Tema V. XML
III. XML Schemas

Desarrollo de Aplicaciones para Internet
Curso 20|21

Índice

Índice

Introducción

- Los *XML Schemas* permiten **definir tipos de documentos**.
- Si un documento cumple las definiciones de un XML schema se dice que es **válido** para dicho XML schema.
- Es una recomendación del W3C posterior a XML.
- También suelen llamarse XSDs o *schemas*.
- Existen otras alternativas similares como DTDs (W3C) o RelaxNG (OASIS).

Ventajas frente a DTDs

- Se definen con **XML**, mientras que los DTDs usan una sintaxis propia.
- Soportan totalmente la recomendación de ***namespaces***.
- Permiten validar el contenido de los elementos de texto en base a **tipos de datos** predefinidos o definidos por el usuario.
- Permiten crear modelos de contenido complejos y reutilizables más fácilmente.
- Permiten modelar conceptos de programación tales como la herencia de objetos o el reemplazo de tipos.

Desventajas frente a DTDs

- No permiten declarar entidades.
- No pueden ser embebidos en documentos XML.
- Las herramientas de programación para XML (p.ej. DOM, SAX, etc.) suelen incorporar características especiales para los tipos de datos de DTD.

Ejemplo

XML Schema

```
<?xml version="1.0"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
   xmlns:target="http://www.example.com/name"
   targetNamespace="http://www.example.com/name"
   elementFormDefault="qualified">
  <element name="name">
    <complexType>
      <sequence>
        <element name="first" type="string"/>
        <element name="middle" type="string"/>
        <element name="last" type="string"/>
      </sequence>
      <attribute name="title" type="string"/>
    </complexType>
  </element>
</schema>
```

Documento XML

```
<?xml version="1.0"?>
<name xmlns="http://www.example.com/name"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.example.com/name name5.xsd"
  title="Mr.">
   <first>John</first>
   <middle>Fitzgerald Johansen</middle>
   <last>Doe</last>
</name>
```

* Ejemplo tomado del libro “Beginning XML” - D. Hunter *et al.*

Índice

<schema>

- Es el **elemento raíz** de un XML Schema.
- Permite declarar información sobre el *namespace*.
- Admite atributos con declaraciones por defecto para el documento.
- Puede incluirse un atributo `version`, con el número de versión del *schema*.

<schema>

- **XML Schema Namespace**
  - Es necesario que el *schema* importe el vocabulario del *namespace* `http://www.w3.org/2001/XMLSchema`.
  - **Ejemplos**

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

<schema>

```
<?xml version="1.0"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
   xmlns:target="http://www.example.com/name"
   targetNamespace="http://www.example.com/name"
   elementFormDefault="qualified">
  <element name="name">
    <complexType>
      <sequence>
        <element name="first" type="string"/>
        <element name="middle" type="string"/>
        <element name="last" type="string"/>
      </sequence>
      <attribute name="title" type="string"/>
    </complexType>
  </element>
</schema>
```

<schema>

- **Elementos y atributos cualificados**
  - Si un elemento o atributo tiene un ***namespace* asociado** se considera que está **cualificado.**
  - **El atributo `xmlns` permite declarar e importar *namespaces* asociados**.
    - Si se declara solo se considera el ***namespace* por defecto**.
    - Si se declara seguido de dos puntos "`:`" y un nombre, el *namespace* se asociará a aquellos **atributos y elementos prefijados** con el nombre seguido de dos puntos "`:`".

<schema>

- **Elementos y atributos cualificados (cont.)**
  - Los atributos `elementFormDefault` y `attributeFormDefault` permiten definir como se deben cualificar elementos y atributos, respectivamente.
    - `unqualified`: indica que solo es necesario cualificar el elemento raíz o el elemento superior del mismo namespace.
    - `qualified`: indica que es necesario cualificar todos los elementos o atributos.

