EJEMPLOS DE USO XSD.

Ejercicio: edad de los trabajadores

Se desea crear un esquema que permita validar la edad de un trabajador, que debe tener un valor entero de entre 16 y 65.

Por ejemplo, este XML debería validarse:

```
<edad>28</edad>
```

Pero este no debería validarse:

```
<edad>-3</edad>
```

La solución del esquema XSD podría ser algo así:

```
<xsd:schema
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <xsd:element name="edad"
              type="tipoEdad"/>
   <xsd:simpleType name="tipoEdad">
      <xsd:restriction base="xsd:integer">
         <xsd:minInclusive value="16"/>
         <xsd:maxInclusive value="65"/>
      </xsd:restriction>
   </xsd:simpleType>
</xsd:schema>
```

Ejercicio: peso de productos

Se desea crear un esquema que permita validar un elemento peso, que puede tener un valor de entre 0 y 1000 pero aceptando valores con decimales, como por ejemplo 28.88

Una posible solución de esquema XSD sería:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="peso" type="tipoPeso"/>
  <xsd:simpleType name="tipoPeso">
    <xsd:restriction base="xsd:decimal">
      <xsd:minInclusive value="0"/>
```

```
      <xsd:maxInclusive value="1000"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:schema>
```

Ejercicio: pagos validados

Crear un esquema XSD que permita validar un elemento pago en el cual puede haber cantidades enteras de entre 0 y 3000 euros.

```
<xsd:schema
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="pago" type="tipoPago"/>
  <xsd:simpleType name="tipoPago">
    <xsd:restriction base="xsd:integer">
      <xsd:minInclusive value="0"/>
      <xsd:maxInclusive value="3000"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:schema>
```

Ejercicio: validación de DNI

Crear un esquema XSD que permita validar un único elemento dni que valide el patrón de 7-8 cifras + letra que suelen tener los DNI en España:

```
<xsd:schema
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="dni" type="tipoDNI"/>
  <xsd:simpleType name="tipoDNI">
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="[0-9]{7,8}[A-Z]"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:schema>
```

Uniendo la herencia y el sistema de tipos

Llegados a este punto ocurre lo siguiente:

- Por un lado tenemos que especificar si nuestros tipos serán simples o complejos (los cuales a su vez pueden ser complejos con contenido simple o complejos con contenido complejo).
- Por otro lado se puede hacer herencia ampliando cosas (extensión) o reduciendo cosas (restricciones a los valores).

Se deduce por tanto que no podemos aplicar todas las «herencias» a todos los tipos:

1. Los tipos simples no pueden tener atributos ni subelementos, por lo tanto **les podremos aplicar restricciones pero nunca la extensión**.
2. Los tipos complejos (independientemente del tipo de contenido) sí pueden tener otras cosas dentro por lo que **les podremos aplicar tanto restricciones como extensiones**.

Restricciones

Como se ha dicho anteriormente la forma más común de trabajar es crear tipos que en unos casos aplicarán modificaciones en los tipos ya sea añadiendo cosas o restringiendo posibilidades. En este apartado se verá como aplicar restricciones.

Si queremos aplicar restricciones para un tipo simple las posibles restricciones son:

- minInclusive para indicar el menor valor numérico permitido.
- maxInclusive para indicar el mayor valor numérico permitido.
- minExclusive para indicar el menor valor numérico que ya no estaría permitido.
- maxExclusive para indicar el mayor valor numérico que ya no estaría permitido.
- totalDigits para indicar cuantas posibles cifras se permiten.
- fractionDigits para indicar cuantas posibles cifras decimales se permiten.
- length para indicar la longitud exacta de una cadena.
- minLength para indicar la longitud mínima de una cadena.
- maxLength para indicar la longitud máxima de una cadena.
- enumeration para indicar los valores aceptados por una cadena.

- pattern para indicar la estructura aceptada por una cadena.

Si queremos aplicar restricciones para un tipo complejo con contenido las posibles restricciones son las mismas de antes, pero además podemos añadir el elemento <attribute> así como las siguientes.

Atributos

En primer lugar es muy importante recordar que **si queremos que un elemento tenga atributos entonces ya no se puede considerar que sea de tipo simple. Se debe usar FORZOSAMENTE un complexType**. Por otro lado en los XML Schema todos los atributos **son siempre opcionales, si queremos hacerlos obligatorios habrá que añadir un «required».**

Un atributo se define de la siguiente manera:

```
<xsd:attribute
    name="fechanacimiento"type="xsd:date"use="required"/>
```

Esto define un atributo llamado nombre que aceptará solo fechas como valores válidos y que además es obligatorio poner siempre.

Ejercicios de XML Schemas

Cantidades limitadas

Crear un esquema que permita verificar algo como lo siguiente:

```
<cantidad>20</cantidad>
```

Se necesita que la cantidad tenga solo valores aceptables entre -30 y +30.

Solución a las cantidades limitadas

La primera pregunta que debemos hacernos es ¿necesitamos crear un tipo simple o uno complejo?. Dado que nuestro único elemento no tiene subelementos ni atributos dentro podemos afirmar que solo necesitamos un tipo simple.

Como aparentemente nuestro tipo necesita usar solo valores numéricos y además son muy pequeños nos vamos a limitar a usar un short. Sobre ese short pondremos una restricción que permita indicar los valores mínimo y máximo.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:element name="cantidad">
      <xs:simpleType>
         <xs:restriction base="xs:short">
            <xs:minInclusive value="-30"/>
            <xs:maxInclusive value="30"/>
         </xs:restriction>
      </xs:simpleType>
   </xs:element> </xs:schema>
```

Este esquema dice que el elemento raíz debe ser cantidad. Luego indica que es un tipo simple y dentro de él indica que se va a establecer una restricción teniendo en mente que se va a «heredar» del tipo short. En concreto se van a poner dos restricciones, una que el valor mínimo debe ser -30 y otra que el valor máximo debe ser 30.

Existe una alternativa más recomendable, que es separar los elementos de los tipos. De esa manera, se pueden «reutilizar» las definiciones de tipos.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:element name="cantidad" type="tipoCantidades">
   </xs:element>
   <xs:simpleType name="tipoCantidades">
         <xs:restriction base="xs:short">
            <xs:minInclusive value="-30"/>
            <xs:maxInclusive value="30"/>
         </xs:restriction>
      </xs:simpleType>
</xs:schema>
```

Obsérvese que hemos puesto el tipo por separado y le hemos dado el nombre tipoCantidades. El elemento raíz tiene su nombre y su tipo en la misma línea.

Cantidades limitadas con atributo divisa

Se desea crear un esquema para validar XML en los que haya un solo elemento raíz llamado cantidad en el que se debe poner siempre un atributo «divisa» que indique en qué moneda está una cierta cantidad. El atributo divisa siempre será una cadena y la cantidad siempre será un tipo numérico que acepte decimales (por ejemplo float). El esquema debe validar los archivos siguientes:

```
<cantidad>divisa="euro">20</cantidad>

<cantidad>divisa="dolar">18.32</cantidad>
```

Pero no debe validar ninguno de los siguientes:

```
<cantidad>20</cantidad>

<cantidad>divisa="dolar">abc</cantidad>
```

Solución a las cantidades limitadas con atributo divisa¶

Crearemos un tipo llamado «tipoCantidad». Dicho tipo *ya no puede ser un simpleType ya que necesitamos que haya atributos*. Como no necesitamos que tenga dentro subelementos entonces este complexType llevará dentro un simpleContent (y no un complexContent).

Aparte de eso, como queremos «ampliar» un elemento para que acepte tener dentro un atributo obligatorio «cantidad» usaremos una <extension>. Así, el posible esquema sería este:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <xsd:element name="cantidad" type="tipoCantidad"/>
   <xsd:complexType name="tipoCantidad">
      <xsd:simpleContent>
         <xsd:extension base="xsd:float">
            <xsd:attribute name="divisa" type="xsd:string" use="required"/>
```

```
        </xsd:extension>
      </xsd:simpleContent>
   </xsd:complexType>
</xsd:schema>
```

