# LENGUAJES DE MARCAS Y SISTEMAS DE GESTIÓN DE LA INFORMACIÓN

U.T.3 . XML. DEFINICION DE ESQUEMAS Y VOCABULARIOS. XML SCHEMA XSD.

# XML SCHEMA. XSD

- XSD = Xml Schema Definition
- PERMITE SUPERAR LAS LIMITACIONES DE LOS DTD
- MAS DEFINICIONES DE TIPOS BASICOS
- MAS CONTROL SOBRE EL NUMERO DE OCURRENCIAS DE CADA ELEMENTO
- SINTAXIS TIPO XML A DIFERENCIA DE DTDS QUE NO LA SEGUIA

# XML SCHEMA. XSD

- VIDEO MOSTRANDO UNA INTRODUCCION A XSD
- https://www.youtube.com/watch?v=O28ZrZTCAA4
- VIDEO SOBRE LA ESTRUCTURA DE UN XSD
- https://www.youtube.com/watch?v=JKhfLpkVh3o
- VIDEO SOBRE EL DISEÑO DE UN XSD
- https://www.youtube.com/watch?v=wC1r_CGutNk

# XML SCHEMA. XSD

- ESTRUCTURA:
- <?xml version="1.0" encoding="UTF-8" ?>
  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  ...
  ELEMENTOS
  ...
</xs:schema>

# XML SCHEMA. XSD

- ESTRUCTURA:
- <?xml version="1.0" encoding="UTF-8" ?>
  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  ...
  ELEMENTOS
  ...
</xs:schema>

VALOR FIJO

# EJEMPLO DE XSD

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema”>
   <xsd:element name="note">
          <xsd:complexType>
                 <xsd:sequence>
                 <xsd:element name="to" type="xsd:string"/>
                 <xsd:element name="from" type="xsd:string"/>
                 <xsd:element name="heading" type="xsd:string"/>
                 <xsd:element name="body" type="xsd:string"/>
                 </xsd:sequence>
          </xsd:complexType>
< /xsd:element>
</xsd:schema>
```

# QUE SE PUEDE ENCONTRAR EN UN XSD

- Un esquema XML define la estructura válida para un tipo de documento XML (al igual que las DTD), es decir:
- Los elementos que pueden aparecer en el documento
- Los atributos que pueden utilizarse junto a cada elemento
- Cómo se pueden anidar los elementos (padres e hijos)
- El orden en el que deben aparecer los elementos hijos de un mismo padre
- El número permitido de elementos hijos
- Si un elemento puede ser vacío o no
- Tipos de datos para elementos y atributos
- Valores por defecto y fijos para elementos y atributos

# VENTAJAS DE XSD

- Mayor precisión en la definición de tipos de datos mediante formatos y facetas
- Por ejemplo, la fecha:

  <date type="date">1999-03-11</date>

  ¿es el 11 de marzo o el 3 de noviembre?

- Los esquemas se definen como documentos XML, en un documento aparte con extensión .XSD
- En los documentos XML que se basen en ese esquema, incluiremos una referencia al archivo .XSD

# XML – SCHEMA: XSD. ATRIBUTOS DE <xs:schema>

- XMLNS:ALIAS ESPECIFICA EL NAMESPACE DE DONDE PROVIENEN LOS ELEMENTOS Y TIPOS USADOS CUYO VALOR ES FIJO. EL VALOR DE ALIAS SUELE SER “XS” O “XSD” AUNQUE PODRIA USARSE CUALQUIER OTRO.
- elementFormDefault=“qualified”-> INDICA QUE HAY QUE AÑADIR EL ESPACIO DE NOMBRES DELANTE DEL ELEMENTO (unqualified indica lo contrario)
- targetNamespace -> SE USA CUANDO IMPORTAMOS UN XSD DENTRO DE OTRO XSD, PARA INDICAR QUE EN EL “IMPORTADOR” LO QUE SE DEFINA DENTRO PERTENECE AL TARGETNAMESPACE ACTUAL

**(VER: https://www.liquid-technologies.com/xml-schema-tutorial/xsd-namespaces)**

# XSD – ELEMENTOS SIMPLES

- Un elemento simple es un elemento que sólo puede contener texto (cualquier tipo de dato), pero no a otros elementos ni atributos
- Para definir un elemento simple, utilizamos la sintáxis:

```
<xsd:element name="xxx" type="yyy"/>
```

- Ejemplos:

