3. XML SCHEMA

1 INTRODUCCIÓN

Un esquema define la estructura de un documento XML. Es una alternativa a las DTD's. Se piensa que con el tiempo los esquemas reemplazarán a las DTD's.

Ventajas sobre las DTD:

- Usan sintaxis XML.
- Definen tipos de datos.

Los esquemas tienen extensión xsd.

Ejemplo:

Documento nota.xml:

```
<?xml version="1.0" encoding=”ISO-8859-1” ?>

    <nota xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance”
    xsi:noNamespaceSchemaLocation="nota.xsd">
    <para>Pedro</para>
    <de>Jose</de>
    <cabecera>Recordatorio</cabecera>
    <mensaje>No te olvides de echar la primitiva</mensaje>
    </nota>
```

Esquema asociado, en un documento llamado nota.xsd:

```
<?xml version="1.0" encoding=”ISO-8859-1”?>

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

<xs:element name="nota">

 <xs:complexType>

  <xs:sequence>

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

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

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

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

  </xs:sequence>

 </xs:complexType>

</xs:element>
```

</xs:schema>

El elemento raíz es schema.

2. ELEMENTOS SIMPLES

Son elementos que contienen sólo texto. No contienen otros elementos o atributos.

Definición de un elemento simple:

<xs:element name="xxx" type="yyy"/>

Dónde xxx es el nombre del element y yyy es el tipo de dato.

XML Schema tiene un montón de tipos de datos, los más comunes son:

- xs:string: Pueden llevar caracteres, retornos de carro, saltos de línea.
- xs:decimal: Para los decimales llevan un punto. Pueden llevar signo.
- xs:integer: Pueden llevar signo.
- xs:boolean:Los valores permitidos son: “true” ó 1 (para verdadero) y “false” ó 0 (para falso).
- xs:date: Formato yyyy-mm-dd
- xs:time: Formato hh:mm:dd

Ejemplo:

<apellido>Rodríguez</apellido>
<edad>36</edad>
<fecha-nacimiento>1970-03-27</fecha-nacimiento>

Las definiciones de los elementos simples serían:

<xs:element name="apellido" type="xs:string"/>
<xs:element name="edad" type="xs:integer"/>
<xs:element name="fecha-nacimiento" type="xs:date"/>

Valor por defecto: es asignado automáticamente al elemento cuando ningún otro valor es especificado. Por ejemplo:

<xs:element name="color" type="xs:string" default="rojo"/>

Valor fijo: es asignado automáticamente al elemento y no se puede asignar otro valor. Por ejemplo:

<xs:element name="color" type="xs:string" fixed="rojo”/>

Ejercicio: Añadir los siguientes datos a la nota del ejemplo anterior: fecha de recepción de la nota, hora, un indicador que me informe de si ha sido leído, el número de mensaje.

Restricciones

Se usan para definir valores aceptables de elementos o atributos. También se les llama "facetas".

Ejemplos:

-Conjunto de valores: Sólo admite uno de los valores especificados.

```
<xs:element name="coche">
 <xs:simpleType>
  <xs:restriction base="xs:string">
   <xs:enumeration value="Audi"/>
   <xs:enumeration value="Golf"/>
   <xs:enumeration value="BMW"/>
  </xs:restriction>
 </xs:simpleType>
</xs:element>
```

-Serie de valores:

Una letra minúscula:

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

Tres dígitos entre 0 y 9:

```
<xs:element name="codigo">
 <xs:simpleType>
  <xs:restriction base="xs:integer">
   <xs:pattern value="[0-9][0-9][0-9]"/>
  </xs:restriction>
 </xs:simpleType>
</xs:element>
```

O así:

```
<xs:element name="codigo">
 <xs:simpleType>
  <xs:restriction base="xs:integer">
   <xs:pattern value="[0-9]{3}"/>
  </xs:restriction>
 </xs:simpleType>
</xs:element>
```

Cero o más ocurrencias de letras minúsculas:

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

Cero o más ocurrencias de letras minúsculas y blancos:

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

Una o más ocurrencias de pares de letras: minúscula, mayúscula:

```
<xs:elemento name="letra">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="([a-z][A-Z])+"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

El elemento genero, solo puede tomar los valores masculino o femenino:

```
<xs:element name="genero">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="masculino|femenino"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

-Longitud:

El elemento password debe tener exactamente 8 caracteres:

```
<xs:element name="password">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:length value="8"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

El elemento password tiene al menos 5 caracteres y como máximo 8 caracteres:

```
<xs:element name="password">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:minLength value="5"/>
      <xs:maxLength value="8"/>
    </xs:restriction>
  </xs:simpleType>
```

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

También se utilizan:

maxExclusive: Valor máximo permitido excluido.

maxInclusive: Valor máximo permitido incluido.

minExclusive: Valor mínimo permitido excluido.

minInclusive: Valor mínimo permitido incluido.

totalDigits: Número exacto de dígitos permitidos (mayor que 0).

fractionDigits: Máximo número de decimales permitidos (mayor o igual a 0).

3. ELEMENTOS COMPLEJOS

Un elemento complejo es un elemento que contiene otros elementos, que contiene atributos o ambas cosas.

3.1 Elementos que contienen otros elementos

Secuencia de elementos respetando el orden: sequence

Ejemplo:

```
<persona>
  <nombre>John</nombre>
  <apellido>Smith</apellido>
</persona>
```

En Schema:

```
<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>
```

Número de ocurrencias: maxOccurs, minOccurs

Indican respectivamente el número máximo y mínimo de veces que puede aparecer un elemento. Por defecto el valor de maxOccurs y minOccurs es de 1. Para indicar un número ilimitado de ocurrencias se utiliza maxOccurs=unbounded.

Ejemplo:

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

Validaría el siguiente xml:

```
<persona>

    <nombre-completo>Federico Sargazo</nombre-completo>

</persona>
```

Y el siguiente:

```
<persona>

    <nombre-completo>Federico Sargazo</nombre-completo>

   <nombre-hijo>Lourdes</nombre-hijo>

   <nombre-hijo>Jose</nombre-hijo>

</persona>
```

También admitiría un xml con hasta 10 hijos, pero ni uno más.

Ejemplo de como funciona xmlcopy:

myfamily.xml:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<persons xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="family.xsd">

<person>
 <full_name>Hege Refsnes</full_name>
 <child_name>Cecilie</child_name>
</person>

<person>
 <full_name>Tove Refsnes</full_name>
 <child_name>Hege</child_name>
 <child_name>Stale</child_name>
 <child_name>Jim</child_name>
 <child_name>Borge</child_name>
</person>

<person>
 <full_name>Stale Refsnes</full_name>
</person>

</persons>
```

Fichero family.xsd:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="persons">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="person" maxOccurs="unbounded">
    <xs:complexType>
     <xs:sequence>
      <xs:element name="full_name" type="xs:string"/>
      <xs:element name="child_name" type="xs:string"
      minOccurs="0" maxOccurs="5"/>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
  </xs:sequence>
 </xs:complexType>
</xs:element>
</xs:schema>
```

Fichero family.xsd según XMLCopy:

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

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

 <xs:element name="persons">

  <xs:complexType>

   <xs:sequence>

    <xs:element ref="person" maxOccurs="unbounded"/>

   </xs:sequence>

  </xs:complexType>

 </xs:element>

 <xs:element name="person">

  <xs:complexType>

   <xs:sequence>

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

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

   </xs:sequence>

  </xs:complexType>

 </xs:element>

</xs:schema>
```

Mediante el atributo ref hace referencia al elemento person que luego define más adelante.

Secuencia de elementos en cualquier orden: all

Cada elemento sólo puede ocurrir como máximo una vez.

```
<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>
```

Valida:

```
<persona>

   <nombre>Gustavo </nombre>

   <apellido>Frog</apellido>

</persona>
```

También valida:

```
<persona>

   <apellido>Frog</apellido>

   <nombre>Gustavo </nombre>

</persona>
```

Mezclar texto y otros elementos: mixed

En el siguiente ejemplo “Estimado Sr.” es texto y luego aparecen otros elementos:

```
<carta>
 Estimado Sr.<nombre>John Smith</nombre>.
 Su pedido <numeropedido>1032</numeropedido>
 sera enviado <fecha-envio>2001-07-13</fecha-envio>.
</carta>
```

En Schema:

```
<xs:element name="carta">
 <xs:complexType mixed="true">
  <xs:sequence>
   <xs:element name="nombre" type="xs:string"/>
   <xs:element name="numeropedido" type="xs:positiveInteger"/>
   <xs:element name="fecha-envio" type="xs:date"/>
  </xs:sequence>
```

</xs:complexType>
</xs:element>

Elegir entre varios elementos: choice

Este Schema:

```
<xs:element name="persona">
  <xs:complexType>
    <xs:choice>
      <xs:element name="empleado" type="xs:string"/>
      <xs:element name="miembro" type="xs:string"/>
    </xs:choice>
  </xs:complexType>
</xs:element>
```

Se podría aplicar a los siguientes XML:

```
<persona>

    <empleado>Juan</empleado>

</persona>
```

Y a:

```
 <persona>

     <miembro>Luis</miembro>

</persona>
```

Grupo de elementos:Group

Se utiliza para definir grupos de elementos.

Ejemplo este fragmento:

```
<xs:group name="persongroup">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
    <xs:element name="birthday" type="xs:date"/>
  </xs:sequence>
</xs:group>


<xs:element name="person" >
<xs:complexType >
  <xs:sequence>
    <xs:group ref="persongroup"/>
    <xs:element name="country" type="xs:string"/>
  </xs:sequence>
</xs:complexType>

</element>
```

Es igual que:

```
<xs:element name="person" >
  <xs:complexType >
    <xs:sequence>

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

   <xs:element name="lastname" type="xs:string"/>
      <xs:element name="birthday" type="xs:date"/>
      <xs:element name="country" type="xs:string"/>
   </xs:sequence>
 </xs:complexType>

</xs:element>
```

3.2.Elementos con atributos

Atributos

La sintaxis es:

<xs:attribute name="xxx" type="yyy"/>

Dónde xxx es el nombre del atributo y yyy es el tipo.

