TEMA5: DEFINICIÓN DE ESQUEMAS EN XML

Contenido del tema:

- ✓ Definición de XML Schema
- ✓ Estructura y sintaxis de XML
- ✓ XML Schemas bien formados

Definición de XML Schema:

Los DTD permiten diseñar un vocabulario para ficheros XML, pero, ¿qué sucede cuando los valores de los elementos y atributos de esos ficheros han de corresponder a datos de un tipo determinado, o cumplir determinadas restricciones que no pueden reflejarse en los DTD? Para ello se definen XML Schemas.

En este caso la extensión de los archivos es .xsd. El XSD está basado en XML, por tanto, debe cumplir con las reglas de XML:

- Siempre se empieza el fichero con la declaración XML.
- Sólo hay un elemento raíz, que en este caso es <schema>.

El ejemplar de estos ficheros es <xs:schema> y contiene declaraciones para todos los elementos y atributos que puedan aparecer en un documento XML asociado válido. Los elementos hijos inmediatos de este ejemplar se definen con la etiqueta <xs:element> y con atributos se especifica, como mínimo, el nombre y, opcionalmente, el tipo de datos que contendrá el elemento.

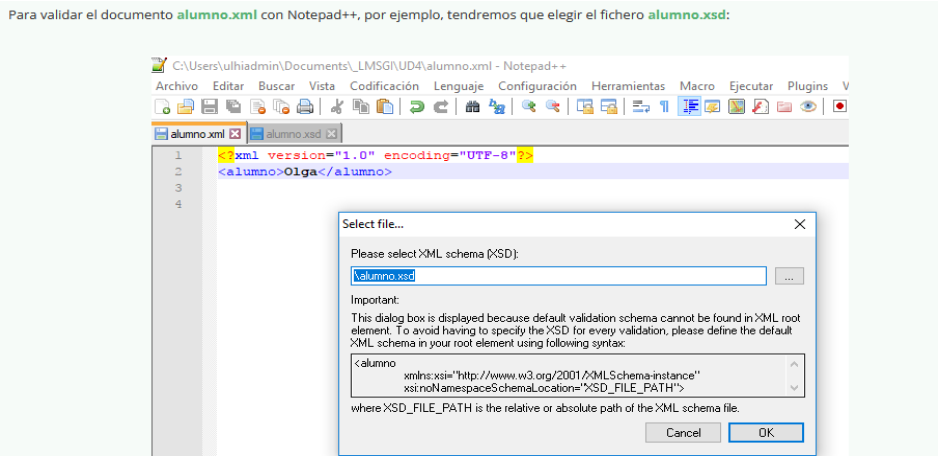
Creación de un esquema correspondiente al siguiente documento XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<alumno>Olga Velarde Cobo</alumno>
```

Si creamos el siguiente documento "**alumno.xsd**":

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="alumno" type="xs:string"/>
</xs:schema>
```

Para validar el documento **alumno.xml** con Notepad++, por ejemplo, tendremos que elegir el fichero **alumno.xsd**:



Una vez que tenemos creado el fichero XSD ¿cómo lo asociamos a un fichero XML?

El modo de asociar un esquema a un documento XML es un espacio de nombres (ns) al ejemplar del documento, donde se indica la ruta de localización de los ficheros esquema mediante su URI, precedida del prefijo " xsi:".

Además, se deberá especificar el espacio de nombres de XSD con el atributo:

- xmlns: "http://www.w3.org/2001/XMLSchema-instance".

