XML - Schema

¿Qué es un XML Schema?

- Un esquema XML, al igual que un DTD, describe la estructura de un documento XML.
- Los esquemas XML están escritos en XML.
- El esquema XML incluye una descripción de su propio formato.
- Los esquemas XML son extensibles:
  - Permite describir el contenido de un documento.
  - Es más fácil definir restricciones sobre los datos.
  - Es más fácil para validar la exactitud de los datos
  - Permite convertir entre diferentes tipos de datos

XML Schema

- Un XML Schema es una alternativa XML a una DTD.

Ejemplo 1

```
<xs:element name="nota">
  <xs:complexType>
  <xs:sequence>
     <xs:element name="de" type="xs:string"/>
     <xs:element name="para" type="xs:string"/>
     <xs:element name="asunto" type="xs:string"/>
     <xs:element name="mensaje" type="xs:string"/>
  </xs:sequence>
  </xs:complexType>
</xs:element>
```

`<xs: element name="nota">` **define el elemento llamado "nota"**
`<xs: complexType>` **El elemento "nota" es un tipo complejo**
`<xs: sequence>` **tipo complejo, secuencia de elementos**
`<xs: element name="" type="">` **un elemento y su tipo**

XML Schema

Ejemplo 1 (cont.)

```
<xs:element name="de" type="xs:string"/>
```
el elemento "de" es de tipo string (texto)

```
<xs:element name="para" type="xs:string"/>
```
el elemento "para" es de tipo string (texto)

```
<xs:element name="asunto" type="xs:string"/>
```
el elemento "asunto" es de tipo string (texto)

```
<xs:element name="mensaje" type="xs:string"/>
```
el elemento "mensaje" es de tipo string (texto)

- El lenguaje de esquema XML también se conoce como definición de esquema XML (XSD).

XML Schema

nota.xsd

```
<?xml version="1.0"?>
<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="asunto" type="xs:string"/>
   <xs:element name="cuerpo" type="xs:string"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>

</xs:schema>
```

nota.xml
(sin referencia al esquema)

```
<?xml version="1.0"?>
<nota>
 <para>Antonio</para>
 <de>Juan</de>
 <asunto>Recuerdo</asunto>
 <cuerpo>¡No me olvides!</cuerpo>
</nota>
```

XML Schema

- Un XML Schema:
  - Define los elementos y los atributos que pueden aparecer en un documento.
  - Define los elementos hijos de otros.
  - Define el número y el orden de los elementos secundarios.
  - Define los contenidos de un elemento: vacío, texto u otro valor.
  - Define los tipos de datos de elementos y atributos
  - Define los valores por defecto y fijos para elementos y atributos.

Estructura de un Esquema XML

- En el XSD (Definición del esquema)
  - Para crear un esquema (xsd) hay que incluir en el elemento raíz el espacio de nombres del esquema.

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

- En el XML:
  - Para instanciar un esquema (xml) hay que expresar en el elemento raíz la instancia del esquema y el documento del esquema:

```
<raíz
xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
xsi:noNamespaceSchemaLocation="archivo.xsd">
```

Estructura de un Esquema XML

- En el XML: Esquema externo o esquema local

nota.xml
(con referencia al esquema)

```
<?xml version="1.0“ encoding=“ISO-8859-1?>
<nota xmlns="http://www.w3schools.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xsi:schemaLocation="
http://roble.pntic.mec.es/jtal0007/xmlfp/XSD/
nota.xsd" >
 <para>Antonio</para>
 <de>Juan</de>
 <asunto>Recuerdo</asunto>
 <cuerpo>No me olvides este  weekend!</cuerpo>
</nota>
```

Caso de un esquema local

```
xmlns="http://www.w3schools.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation=“nota.xsd”
```

Estructura de un Esquema XML

- Un esquema contiene elementos que pueden ser
  - **Simples** (simpleType): No pueden tener ni elementos ni atributos.
  - **Complejos** (complexType): Pueden contener otros elementos y atributos.

```
<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="asunto" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
</xs:schema>
```

Elemento complejo

Elementos simples

XSD: Elementos simples

- Un elemento simple es un elemento XML que sólo contiene datos. (***No puede contener otros elementos o atributos***).

```
<xs:element nombre="xxx" type="yyy"/>
```

