SCHEMA

Curso: 2DAW.

Lenguajes de Marcas y sistemas de gestión de información

Validación de documentos XML con esquemas XML.

Esquemas XML

- Mecanismo para comprobar la validez de un documento XML.
- Alternativa a los DTD
- **Ventajas**
 - Sintaxis XML ➔ son analizables como cualquier otro documento XML
 - Mayor facilidad para crear **validaciones complejas** y reutilizables.
 - Permiten concretar con precisión la cardinalidad de un elemento (veces que puede aparecer en un documento XML).
 - Soportan íntegramente los espacios de nombres.
- **Desventajas**
 - Más complejas que las DTD.
 - Presentan más incompatibilidades con software que las DTD.
 - No permiten definir entidades.
 - Tecnologías como SAX o DOM tienen utilidades especiales para las DTD, pero no para los esquemas.

Esquemas XML

- Los documentos XML que se validan contra un esquema se llaman **instancias del esquema**.
 - Esquema → molde
 - Documentos XML → objetos que tratan de ajustarse a ese molde.
- Analogía con la POO.
 - esquemas → clases
 - documentos XML → objetos.

Estructura de un esquema XML

- Reglas:
 - Elemento raíz → **<schema>**.
 - Espacio de nombres → **<http://www.w3.org/2001/XMLSchema>**.
 - Se puede ...
 - no definir un prefijo
 - usar uno de los más comunes: **xs/xsd**.

```
<schema xmlns="http://www.w3.org/2001/XMLSchema">
```

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
```

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

Esto indica que los elementos y los tipos de datos que se usan en el Schema se encuentran en el espacio de nombres <http://www.w3.org/2001/XMLSchema> y que deberían llevar el prefijo xs: (en el primer caso) o xsd: (en el segundo)

Estructura de un esquema XML

- El esquema tiene al menos un componente de declaración de un elemento raíz identificado con
 - Elemento **<xs:element>**
 - Atributo **name = “nombre del elemento raíz”**

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

```
<simple>Es difícil escribir un documento más simple</simple>
```

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

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
```

```
    <xs:element name = "simple" />
```

```
</xs:schema>
```

Estructura de un esquema XML

- Puede haber un esquema con **múltiples** elementos raíz.

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

Estructura de un esquema XML

- **Esquema** → conjunto de declaraciones de elementos, atributos y definición de tipos para fijar la estructura que deben tener sus documentos instancia XML.
- El **orden** de declaración de los **elementos** (componentes), **no** es significativo ni afecta al funcionamiento del mismo.

Estructura de un esquema XML

- Vinculación esquema con el documento XML se hace en el propio documento XML.
 - **xmlns:xsi** y **xsi:noNamespaceSchemaLocation**, como **atributos** del elemento raíz. Esquemas **no** asociados a espacios de nombres.
 - **xsi:schemaLocation** .- Esquema asociado a espacios de nombres.

Estructura de un esquema XML

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

Vincula el
documento XML con un
esquema de nombre
simple.xsd

Declaración del
espacio de nombres
asociada al prefijo xsi.

Atributos de los documentos instancia del esquema o documento XML

- **xsi:noNamespaceSchemaLocation="simple.xsd"** para vincular el documento XML con un esquema de nombre **"simple.xsd"**.
- **xsi:nil** → elemento vacío de contenido.
 - **false** por defecto.
 - **true**, activar el atributo **nillable** en la definición del elemento en el esquema asociado. + adelante.

```
<xs:element name="fechaCompra" type="xs:date" nillable="true"/>
```

```
<fechaCompra xsi:nil="true"></fechaCompra>
```

El contenido de
<fechaCompra> debe ser
vacío

Atributos de los documentos instancia del esquema o documento XML

- **xsi:type**: puede ser la definición de un tipo de un elemento. P.e. **xs:string**
- **xsi:schemaLocation**: localización de un esquema **con** un espacio de nombres asociado. Se pueden indicar varios esquemas separados por espacios.
- **xsi:noNamespaceSchemaLocation**: localización de un esquema **sin** un espacio de nombres asociado. Se pueden indicar varios esquemas separados por espacios.

Componentes básicos de un esquema

- **xs:schema**
- xs:element
- xs:attribute

xs:schema

- Componente de declaración de esquema y elemento raíz de todo el esquema XML.
- Permite diferenciar las etiquetas XML del esquema, respecto a las del documento XML.
- Atributos **optativos**:
 - **xmlns:referencia_URI** uno o más espacios de nombres a usar en el esquema. Si no se indica ningún prefijo, los componentes del esquema del espacio de nombres pueden usarse de manera **no cualificada (no prefijo)**.
 - **id**: identificador único para el elemento.
 - **targetNamespace**: referencia URI al espacio de nombres del esquema (dónde se encuentran definidos los elementos del esquema).

xs:schema

- **elementFormDefault**: formato de los nombres de los elementos en el espacio de nombres del esquema.
 - **unqualified**, que indica que los elementos **no llevan prefijo**.
 - **qualified**, indica que los elementos deben **llevar el prefijo**.
- **attributeFormDefault**: formato de los nombres de los atributos en el espacio de nombres del esquema. Funciona igual **elementFormDefault**.
 - Posibles valores: **unqualified**, **qualified**
- **version**: la versión del esquema.