Vamos a crear el esquema correspondiente al siguiente documento XML **alumnos.xml** para estructurar la información personal sobre los alumnos de un centro

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

  <alumno>
    <nombre>Jose</nombre>
    <apellidos>García González</apellidos>
    <direccion>
      <domicilio>Urbieta, 12</domicilio>
      <codigo_postal>20120</codigo_postal>
      <localidad>Hernani</localidad>
      <provincia>Gipuzkoa</provincia>
    </direccion>
    <contactar>
      <telf_casa>943623165</telf_casa>
      <telf_movil>611233544</telf_movil>
      <telf_trabajo>943847536</telf_trabajo>
      <email>pepito@ulhi.net</email>
    </contactar>
  </alumno>
  <alumno>
    <nombre>Aitana</nombre>
    <apellidos>Artola Agirre</apellidos>
    <direccion>
      <domicilio>Puga Kalea, 25</domicilio>
      <codigo_postal>01003</codigo_postal>
      <localidad>Gasteiz</localidad>
      <provincia>Araba</provincia>
    </direccion>
    <contactar>
      <telf_casa>945132565</telf_casa>
      <telf_movil>623863544</telf_movil>
      <telf_trabajo>945657536</telf_trabajo>
      <email>aitana@ulhi.net</email>
    </contactar>
  </alumno>
 </alumnos>
```

El esquema alumnos.xsd sería el siguiente:

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

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

<!-- elemento raiz -->

<xs:element name="alumnos" type="datosAlum"/>

<!-- Definicion del tipo datosAlum -->

  <xs:complexType name="datosAlum">
    <xs:sequence>
      <xs:element name="alumno" type="datos" minOccurs="1" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

<!-- Definicion del tipo datos -->

<xs:complexType name="datos">
  <xs:sequence>
    <xs:element name="nombre" type="xs:string" minOccurs="1"  maxOccurs="1"/>
    <xs:element name="apellidos" type="xs:string" minOccurs="1" maxOccurs="1"/>
    <xs:element name="direccion" type="datosDireccion" minOccurs="1" maxOccurs="1"/>
    <xs:element name="contactar" type="datosContactar" minOccurs="1" maxOccurs="1"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="datosDireccion">
  <xs:sequence>
    <xs:element name="domicilio" type="xs:string" minOccurs="0" maxOccurs="1"/>
    <xs:element name="codigo_postal" minOccurs="0" maxOccurs="1" />
    <xs:element name="localidad" type="xs:string" minOccurs="0" maxOccurs="1"/>
    <xs:element name="provincia" type="xs:string" minOccurs="0" maxOccurs="1"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="datosContactar">
  <xs:sequence>
    <xs:element name="telf_casa" type="xs:string" minOccurs="0" maxOccurs="1"/>
    <xs:element name="telf_movil" type="xs:string" minOccurs="0" maxOccurs="1"/>
    <xs:element name="telf_trabajo" type="xs:string" minOccurs="0" maxOccurs="1"/>
    <xs:element name="email" type="xs:string" />
  </xs:sequence>
</xs:complexType>
</xs:schema>
```

Elementos del lenguaje:

También se diferencian entre elementos terminales y no terminales. En este caso se llaman datos simples y datos complejos, es decir, compuestos por el anidamiento de otros datos.

Además de las etiquetas xs:schema y xs:element, algunos de los elementos más usados son:

xs:complexType, define tipos complejos.

xs:simpleType, permite definir un tipo simple restringiendo sus valores.

xs:restriction, permite establecer una restricción sobre un elemento de tipo base.

xs:group, permite nombrar agrupaciones de elementos y de atributos para hacer referencia a ellas.

xs:secuence, permite construir elementos complejos mediante la enumeración de los que les forman.

xs:choice, representa alternativas, hay que tener en cuenta que es una o-exclusiva.

xs:all, representa a todos los elementos en cualquier orden, secuencias no ordenadas.

Contenido mixto, definido dando valor true al atributo mixed del elemento xs:complexType, permite mezclar texto con elementos.

