Apuntes de XSD
(*XML Schema*)

Contenidos del tutorial de XSD

1. Qué es XSD

2. Validación de un documento XML con XSD

Definición de un espacio de nombres

Validación de un sitemap XML

3. Elementos simples

Tipos de declaración de elementos simples (**`fixed`**, **`default`**)

4. Atributos

Tipos de declaración de atributos (**`fixed`**, **`default`**, **`optional`**, **`required`**)

5. Restricciones (facetas)

`xs:minExclusive` y **`xs:maxInclusive`**

`xs:enumeration`

`xs:pattern`

`xs:length`

`xs:whiteSpace`

6. Extensiones

`xs:extension` (**`complexContent`**)

`xs:extension` (**`simpleContent`**)

7. Elementos complejos

Elemento vacío

Contenido mixto

8. Indicadores

Indicadores de orden (**`xs:sequence`**, **`xs:all`**, **`xs:choice`**)

Indicadores de ocurrencia (**`maxOccurs`**, **`minOccurs`**)

Indicadores de grupo (**`xs:group`**, **`xs:attributeGroup`**)

1. Qué es XSD

XSD (*XML Schema Definition*) es un lenguaje, también llamado simplemente ***XML Schema***, que sirve para definir la estructura de un documento XML, permitiendo su validación.

En los siguientes enlaces se puede consultar la *W3C Recommendation* de XSD, en la cual están basados estos apuntes de introducción a XML Schema:

- ***XML Schema Part 0: Primer***
  www.w3.org/TR/xmlschema-0/
- ***XML Schema Part 1: Structures***
  www.w3.org/TR/xmlschema-1/
- ***XML Schema Part 2: Datatypes***
  www.w3.org/TR/xmlschema-2/

2. Validación de un documento XML con XSD

EJEMPLO Se quiere almacenar una lista de marcadores de páginas web, guardando de cada uno de ellos su nombre, una descripción y su URL. Para ello, se ha escrito el siguiente documento XML (***"marcadores.xml"***) asociado al archivo ***"marcadores.xsd"***:

```
<?xml version="1.0" encoding="UTF-8"?>
<marcadores xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="marcadores.xsd">
   <pagina>
      <nombre>Abrirllave</nombre>
      <descripcion>Tutoriales de informática.</descripcion>
      <url>http://www.abrirllave.com/</url>
   </pagina>
   <pagina>
      <nombre>Wikipedia</nombre>
      <descripcion>La enciclopedia libre.</descripcion>
      <url>http://www.wikipedia.org/</url>
   </pagina>
   <pagina>
      <nombre>W3C</nombre>
      <descripcion>World Wide Web Consortium.</descripcion>
      <url>http://www.w3.org/</url>
   </pagina>
</marcadores>
```

- Para vincular un esquema a un documento XML, es obligatorio que este último haga referencia al espacio de nombres **`http://www.w3.org/2001/XMLSchema instance`**. Para ello, habitualmente se utiliza el prefijo **`xsi`**.
- El atributo **`noNameSchemaLocation`** permite referenciar a un archivo con la definición de un esquema que no tiene ningún espacio de nombres asociado. En este caso, dicho archivo es ***"marcadores.xsd"***.

El esquema XML guardado en ***"marcadores.xsd"*** y que permita validar el documento XML ***"marcadores.xml"*** podría ser:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="marcadores">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="pagina" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="nombre" type="xs:string"/>
              <xs:element name="descripcion" type="xs:string"/>
              <xs:element name="url" type="xs:string"/>
             </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Para estar bien formado, un esquema XML tiene que cumplir las mismas reglas de sintaxis que cualquier otro documento XML.

Por otra parte, hay que tener en cuenta que, en todos los esquemas XML, el elemento raíz es "schema". Ahora bien, para escribirlo, es muy común utilizar el prefijo **`xsd`** o **`xs`**.

Con **`xmlns:xs="http://www.w3.org/2001/XMLSchema"`** se ha indicado que:

- Los elementos y tipos de datos utilizados en el esquema pertenecen al espacio de nombres: **`http://www.w3.org/2001/XMLSchema`**.
- Dichos elementos y tipos de datos deben llevar el prefijo **`xs`** **(`xs:schema`, `xs:element`, `xs:complexType`, `xs:string`…)**.

