4. Definición de esquemas y vocabularios en xml.

Autor	Ⓧ Xerach Casanova
Clase	Lenguajes de Marcas
Fecha	@Feb 3, 2021 7:48 PM

1. Documento XML. Estructura y sintaxis

Los documentos XML están formado por:

- **Prólogo:** informa al intérprete encargado de procesar el documento de todos aquellos datos que necesita para realizar el trabajo.
    - **Definición de XML.** se indica la versión de XML, el código de los datos a procesar y la autonomía del documento.
    - **Declaración del tipo de documento.** Hasta ahora solo hemos usado <!DOCTYPE y separado por al menos un espacio.
- **Ejemplar:** Contiene los datos del documento a procesar. Es el elemento raíz del documento y ha de ser único. Contiene elementos estructurados en árbol. La raíz es el ejemplar y las hojas los elementos terminales, que pueden a su vez estar formados por atributos.

Pero además de lo anterior, se deben definir las cualidades que tiene ese ejemplar. Para ello utilizaremos DTD's o XML Schemas.

2. Definición de tipo de documento, DTD.

Están formadas por una relación precisa de qué elementos pueden aparecer en un documento y dónde, asó como el contenido y atributos del mismo. Garantizan que los datos del XML cumplen las restricciones impuestas en la DTD:

- Especificar la estructura del documento.

- Reflejar una restricción de integridad referencial mínima utilizando (ED e IDREF).
- Utilizar pequeños mecanismos de abstracción comparables a las macros, que son las entidades.
- Incluir documentos externos.

Los principales inconvenientes de los DTD son:

- Su sintaxis no es XML.
- No soportan espacios de nombres.
- No definen tipos para los datos, solo hay un tipo de elementos terminales, que son los datos textuales.
- No se permiten secuencias ordenadas.
- No es posible formar claves a partir de varios atributos o elementos.
- Una vez definido el DTD no se puede añadir nuevo vocabulario.

Cuando se definen dentro del documento XML se ubican entre corchetes después del n nombre del ejemplar en el elemento <!DOCTYPE>, pero cuando se define en fichero externo, se hace en un fichero de texto plano con extensión dtd.

2.1. Declaraciones de tipos de elementos terminales

Los tipos terminales son elementos que se corresponden con hojas de la estructura de árbol formado por los datos del documento XML asociado al DTD.

La declaración de tipos de elementos se hace con la cadena <!ELEMENT> separada por al menos un espacio del nombre del elemento XML que se declara, seguidamente irá la declaración del contenido que puede tener el elemento.

En el caso de elementos terminales (que no contienen más elementos, esta declaración de contenido es dada por uno de los siguientes valores:

- EMPTY. El elemento no es contenedor.
- ANY: Permite que el contenido del elemento sea cualquier cosa.
- (#PCDATA): Indica que los datos son analizados en busca de etiquetas, resultando que el elemento no puede contener elementos. Es decir, solo puede contener datos de tipo carácter exceptuando: < &]] >,

```
<!ELEMENT A EMPTY>
<!ELEMENT A ANY>
<!ELEMENT A (#PCDATA)>
```

Ejemplo.

En la siguiente estructura

```
<alumno>Olga Velarde Cobo</alumno>
```

un DTD puede ajustarse a:

```
<!ELEMENT alumno (#PCDATA)>
```

2.2. Declaraciones de tipos de elementos no terminales

Definen las ramas de un documento, es decir, elementos que están formados por otros elementos.

```
<!ELEMENT A (B,C)>
```

El elemento A está formado por el elemento B, seguido de un elemento C.

Los siguientes operadores nos permiten definir la cardinalidad de un elemento.