Estructura
general

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

```
<xs:schema xmlns:xs=http://www.w3.org/2001/XMLSchema>
```

```
...
```

```
</xs:schema>
```

Componentes básicos de un esquema

- xs:schema
- **xs:element**
- xs:attribute

xs:element. I

- Componente de declaración de elemento y representa la existencia de un elemento en el documento XML.
- Atributos **optativos** principales: **name, ref, type, default, fixed, minOccurs, maxOccurs**
 - **name**: nombre del elemento. Si el elemento padre es **<xs:schema>** → Atributo **obligatorio**
 - **ref**: indica que la descripción del elemento se encuentra en otro lugar del esquema. Si el elemento padre es **<xs:schema>** → Atributo **no** se puede usar.
 - **type**: indica el tipo del elemento.

xs:element

- **default**: valor por defecto. Sólo se puede usar si el contenido del elemento es **textual**.
- **fixed**: valor único. Sólo se puede usar si el contenido del elemento es únicamente **textual**.
- **minOccurs**: indica el número **mínimo de ocurrencias** que del elemento puede haber en el documento XML. Si el elemento padre es **<xs:schema>**
→ Atributo **no** se puede usar.
 - Valores: **0.. unbounded** (ilimitado)
- **maxOccurs**: indica el número **máximo de ocurrencias** que del elemento puede haber en el documento XML. Si el elemento padre es **<xs:schema>**
→ Atributo **no** se puede usar.
 - Valores: **0.. unbounded** (ilimitado)

xs:element

- Otros atributos **optativos**: **id**, **form**, **substitutionGroup**, **nillable**, **abstract**, **final**
 - **id**: identificador único para el elemento.
 - **form**: formato del nombre del atributo.
 - **qualified** con el espacio de nombres como prefijo.
 - **unqualified** sin el espacio de nombres.
 - El valor por defecto es el del atributo **elementFormDefault** del componente **<xs:schema>**.
 - **substitutionGroup**: indica el nombre de otro elemento que puede ser sustituido por este elemento. Sólo se puede usar este atributo si el componente padre es **<xs:schema>**

xs:element

- **nillable**: especifica si puede aparecer el atributo de instancia **xsi:nil**, asociado al elemento en el documento instancia XML. Por defecto es **falso**, lo que significa que no se puede asignar a un elemento el atributo de instancia **xsi:nil**. (+adelante).
- **abstract**: indica si el elemento puede ser usado en un documento XML instancia. Si vale **cierto**, indica que el elemento **no** puede aparecer en una instancia.
- **final**: indica si el elemento se puede derivar de alguna manera por extensión (**extension**) o por restricción (**restriction**) o ambas (**#all**).

<xs:element
name="nombre del elemento"
type="tipo de datos"
ref="declaración (aplazada) del elemento global"
id="identificador"
form="formato" <!--qualified o unqualified-->
minOccurs="número mínimo de veces"
maxOccurs="máximo número de veces"
default="valor por defecto"
fixed="valor fijo">

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

```
<xs:element name="semáforo" type="xs:string" default="verde"/>
```

```
<xs:element name="semáforo" type="xs:string" fixed="rojo"/>
```

Componentes básicos de un esquema

- xs:schema
- xs:element
- **xs:attribute**

xs:attribute

- Es el componente de declaración de atributo y representa la existencia de un atributo de un elemento en el documento XML.

```
<xs:attribute  
  name="nombre del elemento"  
  type="tipo global de datos"  
  ref="declaración del elemento global"  
  form="cualificación" <!--qualified o unqualified -->  
  id="identificador"  
  default="valor por defecto"  
  fixed="valor fijo"  
  use="uso" <!-- prohibited, optional o required -->  
>
```

xs:attribute

- Atributos **optativos principales**:
 - **name**: nombre del atributo. Este atributo no puede aparecer simultáneamente con **ref** .
 - **ref**: referencia a la **descripción** del atributo que se encuentra en otro lugar del esquema. Si aparece este atributo no aparecerán los atributos **type**, ni **form**, ni podrá contener un componente **<xs:simpleType>**.

xs:attribute

- **type**: tipo del elemento.
- **use**: indica si la existencia del atributo es opcional, obligatoria o prohibida (no atributo).
 - Posibles valores: **optional**, **required**, **prohibited**.
- **default**: valor que tomará el atributo al ser procesado por alguna aplicación (habitualmente un analizador de XML o un navegador) cuando en el documento XML no había recibido ningún valor. No puede aparecer simultáneamente con **fixed**.
- **fixed**: único valor que puede contener el atributo en el documento XML. No puede aparecer simultáneamente con **default**.

xs:attribute

- Otros atributos **optativos**:
 - **id**: identificador único para el atributo.
 - **form**: formato del nombre del atributo.
 - Posibles valores: **unqualified** y **qualified**. El valor por defecto es el del atributo **attributeFormDefault** del componente **<xs:schema>**

```
<xs:attribute name="moneda" type="xs:string" default="Euro"/>
```

```
<xs:attribute name="unidad" type="xs:string" fixed="Minutos"/>
```

```
<xs:attribute name="idEmple" type="xs:positiveInteger" use="required"/>
```

Tipos de datos

- **Predefinidos**
 - Integrados en la especificación de los esquemas XML.
- **Construidos o derivados**
 - Generados por el usuario basándose en un tipo predefinido o en un tipo previamente contruidos.

