Definición de Esquemas y Vocabularios en XML

1

Introducción

Se dice que un documento XML está bien formado cuando cumple las reglas de sintaxis definidas en XML.

Sin embargo un documento XML es **válido** cuando cumple con unas normas en cuanto a qué elementos y atributos pueden aparecer, en qué orden, que valores pueden tomar, etc.

Todo documento válido está bien formado pero no todo documento bien formado es válido.

Existen dos formas de definir las reglas de validación: **DTD** y **Esquemas**.

2

DTD (1)

Un DTD (*Document Type Definition* - Definición de Tipo de Documento) es una descripción de la estructura de un documento XML en la que se especificará qué elementos tienen que aparecer, en qué orden, cuáles son obligatorios y cuales optativos, qué atributos tienen esos elementos, qué valores pueden tomar los atributos, etc.

Este mecanismo de validación es anterior a XML ya que proviene de SGML y aunque actualmente se utiliza más la técnica de los esquemas (XSD), aún hay muchos documentos que sólo cuentan con DTD para su validación.

3

DTD (2)

El DTD se encarga de definir:

- **Léxico**: conjunto de palabras que se pueden utilizar.
- **Sintaxis**: conjunto de reglas que establecen la estructura del documento.

La importancia del DTD reside en que permite asegurar que los documentos cumplen con las normas establecidas y de esta forma podemos compartir información entre diferentes aplicaciones que pueden realizar cualquier operación sobre la información que contienen: consulta, inserción, modificación, borrado, etc.

4

DTD (3)

Podemos ubicar las declaraciones del DTD en el propio documento XML o en un documento externo (.dtd).

Se suele utilizar un documento externo por 2 razones:

- **Ahorro de espacio**: si almacenamos las declaraciones dentro del documento, éste ocupará mas espacio y además estaríamos repitiendo la misma información en todos los documentos que utilicen esa especificación.
- **Facilitar las modificaciones**: si almacenamos las declaraciones dentro del documento, cada vez que queramos modificar o añadir una declaración tendríamos que hacerlo en cada documento mientras que si lo hacemos en un documento externo con modificar éste es suficiente.

5

DTD (4)

Declaración del DTD

La declaración es la instrucción donde se indica que DTD se va a utilizar para validar el documento XML:

<!DOCTYPE>

La sintaxis de la declaración varía en función del tipo de DTD y éste depende de su ubicación y carácter:

- **Ubicación**:
  - Interno: las reglas aparecen en el propio documento XML.
  - Externo: las reglas aparecen en un archivo independiente
  - Mixto: mezcla de los anteriores. Las reglas internas tienen prioridad sobre las externas.

6

DTD (5)

- **Declaración del DTD**

- **Carácter**:
  - Uso privado (SYSTEM): el DTD se almacena de forma local y por tanto sólo va a ser utilizado por nosotros mismos.
  - Uso público (PUBLIC): el DTD se almacena en un lugar accesible a través de internet para que otras personas o aplicaciones puedan utilizarlo.

    Requiere un FPI (*Formal Public Identifier* - Identificador Público Formal).

7

DTD (6)

- **Declaración del DTD**

Elementos de la declaración:

- **Raíz**: nombre del elemento raíz del documento XML.
- **Privacidad**: indica si el elemento es público (PUBLIC) o de uso interno (SYSTEM).
- **URL**: ubicación del DTD. Sólo existe cuando todo o parte del DTD se encuentra en un archivo externo.
- **FPI**: identificador del DTD cuando su uso es público.

8

DTD (7)

- **Declaración del DTD**

Formato del FPI:

Está compuesto de 4 campos separados por //.

1) **Tipo de norma**:
   - No aprobada por una norma formal (-).
   - Aprobada por organismo no oficial (+).
   - Aprobada por organismo oficial (referencia al estándar).
2) **Nombre del organismo** responsable del estándar.
3) **Tipo de documento**, suele incluir la versión.
4) **Idioma**.

Ejemplo:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3c.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

9

DTD (8)

- **Declaración del DTD**

URN en lugar de FPI:

Cada vez está más extendido el uso de un URN (*Uniform Resource Name*) en lugar del FPI como identificador.