```
<xsd:element name=“apellido” type="xs:string"/>
<xsd:element name=“edad" type="xs:integer"/>
<xsd:element name=“fecNac" type="xs:date"/>
```

# XSD – ELEMENTOS SIMPLES. TIPOS DE DATOS

- Los tipos de datos más utilizados son:
  - xsd:string
  - xsd:decimal
  - xsd:integer
  - xsd:boolean
  - xsd:date
  - xsd:time
- Un elemento simple puede tener un valor por defecto y un valor “fijo”
- Esto se indica mediante los atributos default y fixed

```
<xsd:element name="color" type="xsd:string" default="red"/>
```

# XSD – ELEMENTOS. ATRIBUTOS

- Los atributos se deben declarar de forma similar a los “elementos simples”
- Si un elemento puede ir acompañado de atributos, el elemento se deberá declarar como un elemento “complejo”
- Un atributo se declara de la siguiente forma:

  <xsd:attribute name="xxx" type="yyy"/>

  **Ejemplo**:

  <xsd:attribute name=“idioma" type="xs:string"/>

- Los atributos tienen un tipo de dato: xsd:string, xsd:decimal, xsd:integer, xsd:boolean, xsd:date, xsd:time

# XSD – ELEMENTOS: ATRIBUTOS

- Los atributos pueden tener valores por defecto y valores fijos:

```
<xsd:attribute name="idioma" type="xsd:string" default="ES"/>
<xs:element name="color" type="xs:string" fixed="red"/>
```

- Por defecto, los atributos son opcionales.
- Para indicar que un atributo debe ser obligatorio, se debe añadir a su declaración en el esquema es atributo "use"

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

- El atributo use puede tomar el valor "optional" si el atributo no es obligatorio (opción por defecto)

# XSD – FACETAS=RESTRICCIONES

- LAS FACETAS O RESTRICCIONES PERMITEN RESTRINGIR EL VALOR QUE SE PUEDE DAR A UN ELEMENTO O ATRIBUTO XML
- MEDIANTE RESTRICCIONES PODEMOS INDICAR QUE UN VALOR DEBE ESTAR COMPRENDIDO EN UN RANGO DETERMINADO, DEBE SER UN VALOR DE UNA LISTA DE VALORES “CERRADA”, O DEBE SER MAYOR O MENOR QUE OTRO VALOR…
- TIPOS DE FACETAS:
  - VALOR COMPRENDIDO EN UN RANGO
  - EL VALOR ESTÁ RESTRINGIDO A UN CONJUNTO DE VALORES POSIBLES
  - RESTRINGIR EL VALOR DE UN ELEMENTO A UNA SERIE DE CARACTERES
  - LONGITUD DE LOS VALORES DE LOS ELEMENTOS…

# XSD – FACETAS. EJEMPLO 1

```
<xsd:element name="age">
  <xsd:simpleType>
        <xsd:restriction base="xsd:integer">
              <xsd:minInclusive value="0"/>
              <xsd:maxInclusive value="100"/>
        </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

# XSD – FACETAS. EJEMPLO 2

```
<xsd:element name="car">
  <xsd:simpleType>
        <xsd:restriction base="xsd:string">
              <xsd:enumeration value="Audi"/>
              <xsd:enumeration value="Golf"/>
              <xsd:enumeration value="BMW"/>
        </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

# XSD – FACETAS. EJEMPLO 2. ALTERNATIVA

```
<xsd:element name="car" type="carType"/>

<xsd:simpleType name="carType"> <!– CREAMOS UN TIPO REUTILIZABLE -->
  <xsd:restriction base="xsd:string">
        <xsd:enumeration value="Audi"/>
        <xsd:enumeration value="Golf"/>
        <xsd:enumeration value="BMW"/>
  </xsd:restriction>
</xsd:simpleType>
```

# XSD – FACETAS. EJEMPLO 3.

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

En este ejemplo, el elemento “letter” debe tomar como valor UNA letra minúscula (y sólo una)

# XSD – FACETAS. EJEMPLO 4.

```
<xsd:element name="initials">
  <xsd:simpleType>
        <xsd:restriction base="xsd:string">
              <xsd:pattern value="[a-zA-Z][a-zA-Z][a-zA-Z]"/>
        </xsd:restriction>
</xsd:simpleType>
</xsd:element>
```

En este ejemplo, el elemento “initials” debe tomar como valor 3 letras mayúsculas o minúscula (sólo 3)

# XSD – FACETAS. EJEMPLO 5.