Tipos de datos. Predefinidos

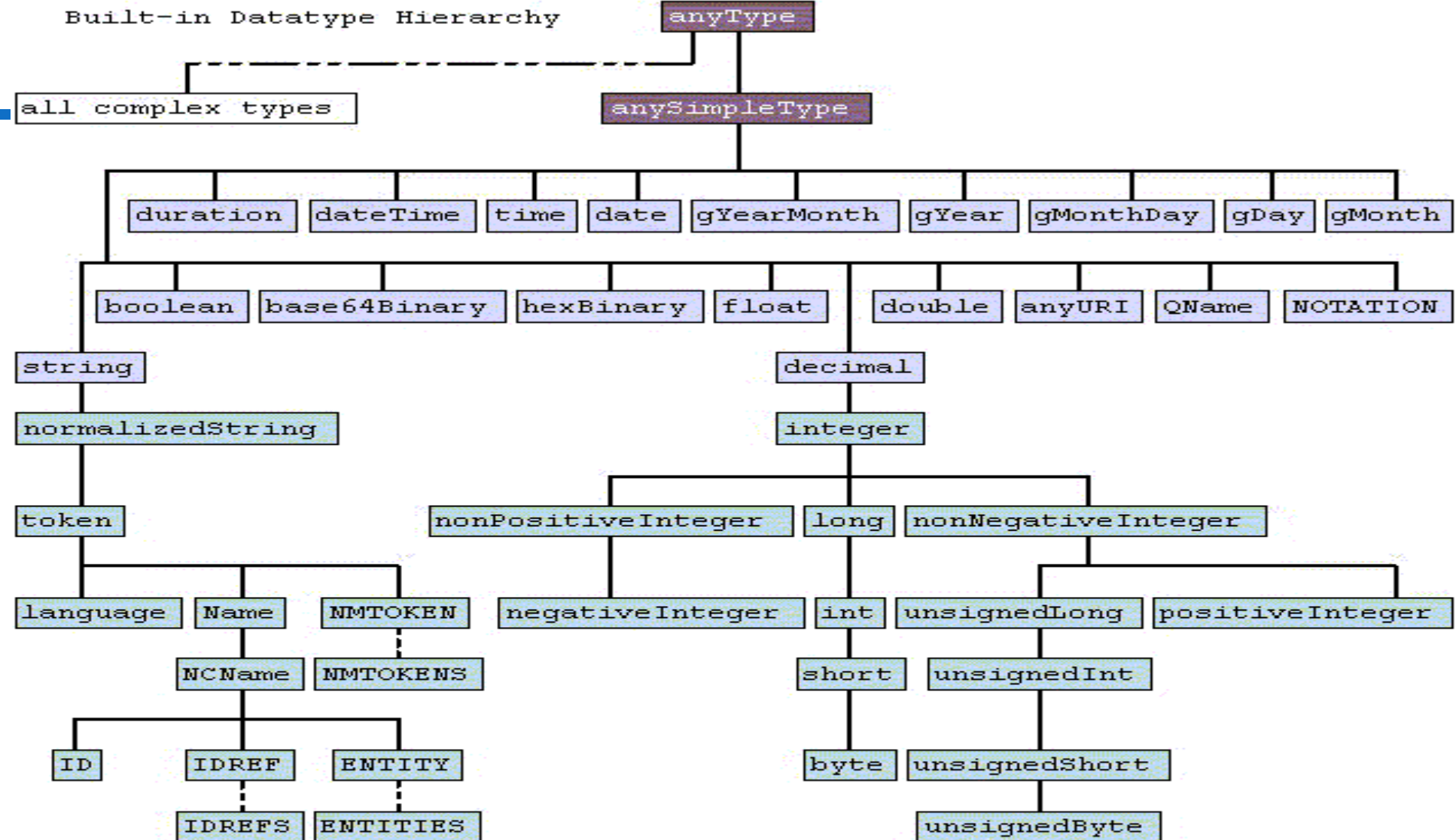
- 44 tipos predefinidos
- Organización jerárquica
- Se especifican en el atributo **type** de **element** o **attribute**








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

Tipos de datos. Predefinidos

- Tipos predefinidos especiales
 - Raíz → **xs:anyType**, del que derivan los demás y se asocia a cualquier elemento sin tipo definido (predefinidos y complejos).
 - **xs:anySimpleType**, cualquier tipo simple.
 - Son demasiado genéricos, se debería especificar más el tipo.

Built-in Datatype Hierarchy



- | | | | |
|---|--------------------------|---|-------------------------------------|
|  | ur types |  | derived by restriction |
|  | built-in primitive types |  | derived by list |
|  | built-in derived types |  | derived by extension or restriction |
|  | complex types | | |

Tipos de datos. Predefinidos

- Predefinidos
 - Primitivos (19)
 - Descienden directamente de **xs:anySimpleType**.
 - No primitivos (25)
 - Derivan de alguno de los primitivos.
- Se agrupan en cinco categorías
 - Numéricos (16)
 - De fecha y hora (9)
 - De texto (16)
 - Binarios (2)
 - Booleanos (1)

Tipo de datos Numéricos	Descripción
float	Número en punto flotante de precisión simple (32 bits)
double	Número en punto flotante de precisión doble (64 bits)
decimal	Números reales que pueden ser representados como $i * 10^{-n}$
integer	Números enteros, de $-\infty$ a $+\infty$
nonPositiveInteger	Números enteros negativos más el 0
negativeInteger	Números enteros menores que 0
nonNegativeInteger	Números enteros positivos más el 0
positiveInteger	Números enteros mayores que 0
unsignedLong	Números enteros positivos (64 bits)
unsignedInt	Números enteros positivos (32 bits)
unsignedShort	Números enteros positivos (16 bits)
unsignedByte	Números enteros positivos (8 bits)
long	Números enteros (64 bits)
int	Números enteros (32 bits)
short	Números enteros (16 bits)
byte	Números enteros (8 bits)

Tipo de datos Fecha y Hora	Descripción
duration	Duración en años + meses + días + horas + minutos + segundos. PnYnMnDThHmMsS.
dateTime	Fecha y hora, en formato aaaa-mm-dd T hh:mm:ss
date	Solamente fecha en formato aaaa-mm-dd
time	Solamente la hora, en formato hh:mm:ss
gDay	Sólo el día, en formato ---dd (g →Gregoriano)
gMonth	Sólo el mes.--mm
gYear	Solamente el año, en formato yyyy
gYearMonth	Sólo el año y el mes, en formato yyyy-mm.
gMonthDay	Sólo el mes y el año, en formato --mm-dd.

Ejemplos	
P1Y	Un año
P1Y2M	Un año y dos meses
P1Y2M3D	Un año, dos meses y 3 días
P3D	Tres días
P1Y2M3DT12H30M40.5S	1 año, 2 meses, 3 días, 12 horas 30 minutos y 40 seg y medio
PT12H30M40.5S	12 horas 30 minutos y 40 seg y medio

Patrón de la norma ISO 8601 ➔	-?\d{4}-\d{2}\d{2} T\d{2}:\d{2}:\d{2} (\\d+)?([+-]\d{2}:\d{2}) Z)?
-?	Un signo negativo (opcional). Para representar fechas anteriores al año 0.
\d{4}-\d{2}\d{2}	La parte de la fecha (obligatoria). Equivale a yyyy-mm-dd
\d{2}:\d{2}:\d{2}	La parte de la hora (obligatoria). Equivale a hh:mm:ss
T	Indicador de hora local
(\\d+)?	La parte decimal de la fracción de segundo (opcional)
([+-]\d{2}:\d{2})(Z)?	La parte de la zona horaria (opcional) La Z permite expresar las horas en formato UTC (Universal Time Coordinated - Tiempo Universal Coordinado), anteriormente GMT (Greenwich Meridian Time – Hora del Meridiano de Greenwich)

❑ **Ejemplo ➔ 3:40 PM**

❑ En la costa este de Estados Unidos, con 5 horas por detrás del UTC.