Ejemplo:

<titulo idioma="francés">Le petit prince</titulo>

El atributo idioma quedaría:

<xs:attribute name="idioma" type="xs:string"/>

También pueden tener valor por defecto o fixed igual que en los elementos.

Por defecto los atributos son opcionales. Si quiero que un atributo sea obligatorio :

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

Elemento vacío con atributo:

<producto identificador="123" />

Schema asociado:

<xs:element name="producto">

```
 <xs:complexType>
  <xs:attribute name="identificador" type="xs:Integer"/>
 </xs:complexType>
</xs:element>
```

Elemento con datos y atributo:

<producto identificador="123" > impresora</producto>

Schema asociado:

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

XMLCopy lo resuelve así:

```
 <xs:element name="producto">
  <xs:complexType mixed="true">
   <xs:attribute name="identificador" type="xs:integer" use="required"/>
  </xs:complexType>
 </xs:element>
```

Esta solución sólo me sirve si el elemento es de tipo texto, sino no.

Ejemplo:

<punto-ebullición unidades="kelvin">20.28</punto-ebullición>

Para este elemento XMLCopy genera el código:

```
<xs:element name="punto-ebullición">
```

```
  <xs:complexType mixed="true">
    <xs:attribute name="unidades" type="xs:string" use="required"/>
  </xs:complexType>
 </xs:element>
```

Este código no me valida que el tipo del elemento sea un decimal. La forma correcta seríala primera que hemos visto:

```
<xs:element name="punto-ebullición" >
        <xs:complexType>
            <xs:simpleContent>
                <xs:extension base="xs:decimal">
                    <xs:attribute name="unidades" type="xs:string" />
                </xs:extension>
            </xs:simpleContent>
        </xs:complexType>
   </xs:element>
```

Elementos con otros elementos y atributos:

Este fragmento xml:

```
<ficha categoria="empresa" zona="periferia">
            <nombre>Ana</nombre>
            <apellido1>Pérez</apellido1>
            <apellido2>Sanz</apellido2>
            <email>qwr-@d.com</email>
 </ficha>
```

Se transforma en:

```
<xs:element name="ficha">
  <xs:complexType>
    <xs:sequence>
```

```
      <xs:element name="nombre" type="xs:string"/>
      <xs:element name="apellido1" type="xs:string"/>
      <xs:element name="apellido2" type="xs:string"/>
      <xs:element name="email" type="xs:string"/>
    </xs:sequence>
    <xs:attribute name="categoria" type="xs:string" use="required"/>
    <xs:attribute name="zona" type="xs:string" use="required"/>
  </xs:complexType>
</xs:element>
```

Elementos con restricciones y atributos:

Más adelante cuando veamos los tipos definidos por el usuario.

4. TIPOS PREDEFINIDOS

Podemos crear nuestros propios tipos simples o complejos, y luego hacer referencia a ellos.

Ejemplo:

```
<xs:simpleType name=”miTipo”>
<xs:restriction base=”xs:string”>
<xs:maxLength value=”32”/>
</xs:restriction>
</xs:simpleType>
```

Estoy definiendo un tipo llamado miTipo que es un string de tamaño 32, luego lo puedo utilizar como tipo en cualquier elemento simple.

Ejemplo con tipos simples y complejos definidos por el programador:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<!-- derinición tipos simples -->
<xs:simpleType name="nombreType">
```

```
            <xs:restriction base="xs:string">
                        <xs:maxLength value="32"/>
            </xs:restriction>
</xs:simpleType>

<xs:simpleType name="codigoType">
            <xs:restriction base="xs:string">
                        <xs:pattern value="[0-9]{10}"/>
            </xs:restriction>
</xs:simpleType>

<!-- definición tipos complejos -->

<xs:complexType name="articuloType">
            <xs:sequence>
                        <xs:element name="titulo" type="nombreType"/>
             <xs:element name="autor" type="nombreType" minOccurs="0" maxOccurs="unbounded"/>
             <xs:element name="numeropalabras" type="xs:integer"/>
             <xs:element name="texto" type="xs:string"/>
            </xs:sequence>
</xs:complexType>

<xs:complexType name="revistaType">
            <xs:sequence>
            <xs:element name="nombre" type="nombreType"/>
              <xs:element name="fechasalida" type="xs:date"/>
              <xs:element name="articulo" type="articuloType" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>

  <xs:attribute name="codigo" type="codigoType" use="required"/>
```

</xs:complexType>

<!--creación o instanciación del elemento revista-->

<xs:element name="revista" type="revistaType"/>

</xs:schema>

Un xml válido para este eschema sería:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<revista xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="revistas.xsd" codigo="0123456789">
        <nombre>El semanal</nombre>
        <fechasalida>2011-02-13</fechasalida>
        <articulo>
                <titulo>Caida de Mubarak</titulo>
                <autor>Pedro Jiménez</autor>
                <numeropalabras>70</numeropalabras>
                <texto>Egipto se ha echado a la calle ante la noticia de que ...bla, bla, bla</texto>
        </articulo>
</revista>
```