Cantidades limitadas con atributo divisa con solo ciertos valores

Queremos ampliar el ejercicio anterior para evitar que ocurran errores como el siguiente:

```
<cantidad>divisa="aaaa">18.32</cantidad>
```

Vamos a indicar que el atributo solo puede tomar tres posibles valores: «euros», «dolares» y «yenes».

Solución al atributo con solo ciertos valores

Ahora tendremos que crear dos tipos. Uno para el elemento cantidad y otro para el atributo divisa. Llamaremos a estos tipos tipoCantidad y tipoDivisa.

La solución comentada puede encontrarse a continuación. Como puede verse, hemos incluido comentarios. Pueden insertarse etiquetas annotation que permiten incluir anotaciones de diversos tipos, siendo la más interesante la etiqueta documentation que nos permite incluir comentarios.

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <xsd:element name="cantidad" type="tipoCantidad"/>
   <xsd:annotation>
      <xsd:documentation>
      A continuación creamos el tipo cantidad
      </xsd:documentation>
   </xsd:annotation>
   <xsd:complexType name="tipoCantidad">
      <xsd:annotation>
         <xsd:documentation>
         Como solo va a llevar atributos debemos
         usar un simpleContent
         </xsd:documentation>
      </xsd:annotation>
      <xsd:simpleContent>
         <xsd:annotation>
```

```
            <xsd:documentation>
            Como queremos "ampliar" un tipo/clase
            para que lleve atributos usaremos
            una extension
            </xsd:documentation>
         </xsd:annotation>
         <xsd:extension base="xsd:float">
            <xsd:attribute name="divisa" type="tipoDivisa"/>
         </xsd:extension>
      </xsd:simpleContent>
   </xsd:complexType>
   <xsd:annotation>
      <xsd:documentation>
      Ahora tenemos que fabricar el "tipoDivisa" que indica
      los posibles valores válidos para una divisa. Estas
      posibilidades se crean con una "enumeration". Nuestro
      tipo es un "string" y como vamos a restringir los posibles
      valores usaremos "restriction"
      </xsd:documentation>
   </xsd:annotation>
   <xsd:simpleType name="tipoDivisa">
      <xsd:restriction base="xsd:string">
         <xsd:enumeration value="euros"/>
         <xsd:enumeration value="dolares"/>
         <xsd:enumeration value="yenes"/>
      </xsd:restriction>
   </xsd:simpleType>
</xsd:schema>
```

1º Ejercicio: lista de códigos

Se nos pide crear un esquema que permita validar un fichero XML como el siguiente:

```
<listacodigos>
  <codigo>AAA2DD</codigo>
  <codigo>BBB2EE</codigo>
  <codigo>BBB2EE</codigo>
</listacodigos>
```

En concreto, todo código tiene la estructura siguiente:

1. Primero van tres mayúsculas
2. Después va exactamente un digito.
3. Por último hay exactamente dos mayúsculas.

Un posible esquema XSD sería el siguiente (obsérvese como usamos maxOccurs para indicar que el elemento puede repetirse un máximo de «infinitas veces»:

```
<xsd:schema
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="listacodigos"
               type="tipoLista"/>
  <xsd:complexType name="tipoLista">
    <xsd:complexContent>
      <xsd:restriction base="xsd:anyType">
        <xsd:sequence>
          <xsd:element name="codigo"
                       type="tipoCodigo"
                       maxOccurs="unbounded"/>
        </xsd:sequence>
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:simpleType name="tipoCodigo">
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="[A-Z]{3}[0-9][A-Z]{2}"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:schema>
```