❑ 15:40:00-05:00

❑ En la zona central de Australia, con 9 horas por delante del UTC

❑ 15:40:00+09:30

Ejemplo

```
<!-- Se usa maxInclusive para indicar un límite final de fecha -->  
<xs:simpleType name="TipoFechaEntregaPractica">  
  <xs:restriction base="xs:date">  
    <xs:maxInclusive value="2009-12-31"/>  
  </xs:restriction>  
</xs:simpleType>
```

**Restricción que
permite generar nuevos tipos de
datos a partir de uno existente,
limitando su rango de valores
posibles.**

Ejemplo

<!-- Se usa minInclusive y maxInclusive para indicar límites iniciales y finales de horas -->

```
<xs:simpleType name="TipoHorarioEmbarque">  
  <xs:restriction base="xs:time">  
    <xs:minInclusive value="05:00:00"/>  
    <xs:maxInclusive value="05:30:00"/>  
  </xs:restriction>  
</xs:simpleType>
```

**Hora en la que un pasajero ha
embarcado en
un determinado vuelo (hora de
comienzo y fin del embarque)**

Tipo de datos de texto	Descripción
string	Cadenas de texto
normalizedString	Cadenas de texto en las que se convierten los caracteres tabulador, nueva línea, y retorno de carro en espacios simples.
token	Cadenas de texto sin los caracteres tabulador, ni nueva línea, ni retorno de carro, sin espacios por delante o por detrás, y con espacios simples en su interior.
language	Valores válidos para xml:lang (según la especificación de XML). en-US
NMTOKEN	Tipo de datos para atributo según XML 1.0, compatible con DTD
NMTOKENS	Lista separada por espacios de NMTOKEN, compatible con DTD. SP FR PR IT
Name	Tipo de nombre según XML 1.0. horaSalida
QName	Nombre cualificado de espacio de nombres XML. cliente:nombre
NCName	QName sin el prefijo ni los dos puntos. nombre
anyURI	Cualquier URI. http://www.w3c.com
ID	Tipo de datos para atributo según XML 1.0, compatible con DTD. Han de ser valores únicos en el documento XML.
IDREF	Tipo de datos para atributo según XML 1.0, compatible con DTD
IDREFS	Lista separada por espacios de IDREF, compatible con DTD
ENTITY	Tipo de datos para atributo en XML 1.0, compatible con DTD
ENTITIES	Lista separada por espacios de ENTITY, compatible con DTD
NOTATION	Tipo de datos para atributo en XML 1.0, compatible con DTD

Tipo de datos Binarios	Descripción
hexBinary	Secuencia de dígitos hexadecimales (0..9, A, B, C, D, E, F)
base64Binary	Secuencia de dígitos en base 64

Tipo de datos Boolean	Descripción
boolean	Puede tener 4 valores: 0, 1, true y false

Tipos de datos simples vs. complejos

Tipos de datos simples. xs:simpleType	Tipos de datos complejos. xs:complexType
Los tipos de datos predefinidos son simples	Asignar a elementos que tengan elementos descendientes y/o atributos .
Representan valores atómicos , no listas	Contiene contenido complejo → elementos descendientes
Asignación a elementos y atributos	Puede tener contenido simple . Diferencia de los tipos de datos simples precisamente en que tiene atributos .
Se pueden construir, derivando de un tipo base predefinido al que se ha introducen restricciones	Pueden no tener contenido
	Pueden tener contenido mixto : combinación de contenido textual con elementos descendientes

Definición de datos simples

- **xs:simpleType**
- xs:restriction
- Facetas para restringir rangos de valores
- xs:pattern
- Más sobre derivación de tipos simples: uniones y listas

xs:simpleType

- **<xs:simpleType>** → definición de tipos simples
 - Es el componente de definición de tipos simples. Atributos **optativos** principales:
 - **name**: nombre del tipo. Este atributo es obligatorio si el componente es hijo de **<xs:schema>**, de lo contrario no está permitido.
 - **id**: identificador único para el componente.

Definición de datos simples

- `xs:simpleType`
- **`xs:restriction`**
- Facetas para restringir rangos de valores
- `xs:pattern`
- Más sobre derivación de tipos simples: uniones y listas

xs:restriction

- **<xs:restriction>** → limitar el rango de valores
 - Atributos **obligatorios**:
 - **base**: nombre del tipo base a partir del cual se construirá el nuevo tipo, sea predefinido o construido.
 - Atributos **optativos principales**:
 - **id**: identificador único para el componente.

Ejemplo. xs:simpleType y xs:restriction

```
<xs:element name="ingresosAnuales" type="xs:float"/>
```

1

```
<xs:element name="ingresosAnuales">  
  <xs:simpleType>  
    <xs:restriction base="xs:float" />  
  </xs:simpleType>  
</xs:element>
```

2

```
<xs:simpleType name="TipoSIngresosAnuales">  
  <xs:restriction base="xs:float" />  
</xs:simpleType>  
<xs:element name="ingresosAnuales" type="TipoSIngresosAnuales"/>
```

3

Ejemplo. xs:simpleType y xs:restriction

- Ejemplo:
 - Se declara un tipo simple que se utilizará para elementos que sean una contraseña. Será de tipo base **xs:string**, con unas facetas que fuercen su tamaño mínimo a 6 y máximo a 12.

```
<xs:simpleType name="TipoContrasenia">  
  <xs:restriction base="xs:string" >  
    <xs:minLength value="6" />  
    <xs:maxLength value="12" />  
  </xs:restriction>  
</xs:simpleType>
```

Se construye un nuevo
tipo de datos

Faceta: se obliga a que la
longitud mínima del elemento
sea de 6

Faceta: se obliga a que la
longitud máxima del elemento
sea de 12

Ejemplo. xs:simpleType y xs:restriction

```
<xs:element name="clave" type="TipoContraseña">
```

```
<clave>claveValida!</clave>
```

```
<clave>mala</clave>
```