```
<xsd:element name="choice">
  <xsd:simpleType>
        <xsd:restriction base="xsd:string">
              <xsd:pattern value="[xyz]"/>
        </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

En este ejemplo, el elemento “choice” debe tomar como valor una de estas letras: x, y o z

# Esquemas XML – facetas (ej. 7)

```
<xsd:element name="letter">
  <xsd:simpleType>
        <xsd:restriction base="xsd:string">
              <xsd:pattern value="([a-z])*"/>
        </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

- Admite modificadores o wildcards, Letter podrá ser cualquier carácter/es en minúsculas, incluso vacío.

# Esquemas XML – facetas (ej. 8)

```
<xsd:element name="password">
  <xsd:simpleType>
        <xsd:restriction base="xs:string">
              <xsd:pattern value="[a-zA-Z0-9]{8}"/>
        </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

En este ejemplo, el valor del campo “password” debe ser 8 caracteres/números

# Esquemas XML – facetas (ej. 9)

```
<xsd:element name="password">
  <xsd:simpleType>
        <xsd:restriction base="xsd:string">
              <xsd:length value="8"/>
        </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

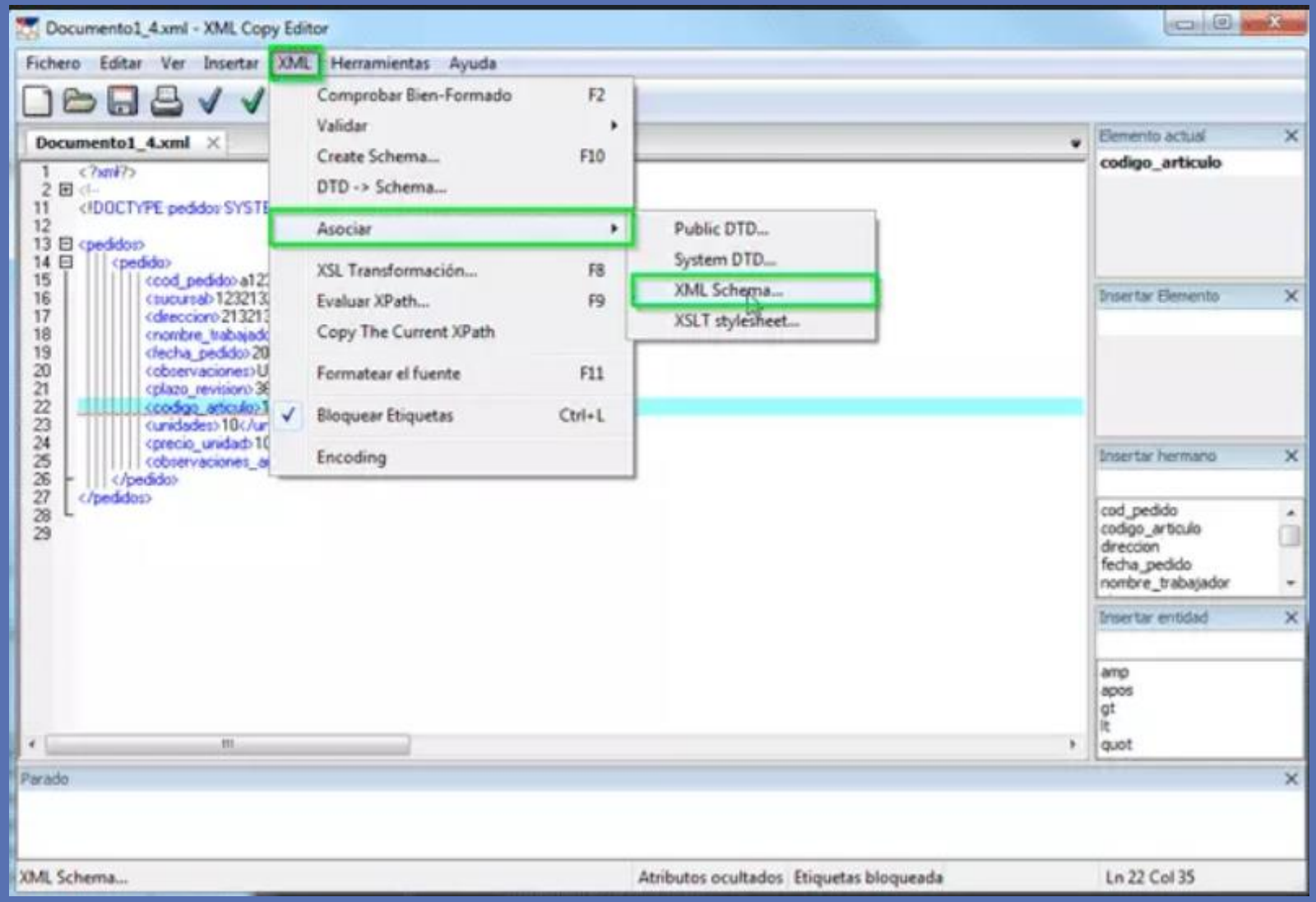
Los elementos length, minLength y maxLength permiten indicar el número exacto, mínimo y máximo de caracteres que puede tener un valor de un elemento.

# Elementos para restricciones o facetas

| | |
|---|---|
| enumeration | Establece una lista de valores "aceptados" |
| fractionDigits | Número de cifras decimales |
| length | Número de caracteres obligatorios |
| maxExclusive y maxInclusive | Valor máximo de un rango |
| minExclusive y minInclusive | Valor mínimo en un rango |
| maxLength y minLength | Número máximo y mínimo de caracteres permitidos |
| pattern | Define una secuencia de caracteres permitida |
| totalDigits | Número exacto de dígitos permitidos |
| whiteSpace | Indica cómo se deben de tratar los espacios en blanco |

# VALIDAR XML CONTRA XSD EN XML COPY EDITOR

- PARA VALIDAR UN XML, CON SU DTD Y CON SU SCHEMA, PRIMERO ES NECESARIO ASOCIAR EL ESQUEMA AL XML DESDE LAS OPCIONES DEL MENÚ MXL -> ASOCIAR -> XML SCHEMA Y SELECCIONAREMOS EL ESQUEMA (XSD). UNA VEZ ASOCIADO SE PUEDE COMPROBAR NORMALMENTE.



# VALIDAR XML CONTRA XSD ONLINE

https://www.freeformatter.com/xml-validator-xsd.html

If the XSD is publicly available using HTTP and referenced through a "schemaLocation" or "noNamespaceSchemaLocation", then the validator will pick it up and it doesn't need to be specified/uploaded.

***The maximum size limit for file upload is 2 megabytes.**

## XML Input

**Option 1:** Copy-paste your XML document here

**Option 2:** Or upload your XML document

Examinar... No se ha seleccionado ningún archivo. UTF-8

## XSD Input (Optional if XSD referred in XML using schemaLocation)

**Option 1:** Copy-paste your XSD document here

**Option 2:** Or upload your XSD document

Examinar... No se ha seleccionado ningún archivo. UTF-8

VALIDATE XML

# EJERCICIO 1

GENERAR EL XSD CORRESPONDIENTE A:

<prueba>esta es una cadena de caracteres con numeros 1234 no permite mayusculas</prueba>

# EJERCICIO 1: SOLUCION

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name="password">
        <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                        <xsd:pattern value="[a-z0-9\s]+"/>
                </xsd:restriction>
        </xsd:simpleType>
</xsd:element>
</xsd:schema>
```