Tipos de datos:

XML Schema dispone de numerosos tipos de datos predefinidos para los elementos o atributos, siendo los más comunes:

- string: se corresponde con una cadena de caracteres UNICODE.
- boolean: representa valores lógicos, es decir que solo pueden tomar dos valores, true o false.
- integer: número entero positivo o negativo.
- positiveInteger, negativeInteger: número entero positivo/negativo
- decimal: número decimal, por ejemplo, 8,97.
- dateTime: representa una fecha y hora absolutas en formato (AAAA-MM-DD T HH:MM:SS)
- time: hora en el formato hh:mm:ss.
- date: fecha en formato (aaaa-mm-dd).
- gYearMonth: representa un mes de un año determinado mediante el formato CCYY-MM.
- gMonth: representa el mes mediante el formato –MM. Por ejemplo, febrero es –02.
- anyURI: representa una URI.
- language: representa los identificadores de lenguaje, sus valores están definidos en RFC 1766.
- ID, IDREF, ENTITY, NOTATION, MTOKEN: Representan lo mismo que en los DTD's .

Aparte de los datos proporcionados por el sistema, a veces es interesante crear tipos de datos nuevos personalizados que tienen mayor precisión y eficiencia. Para ello,

XSD divide los elementos en dos grandes grupos basándose en los datos que contienen:

- Elementos con contenido de tipo simple: son elementos sin atributos que sólo contienen datos.
- Elementos con contenido de tipo complejo: son elementos que pueden tener atributos, no tener contenido o contener elementos.

Veamos algunos ejemplos de cada tipo:

Dado el siguiente documento XML llamado **Ejemplo.xml**, genera el correspondiente documento XSD para validarlo.

```
<?xml version="1.0" encoding="UTF-8"?>
<libro xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="Ejemplo.xsd">
  <título>XML práctico</título>
  <autor>Sébastien Lecomte</autor>
  <autor>Andrew Watt</autor>
  <editorial>Ediciones ENI</editorial>
</libro>
```

Como se puede observar el elemento libro estaría compuesto de los elementos simples título, autor y editorial. Así, el documento **Ejemplo.xsd** ser:

- el elemento libro, sería de tipo "complexType" porque contiene otros elementos

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="libro" type="Tlibro"/>
<xs:complexType name="Tlibro">
        <xs:sequence>
                <xs:element name="título" type="xs:string"/>
                <xs:element name="autor" type="xs:string"/>
                <xs:element name="autor" type="xs:string"/>
                <xs:element name="editorial" type="xs:string"/>
        </xs:sequence>
</xs:complexType>
</xs:schema>
```

Si para validar el documento Ejemplo.xml anterior, quisiéramos hacerlo añadiendo además elementos simples, un posible documento XSD podría ser:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="libro" type="Tlibro"/>
<xs:complexType name="Tlibro">
        <xs:sequence>
                <xs:element name="título" type="Ttítulo"/>
                <xs:element name="autor" type="Tautor"/>
                <xs:element name="autor" type="Tautor"/>
                <xs:element name="editorial" type="Teditorial"/>
        </xs:sequence>
</xs:complexType>
```

```
<xs:simpleType name="Ttítulo">
        <xs:restriction base="xs:string"/>
</xs:simpleType>
<xs:simpleType name="Tautor">
        <xs:restriction base="xs:string"/>
</xs:simpleType>
<xs:simpleType name="Teditorial">
        <xs:restriction base="xs:string"/>
</xs:simpleType>
</xs:schema>
```

Facetas de los tipos de datos:

¿Cuáles son las restricciones que podemos aplicar sobre los valores de los datos de un elemento o atributo? Están definidos por las facetas, que solo pueden aplicarse sobre tipos simples utilizando el elemento xs:restriction. Se expresan como un elemento dentro de una restricción y se pueden combinar para lograr restringir más el valor del elemento. Son, entre otros:

- length, minlength, maxlength: longitud del tipo de datos.
- enumeration: restringe a un determinado conjunto de valores.
- whitespace: define el tratamiento de espacios en blanco, tabulaciones, saltos de línea:
    - preserve: se mantienen.
    - replace: para sustituirlas por espacios en blanco.
    - collapse: después de reemplazarlas por espacios en blanco, eliminar todos los espacios en blanco únicos y sustituir varios espacios en blanco seguidos por un único espacio en blanco.
- pattern: permite construir máscaras que han de cumplir los datos de un elemento. La siguiente tabla muestra algunos de los caracteres que tienen un significado especial para la generación de las máscaras.
- (max/min)(In/Ex)clusive: límites superiores/inferiores del tipo de datos. Cuando son Inclusive el valor que se determine es parte del conjunto de valores válidos para el dato, mientras que cuando se utiliza Exclusive, el valor dado no pertenece al conjunto de valores válidos.
- totalDigits, fractionDigits: número de dígitos totales y decimales de un número decimal.

Elementos para hacer patrones.

Patrón	Significado
[A-Z a-z]	Letra.
[A-Z]	Letra mayúscula.
[a-z]	Letra minúscula.
[0-9]	Dígitos decimales.
\D	Cualquier carácter excepto un dígito del 0 al 9.
(A)	Cadena que coincide con A.
A | B	Cadena que es igual a la cadena A o a la B.

Elementos para hacer patrones.

Patrón	Significado
AB	Cadena que es la concatenación de las cadenas A y B.
A?	Cero o una vez la cadena A.
A+	Una o más veces la cadena A.
A*	Cero o más veces la cadena A.
[abcd]	Alguno de los caracteres que están entre corchetes.
[^abcd]	Cualquier carácter que no esté entre corchetes.
\t	Tabulador.

Dado el siguiente documento XML llamado **Ejemplo.xml**, genera el correspondiente documento XSD para validarlo con las siguientes condiciones:

- el elemento "estado" debe ser una cadena de texto con una longitud máxima de 9 caracteres y dos valores posibles: "conectado" y "ocupado".

```
<?xml version="1.0" encoding="UTF-8"?>
<usuarios xmlns:xs="http://www.w3.org/2001/XMLSchema-instance" xs:noNamespaceSchemaLocation="Ejemplo.xsd">
  <usuario>
    <nombre>Peter Green</nombre>
    <departamento>Finanzas</departamento>
    <puntuación>5</puntuación>
    <estado>conectado</estado>
  </usuario>
</usuarios>
```

Un posible documento **Ejemplo.xsd** sería el siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="usuarios">
        <xs:complexType>
                <xs:sequence>
                        <xs:element name="usuario">
                                <xs:complexType>
                                        <xs:sequence>
                                                <xs:element name="nombre" type="xs:string"/>
                                                <xs:element name="departamento" type="xs:string"/>
                                                <xs:element name="puntuación" type="xs:integer"/>
                                                <xs:element name="estado" type="Testado"/>
                                        </xs:sequence>
                                </xs:complexType>
                        </xs:element>
                </xs:sequence>
        </xs:complexType>
</xs:element>
<xs:simpleType name="Testado">
        <xs:restriction base="xs:string">
                <xs:maxLength value="9"/>
                <xs:enumeration value="conectado"/>
                <xs:enumeration value="ocupado"/>
        </xs:restriction>
</xs:simpleType>
</xs:schema>
```

Dado el documento XML **Ejemplo.xml**, genera el correspondiente documento XSD para validarlo con las siguientes condiciones:

- el elemento "nombre" debe admitir nombres compuestos que comiencen con mayúscula

```
<xs:simpleType name="Tnombre">
        <xs:restriction base="xs:string">
                <xs:pattern value="(([A-Z]+[a-zñáéíóú]*)+[\s]*)*"/>
        </xs:restriction>
</xs:simpleType>
```

Dado el documento XML **Ejemplo.xml**, genera el correspondiente documento XSD para validarlo con las siguientes condiciones:

- el elemento "puntuación" debe ser un número entero comprendido entre 1 y 10, ambos inclusive.

