UNIDAD 4 DEFINICIÓN DE ESQUEMAS Y VOCABULARIOS EN XML
(Parte 1 DTD)

1. INTRODUCCIÓN

En la Unidad 1 vimos las reglas para construir un documento XML de forma correcta, aunque os he preparado un Anexo I, donde vienen las principales características de XML. Es decir construimos documentos XML bien formados, estos documentos deben:

- Seguir la sintaxis básica de XML
- Contener cualquier combinación correcta de elementos y atributos

Pero igual que para construir una frase en castellano de forma correcta no sólo hace falta escribirla sin faltas de ortografía, un documento XML correcto bien formado, no es necesariamente válido.

Pensemos en la siguiente situación, imaginemos que estamos construyendo una aplicación para guardar datos sobre videojuegos y ésta alcanza una gran popularidad, con lo que otras personas quieren realizar contribuciones. ¿Cómo garantizar que todas estas contribuciones tienen todos los elementos necesarios? ¿Cómo saber que están utilizando el mismo vocabulario que tú?

En esta unidad veremos cómo crear las descripciones de validez de documentos y cómo validar los documentos usando estas descripciones. Un documento XML válido debe:

- Estar bien formado
- Referenciar a un DTD (Document Type Definition) o esquema XML
- Utilizar la gramática definida en el DTD o esquema XML

2. TÉCNICAS DE VALIDACIÓN

Las técnicas de validación que vamos a ver en esta unidad son DTD y esquemas XML. DTD y los esquemas XML son implementaciones competitivas.

- Una **definición de tipo de documento** o **DTD** (D*ocument Type Definition*) es una descripción de estructura y sintaxis de un documento XML. Su función básica es la descripción del formato de datos, para usar un formato común y mantener la consistencia entre todos los documentos que utilicen la misma DTD. Es un estándar establecido, pero tiene algunas limitaciones.
- Un **esquema XML XML Schema** se utiliza para describir la estructura y las restricciones de los contenidos de los documentos XML de una forma muy precisa, más allá de las normas sintácticas impuestas por el propio lenguaje XML. Es mucho más potente que el DTD.

Las DTDs y los Esquemas XML son usados por los analizadores sintácticos o parsers para comprobar si un documento XML es válido.

También existen otras técnicas de validación:

- **Relax NG**, es una notación más sencilla de utilizar que las anteriores. No tiene tantas posibilidades como XML Schema, pero se está utilizando mucho.
- **Schematro**n, permite establecer reglas que facilitan las relaciones que han de cumplir los datos en un documento XML.

3. DTD

3.1. Introducción al DTD

Las DTD se idearon para definir reglas de validación para SGML, de donde deriva el XML. Antes de empezar veamos un sencillo ejemplo el uso de DTD con documentos XML.

Creemos un documento XML sencillo:

```
<?xml version="1.0" encoding="UTF-8" ?>
<videojuego>
      <titulo>Resident Evil</titulo>
</videojuego>
```

La raíz del documento es el elemento videojuego. Dentro de él sólo hay otro elemento llamado título, cuyo contenido es el título del videojuego.

Usando DTD el documento se podría describir de la siguiente manera:

```
<!DOCTYPE videojuego [
       <!ELEMENT videojuego(titulo)>
       <!ELEMENT titulo(#PCDATA)>
]>
```

El DTD empieza con DOCTYPE, a continuación la raíz del documento. Dentro del documento, según la definición del DTD, debe de haber dos elementos:

- El primero de ellos definido con ELEMENT se llama videojuego y debe contener un elemento llamado título.
- Ahora miramos la defínción del elemento título, éste puede contener texto de cualquier tipo (#PCDATA)

Podemos incluir la definición del documento dentro del mismo documento, aunque no es la manera más recomendable, de esta manera:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE videojuego [
       <!ELEMENT videojuego (titulo)>
       <!ELEMENT titulo (#PCDATA)>
]>
<videojuego>
      <titulo>Resident Evil</titulo>
</videojuego>
```

Para validar un documento podríamos hacer uso de los navegadores web, pero en caso de encontrar un error no nos darán demasiada información. Así, que utilizaremos la página web de validación http://validator.w3.org.

3.2. Estructura de los DTD

Veamos cómo crear una DTD, así que vamos a ver las partes que componen una DTD:

DOCTYPE

El elemento principal de una DTD es su declaración, DOCTYPE declara la raíz del elemento árbol.

ELEMENT

Describe los elementos y sus relaciones con otros elementos o regla. En el ejemplo anterior el primer elemento se llama videojuego y la regla es título. A su vez, existe un elemento llamado titulo cuya regla es #PCDATA.

La DTD determina que el elemento videojuego debe contener, a su vez, otro elemento y sólo ese, llamado

título. La regla asociada a titulo determina que sólo puede contener texto.

La sintaxis de la etiqueta !ELEMENT es la siguiente:

```
<!ELEMENT nombre tipo_o_regla>
```

Reglas o tipos de contenidos en los elementos

ANY: El elemento podrá contener texto, otros elementos o atributos, en cualquier cantidad y orden. Incluso podría estar vacío. El uso de ANY debe ser muy cauteloso.

EMPTY: Obliga a que el elemento esté vacío, es decir, que no contenga ni otros elementos ni texto. Aunque los elementos vacíos si que pueden contener atributos.

Contenido concreto: para indicar que dentro de un elemento tiene que aparecer un contenido concreto tenemos que ponerlo entre paréntesis.

#PCDATA: Se pone para analizar datos de caracteres y significa que el elemento puede contener texto. No puede contener símbolos como < o >. En su lugar hay que utilizar entidades.

Grupos:

Veamos un ejemplo de un documento XML:

```
<pelicula>
    <titulo>Star Wars</titulo>
    <reparto>
        <interprete>Harrison Ford</interprete>
        <interprete>Mark Hamill</interprete>
        <interprete>Carrie Fisher</interprete>
   </reparto>
</pelicula>
```

¿Qué DTD usamos para describir el documento anterior?

```
<!ELEMENT pelicula (titulo, reparto) >
<!ELEMENT titulo (#PCDATA)>
<!ELEMENT reparto (interprete, interprete, interprete | interpreta)>
<!ELEMENT interprete (#PCDATA)>
```

En una DTD, los elementos a los que se hace referencia en las reglas pueden ser agrupados de diferentes formas. La siguiente tabla resume qué características podemos emplear para ello:

Carácter	Significado
()	Conjunto de elementos
,	Separador de elementos. El orden debe respetarse
|	Separador de alternativas: uno u otro

Cantidad o cardinalidad

Los caracteres +, * e ? son indicadores de ocurrencia. Indican qué elementos se pueden repetir y cómo dentro de la lista de elementos hijos.

Veamos en la siguiente tabla cuál es su significado:

Carácter	Significado
*	Un elemento seguido por un * puede aparecer cero o varias veces dentro del elemento en el cual fue definido.
+	Un elemento seguido por un + puede aparecer una o varias veces dentro del elemento en el cual fue definido.
?	Un elemento seguido por un ? puede aparecer una vez o no aparecer dentro del elemento en el cual fue definido.

Ejemplo:

```
<!ELEMENT pelicula (titulo, reparto?) >
<!ELEMENT titulo (#PCDATA)>
<!ELEMENT reparto (interprete+)>
<!ELEMENT interprete (#PCDATA)>
```

ATTLIST

Describe los atributos que tienen los elementos. Los atributos permiten añadir información a un elemento. Un atributo no puede constar de más atributos y cada atributo sólo puede aparecer una vez en cada elemento. El patrón que se sigue para los atributos es el siguiente:

```
<!ATTLIST elemento atributo tipo presencia valor_predeterminado>
```

Donde:

- `elemento`, el nombre del elemento que podrá utilizar dicho atributo.
- `atributo`, es el nombre del atributo.
- `tipo`, es el tipo de valores que podemos asignar al atributo.
- `presencia`, indica las características de los valores que puede tomar el atributo, si es obligatorio, si hay valor por defecto,…
- `valor_predeterminado`, permite especificar un valor que el atributo tomará si no especifica otro explicitamente en el documento XML.

Veamos un ejemplo:

```
<videojuego titulo=”Resident Evil” dificultad=”Avanzado”/>
```

Un solo elemento, con dos atributos, el nombre del videojuego y su dificultad. El DTD de este elemento, sin tener en cuenta sus atributos sería:

```
<!ELEMENT videojuego EMPTY>
<!ATTLIST videojuego titulo CDATA “ “>
<!ATTLIST videojuego dificultad CDATA “ “>
```

Los tipos de atributos pueden tomar cualquiera de los siguientes valores:

Tipo	Significado
CDATA	Para atributos de cadena de caracteres. A diferencia que con #PCDATA su contenido no es procesado, por lo que puede contener cualquier valor, incluidos elementos como <, >

ID	Para identificador. Un identificador es un nombre que es único dentro del documento. Para cada elemento sólo puede indicarse un atributo como ID. Además sólo pueden indicar #IMPLIED o #REQUIRED en el apartado de presencia
IDREF	Debe ser el valor de un ID usado en otra parte del mismo documento. Se utiliza para crear vínculos dentro de un documento.
IDREFS	Es una lista de IDREF separados por espacios.
NMTOKEN	El valor del atributo será un texto que ha de cumplir unas reglas más estrictas. Cumplira las reglas para los nombres de elementos de XML.
NMTOKENS	El atributo puede contener varios valores de tipo NMTOKEN separados por comas.
(*en1*|*en2*|..)	Lista de tipos enumerados
ENTITY	El valor es una entidad, de la cual se indica el nombre.
ENTITIES	El valor del atributo será una lista de entidades separadas por espacio.

La presencia que los atributos pueden tomar son las siguientes:

Carácter	**Significado**
#IMPLIED	Es opcional asignar un valor al atributo. Indica que dicho atributo podría quedarse sin valor.
#REQUIRED	Es obligatorio asignar un valor al atributo.
#FIXED valor	El valor del atributo debe ser el indicado, y no es posible cambiarlo. No se usa demasiado.
Valor_predeterminado	El valor asignado es el predeterminado

Ejemplos:

Ejemplo 1: En este ejemplo, el elemento cuadrado se define com un elemento vacío con el atributo anchura de tipo CDATA. Si la anchura no es especificada, tiene un valo por defecto de 0.

