ESQUEMAS

Tema 7 – Lenguajes de Marcas

LOS ESQUEMAS Y XML

Son una sintáxis alternativa para las DTDs.

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

Características de los esquemas

- Se definen como documentos XML, en un documento aparte, con extensión .XSD.
- Los documentos XML basados en esquemas, incluyen una referencia al archivo .XSD.
- El esquema contiene
  - Un **prólogo** como cualquier documento xml.
  - Un **elemento raíz** que se llama (siempre) **schema** acompañado del **atributo xmlns** (indica cuál es el espacio de nombres en el que se basa)

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

Elementos

- Cada elemento del documento .xml se declara dentro del esquema como

  <element name=.....>

  es decir, será el valor del atributo ***name***.

- Se comenzará obligatoriamente por el elemento raíz:

  <element name="**pelicula**">

Elementos simples

- Es aquél elemento que sólo puede contener cualquier tipo de dato, y no elementos hijos asociados ni atributos.
- Para definir un elemento simple, incluimos dos atributos en su definición:

  <element **name**="xxx" **type**="yyy"/>

  - el atributo ***name*** contiene el identificador del elemento
  - el atributo ***type*** contiene el tipo de datos que permitiremos que contengan estos elementos.
- Ejemplos:

  <element name=“titulo” type="string"/>

  <element name=“director" type="integer"/>

  <element name=“fecEstreno" type="date"/>

Elementos simples

Un elemento simple puede tener

- un **valor por defecto: default**

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

- un **valor “fijo”: fixed**

  <element name=”cuotaPagada” type=”boolean” fixed=”false”/>

Elementos complejos

Cuando el elemento esté formado por otros elementos y/o atributos propios, lo indicaremos con la etiqueta ***complexType. (Elemento complejo)***

<complexType>

</complexType>

Después le sigue otra etiqueta que indica su forma de agruparse, son los **indicadores de orden** y que pueden ser:

- **<sequence>**
- **<all>**
- **<choice>**

<sequence>

Indica un conjunto de elementos hijos que deberán incluirse en los documentos, estrictamente en el orden en que están declarados.

```
<element name="pelicula">
  <complexType>
    <sequence>
      <element name="titulo" type="string"/>
      <element name="director" type="string"/>
    </sequence>
  </complexType>
</element>
```

<all>

Contiene todo el conjunto de elementos hijos que se describen a continuación, pero sin importar el orden en el que se incluyen.

```
<element name="pelicula">
    <complexType>
      <all>
            <element name="titulo" type="string"/>
           <element name="director" type="string"/>
      </all>
    </complexType>
  </element>
```

<choice>

Indica que del conjunto de elementos hijos que se describen, el documento solo elegirá uno de ellos.

```
<element name="pelicula">
    <complexType>
      <choice>
        <element name="titulo" type="string"/>
        <element name="director" type="string"/>
      </choice>
    </complexType>
  </element>
```

Atributos

Son complementos de información que se pueden asignar a un elemento previamente declarado.

El elemento <complexType> (que contiene la agrupación de los elementos) incluye también la descripción de los atributos que pertenecen a un elemento, mediante la siguiente sintaxis:

<**attribute** name="xxx" type="yyy">

</**attribute**>

Ejemplo:

<attribute name="idioma" type="string">

</attribute>

que contiene la definición del nombre y el tipo de datos que podrá tener cada atributo.

Casos que nos podemos encontrar

Podemos encontrarnos tres casos distintos al usar los **atributos** en los elementos:

a) Elementos que contienen grupos de **elementos y atributos.**

b) Elementos **vacíos** que contienen **atributos**.

c) Elementos que contienen directamente **texto y atributos.**

Y sin atributos:

Elementos que contienen otros elementos y texto

Elementos que contienen grupos de elementos y atributos

La definición de los atributos se incluirán siempre después de la correspondiente al grupo de los elementos que lo forman.

```
<element name="pelicula">
  <complexType>
    <sequence>
      <element name="titulo" type="string"/>
      <element name="director" type="string"/>
    </sequence>
    <attribute name="estreno" type="string" > </attribute>
    <attribute name="minutos" type="integer"
                                        use="optional" default ="100" >
      </attribute>
  </complexType>
</element>
```

Elementos vacíos que contienen atributos

Se trata también de un caso complejo, por tanto lo especificamos con *complexType.*

Si queremos validar <clase genero="drama">, la estructura del esquema será similar a la declarada cuando contenía elementos pero solo se especifica el atributo anidado dentro de *complexType:*

```
<element name="clase">
        <complexType>
                <attribute name="genero" type="string"
                                                use="required"/>
        </complexType>
</element>
```

Elementos que contienen texto y atributos

Si queremos validar un contenido como el siguiente:

<titulo genero="drama">Casablanca</titulo>

en el que hay un atributo en el elemento que contiene el título de la película, usaremos dos nuevas etiquetas :

- ***simpleContent***
- ***extension*** con su atributo ***base*** para especificar el tipo de datos del elemento y después se describen los atributos que contiene y se relacionan anidados.