Qualified vs Unqualified

Qualified

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:libros xmlns:tns="http://www.esei.uvigo.es/dai/libros"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.esei.uvigo.es/dai/libros libros.xsd ">

   <tns:libro titulo="Ejemplos">
      <tns:capitulo numero="1">Ejemplo 1</tns:capitulo>
      <tns:capitulo numero="2">Ejemplo 2</tns:capitulo>
      <tns:capitulo numero="3">Ejemplo 3</tns:capitulo>
   </tns:libro>
</tns:libros>
```

Unqualified

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:libros xmlns:tns="http://www.esei.uvigo.es/dai/libros"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.esei.uvigo.es/dai/libros libros.xsd ">

   <libro titulo="Ejemplos">
      <capitulo numero="1">Ejemplo 1</capitulo>
      <capitulo numero="2">Ejemplo 2</capitulo>
      <capitulo numero="3">Ejemplo 3</capitulo>
   </libro>
</tns:libros>
```

<schema>

- ***Namespaces* objetivo**
  - El atributo `targetNamespace` indica el *namespace* al que pertenece el vocabulario definido.
  - Es opcional.
  - **Ejemplo:**

```
<schema
   xmlns="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://www.example.com/name"
   xmlns:target="http://www.example.com/name">
```

Índice

Elementos de un Esquema

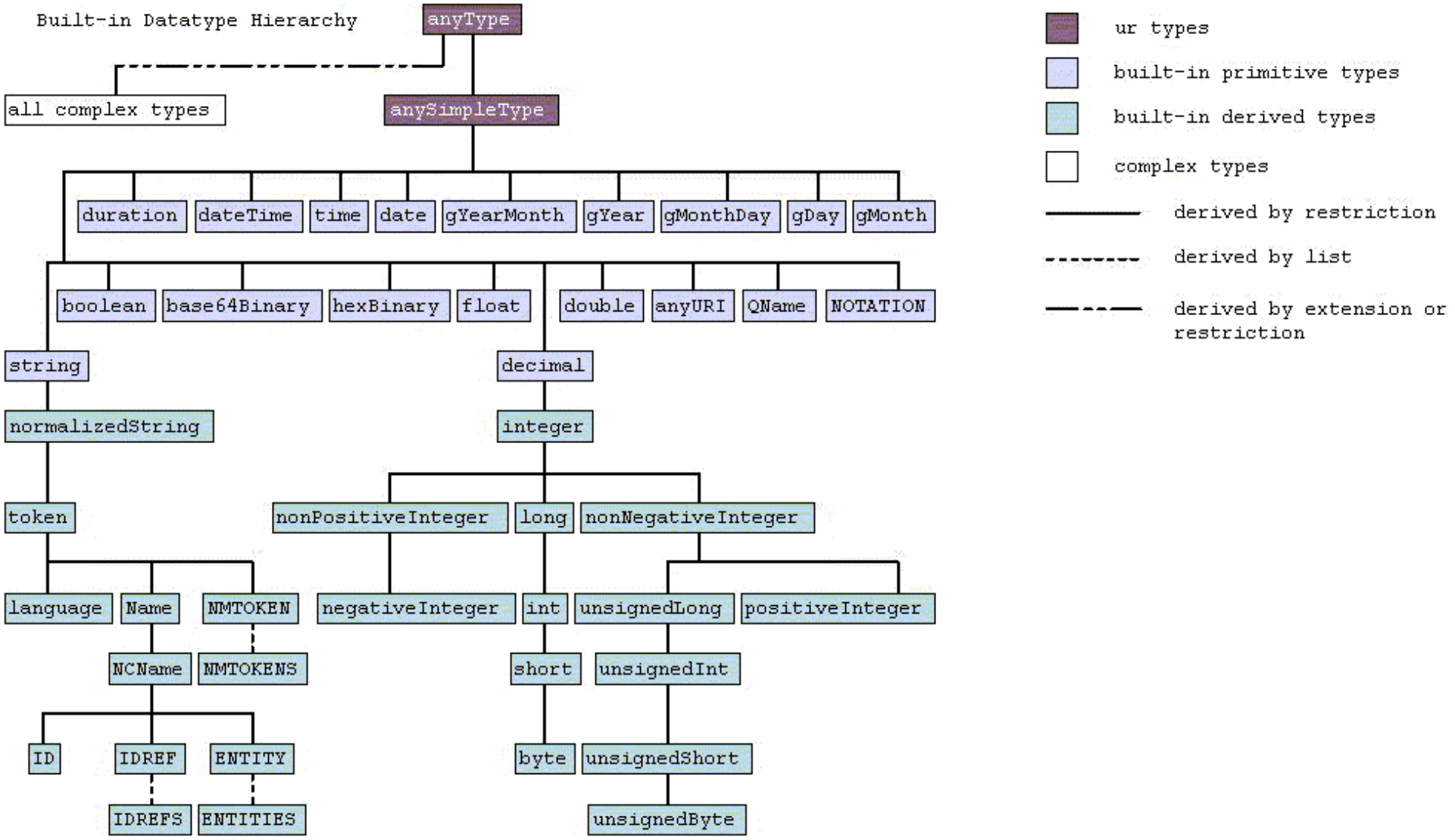
- En un *schema* nos podemos encontrar 5 tipos de componentes:
  - Directivas
  - Tipos (simples y complejos)
  - Elementos
  - Atributos
  - Grupos

