Esquemas XML o XSD

También llamado XSD (XML Schema Definition). Un esquema XML no es más que un documento XML que a su vez describe la estructura de un otro documento XML que es el que se quiere validar.

Al ser un fichero xml tendremos que incluir la primera linea:

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

1. <schema> Es el elemento raiz del esquema. Una vez abierto este tag <schema> dentro se especificarán los elementos que podrán ser simples o complejos. Se puede declara de tres formas:

```
<schema  xmlns="http://www.w3.org/2011/XMLSchema">
<xs:schema
xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xsd:schema
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

2. Los elementos simples son los que solamente contienen texto, no contiene otros elementos hijos ni tampoco atributos.
   Eso si, el texto puede ser cualquier tipo de datos definido en schema (Integer, string, fechahora, booleano, etc) o cualquier tipo definido por nosotros, es posible incluso añadir restricciones (llamadas facetas) u obligar a que se cumpla algún patrón mediante expresiones regulares.

```
<xs:element name="apellidos" type="xs:string"/>
<xs:element name="edad" type="xs:integer"/>
<xs:element name="nacimiento" type="xs:date"/>
```

3. Podemos incluir los siguientes tipos de restricciones:
   a. Sobre los valores. Ejemplo edad entre 0 y 100

```
<xs:element name="edad">
<xs:simpleType>
<xs:restriction base="xs:integer">
      <xs:minInclusive value="0"/>
      <xs:maxInclusive value="100"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
```

   b. Sobre un conjunto de valores. Ejemplo permitimos 3 colores.

```
<xs:element name="color">
<xs:simpleType>
<xs:restriction base="xs:string">
   <xs:enumeration value="Rojo"/>
   <xs:enumeration value="Verde"/>
   <xs:enumeration value="Azul"/>
</xs:restriction>
</xs:simpleType>
```

```
</xs:element>
```

c. Sobre los espacios en blanco. El siguiente ejemplo los mantiene intocables, esto significa que retornos de carro, tabuladores y espacios en blanco no serán alterados dentro del campo de tipo dirección (también podríamos sustituirlos por 1 solo espacio con value="collapse" o reemplazarlos con value="replace"):

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

d. Sobre la longitud. Ejemplo NIF a 9 caracteres (también podríamos poner un máximo y un mínimo con <xs:minLength value="8"/> <xs:maxLength value="255"/>):

```
<xs:element name="nif">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:length value="9"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
```

e. Sobre patrones. Contenido en mayúsculas:

```
<xs:element name="mayusculas">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:pattern value="([A-Z])*"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
```

4. Atributos. Los atributos serán utilizados en los tipos complejos y serán declarados en si mismos como un tipo simple. Los tipos simples no pueden tener atributos. La forma general de declarar un atributo es como sigue: <xs:attribute name="xxx" type="yyy"/>

5. Distintas formas de declarar los elementos complejos:

- Tipo de complejo local al elemento y no se puede reutilizar

```
<xsd:element name= "vehículos">
      <xsd:complesType>
            <xsd:sequence>
                  <xsd:element name= "nombre" type= "xsd:string"
                  MaxOccurs= "unbounded"/>
            </xsd:sequence>
```

```
      </xsd:complesType>
</xsd:element name= "vehículos">
```

- Tipo de complejo global y puede ser utilizado en la definición de cualquier elemento

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

- Se puede utilizar un mecanismo de "herencia" para reutilizar o extender tipos definidos previamente

```
<xsd:complexType name="nombre_tipo">
      <xsd:complexContent>
            <xsd:extension base="nombre_tipo_base">
                  …Información adicional sobre el tipo de base…
            </xsd:extensión>
      </xsd:complexContent>
</xsd:complexType>
```

6. Elementos compuestos. Un elemento complejo es aquel que contenga otro elemento y/o atributos. Hay cuatro tipos de elementos complejos:
   a. Elementos que contienen solo otros elementos

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

Un contenido xml válido sería:

```
<persona>
<apellido>Garcia</apellido>
<nombre>Juan</nombre>
</persona>
```

   b. Elementos vacíos. Elemento puesto de trabajo (que es vacío) y tiene un atributo categoría.

```
<xs:element name="puesto"
type="tipopuesto"/>
<xs:complexType name="tipopuesto">
<xs:attribute name="categoria"
type="xs:string"/>
</xs:complexType>
```

Un contenido xml válido sería:

```
<puesto
categoria="peon"/>
```

   c. Elementos que contienen solo texto

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

```
<numerozapato
pais="España">41</numerozapato>
```

d. Elementos que contienen otros elementos y texto. El atributo mixed es necesario.

```
<xs:element name="carta"
type="tipoCarta"/>
<xs:complexType
name="tipoCarta" mixed="true">
<xs:sequence>
<xs:element name="nombre"
type="xs:string"/>
<xs:element name="pedido"
type="xs:positiveInteger"/>
<xs:element name="fechaEntrega"
type="xs:date"/>
</xs:sequence>
</xs:complexType>
```

Un contenido xml válido sería:

```
<carta>
Querido Señor:<nombre>Juan
Garcia</nombre>.
Su pedido
<pedido>1111</pedido>
será entregado el
<fechaEntrega>2011-11-
11</fechaEntrega>.
</carta>
```

7. Indicadores. Sirven para controlar como utilizar los elementos en los documentos.
   a. All para indicar que los elementos hijo pueden aparecer en cualquier orden

```
<xs:element
name="persona">
<xs:complexType>
<xs:all>
   <xs:element
name="nombre"
type="xs:string"/>
   <xs:element
name="apellido"
type="xs:string"/>
</xs:all>
</xs:complexType>
</xs:element>
```

Un contenido xml válido sería:

<persona> <nombre>Juan</nombre>
<apellido>Garcia</apellido> </persona>

y también sería válido:
<persona> <apellido>Garcia</apellido>
<nombre>Juan</nombre></persona>

   b. Choice. Pueden aparecer cualquiera de las alternativas

```
<xs:element
name="persona">
<xs:complexType>
<xs:choice>
   <xs:element
```

Un contenido xml válido sería:

<persona>

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

<nombre>Juan</nombre></persona>

y también sería válido:
<persona> <nombrecompleto>Juan Garcia</nombrecompleto></persona>

c. Sequence. orden exacto de aparición

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

Un contenido xml válido sería:

<persona> <nombre>Juan</nombre>
<apellido>Garcia</apellido> </persona>

y NO sería válido:
<persona> <apellido>Garcia</apellido>
<nombre>Juan</nombre></persona>

d. maxOccurs y minOccurs. Indica respectivamente el máximo y mínimo número de veces que aparecerá un elemento

```
<xs:element name="persona">
<xs:complexType>
<xs:sequence>
<xs:element name="nombre" type="xs:string"/>
    <xs:element name="nombreHijos" type="xs:string"
maxOccurs="10" minOccurs="0"/>
</xs:sequence>
</xs:complexType>
</xs:element>
```

e. Grupos sirven para agrupar varios elementos y asi poderlos usar posteriormente sin tener que volver a enumeralos. También se puede hacer con los atributos con la etiqueta <xs:attributeGroup>

```
<xs:group name="grupoPersona">
<xs:sequence>
<xs:element name="nombre" type="xs:string"/>
<xs:element name="apellido" type="xs:string"/>
<xs:element name="cumpleaños" 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>
```