Para crear una URN a partir de un FPI simplemente añadimos al principio "urn:pubicid:" y a continuación el FPI sustituyendo // por : y los espacios en blanco por +

Ejemplo:

"-//W3C//DTD XHTML 1.0 Transitional//EN"

Convertido a URN sería:

"urn:publicid:-:W3C:DTD+XHTML+1.0+Transitional:EN"

10

DTD (9)

- **Declaración del DTD**

Las posibles combinaciones son:

Tipo DTD	Sintaxis
Privado e interno	<!DOCTYPE raíz [reglas]>
Privado y externo	<!DOCTYPE raíz SYSTEM URL>
Privado y mixto	<!DOCTYPE raíz SYSTEM URL [reglas]>
Público e interno	No tiene sentido (*)
Público y externo	<!DOCTYPE raíz PUBLIC FPI URL>
Público y mixto	<!DOCTYPE raíz PUBLIC FPI URL [reglas]>

(*) La combinación Público e interno no tiene sentido, ya que si lo almacenamos en nuestro sistema no puede estar accesible para otras personas o aplicaciones.

11

DTD (10)

- **Componentes del DTD**

En la definición del DTD se pueden utilizar 4 tipos de componentes:

- Elemento.
- Atributo.
- Entidad.
- Notación.

12

DTD (11)

- **Componentes del DTD: Elemento**

Indica la existencia de un elemento en el documento XML. Su definición se puede realizar mediante una categoría o indicando su contenido:

<!ELEMENT nombre_elemento categoría>

<!ELEMENT nombre_elemento (contenido)>

- **nombre_elemento**: es el nombre del elemento tal y como debe aparecer en el documento XML pero sin los signos de menor (<) y mayor (>).

  Ejemplos:

  <!ELEMENT catalogo (…) >

  <!ELEMENT libro (…) >

13

DTD (12)

- **Componentes del DTD: Elemento**

<!ELEMENT nombre_elemento categoría>

- **categoría**:
  - Cualquier cosa (**ANY**): establece como válido cualquier contenido eliminando comprobaciones. Suele utilizarse durante las pruebas y desaparecer en las versiones definitivas del DTD.

    Ejemplo: <!ELEMENT catalogo ANY>
  - Vacío (**EMPTY**): describe un elemento sin descendientes.

    Ejemplo: <!ELEMENT separador EMPTY>

14

DTD (13)

- **Componentes del DTD: Elemento**

<!ELEMENT nombre_elemento (contenido)>

- **contenido**:
  - Datos (**#PCDATA**): *Parsed Character Data*
    Cualquier conjunto de caracteres ya sean textuales, numéricos u otro formato que no contenga marcas.

    Ejemplo: <!ELEMENT nombre (#PCDATA)>
  - Descendientes (lista_elementos):
    Establece cuales son los elementos descendientes del elemento que se está describiendo.
  - Mixto: el contenido es una combinación de datos y elementos descendientes.

15

DTD (14)

- **Componentes del DTD: Elemento**

A la hora de definir la lista de descendientes de un elemento se puede establecer secuencia y cardinalidad.

- Secuencia: indica el orden en que deben aparecer los elementos (,) y si son excluyentes (|).
- Cardinalidad: indica el número de veces que puede aparecer un elemento o una secuencia de ellos.

Símbolo	Significado
A, B	Aparece el elemento A y a continuación el elemento B
A | B	Aparecerá el elemento A o el B, pero no ambos
?	El elemento (o secuencia) puede aparecer o no
*	El elemento (o secuencia) aparece de 0 a N veces
+	El elemento (o secuencia) aparece de 1 a N veces

16

DTD (15)

- **Componentes del DTD: Elemento**

Ejemplo:

Un correo electrónico estaría compuesto de los siguientes campos: de, para, cc (optativo), cco (optativo), asunto y mensaje.

```
<!ELEMENT correo_electronico (de, para, cc?, cco?, asunto, mensaje)>
<!ELEMENT de (#PCDATA)>
<!ELEMENT para (#PCDATA)>
<!ELEMENT cc (#PCDATA)>
<!ELEMENT cco (#PCDATA)>
<!ELEMENT asunto (#PCDATA)>
<!ELEMENT mensaje (#PCDATA)>
```

17

DTD (16)

- **Componentes del DTD: Atributo**

Indica la existencia de un atributo en un elemento y sus características.