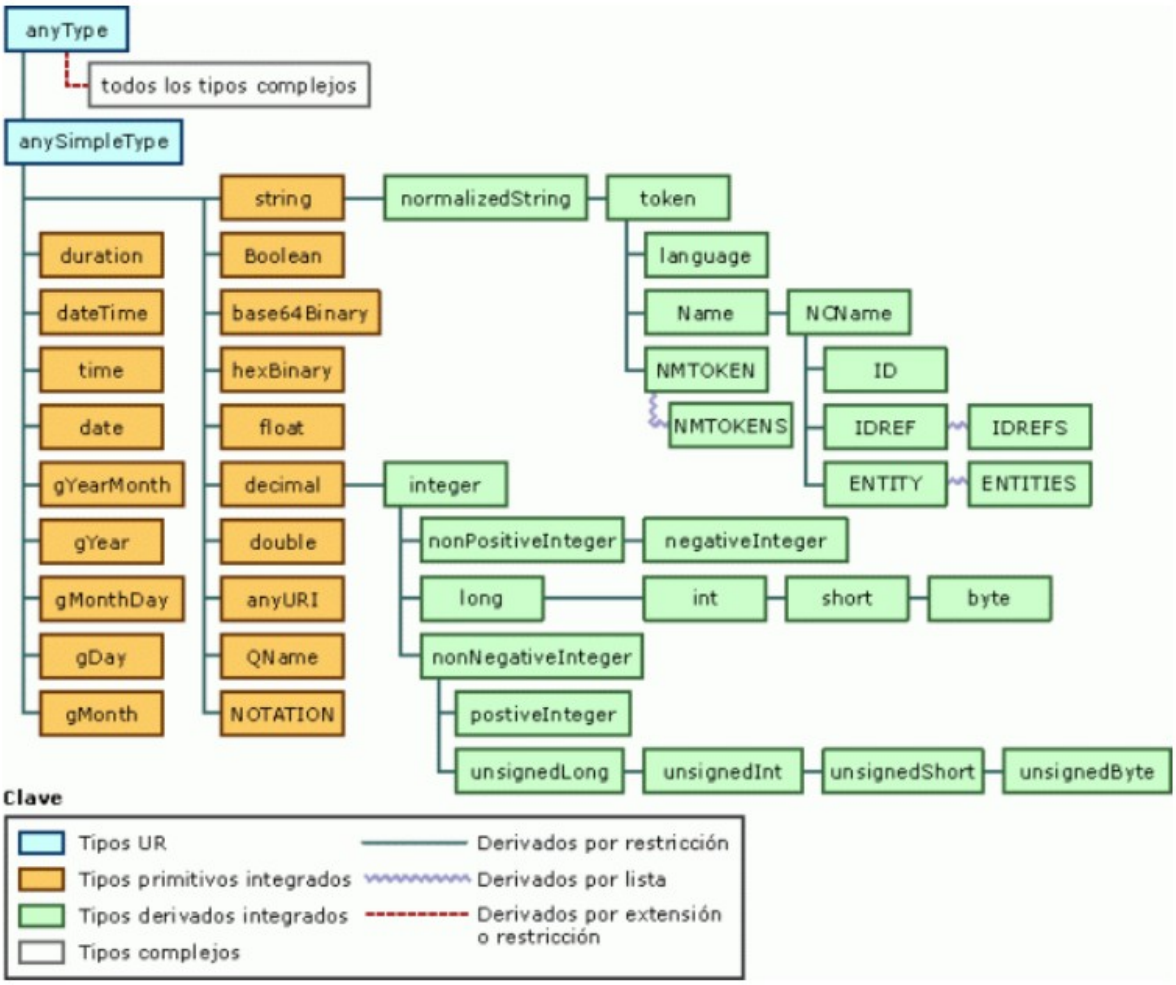
Fíjese también que:

- Los elementos "marcadores" y "página" son de tipo complejo (**`complexType`**), ya que, contienen a otros elementos.
- **`sequence`** indica que los elementos hijo deben aparecer, en el documento XML, en el mismo orden en el que sean declarados en el esquema.
- Los elementos "nombre", "descripción" y "url" son de tipo simple (**`string`** en este caso) y no pueden contener a otros elementos.
- Mediante **`maxOccurs="unbounded"`** se ha indicado que pueden aparecer ilimitados elementos "página" en el documento XML.