- **Operador ?:** indica que el elemento no es obligatorio.

```
<!ELEMENT telefono (trabajo?, casa)>
```

- **Operador uno-o-más, +.** El componente estará presente al menos una vez.

```
<!ELEMENT provincia (nombre, (cp, ciudad)+)>
```

- **Operador cero-o-más, *.** Define un componente presento, cero, una o más veces.

```
<!ELEMENT provincia (nombre, (cp, ciudad)*>
```

- **Operador de elección |.** Si se utiliza sustituyendo las comas en la declaración de grupos indica que para formar el documento XML debes elegir entre elementos separados por ese operador.

```
<!ELEMENT provincia (nombre, (cp | ciudad) )>
```

Para la siguiente estructura:

```
<alumno>
<nombre>Olga</nombre>
<dirección>El Percebe 13</dirección>
</alumno>
```

Un DTD puede ser:

```
<!ELEMENT alumno (nombre, apellidos, direccion)>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT dirección (#PCDATA))>
```

2.3. Declaraciones de listas de atributos para los tipos de elementos

Para declarar los atributos asociados a un elemento se utiliza la cadena <!ATTLIST seguida del nombre del elemento asociado al atributo y luego el nombre del atributo, seguido del tipo y del modificador.

Este elemento puede usarse para declarar una lista de atributos asociada a un elemento, o repetirse el número de veces necesario para asociarlo a dicho elemento de manera individual.

No todos los atributos son del mismo tipo, los más destacados son:

- **Enumeración.** El atributo solo toma uno de los valores determinados dentro de un paréntesis separados por |

```
<!ATTLIST fecha dia_semana (lunes | martes | miércoles | jueves |viernes |sábado |domingo) #REQUIRED)>
```

- **CDATA** se utiliza cuando un atributo es una cadena de texto.
- **ID** permite declarar un atributo identificador de un elemento. El valor ha de ser único y además se deben tener en cuenta que los números no son nombres válidos en XML.
- **IDREF** permite hacer referencia a identificadores. En este caso el valor del atributo ha de corresponder con el identificador de un elemento existente.
- **NMTOKEN** permite declarar que el valor de un atributo ha de ser una sola palabra compuesta por los caracteres permitidos en XML.

También debemos indicar si un atributo es obligatorio o no con los siguientes modificadores:

- **#IMPLIED** el atributo sobre el que se aplica es opcional.
- **#REQUIRED** obligatorio
- **#FIXED** define un valor fijo para un atributo independientemente de que ese atributo se defina explícitamente en una instancia del elemento en el documento.
- **Literal,** asigna a un atributo el valor dado por una cadena entre comillas.

Ejemplo. para la siguiente estructura XML

```
<alumno edad=15>
<nombre>Olga</nombre>
<apellidos>Velarde Cobo</apellidos>
<dirección>El Percebe 13</dirección>
</alumno>
```

Un DTD válido podría ser:

```
<!ELEMENT alumno (nombre, apellidos, direccion)>
<!ATTLIST alumno edad CDATA #REQUIRED>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT apellidos (#PCDATA)>
<!ELEMENT dirección (#PCDATA)>
```

2.4. Declaraciones de entidades

Si queremos declarar valores constantes dentro de documentos debemos usar entidades. Se limitan por & y ;, por ejemplo &entdidad;

El intérprete sustituye la entidad por el valor asociado en el DTD. No se admite recursividad, por tanto una entidad no puede hacer referencia a ella misma. Se definen con el elemento <!ENTITY> y pueden ser de cuatro tipos:

- **Internas.** Existen cinco entidades predefinidas en el lenguaje que son:
  - **<**: corresponde al signo menor que, <
  - **>**: corresponde al signo mayor que, >
  - **"**: corresponde a las comillas rectas dobles, ".
  - **'**: corresponde a comilla simple, '.
  - **&**: corresponde al ampersand, &.

  Para definir una entidad diferente podemos usar la siguiente sintaxis:

  <!ENTITY nombre_entidad "valor de la entidad>

  ```
  <!ENTITY dtd "Definiciones de Tipo de Documento">
  ```

- **Externas:** Permiten establecer una relación entre el documento XML y otro documento a través de la URL de éste último.

```
<!ENTITY nombre_entidad SYSTEM "http://localhost/docsxml/fichero_entidad.xml">
```

El contenido de los ficheros es analizado y debe seguir sintaxis XML. Cuando se incluyen ficheros de formato binario, se utiliza la palabra reservada NDATA en la definición de la entidad y se asocia a dicha entidad una declaración de notación.