Se puede hacer una declaración para cada atributo, pero lo habitual es utilizar una única declaración para todos los atributos de un elemento.

```
<!ATTLIST nombre_elemento
        nombre_atributo tipo_atributo carácter
        nombre_atributo tipo_atributo carácter
        …
>
```

18

DTD (17)

- **Componentes del DTD: Atributo**

- **nombre_atributo**: debe ser un nombre XML válido.
- **carácter**:
  - **"valor"**: establece un valor por defecto para el atributo.
  - **#IMPLIED**: el atributo es opcional y no se le asigna ningún valor por defecto.
  - **#REQUIRED**: el atributo es obligatorio pero no se le asigna un valor por defecto.
  - **#FIXED "valor"**: el atributo es obligatorio y su valor no puede ser otro más que el establecido.

19

DTD (18)

- **Componentes del DTD: Atributo**

- **tipo_atributo**:
  - **CDATA**: caracteres que no contienen etiquetas.

```
<!ATTLIST precio
        moneda  CDATA  #REQUIRED>
```

  - **Enumeración** (valor1 | valor2 | …): lista de valores de entre los cuales el atributo deberá tomar uno.

```
<!ELEMENT semaforo EMPTY>
<!ATTLIST semaforo
        color  ( rojo | naranja | verde )  "verde">
```

20

DTD (19)

- **Componentes del DTD: Atributo**

- **tipo_atributo**:
  - **ID**: identificador que permite identificar a un elemento de forma única. Para ello cada elemento sólo puede tener un atributo de tipo ID, y además dos elementos no pueden tener el mismo valor en atributos de tipo ID.
    Los valores asignados a estos atributos deben ser nombres XML válidos.
  - **IDREF**: representa el valor de un atributo ID de otro elemento, para ser válido debe coincidir con el valor del atributo ID de otro elemento.
  - **IDREFS**: múltiples IDs de otros elementos separados por espacios.

21

DTD (20)

- **Componentes del DTD: Atributo**

- **tipo_atributo**:

Ejemplo:

```
<!ELEMENT empleado (nombre, apellido) >
<!ATTLIST empleado
          idEmpleado  ID  #REQUIRED
          idJefe  IDREF  #IMPLIED >
```

```
<personal>
   <empleado idEmpleado="e_001"> … </empleado>
   <empleado idEmpleado="e_006"  idJefe="e_002">
   …
   </empleado>
</personal>
```

22

DTD (21)

- **Componentes del DTD: Atributo**

- **tipo_atributo**:
  - **NMTOKEN**: nombre XML válido, es decir cualquier nombre sin espacios en su interior.
  - **NMTOKENS**: lista de nombres XML válidos separados entre sí por espacios.

```
<!ATTLIST  rio  pais  NMTOKEN  #REQUIRED >
```

```
<rio pais="EEUU">
   <nombre>Mississippi</nombre>
</rio>
```

```
<rio pais="Estados Unidos">
   <nombre>Mississippi</nombre>
</rio>
```

23

DTD (22)

- **Componentes del DTD: Atributo**

- **tipo_atributo**:
  - **ENTITY**: nombre de una entidad (debe estar declarada en el DTD).
  - **ENTITIES**: lista de nombres de entidades separadas por espacios (declaradas en el DTD).

```
<!ATTLIST operacion simbolos ENTITIES #REQUIRED>
<!ENTITY … >
<!ENTITY … >
```

  - **NOTATION**: nombre de una notación (debe estar declarada en el DTD).

24

DTD (23)

- **Componentes del DTD: Entidad**

Las entidades se utilizan para definir referencias a valores u objetos (ficheros, sitios web, imágenes, etc.), de manera que una vez definida una entidad podemos sustituir su valor por un nombre.