3. Elementos simples

Los elementos simples solamente pueden contener texto (caracteres). Dicho de otro modo, los elementos simples no pueden contener a otro u otros elementos (hijos), ni tampoco pueden tener atributos. Ahora bien, el texto contenido en un elemento simple, puede ser de diferentes tipos de datos predefinidos en ***W3C XML Schema*** o definidos por el usuario (programador).

Los tipos de datos predefinidos pueden ser primitivos (**string**, **boolean**, **decimal**...) o derivados de estos (**integer**, **ID**, **IDREF**...). Véase en la siguiente imagen donde se puede ver la relación que existe entre todos ellos:



Tipos de datos predefinidos (imagen extraída de Microsoft)

Los tipos de datos mas comunes, y que son los que emplearemos nosotros son:

- **xs:string** - Cadena de caracteres.
- **xs:decimal** - Números reales.
- **xs:integer** - Representa una secuencia de dígitos decimales con un signo inicial (+ o -) opcional. Este tipo de datos se deriva dedecimal.
- **xs:boolean** - Valores booleanos que pueden ser true o false.

- **xs:date** - Fecha en formato YYYY-MM-DD.
- **xs:time** - Hora en formato hh:mm:ss.sss.

3.1 xs:element

Este componente permite declarar los elementos del documento XML. Entre otros, los principales atributos que podemos utilizar en la declaración son los siguientes:

- **name**: Indica el nombre del elemento. Obligatorio si el elemento padre es <xs:schema>.
- **ref**: Indica que la declaración del elemento se encuentra en otro lugar del esquema. No se puede usar si el elemento padre es <xs:schema>. No puede aparecer junto con **name**.
- **type**: Indica el tipo de dato que almacenará el elemento. No puede aparecer junto con ref.
- **default**: Es el valor que tomará el elemento al ser procesado si en el documento XML no ha recibido ningún valor. Sólo se puede usar con tipo de dato textual.
- **fixed**: Indica el único valor que puede contener el elemento en el documento XML. Sólo se puede usar con tipo de dato textual.
- **minOccurs**: Indica el mínimo número de ocurrencias que deben aparecer de ese elemento en el documento XML. No se puede usar si el elemento padre es <xs:schema>. Va desde 0 hasta ilimitado (**unbounded**). **Por defecto 1.**
- **maxOccurs**: Indica el máximo número de ocurrencias que pueden aparecer de ese elemento en el documento XML. No se puede usar si el elemento padre es <xs:schema>. Va desde 0 hasta ilimitado (**unbounded**). **Por defecto 1.**

```
<xs:element name="nombre" type="xs:string" default="TicArte"
minOccurs="1" maxOccurs="unbounded" />
```

```
<xs:element ref="contacto" minOccurs="1" maxOccurs="unbounded" />
```

Para definir un elemento simple se puede utilizar la siguiente sintaxis:

```
<xs:element name="nombre_del_elemento" type="tipo_de_dato"/>
```

EJEMPLO Para los siguientes elementos XML:

```
<nombre>Elsa</nombre>

<edad>23</edad>
```

Sus definiciones pueden ser:

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

<xs:element name="edad" type="xs:integer"/>
```

Tipos de declaración de elementos simples (**fixed**, **default**)

Si se quiere indicar que un valor es fijo (**fixed**), se puede escribir, por ejemplo:

```
<xs:element name="mes" type="xs:string" fixed="agosto"/>
```

También, se puede especificar un valor por defecto (**default**), por ejemplo, tecleando:

```
<xs:element name="mes" type="xs:string" default="agosto"/>
```

EJEMPLO

Para los siguientes elementos:

<ciudad>Roma</ciudad>

<fecha-de-nacimiento>1996-12-18</fecha-de-nacimiento>

<hora>18:29:45</hora>

<nota>7.5</nota>

<apto>true</apto>

Escribir sus definiciones de elementos simples correspondientes:

Solución:

<xs:element name="ciudad" type="xs:string"/>

<xs:element name="fecha-de-nacimiento" type="xs:date"/>

<xs:element name="hora" type="xs:time"/>

<xs:element name="nota" type="xs:decimal"/>

<xs:element name="apto" type="xs:boolean"/>

EJEMPLO

Definir un elemento llamado puertaCerrada de tipo lógico, que por defecto tenga el valor "falso", y otro elemento llamado ventanaAbierta también de tipo lógico, que tenga asignado el valor fijo "verdadero".

Solución:

<xs:element name="puertaCerrada" type="xs:boolean" default="false"/>

<xs:element name="ventanaAbierta" type="xs:boolean" fixed="true"/>

4. Atributos

4.1 xs:attribute

Este componente permite declarar los atributos de los elementos del documento XML. Entre otros, los principales atributos que podemos utilizar en la declaración son los siguientes:

- **name**: Indica el nombre del atributo.
- **ref**: Indica que la declaración del atributo se encuentra en otro lugar del esquema. No puede aparecer junto con **name**.
- **type**: Indica el tipo de dato que almacenará el atributo. No puede aparecer junto con **ref**.
- **use**: Indica si la existencia del atributo es opcional (**optional**) o obligatoria (**required)**. **Por defecto opcional.**
- **default**: Es el valor que tomará el elemento al ser procesado si en el documento XML no ha recibido ningún valor. Sólo se puede usar con tipo de dato textual.
- **fixed**: Indica el único valor que puede contener el elemento en el documento XML. Sólo se puede usar con tipo de dato textual.

EJEMPLOS:

```
<xs:attribute name="moneda" type="xs:string" default="euro" use="required" />
```

```
<xs:attribute ref="moneda" use="required" />
```

Para definir un atributo se puede emplear la siguiente sintaxis:

```
<xs:attribute name="nombre_del_atributo" type="tipo_de_dato"/>
```

- Todos los atributos pueden tomar por valor tipos simples.
- Por otra parte, cuando un elemento tiene al menos un atributo dicho elemento se dice que es complejo.

Tipos de declaración de atributos (**fixed**, **default**, **optional**, **required**)

Para indicar que el valor de un atributo es fijo (**fixed**), es posible escribir, por ejemplo:

```
<xs:attribute name="grupo" type="xs:string" fixed="B"/>
```

Para especificar el valor por defecto (**default**) de un atributo, se puede escribir:

```
<xs:attribute name="grupo" type="xs:string" default="B"/>
```

Para indicar que un atributo es obligatorio (**required**) escribirlo, se puede teclear:

```
<xs:attribute name="grupo" type="xs:string" use="required"/>
```

Por defecto, si no se indica nada, el atributo será opcional (**optional**).

EJEMPLO

Dado el siguiente documento XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<fichas xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="fichas.xsd">
    <ficha numero="1">
        <nombre>Ana Sanz Tin</nombre>
        <edad>22</edad>
    </ficha>
    <ficha numero="2">
        <nombre>Iker Rubio Mol</nombre>
        <edad>23</edad>
    </ficha>
</fichas>
```

Escribir el contenido del archivo "fichas.xsd" que permita validarlo.

SOLUCIÓN:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="fichas">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="ficha" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="nombre" type="xs:string" />
                            <xs:element name="edad" type="xs:integer" />
                        </xs:sequence>
                        <xs:attribute name="numero" type="xs:integer" />
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

5. Restricciones (facetas)

XML Schema permite definir restricciones a los posibles valores de los tipos de datos. Dichas restricciones se pueden establecer en diferentes aspectos, llamados facetas.

Dicho de otro modo**, las facetas permiten definir restricciones sobre los posibles valores de atributos o elementos**. Las facetas que pueden utilizarse son:

Facetas de XSD	
Faceta	***Descripción***
xs:length	Especifica una longitud fija.
xs:minLength	Especifica una longitud mínima.
xs:maxLength	Especifica una longitud máxima.
xs:pattern	Especifica un patrón de caracteres admitidos.
xs:enumeration	Especifica una lista de valores admitidos.
xs:whiteSpace	Especifica cómo se debe tratar a los posibles espacios en blanco, las tabulaciones, los saltos de línea y los retornos de carro que puedan aparecer.
xs:maxInclusive	Especifica que el valor debe ser menor o igual que el indicado.

xs:maxExclusive	Especifica que el valor debe ser menor que el indicado.
xs:minExclusive	Especifica que el valor debe ser mayor que el indicado.
xs:minInclusive	Especifica que el valor debe ser mayor o igual que el indicado.
xs:totalDigits	Especifica el número máximo de dígitos que puede tener un número.
xs:fractionDigits	Especifica el número máximo de decimales que puede tener un número.