- Los type más usados son:
  - **`xs:string`** cadena de caracteres (`token`)
  - **`xs:decimal`** números más reales (`float, double`)
  - **`xs:integer`** números enteros (`short, long, byte`)
  - **`xs:boolean`** booleano
  - **`xs:date`** fecha
  - **`xs:time`** hora

XSD: Elementos simples

Ejemplo 2

```
<xs:element name="nombre" type="xs:string"/>
<xs:element name="edad" type="xs:integer"/>
<xs:element name="fecha" type="xs:date"/>
```

```
<nombre>Jose</nombre>
<edad>25</edad>
<fecha>15-03-2012</fecha>
```

XSD: Elementos simples

Ejercicio simple1

Crear un documento simple1.xml que contenga los datos personales de un alumno:

```
<?xml version="1.0" encoding="UTF-8"?>
<alumno xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
xsi:noNamespaceSchemaLocation="simple1.xsd">
    <nombre>Roberto</nombre>
    <dni>28542345</dni>
    <direccion>San José 2</direccion>
    <edad>22</edad>
    <telefono>9659434523</telefono>
</alumno>
```

Crear el documento simple1.xsd para validar simple1.xml

XSD: Elementos simples

Ejercicio simple2

Crear un documento simple2.xml que contenga los datos personales de un alumno:

```
<?xml version="1.0" encoding="UTF-8"?>
<Libro xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="simple2.xsd">
    <Titulo>Programming Bye</Titulo>
    <Autor>John C. Smith</Autor>
    <Fecha>2014</Fecha>
    <ISBN>84-202-77678-3</ISBN>
    <Editorial>Mc Graw Hill</Editorial>
</Libro>
```

Crear el documento simple2.xsd para validar simple2.xml

XSD: Valores de elementos

- Podemos asignar valores por defecto (`default`) y valores fijos (`fixed`) a elementos simples

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

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

XSD: Restricciones

- Podemos restringir valores (restriction), tanto para elementos o como atributos XML.

Ejemplo 3

El valor de la temperatura no puede ser inferior a -20 o superior a 40: (como restricción a un único elemento)

```
<xs:element name="temperatura">
    <xs:simpleType>
        <xs:restriction base="xs:integer">
            <xs:minInclusive value="-20"/>
            <xs:maxInclusive value="40"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
```

Elemento simple. Restricción especificada dentro del elemento

XSD: Restricciones

- Podemos limitar el contenido a un conjunto de valores aceptables, usando la restricción de **enumeración**.

Ejemplo 4

Los únicos valores aceptables son: Lunes, Miércoles y Viernes: (**como TIPO**)

```
<xs:simpleType name="diaSemana">
    <xs:restriction base="xs:string">
        <xs:enumeration value="Lunes"/>
        <xs:enumeration value="Miércoles"/>
        <xs:enumeration value="Viernes"/>
    </xs:restriction>
</xs:simpleType>

<xs:element name="semana" type="diaSemana" />
```

Definición de tipo

XSD: Restricciones

- **TIPOS**:
  - Se trata de un tipo de datos personalizado
  - La ventaja es que podemos utilizarlo en otras partes del esquema.

- En el ejemplo anterior, **diaSemana** (con la restricción de valores Lunes, Miércoles y Viernes) podemos utilizarlo en cualquier parte del esquema si vuelve a aparecer sin tener que volver a crear la restricción. Simplemente indicando el tipo diaSemana

XSD: Restricciones

- Podemos usar expresiones regulares o **patrones** para afinar la restricción.

Ejemplo 5

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

Otros patrones:

```
<xs:pattern value="\d"/>  cualquier dígito
<xs:pattern value="[0-9]+[A-Z]"/> 1 o más números y
seguido de una letra mayúscula
```

XSD: Restricciones

- Podemos restringir la **longitud** de un elemento.