# EJERCICIO 1: SOLUCION

El fichero xml tiene que hacer referencia al XSD:

<password xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="file:/C:/Users/fichero.xsd">
esta es una cadena de caracteres con numeros 1234 no permite mayusculas </password>

# EJERCICIO 2: CREAR XSD

DADO EL XML SIGUIENTE CREA UN XSD QUE LO VALIDE:

```
<prueba atributo="1"></prueba>
```

# EJERCICIO 2: SOLUCION

AL TENER UN ATRIBUTO SE DECLARA COMO COMPLEJO

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="prueba">
  <xs:complexType>
   <xs:attribute name="atributo" use="required" type="xs:integer"/>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

# Elementos complejos

- SON ELEMENTOS QUE CONTIENEN A OTROS ELEMENTOS HIJOS, O QUE TIENEN ATRIBUTOS
- SE SUELEN DIVIDIR EN 4 TIPOS:
  - ELEMENTOS VACÍOS
  - ELEMENTOS NO VACÍOS CON ATRIBUTOS
  - ELEMENTOS CON ELEMENTOS HIJOS
  - ELEMENTOS CON ELEMENTOS HIJOS Y CON “TEXTO” O VALOR PROPIO (COMO EL CONTENIDO MIXTO DE LAS DTD)

# Elementos complejos. Sintaxis (I)

- <xs: element name="nombreElementoCompuesto">

<xs:complexType>

<Indicador de orden>

Elementos / Atributos

</Indicador de orden>

</xs:complexType>

</xs:element>

Donde <Indicador de orden> puede ser: sequence, choice, all.

# Elementos complejos. Sintaxis (II)

<xs:sequence> ---> Han de aparecer todos los elementos en el orden indicado en la secuencia => Equivale a (x, y, z) en DTD.

<xs:choice> ---> Solo uno de los elementos hijos especificados debe de aparecer. => Equivale a (x|y|z) en DTD.

<xs:all> ---> Los hijos deben de aparecer todos pero en cualquier orden, y una sola vez. => No tiene equivalente en DTD.