Seguidamente, se muestran algunos ejemplos de restricciones definidas con una o más facetas:

xs:minExclusive y xs:maxInclusive

EJEMPLO En el siguiente código se define un elemento llamado "mes" con la restricción de que el valor que tome no pueda ser menor que 1 ni mayor que 12:

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

- **xs:simpleType** permite definir un tipo simple y especificar sus restricciones.
- **xs:restriction** sirve para definir restricciones de un **xs:simpleType** (como se ha hecho en este ejemplo). También sirve para definir restricciones de un **xs:simpleContent** o de un **xs:complexContent**. Estos elementos se estudiarán más adelante.
- En el atributo **base** se indica el tipo de dato a partir del cual se define la restricción.
- **xs:minInclusive** sirve para especificar que el valor debe ser mayor o igual que el indicado en su atributo **value**, (en este caso, mayor o igual que 1).
- **xs:maxInclusive** sirve para especificar que el valor debe ser menor o igual que el indicado en su atributo **value**, (en este caso, menor o igual que 12).

También se podría haber escrito:

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

```
</xs:simpleType>
```

Haciendo esto, el tipo **numeroMes** definido, podría ser utilizado por otros elementos, ya que, no está contenido en el elemento "mes".

Ejercicios resueltos

- Edad entre 0 y 130 años
- Precios de tres dígitos

xs:enumeration

EJEMPLO En el siguiente ejemplo se define un elemento llamado "color" con la restricción de que los únicos valores admitidos son: **"verde"**, **"amarillo"** y **"rojo"**.

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

- **xs:enumeration** sirve para definir una lista de valores admitidos.

Ejercicio resuelto

- Tipo de vehículo

xs:pattern

EJEMPLO En el siguiente ejemplo se define un elemento llamado "letra" con la restricción de que el único valor admitido es una de las letras minúsculas de la **"a"** a la **"z"**:

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

- **xs:pattern** sirve para definir un patrón de caracteres admitidos (en este caso se admite una única letra minúscula de la **"a"** a la **"z"**). **El valor del patrón tiene que ser una expresión regular.**

Ejercicios resueltos

- Iniciales de personas famosas
- Iniciales al revés
- Respuestas admitidas
- Números y letras
- Escribir expresiones regulares
- Letras admitidas

xs:length

EJEMPLO En el siguiente ejemplo se define un elemento llamado "clave" con la restricción de que su valor tiene que ser una cadena de, exactamente, doce caracteres:

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

- **xs:length** sirve para especificar una longitud fija.

Ejercicios resueltos

- Longitud fija de una clave
- Longitud mínima y máxima de una clave

xs:whiteSpace

EJEMPLO En el siguiente ejemplo se define un elemento llamado "dirección" con la restricción de que los espacios en blanco, las tabulaciones, los saltos de línea y los retornos de carro que aparezcan en él, se deben mantener (**preserve**):

```
<xs:element name="direccion">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:whiteSpace value="preserve" />
        </xs:restriction>
    </xs:simpleType>
</xs:element>
```

- **xs:whiteSpace** sirve para especificar cómo se debe tratar a los posibles espacios en blanco, las tabulaciones, los saltos de línea y los retornos de carro que puedan aparecer.

En vez de **`preserve`** también se puede utilizar:

- **`replace`** para sustituir todas las tabulaciones, los saltos de línea y los retornos de carro por espacios en blanco.

6. Extensiones

6.1 Tipos de datos Simples y Complejos

Existe dos grandes grupos de tipos de datos que se pueden utilizar en los esquemas XSD:

- Tipos de datos simples **(xs:simpleType)**: Se dividen en los siguientes.
  - Tipos de datos predefinidos
  - Tipos de datos construidos con nuestras propias restricciones y basados en los tipos de datos predefinidos.
- Tipos de datos complejos **(xs:complexType)**: Se dividen en los siguientes.
  - Elementos dentro de otros elementos.
  - Elementos que tienen atributos.
  - Elementos mixtos que tienen datos y otros elementos.
  - Elementos vacíos.