Ejemplo 6

```
<xs:element name="dni">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:length value="10"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
```

Otras restricciones:

```
xs:minLength   establece un mínimo en la longitud
xs:maxLength   establece un máximo en la longitud
```

XSD: Restricciones

`enumeration:`	Define una lista de valores aceptables
`fractionDigits:`	Número máximo de decimales permitidos. igual o mayor que cero
`length:`	Especifica el número exacto de caracteres o elementos de lista permitidos. Debe ser igual o mayor que cero
`maxExclusive:`	Especifica límite superior a un valor numérico (excluido)
`maxInclusive:`	Especifica límite superior a un valor numérico (incluido)
`maxLength:`	Especifica el número máximo de caracteres o elementos de lista permitidos. Debe ser igual o mayor que cero
`minExclusive:`	Especifica límite inferior a un valor numérico (excluido)
`minInclusive:`	Especifica límite inferior a un valor numérico (incluido)
`minLength:`	Especifica el número mínimo de caracteres o elementos de lista permitidos. Debe ser igual o mayor que cero
`pattern:`	Define la secuencia exacta de caracteres que son aceptables
`totalDigits:`	Especifica el número máximo de dígitos permitidos. Debe ser mayor que cero
`whiteSpace:`	Especifica cómo se manejan los espacios en blanco (los saltos de línea, tabuladores, espacios y retornos de carro). Valores: preserve, replace y collapse.

XSD: Restricciones

Ejercicio restricciones 1

A. **Modificar `simple1.xml` escribiendo como contenido de `edad` 20 y guardarlo como `restriccion1.xml`.**

B. **Guardar `simple1.xsd` como `restriccion1.xsd` y modificarlo aplicando una restricción sobre el valor numérico de la edad (por ejemplo, debe estar comprendida entre 16 y 25).**

XSD: Restricciones

Ejercicio restricciones 2

A. **Modificar** `restriccion1.xml` **añadiendo el elemento** sexo **con el contenido M y guardarlo como** `restriccion2.xml`.

B. **Modificar** `restriccion1.xsd` **aplicando una restricción sobre un conjunto de valores. El campo sexo será M o H. Guardarlo como** `restriccion2.xsd`.

C. **Modificar** `restriccion2.xsd` **creando el tipo personalizado** `tipoGenero`, **es decir que el elemento debe aparecer en la secuencia como** `<xs:element name="genero" type="tipoGenero"/>`

XSD: Restricciones

Ejercicio restricciones 3

A. **Modificar `restriccion2.xsd` aplicando una restricción sobre series de valores y guardarlo como `restriccion3.xsd`. Por ejemplo, vamos a obligar a que el formato del `dni` sean números y una letra y el `telefono` esté formado por números.**

B. **Validar el documento xml escribiendo números incorrectos y que no se adapten al patrón.**

XSD: Restricciones

Ejercicio restricciones 4

A. **Modificar `restriccion3.xsd` aplicando restricciones sobre la longitud de los elementos. Lo guardamos como `restriccion4.xsd`. Por ejemplo el DNI estará formado por un máximo de 8 dígitos y una letra y el teléfono lo escribiremos con mayor y menor longitud**

B. **Validar el documento xml escribiendo números de DNI o el teléfono.**

XSD: Atributos

- Los elementos con atributos son elementos complejos. Los atributos se declaran como tipos simples (`xs:string`, `xs:decimal`, `xs:integer`,...)
- Los atributos tienen el formato:

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

Ejemplo 7

```
<curso letra="A">1</curso>
```

```
<xs:element name="curso" type="xs:integer"/>
```

```
<xs:attribute name="letra" type="xs:string"/>
```

XSD: Atributos

- Los atributos pueden tener un valor fijo:

```
<xs:attribute name="letra" type="xs:string" fixed="A"/>
```

- Pueden tener un valor por defecto:

```
<xs:attribute name="letra" type="xs:string" default="A"/>
```

- Los atributos pueden ser obligatorios:

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

- Por defecto el atributo será opcional (**optional**)

XSD: Elementos complejos

- Son elementos XML que contienen otros elementos y/o atributos.

- Hay cuatro tipos de elementos complejos:
  1. Elementos con contenido vacío, pero con atributos.
  2. Elementos con contenido y atributos.
  3. Elementos que solo contienen elementos hijos.
  4. Elementos que contienen elementos hijos y atributos
  5. Elementos que contienen elementos, texto y atributos (**mixto**).