Es válida

No es válida

Faceta	Uso	Tipos de datos donde se usa
xs:minInclusive	Especifica el límite inferior del rango de valores aceptable. El propio valor está incluido.	Numéricos, fecha/horas
xs:maxInclusive	Especifica el límite superior del rango de valores aceptable. El propio valor está incluido.	Numéricos, fecha/horas
xs:minExclusive	Especifica el límite inferior del rango de valores aceptable. El propio valor no está incluido.	Numéricos, fecha/horas
xs:maxExclusive	Especifica el límite superior el rango de valores aceptable. El propio valor no está incluido.	Numéricos, fecha/horas
xs:enumeration	Especifica una lista de valores aceptables.	Todos
xs:pattern	Especifica un patrón o expresión regular que deben cumplir los valores válidos.	Texto
xs:whiteSpace	Especifica cómo se tratan los espacios en blanco, entendiéndose como tales los saltos de línea, tabuladores y los propios espacios. Sus posibles valores son preserve (conserva), replace (sustituye por blancos) y collapse (elimina excepto los blancos que los sustituye por un blanco).	Texto
xs:length	Especifica el número exacto de caracteres (o elementos de una lista) permitidos. Ha de ser mayor o igual que 0.	Texto
xs:minLength	Específica el mínimo número de caracteres (o elementos de una lista) permitidos. Ha de ser mayor o igual que 0.	Texto
xs:maxLength	Especifica el máximo número de caracteres (o elementos de una lista) permitidos. Ha de ser mayor o igual que 0.	Texto
xs:fractionDigits	Especifica el máximo número de posiciones decimales permitidas en números reales. Ha de ser mayor o igual que 0.	Numéricos con parte decimal
xs:totalDigits	Especifica el número exacto de dígitos permitidos en números. Ha de ser mayor que 0.	Numéricos

Ejemplo con faceta

- El elemento **<edadLaboral>**
 - Entero no negativo
 - Valor mínimo=16
 - Máximo = 70
 - [16, 70)


```
<xs:element name="edadLaboral">
  <xs:simpleType>
    <xs:restriction base="xs:nonNegativeInteger">
      <xs:minInclusive value="16"/>
      <xs:maxExclusive value="70"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

```
<xs:simpleType name="TipoEdadLaboral">
  <xs:restriction base="xs:nonNegativeInteger">
    <xs:minInclusive value="16"/>
    <xs:maxExclusive value="70"/>
  </xs:restriction>
</xs:simpleType>

<xs:element name="edadLaboral" type="TipoEdadLaboral"/>
```

xs:simpleType. Ejemplo

- Se quiere definir un tipo de dato simple, TipoEstaciones, basado en el tipo predefinido **xs:token**, y que solamente permita como valores los nombres de las cuatro estaciones.
- Se quiere un tipo de datos, TipoCantidad, basado en **xs:decimal**, que permita números con 2 dígitos decimales y 11 dígitos totales. El rango de valores implícito sería desde -999.999.999,99 a 999.999.999,99.

```
< xs:simpleType name="TipoEstaciones">
  <xs:restriction base="xs:token">
    <xs:enumeration value="Primavera" />
    <xs:enumeration value="Verano" />
    <xs:enumeration value="Otoño" />
    <xs:enumeration value="Invierno" />
  </xs:restriction>
</xs:simpleType>
```

Cadena sin
espacios
iniciales y/o
finales

```
<xs:simpleType name="TipoCantidad">
  <xs:restriction base="xs:decimal">
    <xs:totalDigits value="11" />
    <xs:fractionDigits value="2"/>
  </xs:restriction>
</xs:simpleType>
```

Ejemplo de faceta

- Se define un tipo basado en **xs:string**, usado para registrar direcciones en el que se quiere que todos los caracteres de espaciado (espacio, tabulador, nueva línea y retomo de carro) se colapsen, lo que significa que si estos caracteres aparecen al principio o final de la cadena se eliminan, y si lo hacen en su interior, formando una hilera, se sustituyen en un solo espacio. La definición de este tipo coincide con el tipo predefinido **xs:token**, derivado de **xs:string**.

```
<xs:simpleType name="TipoDireccionFormateada">  
  <xs:restriction base="xs:string" >  
    <xs:whiteSpace value="collapse"/>  
  </xs:restriction>  
</xs:simpleType>
```

**Sustituye los
blancos por
un espacio
en blanco**

Ejercicio

- Define diferentes tipos simples a partir de las siguientes especificaciones. A continuación crea un esquema en el que sólo haya un elemento al que se le asigne un tipo simple de los definidos. Por último, escribe un documento instancia XML que sea válido con respecto a ese esquema. Repítelo para los distintos tipos definidos:
- a. Un número real con tres decimales que represente las temperaturas posibles en la Tierra suponiendo que van desde -75° a 75° , ambas inclusive.
- b. Un **xs:token** que sólo pueda valer las siglas de los países vecinos de España, incluyendo a la propia España: ES, PR, FR, AN.
- c. Un número real que represente salarios, con 5 dígitos enteros y 2 decimales.

Definición de datos simples

- `xs:simpleType`
- `xs:restriction`
- Facetas para restringir rangos de valores
- **`xs:pattern`**
- Más sobre derivación de tipos simples: uniones y listas

xs:pattern

- Representa un patrón o expresión regular que debe de cumplir la cadena de texto a la que se aplique.
- En las facetas se usan los **cuantificadores**
- Anteponerle el **carácter de escape** \ en las expresiones regulares: {, }, [,], (,), ?, *, +, -, |, ^, ., \

Patrón	Significado	Ejemplo
.	Cualquier carácter	<i>;</i>
\w	Cualquier letra	<i>M</i>
\d	Un dígito	<i>7</i>
\D	Cualquier carácter no dígito	<i>i</i>
\s	Cualquier carácter de espaciado (tabulador, espacio...)	
\S	Cualquier carácter de no espaciado	<i>C</i>
{n}	n dígitos exactamente (Ej. cuatro dígitos exactamente)	<i>3856</i>
{n,m}	De n a m dígitos (Ej. de dos o tres dígitos)	<i>91</i>
{n,}	n o más dígitos (Ej. Tres o más dígitos)	<i>3500</i>
[xyz]	Uno de los caracteres x, y o z (en minúscula)	<i>y</i>
[A-Z]	Uno de los caracteres de la A a la Z (en mayúscula)	
[^abc]	Negación de un grupo de caracteres	<i>D</i>
[F-J-[H]]	Sustracción de un carácter de un rango	<i>G</i>
(a b)	Alternativa entre dos expresiones	<i>b</i>
b?	Sucesión de 0 o una ocurrencias de una cadena	<i>B</i>
1*	Sucesión de 0 o más ocurrencias de una cadena	<i>1 1 1</i>
(cd)+	Sucesión de 1 o más ocurrencias de una cadena	<i>cdcd</i>

Ejemplos xs:pattern

- La expresión regular para una cadena que contenga las letras a o b o el carácter + sería:

[a b \+]

- Un número de teléfono que se quiere se represente como 3 dígitos, un punto, 3 dígitos, un punto, 3 dígitos y un punto. Por ejemplo 912.345.678:

```
<xs:simpleType name="TipoTelefono">  
  <xs:restriction base="xs:string">  
    <xs:pattern value="\d{3}\.\d{3}\.\d{3}"/>  
  </xs:restriction>  
</xs:simpleType>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="Libro">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Título" type="xsd:string"/>
        <xsd:element name="Autores"
          type="xsd:string" maxOccurs="10"/>
        <xsd:element name="Editorial" type="xsd:string"/>
      </xsd:sequence>
      <xsd:attribute name="precio" type="xsd:double"/>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Todo esquema debe comenzar con una declaración XML

Declaración del elemento esquema: elemento raíz, atributo xmlns (espacio de nombres)

El elemento raíz se llama Libro

El elemento raíz se llama Libro, con tres elementos hijo y un atributo

Hijos: "Título", "Editorial", que deben aparecer ambos una vez, y "Autores" que puede aparecer de una a diez veces

Atributo: "precio" y es de tipo "double".

Sequence indica que los elementos deben aparecer en ese orden y son de tipo string

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Libro xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="libro.xsd" precio="20">
    <Título>Fundamentos de XML Schema</Título>
    <Autores>Allen Wyke</Autores>
    <Autores>Andrew Watt</Autores>
    <Editorial>Wiley</Editorial>
</Libro>
```

Definición de datos simples

- `xs:simpleType`
- `xs:restriction`
- Facetas para restringir rangos de valores
- `xs:pattern`
- **Más sobre derivación de tipos simples: uniones y listas**

Uniones y listas

- **<xs:union>**: tipo de datos creado a partir de una colección de tipos de datos base.
- Válido para al menos uno de los tipos de datos que forman la unión. Se construye con el componente.
- **Ejemplo:** Tallas de ropa se pueden identificar por números (38, 40, 42) o por letras (S, M, L).

```
<xs:element name="talla" >
  <xs:simpleType>
    <xs:union memberTypes="TipoTallaNumerica TipoTallaTextual" />
  </xs:simpleType>
</xs:element>
```

```
<xs:simpleType name="TipoTallaNumerica" >
  <xs:restriction base="xs:positiveInteger">
    <xs:enumeration value="38"/>
    <xs:enumeration value="40"/>
    <xs:enumeration value="42"/>
  </xs:restriction>
</xs:simpleType>
```

```
<xs:simpleType name="TipoTallaTextual">
  <xs:restriction base="xs:string">
    <xs:enumeration value="S"/>
    <xs:enumeration value="M"/>
    <xs:enumeration value="L"/>
  </xs:restriction>
</xs:simpleType>
```

Uniones y listas

- **<xs:list>**: es un tipo de datos compuesto por listas de valores de un tipo de datos base. La lista de valores debe aparecer separada por espacios. Se construye con el componente.
- **Ejemplo:** Se define un tipo TipoListaNumerosBingo como una lista de valores enteros positivos que empieza en el 1 y acaba en el 90.


```
<xs:simpleType name="TipoListaNumerosBingo">
  <xs:list>
    <xs:restriction base="xs:positiveInteger">
      <xs:maxInclusive value="90"/>
    </xs:restriction>
  </xs:list>
</xs:simpleType>

<xs:element name="numerosGanan" type="TipoListaNumerosBingo"/>
```

```
<numerosGanan>2 7 17 19 20 21 34 43 44 50 51 60 73 77 80</numerosGanan>
```

Definición de datos complejos

- **Contenido simple/contenido complejo**
- Modelos de contenido:
 - **xs:sequence**
 - **xs:choice**
 - **xs:all**
- **xs:complexType**

Contenido simple/contenido complejo

- **<xs:complexType>** elemento con atributos y/o descendientes.
- **Contenido.** Lo que va entre sus etiquetas de apertura y de cierre
 - **Contenido simple** (**xs:simpleContent**) contenido textual, sin elementos descendientes.
 - **Contenido complejo** (**xs:complexContent**): tiene elementos descendientes. Puede tener o no contenido textual.

Definición de tipos de datos complejos

- Modelos de contenido.
 - **Secuencia** `<xs:sequence>`. Uno detrás de otro con minOccurs..maxOccurs (1 por defecto).
 - **Alternativa** `<xs:choice>`. Sólo se elije uno con minOccurs..maxOccurs (1 por defecto).
 - **Todo** `<xs:all>`. Los elementos puede aparecer 0 o 1 vez configurable con los atributos minOccurs y maxOccurs en cualquier orden (ambos valen 1 por defecto).