# Elementos complejos. EJEMPLOS

<product pid="1345"/> <! --TIENE UN ATRIBUTO -->

<food type="dessert">Ice cream</food>  <! -- ATRIBUTO -->

<description>Sucedió el <date>03.03.99</date> .... </description>  <! - - MEZCLA CONTENIDO Y ELEMENTOS -->

<employee> <! -- TIENE HIJOS -->

  <firstname>John</firstname>

  <lastname>Smith</lastname>

</employee>

# DECLARAR ELEMENTOS COMPLEJOS

- Para definir elementos complejos  de tipo secuencia se utiliza la siguiente sintáxis:

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

# DECLARAR ELEMENTOS COMPLEJOS

- Podemos usar otra sintáxis para reutilizar la “definición” de los elementos hijos en varios elementos:

```
<xsd:element name="employee" type="personinfo"/>
<xsd:element name="student" type="personinfo"/>
<xsd:complexType name="personinfo">
  <xsd:sequence>
      <xsd:element name="firstname" type="xsd:string"/>
      <xsd:element name="lastname" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
```

EMPLOYEE ES DE TIPO PERSONINFO QUE SE DEFINE AHI

# DECLARAR ELEMENTOS COMPLEJOS

- EN LA DECLARACIÓN DE ELEMENTOS COMPLEJOS, ES POSIBLE UTILIZAR UN MECANISMO DE “HERENCIA” PARA REUTILIZAR O EXTENDER ELEMENTOS DEFINIDOS CON ANTERIORIDAD (VER LA SIGUIENTE PÁGINA)

# DECLARAR ELEMENTOS COMPLEJOS

```
<xsd:element name="employee" type="fullpersoninfo"/>
<xsd:complexType name="personinfo">
   <xsd:sequence>
         <xsd:element name="firstname" type="xsd:string"/>
         <xsd:element name="lastname" type="xsd:string"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="fullpersoninfo">
   <xsd:complexContent>
         <xsd:extension base="personinfo">
               <xsd:sequence>
                     <xsd:element name="address" type="xsd:string"/>
                     <xsd:element name="city" type="xsd:string"/>
                     <xsd:element name="country" type="xsd:string"/>
               </xsd:sequence>
         </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```

DECLARO EL TIPO PERSONINFO

DECLARO OTRO TIPO

QUE AMPLIA PERSONINFO

# DECLARAR ELEMENTOS COMPLEJOS

- Para declarar un elemento vacío con atributos, se utilizará la siguiente sintáxis:

```
<xsd:element name="product">
<xsd:complexType>
<xsd:attribute name="prodid" type="xsd:positiveInteger"/>
</xsd:complexType>
</xsd:element>
```

- <product prodid="1345" />

# DECLARAR ELEMENTOS COMPLEJOS

- Para declarar un elemento no vacío con atributos, y sin elementos hijos, se utilizará la siguiente sintáxis:

```
<xsd:element name="shoesize">
   <xsd:complexType>
         <xsd:simpleContent>
               <xsd:extension base="xsd:integer">
                     <xsd:attribute name="country" type="xsd:string" />
               </xsd:extension>
         </xsd:simpleContent>
   </xsd:complexType>
</xsd:element>
```

# DECLARAR ELEMENTOS COMPLEJOS

- Para declarar un elemento con contenido “mixto”, basta con añadir un atributo “mixed” al elemento xsd:complexType: (equivale a (#PCDATA, TIPO1, .., TIPO_N)* ).

```
<xsd:element name="letter">
   <xsd:complexType mixed="true">
         <xsd:sequence>
               <xsd:element name="name" type="xsd:string"/>
               <xsd:element name="orderid" type="xsd:positiveInteger"/>
               <xsd:element name="shipdate" type="xsd:date"/>
         </xsd:sequence>
   </xsd:complexType>
</xsd:element>
```

# DECLARAR ELEMENTOS COMPLEJOS

- La declaración anterior permitiría un texto como el siguiente, es decir, intercalar información y tags:

<LETTER>ESTIMADO CLIENTE:

<NAME>JUAN PEREZ</NAME>

. SU PEDIDO NÚMERO

<ORDERID>1032</ORDERID>

SE ENVIARÁ EL DÍA

<SHIPDATE>2001-07-13 </SHIPDATE>.

</LETTER>

# DECLARAR ELEMENTOS COMPLEJOS: INDICADORES

- EN LOS EJEMPLOS ANTERIORES HEMOS UTILIZADO EL ELEMENTO XSD:SEQUENCE COMO ELEMENTO HIJO DEL ELEMENTO XSD:COMPLEXTYPE
- XSD:SEQUENCE INDICA QUE LOS ELEMENTOS ANIDADOS EN ÉL DEBEN APARECER EN UN ORDEN DETERMINADO
- LOS ESQUEMAS XML NOS OFRECEN OTRAS ALTERNATIVAS, ADEMÁS DE XSD:SEQUENCE, PARA INDICAR CÓMO SE DEBEN TRATAR LOS ELEMENTOS QUE APARECEN ANIDADOS EN UN ELEMENTO COMPLEJO
- LAS OPCIONES O “INDICADORES” SON: XSD:ALL Y XSD:CHOICE

# DECLARAR ELEMENTOS COMPLEJOS: INDICADOR XSD:ALL

- EL INDICADOR XSD:ALL INDICA QUE LOS ELEMENTOS QUE CONTIENE PUEDEN APARECER EN CUALQUIER ORDEN, PERO COMO MÁXIMO SÓLO UNA VEZ