- **De parámetro:** da nombres a partes de un DTD y hace referencias a ellas a lo largo del mismo. Útiles cuando varios elementos del DTD comparten listas de atributos o especificaciones de contenidos. Se denotan por

%entidad;

```
<!ENTITY %direccion "calle, numero?, ciudad, cp">
<!ENTITY alumno (dni, %direccion;)>
<!ENTITY ies (nombre, %direccion;)>
```

- **De parámetro externas:** permite incluir en un DTD elementos externos.

```
<!ENTITY persona SYSTEM "persona.dtd">
```

2.5. Declaraciones de notación

Cuando se incluyen ficheros binarios en un fichero debemos usar la siguiente sintaxis:

```
<!NOTATION nombre SYSTEM aplicacion>
```

Por ejemplo, una notación llamada gif donde se indica que se hace referencia a un editor de formatos gif:

```
<!NOTATION gif SYSTEM "gifEditor.exe">
```

Para asociar una entidad externa no analizada a esa notación:

```
<!ENTITY dibujo SYSTWM "imagen.gif" NDATA gif>
```

2.6. Secciones condicionales

Permiten incluir o ignorar partes de la declaración de un DTD para ello se usa:

- INCLUDE, permite que se vea parte de la declaración del DTD

```
<![INCLUDE [Declaraciones visibles] ] >
```

Ejemplo:

```
<![INCLUDE [ <!ELEMENT nombre (#PCDATA)>] ]
```

- IGNORE, permite ocultar esa sección de declaraciones del DTD

```
<![IGNORE [Declaraciones ocultas] ] >
```

Por ejemplo

```
<![IGNORE [<!ELEMENT clave (#PCDATA)>] ] >
```

3. XML Schema

Para hacer que los elementos y atributos de los ficheros XML correspondan a datos de un tipo determinado y que cumplan ciertas restricciones se usan los XML Schemas.

Se definen en ficheros planos y son documentos XML con extensión xsd.

Los elementos XML que se utilizan para generar un esquema han de pertenecer al espacio de nombre XML Schema que es: http://www.w3.org/2001/XMLSchema.

El ejemplar de estos ficheros es <xs:schema>, contiene declaraciones para todos los elementos y atributos que puedan pertenecer a un documento XML asociado válido.

Los elementos hijos inmediatos de este ejemplar son <xs:element>, que nos permiten crear globalmente un documento. Esto significa que el elemento creado puede ser el ejemplar del documento XML asociado.

Ejemplo de esquema correspondiente a un documento XML

```
<?xml version="1.0" encoding="UTF-8" standalone="yes?>
<!DOCTYPE alumno>
<alumno edad="22">Olga Velarde Cobo</alumno>
```

Un XML Schema sería:

```
< ?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="alumno" type="xs:string"/>
</xs:schema>
```

3.1. Tipos de datos

Son los distintos valores que puede tomar el atributo type cuando se declara un elemento o atributo.

Algunos de estos valores son:

- **string** caracteres UNICODE
- **boolean**
- **integer**
- **positiveinteger**
- **negativeinteger**
- **decimal** números decimales
- **datetime** fecha y hora absolutas.
- **duration** duración de tiempo expresado en años, meses, días horas, minutos y segundos. El formato utilizado es: PnYnMnDTnHnMnS. Para representar una duración de 2 años, 4 meses, 3 días 5 horas, 6 minutos y 10 segundos se pondría P2Y4M3DT5H6M7S. Se pueden omitir valores nulos, para indicar duración negativa se pone - delante de la P.
- **time** hora en formato hh:mm:ss
- **date** fecha en formato CCYY-MM-DD
- **gYearMonth** mes de un año determinado en formato CCYY-MM
- **gYear** año en formato CCYY
- **gMonthDay** día de un mes en formato -MM-DD.
- **gDay** ordinal del día del mes en formati —DD, el 4º día será -04
- **gMonth** mes en formato -MM
- **anyURI** representa una URI
- **language** representa identificadores de lenguaje, valores definidos en RFC 1766
- **ID, IDREF, ENTITY, NOTATION, MTOKEN** representan lo mismo que en los DTD.

