XML Schema

- Un esquema es un documento XML, y por tanto puede editarse y procesarse con las mismas herramientas que reconocen documentos XML.
- Especifica cómo definir cada tipo de elemento en el esquema y los tipos de datos de los elementos:
  - Tipos predefinidos: string, integer, decimal, date, boolean.
  - Tipos definidos por el usuario.
- Un esquema se almacena en un fichero con extensión .xs o .xsd.
  - Los ficheros XML basados en un esquema deben incluir la referencia a éste.

Características

XSD o «XML Schema Definition»

- Define qué elementos pueden aparecer en un documento XML.
- Define qué atributos pueden aparecer en un documento XML.
- Define qué elementos son compuestos, indicando qué elementos hijos deben aparecer y en qué orden.
- Define qué elementos pueden ser vacios o que pueden incluir texto asociado.
- **Define los tipos que pueden utilizarse en cada elemento o atributo.**
- Define la obligatoriedad, la optatividad de elementos y/o atributos.

Esquema vs DTD

- XML.
- Estándar desde el 2001 de W3C.
- Extensible, más potente y gramáticas más complejas.
- Definición de tipos de datos.

Prefijos

- Se debe anteponer la etiqueta de espacio de nombres (xs o xsd):
  - a las etiquetas XML Schema.
  - a los tipos predefinidos por XML Schema.

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

Definición del esquema

- Los elementos utilizados en la definición de un esquema proceden de un espacio de nombres oficial: "`http://www.w3.org/2001/XMLSchema`".
- Componentes de un esquema XML:
  - Elemento raíz: elemento *schema*.

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

  - *element*: especifica un elemento en un esquema. Se anidan para formar elementos complejos.
    - Tipo simple.
    - Tipo complejo.
  - *attribute*: especifica un atributo dentro de un elemento.

Referencia al esquema

Cada documento XML asociado a un esquema es considerado una instancia de éste.

```
<?xml version="1.0" encoding="UTF-8"?>
<contactos xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation="contactos.xsd">

  <contacto>
      …
  </contacto>
</contactos>
```

`noNamespaceSchemaLocation` ➔ Esquema para elementos sin namespace.
`SchemaLocation` ➔ Lista de namespaces y esquema para cada uno.

```
<p:Persona
   xmlns:p="http://miweb.com/Personas"
   xmlns:v="http://miweb.com /Vehiculos"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation=
     "http://miweb.com/Personas
      http://miweb.com/schemas/personas.xsd
      http://miweb.com/schemas/Vehiculos
      http://miweb.com/schemas/vehiculos.xsd">
```

Generación automática de esquemas

- Generadores instalables: Trang.
- Herramientas on-line: flame-ware, freeformatter.
- Editores XML: XMLSpy.

Modelos de diseño de esquemas XML

Diseño anidado o de muñecas rusas:

- Las declaraciones se anidan unas dentro de otras.
- Se describe cada elemento y atributo en el mismo lugar donde se declaran.

Diseño plano:

- Se declaran los elementos y los atributos y se indica una referencia a su definición, que se realiza en otro lugar del documento.

Diseño con tipos con nombres reutilizables:

- Se definen tipos de datos simples o complejos a los que se identifica con un nombre.
- Al declarar elementos y atributos, se indica que son de alguno de los tipos con nombre previamente definidos.

Elementos

```
<xs:element name="---" type="---"/>
```

- *name*: nombre que recibe el elemento.
- *type*: tipo de datos que almacena.
  - Predefinido: *string, integer, int, positiveInteger, negativeInteger, nonPositiveInteger, nonNegativeInteger, decimal, long, float, boolean, time, date.*
  - Definido por el usuario (uso de restricciones).

Ejemplos:

```
<xs:element name="nombre" type="xs:string"/>
<xs:element name="salario" type="xs:decimal"/>
```

Cardinalidad

- Atributos *minOccurs* y *maxOccurs*.
  - Elemento opcional.
    - Valor "0" en minOccurs.
  - Elemento sin límite de ocurrencias:
    - Valor "unbounded" en maxOccurs.
  - Valor por defecto de ambos: 1.
  - Ninguno puede usarse en el elemento raíz.

Ejemplos:

```
<xs element name="fecha" type="xs:date" minOccurs="0"/>
<xs:element name="telefono" type="xs:string" minOccurs="0" maxOccurs="3"/>
```

Valores por defecto

- *default*: es el valor que tomará el elemento al ser procesado por alguna aplicación cuando en el documento instancia XML no había recibido ningún valor.
- *fixed*: indica el único valor que puede contener el elemento en el documento instancia XML.