```
<XSD:ELEMENT NAME="PERSON">
  <XSD:COMPLEXTYPE>
      <XSD:ALL>
          <XSD:ELEMENT NAME="FIRSTNAME" TYPE="XSD:STRING"/>
          <XSD:ELEMENT NAME="LASTNAME" TYPE="XSD:STRING"/>
      </XSD:ALL>
  </XSD:COMPLEXTYPE>
</XSD:ELEMENT>
```

# DECLARAR ELEMENTOS COMPLEJOS: INDICADOR XSD:CHOICE

- EL INDICADOR XSD:CHOICE INDICA QUE PUEDE APARECER SÓLO UNO DE LOS ELEMENTOS QUE CONTIENE

```
<XSD:ELEMENT NAME="PERSON">
  <XSD:COMPLEXTYPE>
      <XSD:CHOICE>
          <XSD:ELEMENT NAME="FIRSTNAME" TYPE="XSD:STRING"/>
          <XSD:ELEMENT NAME="LASTNAME" TYPE="XSD:STRING"/>
      </XSD:CHOICE>
  </XSD:COMPLEXTYPE>
</XSD:ELEMENT>
```

# DECLARAR ELEMENTOS COMPLEJOS: INDICADORES MAXOCCURS Y MINOCCURS

- ESTOS INDICADORES SE UTILIZAN PARA INDICAR EL NÚMERO MÁXIMO Y MÍNIMO DE VECES QUE PUEDE APARECER UN ELEMENTO HIJO DE UN ELEMENTO COMPLEJO
- EL ATRIBUTO MAXOCCURS PUEDE TOMAR EL VALOR “UNBOUNDED”, QUE INDICA QUE NO EXISTE NINGÚN LÍMITE

(NO TIENEN EQUIVALENTE EN DTD)

```
<XSD:ELEMENT NAME="PERSON">
   <XSD:COMPLEXTYPE>
         <XSD:SEQUENCE>
         <XSD:ELEMENT NAME="FULL_NAME" TYPE="XSD:STRING"/>
         <XSD:ELEMENT NAME="CHILD_NAME" TYPE="XSD:STRING" MAXOCCURS="10"/>
         </XSD:SEQUENCE>
   </XSD:COMPLEXTYPE>
</XSD:ELEMENT>
```

# EL MODELO DE CONTENIDO ANY

- EN ESQUEMAS XML TAMBIÉN CONTAMOS CON UN MODELO DE CONTENIDO ANY, QUE PERMITE INCLUIR ELEMENTOS NO DECLARADOS INICIALMENTE EN EL ESQUEMA

```
<XSD:ELEMENT NAME="PERSON">
  <XSD:COMPLEXTYPE>
      <XSD:SEQUENCE>
          <XSD:ELEMENT NAME="FIRSTNAME" TYPE="XSD:STRING"/>
          <XSD:ELEMENT NAME="LASTNAME" TYPE="XSD:STRING"/>
          <XSD:ANY MINOCCURS="0"/>  <!– PERMITE AÑADIR UN ELEMENTO PERO TIENE QUE ESTAR DEFINIDO EN EL XSD, NO VALE CUALQUIER COSA -- >
      </XSD:SEQUENCE>
  </XSD:COMPLEXTYPE>
</XSD:ELEMENT>
```

# EL MODELO DE CONTENIDO ANY PARA ATRIBUTOS

- También contamos con un elemento que permite extender el número de atributos de un elemento:

```
<xsd:element name="person">
   <xsd:complexType>
            <xsd:sequence>
                     <xsd:element name="firstname" type="xsd:string"/>
                     <xsd:element name="lastname" type="xsd:string"/>
            </xsd:sequence>
            <xsd:anyAttribute/>
   </xsd:complexType>
</xsd:element>
```

EJERCICIO 1

Obtener el xsd del siguiente XML

```
<?xml version="1.0" encoding="UTF-8"?>
<student>
<firstName>Carlos</firstName>
<lastName>Fernandez</lastName>
<id>1000</id>
<plan>
<courses year="3">
<course>
<name> Programacion Orientada a Objetos</name>
<shortName>POO</shortName>
<record>
<grade>30</grade>
<date>13/06/11</date>
</record>