3.1.1. Facetas de los tipos de datos

Las restricciones que se pueden aplicar sobre los valores de los datos de elementos o atributos se definen por las facetas y se aplican sobre tipos simples, utilizando el elemento xs:restriction.

Se expresan como un elemento dentro de una restricción y se combinan para lograr restringir más el valor del elemento:

- **length, minlength, maxlength:** longitud del tipo de dato.
- **enumeration:** restringe a un determinado conjunto de valores
- **whitespace:** define el tratamiento de espacios (preserve/replace, collapse).
- **(max/min) (Inclusive/Exclusive):** límites superiores e inferiores de tipos de datos. Cuando son Inclusive, el valor que se determine es parte del conjunto de valores válidos.
- **totalDigits, fractionDigits:** número de dígitos totales y decimales de un número decimal.
- **pattern:** permite construir máscaras que han de cumplir los datos de un elemento:

Patrón	Significado
`[A-Z a-z]`	Letra.
`[A-Z]`	Letra mayúscula.
`[a-z]`	Letra minúscula.
`[0-9]`	Dígitos decimales.
`\D`	Cualquier carácter excepto un dígito decimal.
`(A)`	Cadena que coincide con A.
`A | B`	Cadena que es igual a la cadena A o a la B.
`AB`	Cadena que es la concatenación de las cadenas A y B.
`A?`	Cero o una vez la cadena A.
`A+`	Una o más veces la cadena A.
`A*`	Cero o más veces la cadena A.
`[abcd]`	Alguno de los caracteres que están entre corchetes.
`[^abcd]`	Cualquier carácter que no esté entre corchetes.
`\t`	Tabulación.

3.1.2. Facetas: ejercicios.

Creación de una cadena de texto con una longitud máxima de 9 caracteres y dos valores posibles.

```
<xs:simpleType name="estado">
  <xs:restriction base="xs:string">
    <xs:maxLength value="9"/>
    <xs:enumeration value="conectado"/>
    <xs:enumeration value="ocupado"/>
  </xs:restriction>
</xs:simpleType>
```

Creación de un elemento en el que se respetan los espacios tal y como se han introducido.

```
<xs:simpleType name="nombre">
  <xs:restriction base="xs:string">
    <xs:whitespace value="preserve"/>
  </xs:restriction>
</xs:simpleType>
```

Creación de un elemento calificaciones de dos dígitos cuyo valor es un número entero comprendido entre 1 y 10, ambos inclusive.

```
<xs:simpleType name="calificaciones">
  <xs:restriction base="xs:integer">
    <xs:totalDigits value="2"/>
    <xs:minExclusive value="0"/>
    <xs:maxInclusive value="10"/>
  </xs:restriction>
</xs:simpleType>
```

Creación de la máscara de un DNI mediante pattern.

```
<xs:simpleType name="dni">
  <xs:restriction base="xs:string">
    <xs:pattern value="[0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [A-Z]"/>
  </xs:restriction>
</xs:simpleType>
```

3.2. Elementos del lenguaje.

- **Esquema:** xs:schema contiene la definición del esquema.
- **Tipos completos:** xs:complexType define tipos complejos.
- **Tipos simples:** xs:simpleType define un tipo simple restringiendo sus valores.
- **Restricciones:** xs:restriction establece restricciones sobre un elemento de tipo base
- **Agrupaciones:** xs:group nombra agrupaciones de elementos y atributos para hacer referencia a ellas.
- **Secuencias:** xs:secuence construye elementos complejos mediante la enumeración de los que les forman.
- **Alternativa:** xs:choice representa alternativas, teniendo en cuenta que es una o-exclusiva.
- **Contenido mixto:** definido dando valor true al atribituto mixed del elemento xs:complexType, permite mezclar texto con elementos.
- **Secuencias no ordenadas:** xs:all, representa a todos los elementos en cualquier orden.

```
<?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>
        <!-- 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>
```

```
            <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>
```

3.3. Definición de tipos de datos XML Schema

En los XML Schema se diferencian también los elementos terminales y los elementos no terminales:

- **Tipos de datos simples.** Se suelen definir para hacer una restricción sobre un tipo de datos XSD ya definido y establece el rango de valores que se pueden tomar. Se pueden crear también tipos de datos simples basados en listas de valores utilizando derivedBy de simpleType.
- **Tipos de datos compuestos.** El elemento xsd:complexType permite definir estructuras complejas de datos. Su contenido son las declaraciones de elementos y a tributos, o referencias a elementos y atributos declarados de forma global.