```
<xs:simpleType name="Tpuntuación">
        <xs:restriction base="xs:integer">
                <xs:minExclusive value="0"/>
                <xs:maxInclusive value="10"/>
        </xs:restriction>
</xs:simpleType>
```

Documentación del esquema:

Una vez que hemos visto como crear un esquema vamos a ver el modo de incorporar cierta documentación (quién es el autor, limitaciones de derechos de autor, utilidad del esquema, etc.) al mismo.

Podemos pensar que un método para añadir esta información es utilizar comentarios. El problema es que los analizadores no garantizan que los comentarios no se modifiquen al procesar los documentos y por tanto, que los datos añadidos no se pierdan en algún proceso de transformación del documento.

En lugar de usar los comentarios, XML Schema tiene definido un elemento xs:annotation que permite guardar información adicional. Este elemento a su vez puede contener una combinación de otros dos que son:

- xs:documentation, además de contener elementos de esquema puede contener elementos XML bien estructurados.
  También permite determinar el idioma del documento mediante el atributo xml:lang.
- xs:appinfo, se diferencia muy poco del elemento anterior, aunque lo que se pretendió inicialmente era que xs:documentation fuese legible para los usuarios y que xs:appinfo guardase información para los programas de software.
  También es usado para generar una ayuda contextual para cada elemento declarado en el esquema.

Ejemplo de documentación de un esquema:

```
<xs:schema xmlns:xs=http://www.w3.org/2001/XMLSchema>
  <xs:annotation>
    <xs:documentation xml:lang ="es-es">
      Materiales para formación e-Learning
      <modulo>Lenguajes de marcas y sistemas de gestión de información.</modulo>
      <fecha_creación> 2011<f/echa_creacion>
      <autor> BIRT </autor>
    </xs:documentation>
  </xs:annotation>

  <xs:element name="lmsgi" type=xs:string>
    <xs:annotation>
      <xs:appinfo>
        <texto_de_ayuda>Se debe de introducir el nombre completo del tema</texto_de_ayuda>
      </xs:appinfo>
    </xs:annotation>
  </xs:element>
</xs:schema>
```

- Tabla de "facetas" para restringir los valores de elementos y atributos:

Faceta	Uso	Tipos de datos para donde se usa
`xs:minInclusive`	Especifica el límite inferior del rango de valores aceptable. El propio valor está incluido.	Numéricos y de fecha/horas
`xs:maxInclusive`	Especifica el límite superior del rango de valores aceptable. El propio valor está incluido.	Numéricos y de fecha/horas
`xs:minExclusive`	Especifica el límite inferior del rango de valores aceptable. El propio valor no está incluido.	Numéricos y de fecha/horas
`xs:maxExclusive`	Especifica el límite superior del rango de valores aceptable. El propio valor no está incluido.	Numéricos y de fecha/horas
`xs:enumeration`	Especifica una lista de valores aceptables.	Todos
`xs:pattern`	Especifica un patrón o expresión regular que deben cumplir los valores válidos.	Texto
`xs:whiteSpace`	Especifica cómo se tratan los espacios en blanco, entendiéndose como tales los saltos de línea, tabuladores y los propios espacios. Sus posibles valores son *preserve*, *replace* y *collapse*.	Texto
`xs:length`	Especifica el número exacto de caracteres (o elementos de una lista) permitidos. Ha de ser mayor o igual que 0.	Texto
`xs:minLength`	Especifica el mínimo número de caracteres (o elementos de una lista) permitidos. Ha de ser mayor o igual que 0.	Texto
`xs:maxLength`	Especifica el máximo número de caracteres (o elementos de una lista) permitidos. Ha de ser mayor o igual que 0.	Texto
`xs:fractionDigits`	Especifica el número máximo de posiciones decimales permitidas en números reales. Ha de ser mayor o igual que 0.	Números con parte decimal
`xs:totalDigits`	Especifica el número exacto de dígitos permitidos en números. Ha de ser mayor que 0.	Numéricos