AMBOS: Sólo se pueden usar si el contenido del elemento es únicamente textual.

```
<xs:element name="semaforo" type="xs:string" default="rojo"/>
<xs:element name="semaforo" type="xs:string" fixed="rojo"/>
```

Atributos

```
<xs:attribute name="xxx" type="yyyyy"/>
```

- *name*: Nombre que recibe el atributo.
- *type*: tipo de datos (sólo predefinido) que almacena.
- *use*: Indica obligatoriedad u opcionalidad.
  - required, optional (por defecto), prohibited.
- *default /fixed*: como en elementos.

```
<xs:attribute name="dni" type="xs:string" default="0" use="required"/>
```

Modelos de contenido

- Secuencia (`xs:sequence`):

Los elementos aparecen en fila, unos detrás de otros, en un orden determinado.

- Alternativa (`xs:choice`):

Los elementos aparecen como alternativa unos de los otros, sólo se elije uno.

- Todos (`xs:all`):

Los elementos aparecen en cualquier orden.

Pueden combinarse con minOccurs y maxOccurs.

```
<xs:sequence>
  <xs:element name="emisor" type "xs:string" />
  <xs:element name="emisor" type "xs:string" />
  <xs:element name="emisor" type "xs:string" />
</xs:sequence>

<xs:choice>
  <xs:element name="prologo" type "xs:string" />
  <xs:element name="prefacio" type "xs:string" />
  <xs:element name="introduccion" type "xs:string" />
</xs:choice>

<xs:all>
  <xs:element name="alfa" type "xs:string" />
  <xs:element name="omega" type "xs:string" />
</xs:all>
```

Tipos de datos predefinidos

- Existen 44, organizados de forma jerárquica.
- `xs:anyType` es el raíz y el tipo más genérico.
  - Es el tipo que se asigna a los elementos en los que no se define type.
  - No puede asignarse a ningún atributo.
- 5 categorías (ver documento anexo):
  - Numéricos (16).
  - Fecha y hora (9).
  - Texto (16).
  - Binarios (2).
  - Booleanos (1).

Tipos de datos simples

- Todos los tipos de datos predefinidos son simples.
- En general representan valores atómicos.
- Se pueden asignar tanto a elementos que contengan sólo contenido textual como a atributos.
- También se pueden construir, derivando de un tipo base predefinido al que se le introducen restricciones.

Tipos de datos construidos

- Son tipos generados por el usuario basándose en un tipo predefinido o en un tipo previamente construido.
- Se denomina faceta a una restricción que permite generar nuevos tipos a partir de uno existente, limitando su rango de valores posibles.
  - Por ejemplo, a un elemento `xs:date` se le puede poner una faceta `xs:maxInclusive`.

Facetas

`xs:minInclusive` – `xs:maxInclusive`.

`xs:minExclusive` – `xs:maxExclusive`.

`xs:enumeration` (lista de valores aceptables).

`xs:pattern` (expresión regular, patrón).

`xs:whiteSpace` (tratamiento de blancos: *preserve*, *replace*, *collapse*).

`xs:length` (número exacto de caracteres).

`xs:minLength` – `xs:maxLength`.

`xs:fractionDigits` (número máximo de decimales).

`xs:totalDigits` (número exacto de dígitos).

Uso de facetas o restricciones

- `xs:simpleType`, con la posibilidad de incluir `xs:restriction`.
- Para elementos y para atributos.

```
<xs:element name="ingresosAnuales" type="xs:float"/>

<xs:simpleType name="TipoContrasenia">
  <xs:restriction base="xs:string">
    <xs:minLength value="6" />
    <xs:maxLength value="12" />
  </xs:restriction>
</xs:simpleType>
<xs:element name="clave" type="TipoContrasenia">
```

```
<xs:element name="edadLaboral">
  <xs:simpleType>
    <xs:restriction base="xs:nonNegativeInteger">
      <xs:minInclusive value="16" />
      <xs:maxInclusive value="70" />
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

O bien:

```
<xs:simpleType name="TipoEdadLaboral">
  <xs:restriction base="xs:nonNegativeInteger">
    <xs:minInclusive value="16" />
    <xs:maxInclusive value="70" />
  </xs:restriction>
</xs:simpleType>
<xs:element name="edadLaboral" type="TipoEdadLaboral"/>
```

```
<xs:simpleType name="TipoEstaciones">
  <xs:restriction base="xs:token">
    <xs:enumeration value="Primavera" />
    <xs:enumeration value="Verano" />
    <xs:enumeration value="Otoño" />
    <xs:enumeration value="Invierno" />
  </xs:restriction>
</xs:simpleType>