Dentro de los tipos de datos complejos, podemos definir dos tipos de contenidos que pueden ir entre las etiquetas de apertura y cierre de los elementos:

- **Contenido simple** (**xs:simpleContent**): Cuando el elemento declarado sólo tienen contenido textual, sin elementos descendientes.
- **Contenido complejo** (**xs:complexContent**): Cuando el elemento declarado tiene elementos descendientes, pudiendo tener o no contenido textual.

xs:extension (complexContent)

```
<?xml version="1.0" encoding="UTF-8"?>
<fichas xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="fichas.xsd">
    <ficha numero="1">
        <nombre>Eva</nombre>
        <edad>25</edad>
        <ciudad>París</ciudad>
        <pais>Francia</pais>
    </ficha>
    <ficha numero="2">
        <nombre>Giovanni</nombre>
        <edad>26</edad>
        <ciudad>Florencia</ciudad>
        <pais>Italia</pais>
    </ficha>
</fichas>
```

Y el archivo ***"fichas.xsd"*** que permite validarlo:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="fichas">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="ficha" type="infoPersonaAmpliada"
maxOccurs="unbounded" />
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:complexType name="infoPersonaAmpliada">
        <xs:complexContent>
            <xs:extension base="infoPersona">
                <xs:sequence>
                    <xs:element name="ciudad" type="xs:string" />
                    <xs:element name="pais" type="xs:string" />
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="infoPersona">
        <xs:sequence>
            <xs:element name="nombre" type="xs:string" />
            <xs:element name="edad" type="edadPersona" />
        </xs:sequence>
        <xs:attribute name="numero" type="xs:integer" />
    </xs:complexType>
    <xs:simpleType name="edadPersona">
        <xs:restriction base="xs:integer">
            <xs:minExclusive value="-1" />
            <xs:maxExclusive value="131" />
        </xs:restriction>
    </xs:simpleType>
</xs:schema>
```

- Obsérvese que, **`infoPersonaAmpliada`** se basa en **`infoPersona`**, añadiéndole dos elementos: "ciudad" y "país".
- En cuanto a **`xs:complexContent`**, sirve para definir restricciones o extensiones a un tipo complejo (**`complexType`**).

Ejercicio resuelto
· Información de persona ampliada

`xs:extension (simpleContent)`

`xs:simpleContent` permite definir restricciones o extensiones a elementos que solo contienen datos, es decir, no contienen a otros elementos.

EJEMPLO El siguiente archivo ***"precios.xsd"***:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="precios">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="precio" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:simpleContent>
                            <xs:extension base="xs:decimal">
                                <xs:attribute name="moneda" type="xs:string" />
                            </xs:extension>
                        </xs:simpleContent>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Permite validar el siguiente documento XML:

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

    <precio moneda="Euro">5</precio>
    <precio moneda="Dólar">6.2</precio>
    <precio moneda="Libra esterlina">4.3</precio>
</precios>
```

- Nótese que, utilizando **`xs:extension`**, al elemento "precio" se le ha incorporado el atributo **`moneda`**.

Ejercicios resueltos

- Precios de artículos
- Información de ubicaciones
- Colores de muebles

7. Elementos complejos

Un elemento es complejo (**`complexType`**) cuando contiene uno o más elementos y/o atributos. De entre las posibles combinaciones de elementos y/o atributos que puede contener un elemento complejo (1 elemento y 0 atributos, 1 elemento y 1 atributo, 1 elemento y varios atributos, 0 elementos y 1 atributo...) cabe destacar las siguientes:

- **Un elemento complejo puede estar vacío**, es decir, no contener elementos ni texto, pero sí tener al menos un atributo.

- **Un elemento complejo puede contener contenido mixto**, es decir, contener uno o más elementos, además de texto. Por otra parte, podría tener atributos, o no.