3.3.1. Definición de tipos de datos: ejercicios

Creación de un elemento simple de nombre edad que representa la edad de un alumno de la ESO, por tanto su rango está entre los 12 y los 18 años.

```
<xs:element name="edad">
 <xs:simpleType>
     <xs:restriction base="xs:positiveInteger">
         <xs:minInclusive value="12"/>
         <xs:maxInclusive value="18"/>
     </xs:restriction>
 </xs:simpleType>
</xs:element>
```

Creación de una lista con los días de la semana en letras.

```
<xs:simpleType name="dia_semana" base="xs:string" derivedBy="list"/>
    <dia_semana>Lunes Martes Miercoles Jueves Viernes Sabado Domingo<dia_semana>
</xs:simpleType>
```

Creación de un elemento compuesto de nombre de alumno, formado por los elementos nombre, apellidos y web personal.

```
<xs:complexType name="alumno">
    <xs:secuence>
        <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="web" type="xs:string" minOccurs="0" maxOccurs="5">
        <xs:complexType>
            <xs:attribute name="href" type="xs:string"/>
        </xs:complexType>
        </xs:element>
    <xs:secuence>
 </xs:complexType>
```

3.2. Asociación con documentos XML

Una vez creado el fichero xsd a un documento XML es un espacio de nombres al ejemplar del documentos, donde se indica la localización de los ficheros esquema mediante su URI, precedida del prefijo xsi

```
<?xml version="1.0" encoding="ISO-8859-1"? >
<alumnos xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:SchemaLocation="file:/D:/ubicación/del archivo/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>
    <alumno>
        <nombre>Carlos</nombre>
        <apellidos>López Pérez</apellidos>
        <direccion>
            <domicilio>El Cangrejo, 25</domicilio>
            <codigo_postal>86290</código_postal>
            <localidad>Santillana</localidad>
            <provincia>Cantabria</provincia>
        </direccion>
        <contactar>
            <telf._casa>931132565</telf._casa>
            <telf._movil>623863544</telf._movil>
            <telf._trabajo>984657536</telf._trabajo>
            <email>carlos@educadistancia.com</email>
        </contactar>
    </alumno>
</alumnos>
```

3.5. Documentación del esquema

Para incorporar el autor, limitación de derechos de autor, utilidad del esquema etc, se puede realizar con el elemento xs:annotation, y permite guardar información adicional, a su vez se puede utilizar:

- xs:documentation, puede contener elementos XML bien estructurados.
- xs:appinfo, se diferencia poco de la anterior, este guarda información para los programas de software. También se usa para genera ayuda contextual de cada elemento del esquema.

```
<xs:schema xmlns:xsi=http://www.w3.org/2001/XMLSchema>
    <xs:annotation>
        <xs:documentation xml:lang ="es-es">
            Materiales para formación e-Learning
            <modulo>Lenguajes de marcas y sistemas de gestión de información.<modulo>
            <fecha_creación> 2011<fecha_creacion>
            <autor> Nuky La Bruji</autor>
        </xs:documentation>
    </xs:annotation>
    <xs:element name="lmsgi"  type=xs:string>
        <xs:annotation>
            <xs:appinfo>
                <texto_de_ayuda>Se debe de introducir el nombre completo del tema</texto_de_ayuda>
            <xs:appinfo>
        </xs:annotation>
    </xs:element>
</xs:schema>
```

4. Herramientas de creación y validación

- Editix XML Editor (Versiones open source y comercial).

- Microsoft Core XML Services (MSXML) (Gratuito).
- XMLFox (freeware).
- Altova XML Spy Edición Estándar (comercial).
- Editor XML xmlBlueprint. (comercial)
- Stylus Studio 2001 (comercial).
- Oxygen XML Editor (comercial).
- Exchanger XML Editor (comercial)
- XML copy editor (open source).

Mapa conceptual