<xs:simpleType name="TipoCantidad">
  <xs:restriction base="xs:decimal">
    <xs:totalDigits value="11" />
    <xs:fractionDigits value="2" />
  </xs:restriction>
</xs:simpleType>
```

Restricciones con patrones de valores

```
<xsd:element name="grupo">
 <xsd:simpleType>
  <xsd:restriction base="xsd:string">
   <xsd:pattern value="[1-4][A-G]"/>
  </xsd:restriction>
 </xsd:simpleType>
</xsd:element>
```

Un valor numérico de 1 a 4 seguido de una letra mayúscula entre la A y la G.

Elementos de expresiones regulares

.	Cualquier carácter	;
\w	Cualquier letra	M
\d	Un dígito	7
\D	Cualquier carácter no dígito	j
\s	Cualquier carácter de espaciado	
\S	Cualquier carácter de no espaciado	C
\d{n}	n dígitos (ej.4)	3856
\d{n,m}	De n a m dígitos (ej.2 a 3)	91
\d{n,}	n ó más dígitos (ej. 3)	3500

Elementos de expresiones regulares

`[xyz]`	Una x ó una y ó una z minúscula	y
`[A-Z]`	Un carácter de la A a la Z	B
`[^abc]`	Ni a ni b ni c	d
`[F-J-[H]]`	Sustracción de un carácter de un rango	G
`(a|b)`	a ó b	b
`b?`	0 ó 1 b	
`1*`	0, 1 o varios unos	111
`(cd)+`	1 ó más veces la cadena cd	cdcd

Caracteres especiales

- Hay ciertos caracteres que tienen un significado propio en las expresiones regulares: {, }, [,], (,), ?, *, +, -, ^, ., \.
- Para indicar en una expresión regular la aparición literal de estos caracteres, hay que anteponerles el carácter de escape \.

Por ejemplo, la expresión regular para una cadena que contenga las letras a ó b o el carácter + sería: `[ab\+]`.

Tipos de datos complejos

- Sólo se pueden asignar a elementos que tengan elementos descendientes y/o atributos.
- Cuatro tipos:
  - Elementos vacíos.
  - Elementos que contienen sólo otros elementos.
  - Elementos que contienen sólo texto.
  - Elementos que contienen tanto otros elementos como texto.
- Todos estos elementos pueden contener atributos a su vez.

Tipos de datos complejos

`<xs:complexType>`

- Sólo se pueden asignar a elementos.
- Un elemento se considera de tipo complejo si tiene atributos y/o descendientes.
- El contenido de un elemento es aquello que va entre sus etiquetas de apertura y cierre.
  - Contenido simple (`xs:simpleContent`): el elemento declarado sólo tiene contenido textual, sin elementos descendientes.
  - Contenido complejo (`xs:complexContent`): el elemento declarado tiene elementos descendientes. Puede tener o no contenido textual.

```
<xs:element name="persona">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="primerApellido" type="xs:string" />
      <xs:element name="segundoApellido" type="xs:string" />
      <xs:element name="fechaNacimiento" type="xs:date" />
    </xs:sequence>
  </xs:complexType>
</xs:persona>
```

O bien:

```
<xs:complexType name="TipoPersona">
  <xs:sequence>
    <xs:element name="primerApellido" type="xs:string" />
    <xs:element name="segundoApellido" type="xs:string" />
    <xs:element name="fechaNacimiento" type="xs:date" />
  </xs:sequence>
</xs:complexType>
<xs:element name="persona" type="TipoPersona" />
```

Extensión de un tipo complejo

- Añade nuevos elementos y/o atributos a tipos complejos ya existentes.
- Los elementos añadidos aparecerán al final de todos los elementos del tipo base.

```
<xs:extension base="TipoBase">
. . . . .
</xs:extension>
```

```
<xs:complexType name="TipoAlumno">
  <xs:sequence>
    <xs:element name="Apellido" type="xs:string" />
    <xs:element name="Ciclo" type="xs:string" />
  </xs:sequence>
</xs:complexType>

<xs:complexType name="TipoAlumnoExtendido">
  <xs:complexContent>
    <xs:extension base="TipoAlumno">
      <xs:sequence>
        <xs:element name="Curso" type="xs:positiveInteger" />
      </xs:sequence>
      <xs:attribute name="Matricula" type="xs:positiveInteger"
                    use="required" />
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:element name="Alumno" type="TipoAlumnoExtendido" />

<Alumno Matricula="123">
  <Apellido>Marín</Apellido>
  <Ciclo>DAM</Ciclo>
  <Curso>1</Curso>
</Alumno>
```