Unidad 3

DTDs

Tabla de contenidos

1 Introducción

En otros temas se ha visto como se creaban documentos XML bien formados, siguiendo las reglas básicas de XML (nombres de etiquetas y atributos, anidamientos, etc.)

En este tema vamos a dar un paso mas y ver como se pueden declarar vocabularios y esquemas utilizando DTDs. Al definir un vocabulario o esquema mediante DTDs indicamos la estructura que debe cumplir el documento, además de su mera validez como un documento bien formado. Se puede indicar lo siguiente:

- Qué etiquetas se permiten y qué etiquetas no.
- Qué contenido puede tener un elemento (otros elementos, texto o mezclados)
- Qué atributos tienen los elementos
- Restricciones sobre los contenidos de los atributos.

2 Definición de DTD

Para definir un vocabulario o dialecto XML, el mecanismo original es a base de los llamados *Document Type Definitions* (Definición de Tipo de Documento) o DTDs.

Un DTD es un lenguaje específico que indica la estructura y vocabulario de una clase de documentos XML. Es importante destacar que los DTDs indican sólo la estructura y el vocabulario y no intentan controlar los contenidos de los documentos.

3 Asociación de DTD

Para comprobar que un documento XML determinado es válido, o sea que cumple un determinado vocabulario o dialecto, debe existir alguna manera de indicar, para ese documento XML, que DTD es el que se le aplica. Esto se puede hacer de varias maneras: Interno, externo o público.

Un DTD es interno cuando se declara *dentro* del mismo documento XML que se pretende validar. Esto presenta la ventaja de que no es necesario acudir a algún archivo separado que se puede perder o mover. La desventaja es que el DTD no se puede compartir entre varios documentos que usen el mismo idioma o vocabulario.

Un DTD externo, en cambio, se declara dentro de un archivo que sólo contiene DTD. Desde el archivo XML que se pretende validar se indica que archivo se quiere utilizar como DTD del documento. Esto se hace con la declaración:

```
<!DOCTYPE raiz SYSTEM "url">
```

donde `DOCTYPE` y `SYSTEM` son palabras clave y se escriben tal cual, `raiz` es el nombre del elemento raíz del documento y `url` la URL donde se puede localizar el archivo con el DTD. **La declaración debe aparecer después del prólogo XML y antes del elemento raíz**. Por ejemplo:

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE direccion SYSTEM “direccion.dtd”>
<dirección>
  <calle>Ajo Porro, 23</calle>
  <localidad>Málaga</localidad>
  <provincia>Málaga</provincia>
</direccion>
```

La ventaja de tener el DTD en un archivo separado es que el mismo archivo puede ser “reutilizado” para muchos documentos XML que usen ese vocabulario. También, en caso de cambios, todos los documentos XML incorporarán dicho cambio la próxima vez que se valide. Como desventaja hay que decir que, al estar el DTD por

separado hay que llevar un control de él para que no se extravíe, mueva, etc., lo que impediría la validación de los documentos que dependan del él.

Por último, existe un tercer tipo de vinculación, mediante un ID público. Esto supone que hay un repositorio en el que se almacenan los DTDs y mediante un ID se recupera. Se usa sólo para DTDs muy utilizados y nosotros no vamos a usarlo.

4 Declaraciones

Un DTD consiste en *declaraciones*, que proporcionan información sobre:

- **Elementos**. ¿De qué elementos (etiquetas) dispone mi vocabulario? ¿Qué contenido se le permite a cada elemento?
- **Atributos.** ¿Qué atributos tienen los elementos de mi vocabulario? ¿Qué valores se permiten en dichos elementos?
- **Entidades**. ¿Qué entidades propias tendrá mi vocabulario XML?

4.1 Declaraciones de elementos

Un DTD debe incluir tantas declaraciones de elementos como tipos de elementos (etiquetas) haya en el vocabulario.

Para cada elemento habrá que indicar:

- El nombre del elemento (la etiqueta)
- Los posibles contenidos que se permiten dentro del elemento (otros elementos, texto, una combinación de ambos, etc.)

La sintaxis de la declaración de un elemento es:

```
<!ELEMENT nombre (contenido)>
```

donde:

- `ELEMENT` es la palabra clave que indica que la declaración es de un elemento,
- `nombre` es el nombre del elemento (etiqueta) que se está definiendo y,
- `contenido` es el contenido permitido en dicho elemento.

La parte más compleja es el contenido porque existen distintas posibilidades de contenido del elemento. Estas se examinan a continuación.

4.1.1 Elemento con contenido sólo texto

En este caso se indica utilizando la palabra clave `#PCDATA`. Por ejemplo:

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

declara un elemento llamado `em` que sólo puede contener texto. **No podrá contener otros elementos.**

Ejemplo correcto