Elementos que contienen texto y atributos

```
<element name="titulo">
   <complexType>
      <simpleContent>
         <extension base="string">
            <attribute name="genero" type="string" use="required"/>
          </extension>
       </simpleContent>
   </complexType>
</element>
```

Elementos que contienen otros elementos y texto

Para declarar un elemento con contenido “mixto”, basta con añadir un atributo “mixed” al elemento complexType. Cuando utilicemos el atributo “mixed” recurriremos a los espacios de nombres.

```
<xs:element name="productora">
    <xs:complexType mixed="true">
        <xs:sequence>
            <xs:element name="productor" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
<xs:/element>
```

Valores que pueden tomar los atributos

- Los atributos pueden tener valores **por defecto**, **fijos** y **opcionales.**
- Si no se indica nada, los atributos son **opcionales**. En caso de querer obligar su incorporación tendremos que poner **use** = "**required**"

```
<attribute name="letra" type="string" default="A"/>
<attribute name="letra" type="string" fixed="A"/>
<attribute name="letra" type="string" use="required"/>
```

- También podemos poner el valor **use**="**optional**" si el atributo no es obligatorio.

Tipos de datos

Tipos de dato	Significado	Ejemplos
string	Cualquier cadena de texto que contenga letras y/o numeros	“Casablanca” “0005RX” “964 23 25 25”
boolean	Valores lógicos correspondientes a cierto y falso	false, true
integer byte short long ….	Números enteros, positivos o negativos. Se diferencian solo por el rango de valores que pueden representar.	10 -88 0 125
decimal float double ….	Números con parte decimal	15.3 300.0 -80.7777
time	Hora	12:30:00
date	Fecha (formato AAAA-MM-DD)	13/02/11

Indicadores de ocurrencia de los elementos

Nos permiten conocer cuántas veces pueden repetirse cada elemento.

<minOccurs> indica el mínimo número de veces que el elemento hijo puede aparecer en el documento.

- Si no se pone nada, su valor será 1, por lo que el elemento deberá figurar obligatoriamente una vez.
- Si contiene el valor 0 indicará que el valor es opcional.

Indicadores de ocurrencia de los elementos

<maxOccurs> contiene el máximo número de veces que un elemento puede aparecer en el documento.

- Su valor por defecto también es 1.
- El valor ***unbounded*** significa que no tiene límite, por lo que se podrá incluir el elemento un número indefinido de veces.

  <element name=”agenda” maxOccurs=”**unbounded**”>

Restricciones

- Las restricciones nos permiten establecer un rango de valores en los que un elemento o atributo pueden moverse.
- Utilizamos el elemento ***simpleType*** que contiene el atributo name con el que ponemos un identificador a este nuevo tipo, y que anida otro elemento ***restriction*** que a su vez contiene los posibles valores mediante los elementos ***enumeration***
- Tipos de restricciones:
  - 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…

Restricciones más usadas

Elemento	Significado	Ejemplos
pattern	Patrón para los caracteres	patern value= “([0-9\s])*” Validaria todos los dígitos del 0 al 9 y espacios en blanco, sin límite de número de caracteres ---------- pattern value=”{3}-[A-Z]{2} “ Validaria un dato con tres dígitos, un guión y dos letras mayúsculas como: 732-AB ----------
length	Número de caracteres permitidos totales	length value =”9” Nos permite tener un valor de exactamente nueve caracteres
minLength	Número de caracteres mínimos permitidos	MinLength value = “5” Valor con más de 5 caracteres
maxLength	Número de caracteres máximos permitidos	MaxLength value = ”10” Valor con menos de 10 caracteres

Ejemplo 1 de restricciones

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

Ejemplo 2 de restricciones

```
<xs:element name="coche">
<xs:simpleType>
    <xs:restriction base="xs:string">
        <xs:enumeration value="Audi"/>
        <xs:enumeration value="Golf"/>
        <xs:enumeration value="Seat"/>
    </xs:restriction>
</xs:simpleType>
</xs:element>
```

Ejemplo 3 de restricciones

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

En este ejemplo, el elemento “letra” debe tomar como valor 1 letra minúscula (sólo 1)

Ejemplo 4 de restricciones

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

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

Ejemplo 5 de restricciones

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

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

Ejemplo 6 de restricciones

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

Ejemplo 7 de restricciones

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

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

Ejemplo 8 de restricciones

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