Índice

Tipos de Datos

- Existen dos **categorías** de tipos de datos:
  - `complexType`: es la categoría de los **tipos de datos que pueden contener elementos**.
  - `simpleType`: es la categoría de los **tipos de datos que no contienen elementos** (`string`, `int`, `float`, etc.).
- Los tipos de dato pueden ser:
  - **Globales**: si no están asociados directamente a un elemento. Deben tener un nombre.
  - **Locales**: si están asociados a un elemento. No pueden tener nombre (son anónimos).

Jerarquía de Tipos de Datos



* Imagen: http://www.w3.org/TR/xmlschema-2/

Tipos Predefinidos

- Existen dos clases:
  - **Primitivos**: son los tipos base.
  - **Derivados**: creados a partir de los tipos primitivos, a los que añaden restricciones.
- Puede consultarse su definición en la URL `http://www.w3.org/TR/xmlschema-2/#<tipoDeDato>`
  - Por ejemplo:
    http://www.w3.org/TR/xmlschema-2/#int
    http://www.w3.org/TR/xmlschema-2/#string

Índice

<complexType>

- **Formato**

```
<complexType
    mixed="'true' o 'false'"
    name="Nombre del tipo">
```

- El atributo `mixed` indica si el contenido del elemento puede **incluir texto o no** en combinación con el resto de elementos.
- Si `complexType` está vacío, entonces los elementos de dicho tipo no podrán tener contenido (**elementos vacíos**).
- Puede contener **tres tipos de hijos**:
  - Modelos de contenido.
  - Declaraciones de atributos.
  - Tipos de contenido (`complexContent` y `simpleContent`).

<complexType>

- **Modelos de contenido**
  - Existen **cuatro** posibles modelos de contenido:
    - Declaración `<sequence>`
    - Declaración `<choice>`
    - Declaración `<all>`
    - Referencia a un `<group>` global.
  - Cada modelo de contenido puede contener:
    - Otros modelos de contenido.
    - Declaraciones de elementos.
    - Comodines para elementos.

<complexType>

- **Declaraciones `<sequence>`**
  - El elemento debe contener los elementos y/o modelos de contenido declarados en el mismo orden que son declarados.
  - **Formato:**

```
<sequence
   minOccurs="Entero no negativo"
   maxOccurs="Entero no negativo o
'unbounded'">
```

<complexType>

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://www.example.org/dai"
   xmlns:tns="http://www.example.org/dai"
   elementFormDefault="qualified">

   <element name="datosPersonales">
      <complexType>
         <sequence>
            <element name="datos" type="tns:datos"
               minOccurs="0" maxOccurs="unbounded" />
         </sequence>
      </complexType>
   </element>

   <complexType name="datos">
      <sequence>
         <element name="nombre" type="string" />
         <element name="apellidos" type="string" />
         <element name="nacimiento" type="date" />
      </sequence>
      <attribute name="dni" type="string" />
   </complexType>
</schema>
```

<complexType>

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:datosPersonales xmlns:tns="http://www.example.org/dai"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.example.org/dai
                       datosPersonales.xsd ">

   <tns:datos dni="12345678Z">
      <tns:nombre>Pedro</tns:nombre>
      <tns:apellidos>Pérez Pérez</tns:apellidos>
      <tns:nacimiento>1980-01-01</tns:nacimiento>
   </tns:datos>

   <tns:datos dni="87654321A">
      <tns:nombre>Ana</tns:nombre>
      <tns:apellidos>García García</tns:apellidos>
      <tns:nacimiento>1960-01-01</tns:nacimiento>
   </tns:datos>
</tns:datosPersonales>
```

<complexType>

- **Declaraciones `<choice>`**
  - El elemento solo puede contener uno de los elementos o modelos de contenido declarados.
  - **Formato:**

```
<choice
   minOccurs="Entero no negativo"
   maxOccurs="Entero no negativo o
'unbounded'">
```

<complexType>

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://www.example.org/dai"
   xmlns:tns="http://www.example.org/dai"
   elementFormDefault="unqualified">

