Estructura de documentos XML, W3C Esquemas

XML Schemas

- Son una sintáxis alternativa para las DTDs, propuesta inicialmente por Microsoft, ArborText, Inso, etc.
- Utilizan la sintáxis propia de XML
- Ventajas:
  - Fáciles de aprender (se usa también XML)
  - Soportan tipos de datos: numéricos, fechas…
  - Procesables igual que los documentos XML

Qué encontramos en un esquema XML

- 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

XML Schemas

- La propuesta inicial de Microsoft dio lugar a los llamados "esquemas XDR"
- Posteriormente, el W3C diseñó un modelo de esquemas que es la propuesta oficial y la que debemos conocer (llamados "esquemas XSD")
- XSD se publicó como una recomendación el 31 de marzo del 2001 (se considera oficial desde mayo)
- Revisión en Octubre de 2004 (3 partes).
- XSD es más complejo que otras alternativas anteriores, pero supuso un importante paso hacia adelante en la estandarización de XML

Otras ventajas de XML Schemas

- 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

Asociar dtd a documentos XML

```
<?xml version="1.0"?>
<!DOCTYPE note SYSTEM "http://www.us.com/dtd/note.dtd">
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>
    Don't forget me this weekend!
</body>
</note>
```

Asociar esquemas a documentos XML

```
<?xml version="1.0"?>
<note   xmlns="http://www.us.com "
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation= "http://www.us.com note.xsd">
<to>Juan</to>
<from>Dani</from>
<heading>Recuerda</heading>
<body>
   No olvides llamar a Pablo!
</body>
</note>
```

Ejemplo esquema W3C

```
<?xml version="1.0"?>
<xsd:schema schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:target="“www.us.com" targetNamespace="“www.uv.com">
   <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>
```

Esquemas XML – elemento schema

- Los elementos utilizados en la creación de un esquema "proceden" del espacio de nombres:

  http://www.w3.org/2001/XMLSchema

- El elemento *schema* es el elemento raíz del documento en el que se define el esquema:

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

Esquemas XML – 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="xsd:string"/>

  <xsd:element name=“edad" type="xsd:integer"/>

  <xsd:element name=“fecNac" type="xsd:date"/>

Esquemas XML – elementos "simples", tipos de datos

- Los tipos de datos más utilizados son:
  - xsd:string
  - xsd:decimal
  - xsd:integer
  - xsd:boolean
  - xsd:date
  - xsd:time

  (ver http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/datatypes.html)
- 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"/>
```

Esquemas XML – atributos (1)

- 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

Esquemas XML – atributos (2)

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

  <xsd:attribute name="idioma" type="xsd:string" default="ES"/>

- 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)

Esquemas XML – facetas

- 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…

Esquemas XML – facetas (ej. 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>
```

Esquemas XML – facetas (ej. 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>
```

Esquemas XML – facetas (ej. 2, alt.)

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

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

Esquemas XML – facetas (ej. 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 1 letra minúscula (sólo 1)

Esquemas XML – facetas (ej. 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)

Esquemas XML – facetas (ej. 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. 6)

```
<xsd:element name="prodid">
   <xsd:simpleType>
        <xsd:restriction base="xsd:integer">
        <xsd:pattern value="[0-9][0-9][0-9][0-9][0-9]"/>
        </xsd:restriction>
</xsd:simpleType>
</xsd:element>
```

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>
```

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

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

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

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

- Ejemplos:

```
<product pid="1345"/>

<food type="dessert">Ice cream</food>


<description>Sucedió el <date>03.03.99</date> .... </description>


<employee>
    <firstname>John</firstname>
    <lastname>Smith</lastname>
</employee>
```

Declarar elementos complejos

- Para definir elementos complejos 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>
```

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)

```
<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>
```

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:

```
<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:

```
<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

```
<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"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

El modelo de contenido ANY

- 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>
```

Práctica 1

- Diseñar un esquema XML para crear documentos para el 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.