</course>
<course>
<name>Analisis y proyectos de software</name>
<shortName>APS</shortName>
</course>
</courses>
</plan>
</student>
```

# EJERCICIO 1 SOLUCION

- <?xml version="1.0" encoding="UTF-8"?>
- <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
- <xsd:element name="student">
- <xsd:complexType>
- <xsd:sequence>
- <xsd:element name="firstName" type="xsd:string"/>
- <xsd:element name="lastName" type="xsd:string"/>
- <xsd:element name="id" **type="xsd:ID"**/>
- <xsd:element ref="plan"/>
- </xsd:sequence>
- </xsd:complexType>
- </xsd:element>

# EJERCICIO 1 SOLUCION (II)

```
<xsd:element name="record">
<xsd:complexType>
<xsd:sequence>
<xsd:element name="grade" type="xsd:string"/>
<xsd:element name="date" type="xsd:date"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
<xsd:element name="course">
<xsd:complexType>
<xsd:sequence>
<xsd:element name="name" type="xsd:string"/>
<xsd:element name="shortName"
type="xsd:string"/>
<xsd:element ref="record" minOccurs="0"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
```

# EJERCICIO 1 SOLUCION(III)

```
<xsd:element name="courses">
<xsd:complexType>
<xsd:sequence minOccurs="0"
maxOccurs="unbounded">
<xsd:element ref="course"/>
</xsd:sequence>
<xsd:attribute name="year" type="xsd:string"/>
</xsd:complexType>
</xsd:element>
<xsd:element name="plan">
<xsd:complexType>
<xsd:sequence minOccurs="0"
maxOccurs="unbounded">
<xsd:element ref="courses"/>
</xsd:sequence>
</xsd:complexType>
<xsd:unique name="coursesYear">
<xsd:selector xpath="courses"/>
<xsd:field xpath="@year"/>
</xsd:unique>
</xsd:element>
```

# EJERCICIO 1 SOLUCION (IV)

```
<xsd:element name="courses">
<xsd:complexType>
<xsd:sequence minOccurs="0"
maxOccurs="unbounded">
<xsd:element ref="course"/>
</xsd:sequence>
<xsd:attribute name="year" type="xsd:string"/>
</xsd:complexType>
</xsd:element>
<xsd:element name="plan">
<xsd:complexType>
<xsd:sequence minOccurs="0"
maxOccurs="unbounded">
<xsd:element ref="courses"/>
</xsd:sequence>
</xsd:complexType>
<xsd:unique name="coursesYear">
<xsd:selector xpath="courses"/>
<xsd:field xpath="@year"/>
</xsd:unique>
</xsd:element>
```

EJERCICIO 2
DADO EL SIGUIENTE XML DEFINIR UN XSD QUE LO VALIDE:

```
<?xml version="1.0" encoding="UTF-8"?>
<email>
<from> carlos@gmail.com </from>
<to> destiny@Gmail.com </to>
<content>
Estimado <person> Juan </person>,  aquí tienes las facturas que  habias solicitado para el pago del curso <course>Lenguajes de marcas</course>:
<exercises>
<exercise>
<topic> DTD </topic>
<description> Obtener dtds desde insntacias </description>
</exercise>
<exercise>
<topic> XPath </topic>
<description> Encuentra estudiantes con nota mayor que 7</description>
</exercise>
</exercises>
Saludos,
<person> Carlos </person>
</content>
</email>
```

# EJERCICIO 2. SOLUCION

```
<xs:schema
	xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
	<xs:element name="email">
		<xs:complexType>
			<xs:sequence>
				<xs:element ref="from"/>
				<xs:element ref="to"/>
				<xs:element ref="content"/>
			</xs:sequence>
		</xs:complexType>
	</xs:element>
	<xs:element name="from" type="xs:string"/>
	<xs:element name="to" type="xs:string"/>
	<xs:element name="content">
		<xs:complexType mixed="true">
			<xs:choice minOccurs="0" maxOccurs="unbounded">
				<xs:element ref="course"/>
				<xs:element ref="exercises"/>
				<xs:element ref="person"/>
			</xs:choice>
		</xs:complexType>
	</xs:element>
	<xs:element name="course" type="xs:string"/>
	<xs:element name="exercises">
		<xs:complexType>
			<xs:sequence>
				<xs:element maxOccurs="unbounded" ref="exercise"/>
			</xs:sequence>
		</xs:complexType>
	</xs:element>
	<xs:element name="exercise">
		<xs:complexType>
			<xs:sequence>
				<xs:element ref="topic"/>
				<xs:element ref="description"/>
			</xs:sequence>
		</xs:complexType>
	</xs:element>
	<xs:element name="topic" type="xs:NCName"/>
	<xs:element name="description" type="xs:string"/>
	<xs:element name="person" type="xs:NCName"/>
</xs:schema>
```

# EJERCICIO 2. CREAR XSD A PARTIR DE XML (I)