XSD: Elementos complejos

1. Los elementos vacíos con atributos.

Ejemplo 8

```
<xs:element name="producto">
    <xs:complexType>
        <xs:attribute name="id" type="xs:positiveInteger"/>
    </xs:complexType>
</xs:element>

<producto id="1234" />

<producto id="A24" />
```

Incorrecto

XSD: Elementos complejos

2. Los elementos con contenido (texto) y atributos.

Ejemplo 9

```
<xs:element name="nombre" >
     <xs:complexType>
          <xs:simpleContent>
          <xs:extension base="xs:string">
               <xs:attribute name="repetidor" type="xs:string" />
          </xs:extension>
          </xs:simpleContent>
     </xs:complexType>
</xs:element>

<nombre repetidor="no">Andrés</nombre>
```

Contenido simple, solo valor...

… que extendemos con un atributo

En este caso se debe utilizar un contenido simple `simpleContent`, **dentro de un tipo complejo** `complexType`.

El elemento `extension` **sirve para ampliar el tipo base con un atributo** `attribute`.

XSD: Elementos complejos

3. Los elementos que solo contienen otros elementos.

Ejemplo 10

```
<xs:element name="alumno">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="nombre" type="xs:string"/>
            <xs:element name="apellido" type="xs:string"/>
        </xs:sequence> </xs:complexType>
</xs:element>

<alumno>
    <nombre>Pedro</nombre>
    <apellido>Marín</apellido>
</alumno>
```

Una secuencia (orden) de elementos

En ese orden

XSD: Elementos complejos

4. Los elementos que contienen elementos hijos y atributos.

Ejemplo 11

```
<xs:element name="alumno">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="nombre" type="xs:string"/>
            <xs:element name="apellido" type="xs:string"/>
        </xs:sequence>
        <xs:attribute name="clase" type="xs:string" use="required"/>
    </xs:complexType>
</xs:element>

<alumno clase="DAM1">
  <nombre>Pedro</nombre>
  <apellido>Marín</apellido>
</alumno>
```

El atributo hay que declararlo, obligatoriamente, justo antes de la etiqueta de cierre **</xs:complexType>**

XSD: Elementos complejos

5. Los elementos con contenido mixto.

Ejemplo 12

```
<xs:element name="carta">
    <xs:complexType mixed="true" >
    <xs:sequence>
        <xs:element name="nombre" type="xs:string"/>
        <xs:element name="pedido" type="xs:positiveInteger"/>
        <xs:element name="fechaenvio" type="xs:date"/>
    </xs:sequence>
    </xs:complexType>
</xs:element>

<carta>
    Estimado Sr.<nombre>Juan León</nombre>. Su pedido
    <pedido>1032</pedido> será enviado el <fechaenvio>25-03-
    2012</fechaenvio>.
</carta>
```

XSD: Elementos complejos

Ejercicio complejo1. Realizar el esquema complejo1.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<alumno dni="11111111"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="complejo1.xsd">
    <nombre>Lorenzo Pérez</nombre>
    <direccion>
        <calle>Banderas</calle>
        <numero>7</numero>
        <ciudad>Alfafar</ciudad>
        <cp>46910</cp>
        <provincia>Valencia</provincia>
    </direccion>
    <telefono>961555555</telefono>
</alumno>
```

Número entero

Entre 46000 y 46999

XSD: Indicadores

- Los indicadores sirven para controlar cómo los elementos pueden aparecer y en qué número.
- Hay 7 indicadores:

  3 Indicadores de orden:

  Todo (`all`), Elección (`choice`) y Secuencia (`sequence`)

  2 Indicadores de ocurrencia:

  `maxOccurs` y `minOccurs`

  2 Indicadores de grupo:

  `group` y `attributeGroup`

XSD: Indicadores (all)

- Los elementos aparecen una sola vez pero no importa el orden.

Ejemplo 13

```
<xs:element name="persona">
    <xs:complexType>
    <xs:all>
        <xs:element name="nombre" type="xs:string"/>
        <xs:element name="apellidos" type="xs:string"/>
    </xs:all>
    </xs:complexType>
</xs:element>
```

XSD: Indicadores (choice)

- Solo aparece uno de los elementos que contiene.

Ejemplo 14

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

XSD: Indicadores (sequence)

- Los elementos aparecen en el mismo orden al especificado.

Ejemplo 15

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

XSD: Indicadores (maxOcurrs y minOcurrs)

- Sirve para especificar el número máximo y mínimo de veces que puede aparecer un elemento hijo. El atributo `maxOccurs` puede tomar el valor "`unbounded`", que indica que no existe ningún límite.

Ejemplo 16

```
<xs:element name="persona">
    <xs:complexType>
    <xs:sequence>
        <xs:element name="nombre" type="xs:string"/>
        <xs:element name="nombre_hijo" type="xs:string"
                minOccurs="1" maxOccurs="10" />
    </xs:sequence>
    </xs:complexType>
</xs:element>
```

XSD: Indicadores (group)

- Se utiliza para crear conjuntos de elementos relacionados y poder reutilizarlos después solo indicando el nombre.