   <element name="documentos">
      <complexType>
         <sequence>
            <element name="documento" type="tns:documento"
               minOccurs="0" maxOccurs="unbounded"/>
         </sequence>
      </complexType>
   </element>

   <complexType name="documento">
      <choice>
         <element name="dni" type="string"/>
         <element name="pasaporte" type="string"/>
         <element name="libroDeFamilia" type="string"/>
      </choice>
   </complexType>
</schema>
```

<complexType>

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:documentos xmlns:tns="http://www.example.org/dai"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.example.org/dai
                       documento.xsd ">

   <documento>
      <dni>12345678Z</dni>
   </documento>

   <documento>
      <pasaporte>11122233X</pasaporte>
   </documento>

   <documento>
      <libroDeFamilia>
         DNI Padre: 12345678Z
         DNI Madre: 87654321A
      </libroDeFamilia>
   </documento>

   <documento>
      <dni>87654321A</dni>
   </documento>
</tns:documentos>
```

<complexType>

- **Declaraciones `<all>`**
  - El elemento debe contener los elementos y/o modelos de contenido declarados en cualquier orden.
  - Tiene una serie de restricciones:
    - Debe aparecer como único contenido del tipo complejo asociado.
    - Solo pueden contener declaraciones `<element>`.
    - Su cardinalidad y la de los elementos contenidos es 0 o 1.
  - **Formato:**

```
<all
   minOccurs="'0' o '1'"
   maxOccurs="1">
```

<complexType>

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://www.example.org/dai"
   xmlns:tns="http://www.example.org/dai"
   elementFormDefault="qualified">

   <element name="datosPersonales">
      <complexType>
         <sequence>
            <element name="datos" type="tns:datos"
               minOccurs="0" maxOccurs="unbounded"/>
         </sequence>
      </complexType>
   </element>

   <complexType name="datos">
      <all>
         <element name="nombre" type="string" />
         <element name="apellidos" type="string" />
         <element name="nacimiento" type="date" />
      </all>
      <attribute name="dni" type="string" />
   </complexType>
</schema>
```

<complexType>

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:datosPersonales xmlns:tns="http://www.example.org/dai"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.example.org/dai
                              datosPersonalesAll.xsd ">

   <tns:datos dni="12345678Z">
      <tns:nombre>Pedro</tns:nombre>
      <tns:apellidos>Pérez Pérez</tns:apellidos>
      <tns:nacimiento>1980-01-01</tns:nacimiento>
   </tns:datos>

   <tns:datos dni="87654321A">
      <tns:nacimiento>1960-01-01</tns:nacimiento>
      <tns:nombre>Ana</tns:nombre>
      <tns:apellidos>García García</tns:apellidos>
   </tns:datos>

   <tns:datos dni="11112222B">
      <tns:apellidos>Rodríguez Rodríguez</tns:apellidos>
      <tns:nacimiento>1975-01-01</tns:nacimiento>
      <tns:nombre>María</tns:nombre>
   </tns:datos>
</tns:datosPersonales>
```

<complexType>

- **Referencias a `<group>`**
  - Su declaración debe ser global.
  - Permite agrupar conjuntos de elementos y modelos de contenido para su reutilización.
  - Los grupos son similares a `<complexType>` pero no se consideran tipos.
  - **Formato:**

```
<group
   ref=”Nombre del grupo global”>
```

<complexType>

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://www.example.org/dai"
   xmlns:tns="http://www.example.org/dai"
   elementFormDefault="qualified">

   <element name="datosPersonales">
      <complexType>
         <sequence>
            <element name="datos" type="tns:datos"
               minOccurs="0" maxOccurs="unbounded"/>
         </sequence>
      </complexType>
   </element>

   <group name="datosPersonales">
      <sequence>
         <element name="nombre" type="string" />
         <element name="apellidos" type="string" />
         <element name="nacimiento" type="date" />
      </sequence>
   </group>

   <complexType name="datos">
      <group ref="tns:datosPersonales"/>
      <attribute name="dni" type="string" />
   </complexType>
</schema>
```

<complexType>

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:datosPersonales xmlns:tns="http://www.example.org/dai"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.example.org/dai
                       datosPersonalesGroup.xsd ">

   <tns:datos dni="12345678Z">
      <tns:nombre>Pedro</tns:nombre>
      <tns:apellidos>Pérez Pérez</tns:apellidos>
      <tns:nacimiento>1980-01-01</tns:nacimiento>
   </tns:datos>

   <tns:datos dni="87654321A">
      <tns:nombre>Ana</tns:nombre>
      <tns:apellidos>García García</tns:apellidos>
      <tns:nacimiento>1960-01-01</tns:nacimiento>
   </tns:datos>
</tns:datosPersonales>
```