```
<?xml version="1.0" encoding="ISO-8859-1"? >
<alumnos xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:SchemaLocation="file:/C:/Users/Ana_Desktop/alumnos.xsd">
<alumno>
<nombre>Jose Ramón</nombre>
<apellidos>García González</apellidos>
<direccion>
<domicilio>El Pez, 12</domicilio>
<codigo_postal>85620</código_postal>
<localidad>Suances</localidad>
<provincia>Cantabria</provincia>
</direccion>
<contactar>
<telf._casa>985623165</telf._casa>
<telf._movil>611233544</telf._movil>
<telf._trabajo>965847536</telf._trabajo>
<email>pepito@educadistancia.com</email>
</contactar>
</alumno>
```

# EJERCICIO 2. CREAR XSD A PARTIR DE XML (I)

```
<alumnos xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:SchemaLocation="file:/C:/Users
/Ana_Desktop/alumnos.xsd">
   <alumno>
      <nombre>Jose Ramón</nombre>
      <apellidos>García González</apellidos>
      <direccion>
         <domicilio>El Pez, 12</domicilio>
         <codigo_postal>85620</codigo_postal>
         <localidad>Suances</localidad>
         <provincia>Cantabria</provincia>
      </direccion>
      <contactar>
         <telf._casa>985623165</telf._casa>
         <telf._movil>611233544</telf._movil>
         <telf._trabajo>965847536</telf._trabajo>
         <email>pepito@educadistancia.com</email>
      </contactar>
   </alumno>
</alumnos>
```

# EJERCICIO 2. SOLUCION (I)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xs:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<!-- elemento raíz -->
<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>
```

# EJERCICIO 2. SOLUCION (II)

```
<!-- Atributos del elemento usuario -->
<xs:attribute name="id" type="xs:string"/>
</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:complexType>
<xs:attribute name="cp" type="xsd:string"/>
</xs:complexType>
</xs:element>
```

# EJERCICIO 2. SOLUCION (III)

```
<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" minOccurs="0" maxOccurs="unbounded" >
<xs:complexType>
<xs:attribute name="href" type="xs:string"/>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:schema>
```

# Práctica 1

- Diseñar un esquema XML para crear documentos para un préstamo.
- En cada documento se indicarán:
  - El nombre y apellidos del bibliotecario
  - Fecha del préstamo y de devolución
  - Datos del lector (id, nombre, apellidos, teléfono y dirección) La dirección se dividirá en tipo de calle (que puede ser calle, avenida o plaza), nombre calle, número, piso y letra, c.p., localidad y provincia
  - Un máximo de tres ejemplares en préstamo. Para cada uno de ellos: el número de registro, título, autor(es)
  - El préstamo tendrá un atributo numérico que servirá como identificador

# Práctica 2

- Modificar un nuevo esquema, de forma que no todos los elementos estén anidados, utilizando las referencias.
- Primero declaramos los elementos simples. Luego declararemos los elementos complejos indicando su “modelo de contenido” mediante atributos ref.

# Práctica 3

- Crear un esquema xml para codificar datos de un pedido a un proveedor. Se indicarán los datos del proveedor (nif, nombre, dirección, localidad, téléfono), datos de la biblioteca, y el listado de items que se han pedido.
- Para cada item se indicará el número de unidades, precio, y también el precio total del pedido y el número de items.

# RESTRICCIONES AVANZADAS

- RESTRICCIONES SOBRE IDS
- SOLO PUEDEN SER DE TIPO SIMPLE:
- P.E. PERMITE UN IDENTIFICADOR FORMADO SOLO POR LETRAS:
- <xs:simpleType name="idcustom">
- <xs:restriction base="xs:ID">
- <xs:pattern value="[a-z]*" />
- </xs:restriction>
- </xs:simpleType>

# RESTRICCIONES AVANZADAS

- RESTRICCIONES SOBRE IDS
- SOLO PUEDEN SER DE TIPO SIMPLE:
- P.E. PERMITE UN IDENTIFICADOR FORMADO SOLO POR UN CONJUNTO DE IDS:
- <xs:simpleType name=“misacciones" >
- <xs:restriction base="xs:ID">
- <xs:pattern value=“telefonica"/>
- <xs:pattern value=“inditex"/>
- <xs:pattern value=“repsol"/>
- </xs:restriction>
- </xs:simpleType>