```
<xs:sequence>
```

```
  <xs:element name="emisor" type="xs:string"/>
```

```
  <xs:element name="receptor" type="xs:string"/>
```

```
  <xs:element name="contenido" type="xs:string"/>
```

```
</xs:sequence>
```

Los tres y en
ese orden

```
<xs:choice>
```

```
  <xs:element name="prologo" type="xs:string"/>
```

```
  <xs:element name="prefacio" type="xs:string"/>
```

```
  <xs:element name="introducción" type="xs:string"/>
```

```
</xs:choice>
```

Se elije uno de
los tres

```
<xs:all>
```

```
  <xs:element name="alfa" type="xs:string"/>
```

```
  <xs:element name="omega" type="xs:string"/>
```

```
</xs:all>
```

Orden
indiferente

xs:complexType

- **xs:complexType**: definición de tipos complejos.
 - Atributos optativos principales
 - **name**
 - Nombre del tipo.
 - Obligatorio si el componente es hijo de <xs:schema>, de lo contrario no está permitido.
 - **mixed**
 - Indica si se intercala contenido textual con los elementos descendientes:
 - Posibles valores: false, true.
 - **id**
 - Identificador único para el componente.
 - Posibles valores: false, true.

xs:complexType

- **abstract:**

- Indica si se puede usar directamente como tipo de un elemento de un documento instancia XML. Si vale cierto, significa que no puede usarse directamente y el tipo debe derivarse para generar otro tipo que sí se pueda usar.

- **final:**

- indica si el tipo puede ser derivable por extensión por restricción o por ambas.
- Posibles valores: **extensión, restriction, #all.**

```
<xs:element name="persona">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="primerApellido" type="xs:string" />
      <xs:element name="segundoApellido" type="xs:string" />
      <xs:element name="fechaNacimiento" type="xs:date" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

```
<xs:complexType name="TipoPersona">
  <xs:sequence>
    <xs:element name="primerApellido" type="xs:string" />
    <xs:element name="segundoApellido" type="xs:string" />
    <xs:element name="fechaNacimiento" type="xs:date" />
  </xs:sequence>
</xs:complexType>

<xs:element name="persona" type="TipoPersona" />
```


Elementos vacíos

- Opciones.
 - Tipo simple derivado de **xs:string** con longitud 0.
 - Tipo complejo sin contenido, sin elementos descendientes.

```
<xs:element name="br">  
  <xs:complexType />  
</xs:element>
```

```
<xs:element name="br">  
  <xs:complexType>  
    <xs:complexContent>  
      <xs:restriction base="xs:anyType">  
      </xs:complexContent>  
    </xs:complexType>  
  </xs:element>
```

Se desaconseja

Diferentes declaraciones de elementos

Tipo	Contenido	Elementos Descendientes	Atributos	Contenido textual
Simple	Simple			✓
Complejo	Simple		✓	✓
Complejo	Complejo			
Complejo	Complejo		✓	
Complejo	Complejo	✓		
Complejo	Complejo	✓	✓	
Complejo	Complejo Mixto	✓		✓
Complejo	Complejo Mixto	✓	✓	✓

Opciones

- **Declaración de elementos con contenido textual y atributos**
- Declaración de elementos sólo con atributos
- Declaración de elementos sólo con elementos descendientes
- Declaración de elementos con atributos y elementos descendientes
- Declaración de elementos con contenido textual y elementos descendientes
- Declaración de elementos con atributos, elementos descendientes y contenido textual
- Extensión de un tipo complejo

Declaración de elementos con contenido textual y atributos

- **Tipo de datos complejo con contenido simple** que contenga algún atributo con **<xs:extension>**

```
<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>
```

Indica que el elemento complejo no debe tener elementos descendientes

Permite la extensión de atributos

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

Opciones

- Declaración de elementos con contenido textual y atributos
- **Declaración de elementos sólo con atributos**
- Declaración de elementos sólo con elementos descendientes
- Declaración de elementos con atributos y elementos descendientes
- Declaración de elementos con contenido textual y elementos descendientes
- Declaración de elementos con atributos, elementos descendientes y contenido textual
- Extensión de un tipo complejo

Declaración de elementos sólo con atributos

- Se realiza con tipos de datos complejo con contenido complejo.

```
<xs:element name="img">  
  <xs:complexType>  
    <xs:attribute name="src" type="xs:string"/>  
    <xs:attribute name="alt" type="xs:string"/>  
  </xs:complexType>  
</xs:element>
```

```
<img src= "amanecer. jpg" alt="Amanecer, sol naciente" />
```

Opciones

- Declaración de elementos con contenido textual y atributos
- Declaración de elementos sólo con atributos
- **Declaración de elementos sólo con elementos descendientes**
- Declaración de elementos con atributos y elementos descendientes
- Declaración de elementos con contenido textual y elementos descendientes
- Declaración de elementos con atributos, elementos descendientes y contenido textual
- Extensión de un tipo complejo

Declaración de elementos sólo con elementos descendientes

- Tipo de datos complejo con contenido complejo

```
<xs:element name="ul">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="li" type="xs:string" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

```
<ul>
  <li>Primer elemento</li>
  <li>Segundo elemento</li>
</ul>
```


Opciones

- Declaración de elementos con contenido textual y atributos
- Declaración de elementos sólo con atributos
- Declaración de elementos sólo con elementos descendientes
- **Declaración de elementos con atributos y elementos descendientes**
- Declaración de elementos con contenido textual y elementos descendientes
- Declaración de elementos con atributos, elementos descendientes y contenido textual
- Extensión de un tipo complejo

Declaración de elementos con atributos y elementos descendientes

```
<xs:element name="table">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="tr" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="border" type="xs:string"/>
  </xs:complexType>
</xs:element>

<xs:element name="tr">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="td" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Los atributos se definirán siempre al final, después de la secuencia de elementos descendientes

```
<table border="1 ">
  <tr>
    <td>Primera celda</td>
  </tr>
</table>
```

Opciones

- Declaración de elementos con contenido textual y atributos
- Declaración de elementos sólo con atributos
- Declaración de elementos sólo con elementos descendientes
- Declaración de elementos con atributos y elementos descendientes
- **Declaración de elementos con contenido textual y elementos descendientes**
- Declaración de elementos con atributos, elementos descendientes y contenido textual
- Extensión de un tipo complejo

Declaración de elementos con contenido textual y elementos descendientes

- Tipo de datos complejo con contenido complejo

```
<xs:element name="p">
  <xs:complexType mixed="true" >
    <xs:choice minOccurs="0" maxOccurs="unbounded" >
      <xs:element name="b" type="xs:string" />
      <xs:element name="i" type="xs:string" />
    </xs:choice>
  </xs:complexType>
</xs:element>
```

Indica que el contenido es mixto

Para permitir un número indefinido de ocurrencias del elemento o del elemento <i>, en el orden que sea.

```
<p>
```