Elemento vacío

EJEMPLO En el siguiente código se ha definido vacío el elemento "bola", no pudiendo contener ni otros elementos ni texto. Ahora bien, véase que sí tiene un atributo, llamado **`numero`**:

```
<xs:element name="bola">
    <xs:complexType>
        <xs:attribute name="numero" type="numeroDeBola" />
    </xs:complexType>
</xs:element>
<xs:simpleType name="numeroDeBola">
    <xs:restriction base="xs:positiveInteger">
        <xs:minInclusive value="1" />
        <xs:maxExclusive value="90" />
    </xs:restriction>
</xs:simpleType>
```

- **`xs:positiveInteger`** indica que el valor del atributo **`numero`** debe ser un número entero mayor que cero.

Ejercicio resuelto

· Números del bingo

Contenido mixto

Fíjese que, en el siguiente código se ha definido el elemento "persona" de tipo complejo mixto (**`mixed="true"`**):

```
<xs:element name="persona">
    <xs:complexType mixed="true">
        <xs:sequence>
            <xs:element name="nombre" type="xs:string" />
            <xs:element name="ciudad" type="xs:string" />
            <xs:element name="edad"
type="xs:positiveInteger" />
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

Ejercicio resuelto

· Información de personas en contenido mixto

8. Indicadores

Los indicadores permiten establecer cómo se van a escribir –o utilizar– los elementos en un documento XML. Hay siete tipos de indicadores que se pueden clasificar en:

- **Indicadores de orden:** secuencia (**`sequence`**), todo (**`all`**) y elección (**`choice`**).
- **Indicadores de ocurrencia:** **`maxOccurs`** y **`minOccurs`**.
- **Indicadores de grupo:** de elementos (**`group`**) y de atributos (**`attributeGroup`**).

Indicadores de orden (`xs:sequence`, `xs:all`, `xs:choice`)

Mientras que **`xs:sequence`** sirve para especificar el orden en el que obligatoriamente deben aparecer los elementos hijo de un elemento, **`xs:all`** sirve para indicar que dichos elementos pueden aparecer en cualquier orden.

EJEMPLO El siguiente archivo ***"lugar.xsd"***:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="lugar">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="ciudad">
                    <xs:complexType>
                        <xs:all>
                            <xs:element name="nombre" type="xs:string" />
                            <xs:element name="pais" type="xs:string" />
                        </xs:all>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Permite validar el siguiente documento XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<lugar xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="lugar.xsd">
    <ciudad>
        <pais>Italia</pais>
        <nombre>Florencia</nombre>
    </ciudad>
</lugar>
```

Por otra parte, **`xs:choice`** sirve para especificar que solamente se permite escribir uno de los elementos hijo. Por ejemplo, en este caso, se podría utilizar para indicar que habría que elegir entre escribir el "nombre" o escribir el "país" de la "ciudad", pero no ambos.

Indicadores de ocurrencia (`maxOccurs`, `minOccurs`)

`maxOccurs` y **`minOccurs`** permiten establecer, respectivamente, el número máximo y mínimo de veces que puede aparecer un determinado elemento. El valor por defecto para **`maxOccurs`** y **`minOccurs`** es 1.

EJEMPLO Dado el siguiente documento XML ***"paises.xml"***:

```
<?xml version="1.0" encoding="UTF-8"?>
<paises xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="paises.xsd">
    <pais>
        <nombre>Argentina</nombre>
        <ciudad>Buenos Aires</ciudad>
        <ciudad>Rosario</ciudad>
    </pais>
    <pais>
        <nombre>México</nombre>
        <ciudad>Guadalajara</ciudad>
        <ciudad>Monterrey</ciudad>
        <ciudad>Cancún</ciudad>
        <ciudad>Mérida</ciudad>
        <ciudad>Ciudad de México</ciudad>
    </pais>
    <pais>
        <nombre>Colombia</nombre>
    </pais>
</paises>
```

Considerando que se quiere especificar que:

- "país" pueda aparecer una o ilimitadas veces.
- "nombre" tenga que escribirse obligatoriamente, y solo una vez, dentro de "país".
- De cada "país" puedan escribirse de cero a cinco "ciudades".