Existen 3 tipos de entidades:

- Entidades generales (internas o externas)
- Entidades parámetro (internas o externas)
- Entidades no procesadas (*unparsed*)

25

DTD (24)

- **Componentes del DTD: Entidad**

- **Entidades generales internas**:

  <!ENTITY nombre_entidad "valor_entidad">

  Son las más sencillas, permiten definir abreviaturas que serán sustituidas por su valor pasando a formar parte del XML y por tanto siendo procesadas.

  Una vez definida, podemos utilizar la entidad en el documento XML utilizando: **&nombre_entidad;**

```
<!ENTITY rsa "República Sudafricana">
```

```
<pais>
    <nombre>&rsa;</pais>
</pais>
```

26

DTD (25)

- **Componentes del DTD: Entidad**

- **Entidades generales externas**:

  <!ENTITY nombre_entidad privacidad url_archivo>

  Igual que las anteriores pero en este caso el valor se obtiene de un archivo de texto.

```
<!ELEMENT escritores (#PCDATA)>
<!ENTITY autores SYSTEM "autores.txt">
```

```
<escritores>&autores;</escritores>
```

27

DTD (26)

- **Componentes del DTD: Entidad**

- **Entidades no procesadas**:

  <!ENTITY entidad privacidad url NDATA notacion>

  Hacen referencia a entidades generales externas cuando el archivo no contiene texto sino datos binarios: imágenes, videos, ejecutables, etc.

  En estos casos el contenido del archivo no debe ser procesado como si fuera texto XML.

```
<!NOTATION JPG SYSTEM "image/jpeg">
<!ENTITY mediterraneo SYSTEM "mediterraneo.jpg" NDATA JPG>
<!ELEMENT mar…>
<!ATTLIST mar imagen ENTITY #IMPLIED>
```

```
<mar imagen="mediterraneo"> … </mar>
```

28

DTD (27)

- **Componentes del DTD: Entidad**

- **Entidades parámetro**:

  Sólo pueden utilizarse dentro del propio DTD y se utilizan para agrupar elementos del DTD que se repiten con cierta frecuencia.

  Para referenciarlos se utiliza **%** en lugar de &.

  - Entidades parámetro internas:

  <!ENTITY % nombre_entidad "valor_entidad">

  - Entidades parámetro externas:

  <!ENTITY % nombre_entidad privacidad url>

29

DTD (28)

- **Componentes del DTD: Entidad**

- **Entidades parámetro**:

Tras definir la entidad parámetro "dimensiones":

```
<!ENTITY % dimensiones
        "alto CDATA #IMPLIED ancho CDATA #IMPLIED">
```

La declaración:

```
<!ATTLIST objeto %dimensiones>
```

Es equivalente a:

```
<!ATTLIST objeto
                alto CDATA #IMPLIED
                ancho CDATA #IMPLIED>
```

30

DTD (29)

- **Componentes del DTD: Notación**

Las notaciones sirven para definir formatos de datos que no son XML, en muchas ocasiones para describir tipos MIME como image/jpg.

Se utilizan para indicar un tipo de atributo al que se le permite usar un valor que ha sido declarado como notación en el DTD.

La declaración de la notación sería:

```
<!NOTATION nombre_notación privacidad "identificador">
```

Y la del atributo que la utiliza:

```
<!ATTLIST nombre_elemento
        nombre_atributo NOTATION valor_defecto>
```

31

DTD (30)

- **Componentes del DTD: Notación**

```
<!NOTATION GIF SYSTEM "image/gif">
<!NOTATION JPG SYSTEM "image/jpeg">
<!NOTATION PNG SYSTEM "image/png">
<!ELEMENT mar…>
<!ATTLIST mar
            imagen ENTITY #IMPLIED
            formato_imagen NOTATION (GIF | JPG | PNG) #IMPLIED>
<!ENTITY mediterraneo SYSTEM "mediterraneo.jpg" NDATA JPG>
```

```
<mar imagen="mediterraneo" formato_imagen="JPG">
        <nombre>Mediterráneo</nombre>
</mar>
```

32

DTD (31)

- **Secciones condicionales**

Las secciones condicionales sólo se pueden ubicar en DTDs externos y permiten incluir (INCLUDE) o excluir (IGNORE) reglas en función de condiciones.