<complexType>

- **Tipos vacíos**
  - Si un tipo complejo no incluye ningún modelo ni tipo de contenido, entonces será vacío.
  - Puede incluir atributos.

```
<complexType name="gps">
   <attribute name="latitud" type="double"/>
   <attribute name="longitud" type="double"/>
</complexType>
```

<complexType>

- **Tipos con solo texto**
  - Si se desea que un tipo solo contenga texto y, además, tenga atributos, debe hacerse uso de `simpleContent`.
  - Este elemento permite asociar un tipo simple al tipo actual y definir los atributos asociados.
  - El elemento `simpleContent` deberá contener un elemento interno `extension`, con el atributo `base` en el cual se indica el tipo simple asociado.
  - Si se desea asociar atributos al tipo, deberán declararse dentro del elemento `extension`.

<complexType>

```
<complexType name="descripcion">
   <simpleContent>
      <extension base="string">
         <attribute name="autor" type="string"/>
         <attribute name="fecha" type="date"/>
      </extension>
   </simpleContent>
</complexType>
```

```
<descripcion autor="MRJ" fecha="2012-10-01">
   Esto es un ejemplo de un elemento con texto
   y atributos
</descripcion>
```

Índice

<simpleType>

- **Formato:**

```
<simpleType
   name=”Nombre del tipo”
   final=”'#all' o ('list' o 'union' o
'restriction')”>
```

- El atributo `final` prohíbe extender el tipo mediante listas (`list`), uniones (`union`), restricciones (`restriction`) o todos ellos (`#all`).
- Los tipos simples **derivan** de otros tipos mediante:
  - **Restricción**: Limitan los valores del tipo padre.
  - **Lista**: Los valores válidos serán listas de valores del tipo válido.
  - **Unión**: Combinación de los valores de varios tipos padre.

<simpleType>

- **Derivación por restricción**
  - **Formato:**

```
<restriction
    base="nombre del tipo simple padre">
```

  - Permite definir una serie de restricciones sobre los valores del tipo padre.
  - El tipo resultante tendrá un subconjunto de los valores del tipo padre.
  - Las restricciones son denominadas *facets*.
  - No todas las restricciones son aplicables a todos los tipos.

<simpleType>

Lista de *Facets* Disponibles*

Facet	Description
`minExclusive`	Allows you to specify the minimum value for your type that excludes the value you specify
`minInclusive`	Allows you to specify the minimum value for your type that includes the value you specify
`maxExclusive`	Allows you to specify the maximum value for your type that excludes the value you specify
`maxInclusive`	Allows you to specify the maximum value for your type that includes the value you specify
`totalDigits`	Allows you to specify the total number of digits in a numeric type
`fractionDigits`	Allows you to specify the number of fractional digits in a numeric type (e.g., the number of digits to the right of the decimal point)
`length`	Allows you to specify the number of items in a list type or the number of characters in a string type
`minLength`	Allows you to specify the minimum number of items in a list type or the minimum number of characters in a string type
`maxLength`	Allows you to specify the maximum number of items in a list type or the maximum number of characters in a string type
`enumeration`	Allows you to specify an allowable value in an enumerated list
`whiteSpace`	Allows you to specify how whitespace should be treated within the type
`pattern`	Allows you to restrict string types using regular expressions

* Tabla tomada del libro “Beginning XML” - D. Hunter *et al.*

<simpleType>

```
<simpleType name="tipoBillete">
   <restriction base="string">
      <enumeration value="turista"/>
      <enumeration value="preferente"/>
      <enumeration value="business"/>
      <enumeration value="primera"/>
   </restriction>
</simpleType>
```

```
<simpleType name="pasajeros">
    <restriction base="unsignedInt">
        <minInclusive value="1"/>
        <maxInclusive value="5"/>
    </restriction>
</simpleType>
```

```
<simpleType name="coordenada">
    <restriction base="decimal">
    <fractionDigits value="10"/>
    </restriction>
</simpleType>
```

```
<simpleType name="dni">
   <restriction base="string">
      <pattern value="[0-9]{8}[a-zA-Z]"/>
   </restriction>
</simpleType>
```

<simpleType>

- **Derivación por lista**
  - **Formato:**

```
<list itemType=”nombre del tipo simple padre”>
```

  - Los valores de los tipos lista serán una serie de valores del tipo padre separados por un espacio.