El código del archivo ***"paises.xsd"*** que permita validar ***"paises.xml"***, podría ser:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="paises">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="pais" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="nombre" type="xs:string" />
                            <xs:element name="ciudad" type="xs:string"
minOccurs="0" maxOccurs="5" />
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Indicadores de grupo (`xs:group`, `xs:attributeGroup`)

`xs:group` sirve para agrupar un conjunto de declaraciones de elementos relacionados.

EJEMPLO Dado el siguiente documento XML ***"personas.xml"***:

```
<?xml version="1.0" encoding="UTF-8"?>
<personas xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="personas.xsd">
    <persona>
        <nombre>Eva</nombre>
        <edad>25</edad>
        <pais>Francia</pais>
        <telefono>999888777</telefono>
    </persona>
    <persona>
        <nombre>Giovanni</nombre>
        <edad>26</edad>
        <pais>Italia</pais>
        <telefono>111222333</telefono>
    </persona>
</personas>
```

Y el archivo ***"personas.xsd"*** que permite validarlo:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="personas">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="persona" type="datosDePersona"
maxOccurs="unbounded" />
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:complexType name="datosDePersona">
        <xs:sequence>
            <xs:group ref="datosBasicos" />
            <xs:element name="telefono" type="xs:string" />
        </xs:sequence>
    </xs:complexType>
    <xs:group name="datosBasicos">
        <xs:sequence>
            <xs:element name="nombre" type="xs:string" />
            <xs:element name="edad" type="xs:positiveInteger" />
            <xs:element name="pais" type="xs:string" />
        </xs:sequence>
    </xs:group>
</xs:schema>
```

- Observe que, se ha definido el grupo **datosBasicos**, el cual ha sido incorporado a la definición del tipo complejo **datosDePersona**.

Del mismo modo, **attributeGroup** sirve para definir un grupo de atributos. Por ejemplo, para validar el siguiente documento XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<personas xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="personas.xsd">
    <persona nombre="Eva" edad="25" pais="Francia" />
    <persona nombre="Giovanni" edad="26" pais="Italia" />
</personas>
```

Se puede escribir el siguiente código en el archivo ***"personas.xsd"***:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="personas">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="persona" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:attributeGroup ref="datosDePersona" />
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:attributeGroup name="datosDePersona">
        <xs:attribute name="nombre" type="xs:string" />
        <xs:attribute name="edad" type="xs:positiveInteger" />
        <xs:attribute name="pais" type="xs:string" />
    </xs:attributeGroup>
</xs:schema>
```

- En este caso, se ha definido el grupo de atributos **datosDePersona**, el cual ha sido incorporado a la definición del elemento **persona**.

Ejercicios resueltos

- Panel de vuelos
- Factura
- Registro de conexiones
- Personal de departamentos

EJEMPLO COMPLETO

Para el siguiente documento XML definir su correspondiente documento XSD

email.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<email>
    <from> luca.rossi.917@gmail.com </from>
    <to> atzeni@dia.uniroma3.it </to>
    <content> Dear <person> Paolo </person>, here are some
        assignments of <course> Basi di Dati 2 </course>:
        <exercises>
            <exercise>
                <topic> DTD </topic>
                <description> From Instance to DTD </description>
            </exercise>
            <exercise>
                <topic> XPath </topic>
                <description> Find students with average 26 </description>
            </exercise>
        </exercises>
        Best Regards, <person> Luca </person>
    </content>
</email>
```

SOLUCIÓN

email.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:element name="email">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="from" type="xsd:string" />
                <xsd:element name="to" type="xsd:string" />
                <xsd:element name="content">
                    <xsd:complexType mixed="true">
                        <xsd:sequence>
                            <xsd:element name="person" type="xsd:string" />
                            <xsd:element name="course" type="xsd:string" />
                            <xsd:element name="exercises">
                                <xsd:complexType>
                                    <xsd:sequence minOccurs="0" maxOccurs="unbounded">
                                        <xsd:element name="exercise">
                                            <xsd:complexType>
                                                <xsd:sequence>
                                                    <xsd:element name="topic" type="xsd:string" />
                                                    <xsd:element name="description"
                                                        type="xsd:string" />
                                                </xsd:sequence>
                                            </xsd:complexType>
                                        </xsd:element>
                                    </xsd:sequence>
                                </xsd:complexType>
                            </xsd:element>
                            <xsd:element name="person" type="xsd:string" />
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```