# Apuntes XSD

## ¿Qué es y para qué sirve?

Los XML Schema son ficheros en texto plano con la extensión XSD que describen la estructura de un archivo XML. De forma que podemos determinar si un archivo XML cumple la estructura definida o no.

## Asociar XML con un archivo XSD

Se hace añadiendo la siguiente instrucción a su nodo raíz:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<clase xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="alumnos.xsd">
  <alumno>
    <nombre>José Ramón</nombre>
    <apellidos>García González</apellidos>
  </alumno>
  <alumno>
    <nombre>Carlos</nombre>
    <apellidos>López Pérez</apellidos>
  </alumno>
</clase>
```

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, aunque se puede utilizar otros como xs.

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 “alumnos.xsd”.

## Estructura básica del archivo XSD

La base de un archivo XSD es la siguiente:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="root-element">
   <!-- Aquí escribes tu código -->
  </xs:element>
</xs:schema>
```

# Definir elementos

Para definir un elemento debemos tener claro que existen dos tipos de elementos simples y complejos.

Un **elemento simple** no contiene atributos y no contiene otros elementos (como hijos). Ejemplo:

```
<lastname>Refsnes</lastname>
<age>36</age>
<dateborn>1970-03-27</dateborn>
```

Para escribir su código XSD se pondrá: *(especificamos el nombre del elemento y el tipo de dato)*

```
<xs:element name="lastname" type="xs:string"/>
<xs:element name="age" type="xs:integer"/>
<xs:element name="dateborn" type="xs:date"/>
```

Un **elemento complejo** contiene atributos y/o otros elementos. Ejemplos:

```
<employee rol="admin">
    <firstname>John</firstname>
    <lastname>Smith</lastname>
</employee>
```

El código XSD para validar este elemento será:

```
<xs:element name="employee">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="firstname" type="xs:string"/>
            <xs:element name="lastname" type="xs:string"/>
        </xs:sequence>
        <xs:attribute name="rol" type="xs:string"/>
    </xs:complexType>
</xs:element>
```

# Cuando un elemento se repite

Si un elemento se repite deberemos utilizar minOccurs y maxOccurs para determinar su cardinalidad

```
<propiedades>
    <casa>Calle Batalla de Lepanto</casa>
    <casa>Avenida Tenor Fleta</casa>
    <casa>Calle Rio Ebro</casa>
</propiedades>
```

Su XSD para determinar su cardinalidad puede ser: *(indica que puede aparecer al menos 1 y como máximo ilimitado)*

```
<xs:element name="properties">
    <xs:complexType>
      <xs:sequence>
          <xs:element name="casa" type="xs:string" minOccurs="1" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
</xs:element>
```

Sino se especifica por defecto tomando los valores minOccurs 1 y maxOccurss 1. También se utiliza para definir que un elemento es opcional.

# Tipos de elementos

El contenido de un elemento o atributo puede tener distintos valores según el tipo de dato:

- xs:**string** Cadena de texto alfanumérica.
- xs:**decimal** Números decimales.
- xs:**integer** Números enteros.
- xs:**boolean** Booleano true o false.
- xs:**date** Tipo fecha, también existe para year, month, day, time, etc.
- xs:**ID** Una clave primaria que ningún otro elemento podrá repetir.
- xs:**IDREF** Hace referencia a una campo ID que ya exista en el documento.

# Atributos

Ya hemos visto un ejemplo. Se puede especificar el valor use con required o optional. Para indicar si el atributo es obligatorio o no. Por defecto es obligatorio.

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

# El orden de los hijos de un elemento

Para definir los hijos de un elemento y el orden en el que deben aparecer utilizados **sequence**

```
<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>
```

Para definir que pueden aparecer desordenados usamos **all** (no se pueden repetir los elementos con maxOccurs. Al menos de forma facil)

```
<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>
```

Para definir que solo puede aparecer uno de los especificados ponemos **choice**

```
<xs:element name="persona">
    <xs:complexType>
        <xs:choice>
            <xs:element name="alumno" type="xs:string"/>
            <xs:element name="profesor" type="xs:string"/>
        </xs:choice>
    </xs:complexType>
</xs:element>
```

# Restricciones de los elementos

Además del tipo de dato podemos especificar restricciones sobre ese datos. Por ejemplo, establecer rangos en un número, longitud máxima en una cadena de texto, o expresiones regulares.

Para añadir una restricción a un elemento simple debemos eliminar el atributo type y escribirlo de la siguiente forma:

```
<xs:element name="nota">
    <xs:simpleType>
        <xs:restriction base="xs:integer">
            <xs:minInclusive value="0"/>
            <xs:maxInclusive value="10"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
```

Alguna de las restricciones que podemos utilizar son:

- **maxInclusive** y **minInclusive** Maximo y mínimo valores numéricos permitidos
- **maxLength** y **minLength** Maximo y mínimo número de caracteres permitido
- **pattern** Patrón en expresión regular
- **totalDigits** Número total de digitos
- **fractionDigits** El número de decimales permitidos
- **enumeration** Una lista de valores aceptados

Ejemplos:

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

# Casos especiales

Los **elementos vacíos** se pueden especificar de muchas maneras distintas:

Forma 1:

```
<xs:element name="myEmptyElement">
    <xs:complexType/>
</xs:element>
```

Forma 2:

```
<xs:element name="myEmptyElement">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:maxLength value="0"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
```

Forma 3:

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

Forma 4:

```
<xs:element name="myEmptyElement"/>
```

Elementos con **contenido simple, pero con un atributo**:

```
<employee id="1">John Smith</employee>
```

La forma correcta sería:

```
<xs:element name="employee">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="xs:ID"/>
            </xs:extension  >
        </xs:simpleContent >
    </xs:complexType >
</xs:element>
```

# Expresiones regulares

Una expresión regular o RegEx es una secuencia de caracteres que forman un patrón. Se puede utilizar para búsquedas o validaciones, de tal forma que podremos evaluar si se cumple o no dicho patrón.

## Expresiones regulares más utilizadas

**Blackslash o barra invertida: **
Se utiliza para escapar el siguiente carácter de la expresión de búsqueda de forma que este adquiera un significado especial o deje de tenerlo. O sea, la barra inversa no se utiliza nunca por sí sola, sino en combinación con otros caracteres. Al utilizarlo por ejemplo en combinación con el punto "\." este deja de tener su significado normal y se comporta como un carácter literal.

**Pipe o barra: |**
Sirve para indicar una de varias opciones. Por ejemplo la expresión regular (indexingdata|seo|sem) permitirá encontrar cualquiera de esas palabras.

**Punto: .**
El punto se interpreta como cualquier carácter, es decir, busca cualquier carácter o incluso un espacio SIN incluir los saltos de línea.

**Asterisco: ***
Sirve para encontrar algo que se encuentra repetido 0 o más veces. Su utilización más frecuente es junto al punto .* para indicar "cualquier cadena alfanumérica"

**Circunflejo: ^**
Permite indicar cuál es el inicio de una cadena de texto.

**Dólar: $**
Permite indicar cuál es el final de una cadena de texto.

**Paréntesis: ( )**
Permiten agrupar una consulta, de manera que podamos construir partes de una RegEx mayor, o garantizar que no se mezclan las cadenas.

**Corchetes: [ ]**
Permiten indicar un intervalo de números o caracteres, de manera que cualquiera incluido en el intervalo valide la consulta.

**Interrogación: ?**
Indica que el carácter o cadena de delante del signo es opcional.

**[abc] :**
Uno de los caracteres" a", "b", o "c"

**[^abc] :**
Cualquier carácter menos: "a", "b", o "c"

**[a-z] :**
Cualquier minúscula entre la ”a” y la “z”

**[0-9] :**
Cualquier número entre el “0” y el “9”

**[a-zA-Z] :**
Cualquier mayúscula o minúscula entre la “aA” y la “zZ”

**[a-zA-Z0-9-] :**
Cualquier letra o número o el guión (ideal para URLs)

**.* :**
Lo de antes puede existir ninguna vez o más

**.+ :**
Lo de antes puede existir una vez o más

**{X}**
Lo de antes debe repetirse X número de veces

**{X, Y}**
Lo de antes debe repetirse un número de veces entre X e Y

| Anchors | |
|---|---|
| ^ | Start of line |
| $ | End of line |

| Character Classes | |
|---|---|
| \s | White space character |
| \S | Non-white space character |
| \d | Digit character |
| \D | Non-digit character |
| \w | Word |
| \W | Non-word (e.g. punctuation, spaces) |

| Metacharacters (must be escaped) | | |
|---|---|---|
| ^ | [ | ] |
| $ | ( | ) |
| . | { | } |
| * | + | ? |
| \ | \| | - |

| GA Filter group accessors | |
|---|---|
| $Ax | Access group x in field A (e.g. $A1) |
| $Bx | Access group x in field B (e.g. $B1) |

| Quantifiers | |
|---|---|
| * | Zero or more (greedy) |
| *? | Zero or more (lazy) |
| + | One or more (greedy) |
| +? | One or more (lazy) |
| ? | Zero or one (greedy) |
| ?? | Zero or one (lazy) |
| {X} | Exactly X (e.g. 3) |
| {X,} | X or more, (e.g. 3) |
| {X, Y} | Between X and Y (e.g. 3 and 5) (lazy) |

| Ranges and Groups | |
|---|---|
| . | Any character |
| (a\|b) | a or b (case sensitive) |
| (...) | Group, e.g. (keyword) |
| (?:...) | Passive group, e.g. (?:keyword) |
| [abc] | Range (a or b or c) |
| [^abc] | Negative range (not a or b or c) |
| [A-Z] | Uppercase letter between A and Z |
| [a-z] | Lowercase letter between a and z |
| [0-7] | Digit between 0 and 7 |

**Sample Patterns**

**^/directory/(.*)**
*Any page URLs starting with /directory/*
**(brand\s*?term)**
*Brand term with or without whitespace between words*
**^brand\s+[^cf]**
*Key phrases beginning with 'brand' and the second word not starting with c or f*
**\.aspx$**
*URLs ending in '.aspx'*
**ORDER\-\d{6}**
*"ORDER-" followed by a six digit ID*
**(?:\?|&)utm=([^&$]+)**
*Value of 'utm' querystring parameter*