```
<xs:group name="Nombre_grupo">
...
</xs:group>
...
<xs:group ref=" Nombre_grupo"/>
```

1. Declaración del grupo de elementos

2. Utilización del grupo

Ejemplo 17

```
<xs:group name="grupoAlumno">
    <xs:sequence>
        <xs:element name="nombre" type="xs:string"/>
        <xs:element name="apellido" type="xs:string"/>
        <xs:element name="cumpleanyos" type="xs:date"/>
    </xs:sequence>
</xs:group>
```

1. Declaración del grupo de elementos grupoAlumno con 3 elementos

XSD: Indicadores (group)

Ejemplo 17 (cont.)

2. Utilización del grupo grupoAlumno. Como si hubiéramos declarado los tre elementos nombre, apellidos y cumpleanyos…

```
<xs:element name="alumno" >
    <xs:complexType>
    <xs:sequence>
        <xs:group ref="grupoAlumno"/>
        <xs:element name="ciudad" type="xs:string"/>
    </xs:sequence>
    </xs:complexType>
</xs:element>
```

… en el XML

```
<alumno>
    <nombre>Alberto</nombre>
    <apellido>Gil</apellido>
    <cumpleanyos>25-07-2017</cumpleanyos>
    <ciudad>Cuenca</ciudad>
</alumno>
```

XSD: Indicadores (attributeGroup)

- Sirve para crear grupos de atributos.

```
<xs:attributeGroup name="Nombre_grupo">
  ...
</xs:attributeGroup>
  ...
<xs:attributeGroup ref="Nombre_grupo"/>
```

1. Declaración del grupo de elementos

2. Utilización del grupo

Ejemplo 18

```
<xs:attributeGroup name="grupoAtributosAlumno">
    <xs:attribute name="nombre" type="xs:string"/>
    <xs:attribute name="apellido" type="xs:string"/>
    <xs:attribute name="cumpleanyos" type="xs:date"/>
</xs:attributeGroup>
```

1. Declaración del grupo de atributos `grupoAtributosAlumno`

XSD: Modelo de contenido (any)

- **any**: Permite añadir elementos sin especificar.

Ejemplo 19

```
<xs:element name="persona">
    <xs:complexType>
    <xs:sequence>
        <xs:element name="nombre" type="xs:string"/>
        <xs:element name="apellido" type="xs:string"/>
        <xs:any minOccurs="0"/>
    </xs:sequence>
    </xs:complexType>
</xs:element>

<persona>
<nombre>Yolanda<nombre>
<apellido>Martos<apellido>
<ciudad>Badajoz<ciudad>
</persona>
```

Permite añadir un nuevo elemento detrás del apellido

XSD: Modelo de contenido (anyAttribute)

- **any**: Permite añadir elementos sin especificar.

Ejemplo 20

```
<xs:element name="persona">
    <xs:complexType>
        <xs:attribute name="nombre" type="xs:string"/>
        <xs:anyAttribute minOccurs="0"/>
    </xs:complexType>
</xs:element>

<persona nombre="Yolanda"
         apellido="Martos"
         ciudad="Badajoz" />
```

Permite añadir más atributos

XSD: Ejercicios complejos

Ejercicio complejo2

Modificar `complejo1.xsd` creando un grupo de elementos con el elemento `direccion` y usarlo en el esquema. Entrega `complejo2.xsd` y `comlejo2.xml`

Ejercicio complejo3

Añade al ejercicio algunas clausulas `all` y `choice`. Entrega `complejo3.xsd` y `comlejo3.xml`

Ejercicio complejo4

Añade al ejercicio lo necesario para que aparezca un teléfono fijo o móvil o los dos. Entrega `complejo4.xsd` y `comlejo4.xml`