UT7:
Definición de esquemas y vocabularios en XML

Módulo: Lenguajes de marcas y sistemas de gestión de información

XML Schemas

- Son una sintaxis alternativa para las DTDs,
- Utilizan la sintaxis 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 DTDs), 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

Otras ventajas de XML Schemas

- Mayor precisión en la definición de tipos de datos mediante formatos y facetas
- 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 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)
- XSD es más complejo que otras alternativas anteriores, pero supuso un importante paso hacia adelante en la estandarización de XML

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/schema/note.xsd">
      <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:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="“note.xsd">
      <to>Tove</to>
      <from>Jani</from>
      <heading>Reminder</heading>
      <body>
          Don't forget me this weekend!
      </body>
</note>
```

Ejemplo esquema W3C

```
<?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>
```

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=“apellidos” type="xsd:string"/>

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

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

Esquemas XML – elementos “simples”, tipos de datos

Tenemos dos posibles variantes:

- **Primitivos**: tipos básicos de XML, están ya definidos por el propio lenguaje XML.
- **Derivados**: Tipos de datos más complejos creados a partir de los anteriores.

Esquemas XML – elementos “simples”, tipos de datos

- Los tipos de datos primitivos 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

Ejemplo:

```
<xsd:element name="color" type="xsd:string" default="red"/>
<xsd:element name=“ciudad" type="xsd:string" fixed=“Madrid"/>
```

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"/>

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

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

Esquemas XML – atributos (2)

- Los atributos pueden tener valores y se debe añadir a su declaración el atributo “use” para especificarlos.
  - Por defecto, “default”
  - Fijos, “fixed”
  - Opcionales, “optional”
  - Obligatorios, “required”

```
<xsd:attribute name=“idioma" type="xsd:string" default=“ES"/>
<xsd:attribute name="lang" type="xsd:string" use="required"/>
<xsd:attribute name="lang" type="xsd:string" use=“fixed“
    value=“español”/>
```

- 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…

Esquemas XML – facetas

- Tipos de facetas:
  a) Valor comprendido en un rango
  b) El valor está restringido a un conjunto de valores posibles
  c) Restringir el valor de un elemento a una serie de caracteres
  d) Longitud de los valores de los elementos…

Esquemas XML – facetas

Establecer tipos simples por restricción

Permiten establecer reglas complejas que deben de cumplir los datos. En este caso dentro de la etiqueta **simpleType** se indica una etiqueta **restriction**, dentro de la cual se establecen las posibles restricciones. Sintaxis:

<xsd:**simpleType name**="nombre">

<xsd:**restriction base**="tipo">

...definición de la restricción...

<xsd:**restriction**>

</xsd:**simpleType**>

El atributo **base** sirve para indicar en qué tipo nos basamos al definir la restricción (Es decir de que tipo estamos creando este derivado)

Esquemas XML – facetas (ej. 1)

a) Valor comprendido en un rango

```
<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)

b) El valor está restringido a un conjunto de valores posibles

```
<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)

c) Restringir el valor de un elemento a una serie de caracteres

```
<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 (sólo 1), valores válidos de la “a” a la “z”

Esquemas XML – facetas (ej. 4)

c) Restringir el valor de un elemento a una serie de caracteres

```
<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)

c) Restringir el valor de un elemento a una serie de caracteres

```
<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*.

[^xyz] prohibe usar cualquiera de los caracteres entre corchetes.

Esquemas XML – facetas (ej. 6)

c) Restringir el valor de un elemento a una serie de caracteres

```
<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>
```

En este ejemplo, el elemento “prodid” esta formado por una combinación de 5 dígitos con valores del “0” al “9” .

También se puede definir así: <xsd:pattern value="[0-9]{5}"/>

Esquemas XML – facetas (ej. 7)

c) Restringir el valor de un elemento a una serie de caracteres

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

El valor aceptable es una o más repeticiones de las letras minúsculas de la “a” a la “z”

?, acepta el carácter precedente cero o más veces.

*, acepta el carácter precedente una o más veces.

Esquemas XML – facetas (ej. 8)

d) Longitud de los valores de los elementos

```
<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

{n} acepta exactamente n repeticiones del carácter precedente.

{n,} acepta al menos n repeticiones del carácter precedente.

{n,m} acepta entre n y m repeticiones del carácter precedente.

Esquemas XML – facetas (ej. 9)

d) Longitud de los valores de los elementos

```
<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:
  a) Elementos con elementos hijos
  b) Elementos vacíos
  c) Elementos no vacíos con atributos
  d) Elementos con elementos hijos y con “texto” o valor propio

Elementos complejos

a) Elemento con elementos hijos

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

b) Elemento vacío

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

c) Elemento no vacio con atributo

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

d) Elemento con elemento hijo y con texto

```
<description>Sucedió el <date>03/03/99</date> ....</description>
```

Declarar elementos complejos

Tipo a) Elemento con elementos hijos, sintaxis:

```
<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 sintaxis 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

Tipo b) Para declarar un elemento vacío con atributos, se utilizará la siguiente sintáxis:

Ejemplo: <product prodid="1345" />

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

Declarar elementos complejos

Tipo c) Para declarar un elemento no vacío con atributos y sin elementos hijos, se utilizará la siguiente sintaxis:

Ejemplo: <shoesize country="ES">40</shoesize>

```
<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

Tipo d) 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

Tipo d) 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>
```

Declarar elementos complejos: Ejemplo

```
<xsd:element name="correo">
<xsd:complexType>
    <xsd:sequence>
        <xsd:element name="remite" type="xsd:string"/>
        <xsd:element name="para" type="xsd:string"
          minOccurs="1" maxOccurs="unbounded"/>
     <xsd:choice>
            <xsd:element name="cc" type="xsd:string"
          minOccurs="1" maxOccurs="unbounded"/>
          <xsd:element name="cco" type="xsd:string"
          minOccurs="1" maxOccurs="unbounded"/>
    </xsd:choice>
  </xsd:sequence>
</xsd:complexType>
</xsd:element>
```

En este ejemplo, el elemento *correo* consta de tres elementos: *remite*, *para* y un tercero que puede ser *cc o cco*.

Herramientas para XML

- Herramienta de validación on-line: comprueba si un documento xml está bien formado con respecto a un esquema.

http://www.corefiling.com/opensource/schemaValidate.html

- Herramienta para validar ficheros .xml y para generar esquemas automáticamente a partir de documentos xml:
- http://www.flame-ware.com/products/xml-2-xsd

- http://www.thaiopensource.com/relaxng/trang.html