El perro de San Roque no tiene rabo porque <i>Ramón Ramírez</i> se lo ha cortado.

```
</p>
```

Opciones

- Declaración de elementos con contenido textual y atributos
- Declaración de elementos sólo con atributos
- Declaración de elementos sólo con elementos descendientes
- Declaración de elementos con atributos y elementos descendientes
- Declaración de elementos con contenido textual y elementos descendientes
- **Declaración de elementos con atributos, elementos descendientes y contenido textual**
- Extensión de un tipo complejo

Declaración de elementos con atributos, elementos descendientes y contenido textual

- Tipo de datos complejo con contenido complejo que contenga elementos atributos y contenido textual.

```
<xs:element name="form">
  <xs:complexType mixed="true">
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element ref="input"/>
    </xs:choice>
    <xs:attribute name="name" type="xs:string"/>
    <xs:attribute name="method" type="xs:string"/>
  </xs:complexType>
</xs:element>
<xs:element name="input">
  <xs:complexType>
    <xs:attribute name="type" type="xs:string" use="required"/>
    <xs:attribute name="name" type="xs:string" use="required"/>
  </xs:complexType>
</xs:element>
```

```
<form name="f1" method="post"> "
  Apellido: <input type="text" name="apellido" />
</form>
```

Opciones

- Declaración de elementos con contenido textual y atributos
- Declaración de elementos sólo con atributos
- Declaración de elementos sólo con elementos descendientes
- Declaración de elementos con atributos y elementos descendientes
- Declaración de elementos con contenido textual y elementos descendientes
- Declaración de elementos con atributos, elementos descendientes y contenido textual
- **Extensión de un tipo complejo**

Extensión de un tipo complejo

- **Herencia** en la programación orientada a objeto: se pueden añadir nuevos elementos y atributos a tipos complejos ya existentes.

```
<xs:complexType name="TipoAlumno">  
  <xs:sequence>  
    <xs:element name="Apellido" type="xs:string" />  
    <xs:element name="Ciclo" type="xs:string" />  
  </xs:sequence>  
</xs:complexType>
```

```
<xs:complexType name="TipoAlumnoExtendido">  
  <xs:complexContent>  
    <xs:extension base="TipoAlumno">  
      ...  
    </xs:extension>  
  </xs:complexContent>  
</xs:complexType>
```


Extensión de un tipo complejo

- Elemento **<Curso>** y atributo Matrícula ubicados después de la definición de secuencia, alternativa u otros elementos hijos

```
<xs:complexType name="TipoAlumnoExtendido">
  <xs:complexContent>
    <xs:extension base="TipoAlumno">
      <xs:sequence>
        <xs:element ref="Curso" />
      </xs:sequence>
      <xs:attribute name="Matrícula" type="xs:positiveInteger" use="required" />
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<xs:element name="Curso" type="xs:positiveInteger" />
```

Modelos de diseño de esquemas XML

- **1 Diseño anidado o de muñecas rusas**

- Anidan declaraciones unas dentro de otras.
- Describe cada elemento y atributo en el mismo lugar donde se declaran.
- Produce duplicidades.
- Esquemas son más cortos
- Lectura más compleja

- **2 Diseño plano**

- Declaración de elementos y atributos con referencia a una definición en otro lugar del documento.

Modelos de diseño de esquemas XML

- **3 Diseño con tipos con nombre reutilizables**
 - Definición en tipos de datos simples o complejos (plantillas, como las clases en la programación orientada a objeto).
 - Declaración de elementos y atributos indicando alguno de los tipos.

Grupos de Elementos

- Define una declaración de grupo:
- Ejemplo:

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

```
<xs:group name="grupopersona">  
  <xs:sequence>  
    <xs:element name="nombre" type="xs:string"/>  
    <xs:element name="apellido" type="xs:string"/>  
    <xs:element name="fnac" type="xs:date"/>  
  </xs:sequence>  
</xs:group>
```

Grupos de Elementos

- Referencia a un grupo

```
<xs:group name="grupopersona">
  <xs:sequence>
    <xs:element name="nombre" type="xs:string"/>
    <xs:element name="apellido" type="xs:string"/>
    <xs:element name="fnac" type="xs:date"/>
  </xs:sequence>
</xs:group>

<xs:element name="persona" type="infopersona"/>

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

Grupos de atributos

- Definición de grupo de atributos

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

```
<xs:attributeGroup name="grupoatpersona">  
  <xs:attribute name="nombre" type="xs:string"/>  
  <xs:attribute name="apellido" type="xs:string"/>  
  <xs:attribute name="fnac" type="xs:date"/>  
</xs:attributeGroup>
```

Referencia

```
<xs:attributeGroup name="grupoatpersona">
  <xs:attribute name="nombre" type="xs:string"/>
  <xs:attribute name="apellido" type="xs:string"/>
  <xs:attribute name="fnac" type="xs:date"/>
</xs:attributeGroup>

<xs:element name="persona">
  <xs:complexType>
    <xs:attributeGroup ref="grupoatpersona "/>
  </xs:complexType>
</xs:element>
```

El elemento any

- Para extender en un documento XML con otros elementos que no hemos especificado en el esquema

```
<xs:element name="persona">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="nombre" type="xs:string"/>
      <xs:element name="apellido" type="xs:string"/>
      <xs:any minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```


El elemento any

- Podemos extender persona con hijos, por ejemplo

```
<xs:element name="persona">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="nombre" type="xs:string"/>
      <xs:element name="apellido" type="xs:string"/>
      <xs:any minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

- Si tengo este archivo “hijos.xsd”

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
<xs:element name="hijos">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="nombrehijo" type="xs:string"
maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<personas xmlns="http://www.microsoft.com"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://www.microsoft.com family.xsd
                              https://www.w3schools.com children.xsd">
  <persona>
    <nombre>Hege</nombre>
    <apellido>Refsnes</apellido>
    <hijos>
      <nombrehijo>Cecilie</nombrehijo>
    </hijos>
  </persona>
  <persona>
    <nombre>Stale</nombre>
    <apellido>Refsnes</apellido>
  </persona>
</personas>
```