```
<!ELEMENT cuadrado EMPTY>
<!ATTLIST cuadrado anchura CDATA "0">
<cuadrado anchura="100" />
```

Ejemplo 2:

```
<!ATTLIST persona numero CDATA #REQUIRED>
```

Con el anterior DTD un documento XML válido sería:

```
<persona numero="5677" />
```

En cambio, lo siguiente sería un documento XML no válido

```
<persona />
```

Ejemplo 3:

```
<!ATTLIST contacto fax CDATA #IMPLIED>
```

Con el anterior DTD un documento XML válido sería:

```
<contacto fax="555-667788" />
```

o el siguiente:

```
<contacto />
```

Ejemplo 4:

```
<!ATTLIST cliente empresa CDATA #FIXED "Microsoft">
```

Con el anterior DTD un documento XML válido sería:

```
<cliente empresa="Microsoft" />
```

En cambio, lo siguiente sería un documento XML no válido XML:

```
<cliente empresa="W3Schools" />
```

Ejemplo 5:

```
<!ATTLIST pago tipo (efectivo|tarjeta) "efectivo">
```

Con el anterior DTD un documento XML válido sería:

```
<pago tipo="tarjeta" />
```

o

```
<pago tipo="efectivo" />
```

Ejemplo 6: Declaración de varios atributos en la misma etiqueta.

```
<!ATTLIST persona nacionalidad CDATA "Española"
                  sexo (Hombre | Mujer) #IMPLIED
                  id ID #REQUIRED>
```

ENTITIES

Cuando incluimos dentro del texto de un nodo una entidad, el analizador la sustituye por el valor con el que se corresponda. Todas las entidades están delimitadas por el ampersand (&) y el punto y coma (;).

Las entidades predeterminadas:

Carácter	Alias
&	&amp;
<	&lt;
>	&gt;
'	&apos;
"	&quot;

Pero, nosotros podemos crear nuestras propias entidades. Gracias a las entidades podemos realizar algunas operaciones que pueden sernos de gran utilidad, como:

1. La inserción de caracteres especiales
2. La simplificación de textos largos y repetitivos

Caracteres especiales

Estas entidades se declaran del mismo modo que los elementos o los atributos:

```
<!ENTITY nombre sustitución>
```

Suponer que queremos almacenar el precio en euros de nuestros videojuegos. De entre todas las opciones disponibles para incluir el símbolo del euro, ésta es una de las posibles:

```
<precio>19,95 &#x20ac;</precio>
```

Esta secuencia de números es difícil de recordar, podemos simplificar un poco la tarea definiendo una

entidad, de la siguiente forma:

```
<!ENTITY euro "&#x20ac;">
```

A partir de ese momento, podemos incluir el símbolo del euro así:

```
<precio>19,95 &euro;</precio>
```

Textos que se repiten

Podemos utilizar esta técnica para textos que se repiten. Por ejemplo, si queremos asignar a cada videojuego una frase de recomendación, como "Obra maestra" o "Muy flojo". Creamos las entidades correspondientes:

```
<!ENTITY om "Obra maestra">
<!ENTITY mf "Muy flojo">
```

Un documento que pueda utilizar esas entidades podría ser parecido al siguiente:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE videojuego [
     <!ELEMENT videojuego (titulo, opinion)>
     <!ELEMENT titulo (#PCDATA)>
     <!ELEMENT opinion (#PCDATA)>
     <!ENTITY om "Obra maestra">
     <!ENTITY mf "Muy flojo">
]>
<videojuego>
    <titulo>Resident Evil</titulo>
    <opinion>&om;</opinion>
</videojuego>
```

3.3. Enlanzando DTD con documentos XML

Los DTD se pueden incluir en un documento XML de dos maneras:

- Podemos incluirlo dentro del propio documento, especificando las reglas DTD al principio del documento.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE videojuego [
...
]>
<videojuego>
...
</videojuego>
```

- Especificando reglas DTD en un fichero separado privado

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE videojuego SYSTEM "videojuego.dtd">
<videojuego>
...
```

```
</videojuego>
```

3.4. Limitaciones de DTD

Encontramos las siguientes limitaciones a los DTD:

- Es complejo debido a la larga cantidad de tags.
- No proporciona detalles del tipo de datos.
- No permite insertar nuevos tags.