La utilización de secciones condicionales tiene sentido cuando son combinadas con referencias a entidades parámetro.

- Reglas ignoradas:
  <![IGNORE [….]]>
- Reglas incluidas:
  <![INCLUDE [….]]>

33

DTD (32)

- **Secciones condicionales**

Ejemplo: definir una entidad persona de manera que sus datos personales puedan ser básicos (nombre y edad) o ampliados (nombre, apellidos, edad, ciudad).

En primer lugar creamos el archivo "persona.dtd" con dos secciones: basicos y avanzados:

```
<![ %basicos; [
      <!ELEMENT persona (nombre, edad)>]]>
<![ %ampliados; [
      <!ELEMENT persona (nombre, apellido, edad, ciudad)>]]>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT apellido (#PCDATA)>
<!ELEMENT  edad (#PCDATA)>
<!ELEMENT ciudad (#PCDATA)>
```

34

DTD (33)

- **Secciones condicionales**

En el apartado de declaraciones de nuestro XML establecemos la sección que queremos utilizar asignando INCLUDE e IGNORE a cada una de las secciones.

Si queremos utilizar los datos básicos:

```
<xml version="1.0" encoding="UTF-8" standalone="no">
<!DOCTYPE persona SYSTEM "persona.dtd" [
        <!ENTITY % basicos "INCLUDE">
        <!ENTITY % ampliados "IGNORE">
]>
<persona>
        <nombre>Antonio</nombre>
        <edad>20</edad>
</persona>
```

DTD (34)

- **Secciones condicionales**

Y si queremos utilizar los datos avanzados sólo hay que cambiar INCLUDE e IGNORE a ampliados y básicos respectivamente:

```
<xml version="1.0" encoding="UTF-8" standalone="no">
<!DOCTYPE persona SYSTEM "persona.dtd" [
        <!ENTITY % basicos "IGNORE">
        <!ENTITY % ampliados "INCLUDE">
]>
<persona>
        <nombre>María</nombre>
        <apellido>Hernández</apellido>
        <edad>22</edad>
        <ciudad>Guadalajara</ciudad>
</persona>
```

36

DTD (35)

- **Generación automática de DTD**

Existen herramientas que permiten generar de forma automática un DTD a partir de un documento XML.

Hay que tener en cuenta que estos DTDs generados automáticamente suponen una gran ayuda y son un buen punto de partida, pero que será necesario refinarlos para obtener el DTD correcto.

La razón es que un DTD debe servir para validar varios documentos XML diferentes, y a partir de un caso particular es imposible determinar cada uno de los casos concretos con los que nos podemos encontrar.

Para que el DTD generado sea más preciso se recomienda utiliza un XML lo más completo posible en el que no se omita ningún componente opcional.

37

DTD (36)

- **Generación automática de DTD**

- Elementos/Atributos opcionales: a partir de un solo caso no se puede saber si un elemento o atributo es opcional, ya que si aparece en el documento será tomado como obligatorio y si no lo hace es imposible que se pueda determinar su posible existencia.
- Enumeraciones y valores predefinidos: a partir de un único valor de un atributo no se puede conocer el resto de posibles valores ni si un valor es el valor por defecto, de manera que estos atributos se mostrarán simplemente como #CDATA.
- Identificadores: no se puede saber que un atributo es un identificador, se mostrarán como #CDATA.

38

DTD (37)

- **Limitaciones de los DTD**

1) Un DTD no es un documento XML y no se puede verificar si está bien formado.
2) No se pueden fijar restricciones sobre los valores de elementos y atributos, como tipo de dato, tamaño…
3) No soportan espacios de nombres.
4) Sólo se pueden enumerar los valores de los atributos, no de los elementos.
5) Sólo se pueden establecer valores por defecto para los atributos, no para los elementos.
6) El control de la cardinalidad (número de repeticiones) de los elementos es limitado.

39