<simpleType>

```
<simpleType name="premios">
   <restriction base="string">
      <enumeration value="oscar"/>
      <enumeration value="goya"/>
      <enumeration value="goldenGlobe"/>
      <enumeration value="razzie"/>
   </restriction>
</simpleType>
```

```
<simpleType name="premiosGanados">
   <list itemType="tns:premios"/>
</simpleType>
```

<simpleType>

- **Derivación por unión**
  - **Formato:**

```
<union memberTypes=”nombre de los tipo
simple padres separados por un espacio”>
```

  - Los valores de los tipos unión serán todos los posibles valores de los tipos padre.

```
<simpleType name="dateOrTimestap">
   <union memberTypes="date unsignedLong"/>
</simpleType>
```

Índice

<attribute>

- **Atributos `<attribute>`**
  - Pueden ser **globales o locales**.
  - Solo pueden ser de **tipos simples**.
  - **Formato:**

```
<attribute
  name=”Nombre del atributo”
  type=”Tipo del atributo”
  ref=”Declaración global de atributo”
  form=”'qualified' o 'unqualified'”
  use=”'optional' o 'prohibited' o 'required'”
  default=”Valor por defecto”
  fixed=”Valor fijo”>
```

<attribute>

- **Atributos `<attribute>` (continuación)**
  - El tipo asignado a los atributos puede ser global o local.

Atributo con Tipo Local

```
<attribute name="atributoTipoLocal">
   <simpleType>
      <restriction base="unsignedInt">
         <maxExclusive value="256"/>
      </restriction>
    </simpleType>
</attribute>
```

Atributo con Tipo Global

```
<simpleType name="byteValue">
   <restriction base="unsignedInt">
      <maxExclusive value="255"/>
   </restriction>
</simpleType>

<attribute name="atributoTipoLocal"
           type="ejemplo:byteValue"/>
```

<attributeGroup>

- **Grupos de Atributos `<attributeGroup>`**
  - Permite definir grupos de atributos.
  - De un modo similar a `group` su objetivo es favorecer la reutilización.
  - Deben tener un nombre y su alcance es global.
  - Pueden contener atributos u otros grupos de atributos.
  - **Formato:**

```
<attributeGroup
  name=”Nombre global del grupo de atributos”>
```

Índice

<element>

- Declaración de la **estructura de un elemento**.
- Pueden declararse como **parte de un tipo** o como **elementos raíz**.
- **Formato:**

```
<element
   name=”Nombre del elemento”
   type=”Tipo del elemento”
   ref=”Declaración global del elemento”
   form=”'qualified' o 'unqualified'”
   minOccurs=”Entero no negativo”
   maxOccurs=”Entero no negativo o 'unbounded'”
   default=”Valor por defecto”
   fixed=”Valor fijo”>
```

Índice

Uso de Schemas

- La asociación de un *schema* a un documento se hace mediante atributos en el elemento raíz:
  - **`xmlns`**: permite importar *namespaces*.
  - **`schemaLocation`**: permite indicar donde se encuentran los *schemas* asociados a cada *namespace*.
    - Contiene parejas de URIs, donde el primer elemento es el **nombre del *namespace*** y el segundo su **ubicación**.
- Es necesario que importe el namespace `http://www.w3.org/2001/XMLSchema-instance`

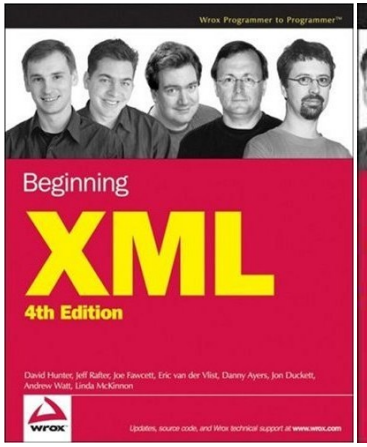
Uso de Schemas

```
<?xml version="1.0"?>
<name xmlns="http://www.example.com/name"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.example.com/name name.xsd"
  title="Mr.">
    <first>John</first>
    <middle>Fitzgerald Johansen</middle>
    <last>Doe</last>
</name>
```

Índice

Bibliografía

- **Beginning XML, 4th Edition** – D. Hunter *et al.*
- **Professional XML** – B. Evjen *et al.*



- **World Wide Web Consortium (W3C)** [http://www.w3.org/ - última visita 01/11/2020]