```
<em>Este texto tiene emphasis</em>
```

Ejemplo incorrecto (contiene otro elemento)

```
<em><b>Texto en negrita</b></em>
```

4.1.2 Elemento que contiene sólo otro elemento

En este caso, se indica colocando el nombre del elemento. **Este elemento deberá declararse también utilizando otra declaración de elemento en el DTD.** Por ejemplo:

```
<!ELEMENT articulos (articulo)>
```

declara un elemento llamado `articulos` que sólo puede contener un elemento `artículo` (que habrá que declarar también). **No podrá contener ni texto, ni otros elementos ni más de un elemento artículo.**

Ejemplo correcto

```
<articulos><articulo>Articulo de deporte</articulo></articulos>
```

Ejemplo incorrecto (contiene texto)

```
<articulos>Articulo deportivo</articulos>
```

Ejemplo incorrecto (contiene otro elemento)

```
<articulos><item>Articulo deportivo</item></articulos>
```

Ejemplo incorrecto (contiene mas de un articulo)

```
<articulos><articulo>Articulo deportivo</articulo><articulo>Artículo
textil</articulo></articulos>
```

4.1.3 Repetición de elementos

Podemos indicar que un elemento se repite añadiendo un *sufijo* al nombre del elemento. Los posibles sufijos y su significado son:

- **? (cierre de interrogación)**. Indica que el elemento puede aparecer 0 ó 1 vez. Dicho de otra manera, el elemento es opcional y de aparecer sólo lo puede hacer una vez.
- *** (asterisco)**. Indica que el elemento puede aparecer 0 ó más veces (sin límite). Dicho de otra manera el elemento puede aparecer o no y de aparecer, puede hacerlo las veces que quiera.
- **+ (signo mas)**. Indica que el elemento debe aparecer al menos una vez pero puede aparecer más veces (sin límite).

Ejemplos utilizando ?:

```
<!ELEMENT articulos (articulo?)>
```

declara un elemento `articulos` que puede contener o no un sólo elemento `artículo`. No puede contener ningún otro elemento ni texto

Ejemplo correcto:

```
<articulos>
  <articulo>Articulo 1</articulo>
</articulos>
```

Otro ejemplo correcto:

```
<articulos</articulos>
```

Ejemplo incorrecto (contiene un elemento distinto a `articulo`):

```
<articulos>
  <item>Item 1</item>
```

```
</articulos>
```

Ejemplo incorrecto (contiene dos elementos `articulo`):

```
<articulos>
  <articulo>Articulo 1</articulo>
  <articulo>Articulo 2</articulo>
</articulos>
```

Ejemplo incorrecto (contiene texto):

```
<articulos>
Hola
</articulos>
```

Ejemplos utilizando *:

```
<!ELEMENT articulos (articulo*)>
```

declara un elemento `articulos` que puede contener 0, 1 ó más elementos `articulo`. No puede contener texto ni otros elementos.

Ejemplo correcto (vacio):

```
<articulos />
```

Ejemplo correcto (un articulo):

```
<articulos>
  <articulo>Articulo 1</articulo>
</articulos>
```

Ejemplo correcto (tres articulos):

```
<articulos>
  <articulo>Articulo 1</articulo>
  <articulo>Articulo 2</articulo>
  <articulo>Articulo 3</articulo>
</articulos>
```

Ejemplo incorrecto (contiene un elemento distinto a `articulo`):

```
<articulos>
  <item>Item 1</item>
</articulos>
```

Ejemplo incorrecto (contiene texto):

```
<articulos>
Hola
</articulos>
```

Ejemplos utilizando +:

```
<!ELEMENT articulos (articulo+)>
```

declara un elemento `articulos` que puede contener 1 ó más elementos `articulo`. No puede contener texto ni otros elementos, ni estar vacío.

Ejemplo correcto (un articulo):

```
<articulos>
  <articulo>Articulo 1</articulo>
</articulos>
```

Ejemplo correcto (tres articulos):

```
<articulos>
  <articulo>Articulo 1</articulo>
  <articulo>Articulo 2</articulo>
  <articulo>Articulo 3</articulo>
</articulos>
```

Ejemplo incorrecto (contiene un elemento distinto a `articulo`):

```
<articulos>
  <item>Item 1</item>
</articulos>
```

Ejemplo incorrecto (contiene texto):

```
<articulos>
Hola
</articulos>
```

Ejemplo incorrecto (vacio):

```
<articulos />
```

4.1.4 Secuencias de elementos

También podemos especificar que el contenido de un elemento es *una secuencia de elementos*. En este caso, el contenido sólo puede ser los elementos indicados **en el mismo orden en que aparecen en la declaración**. Esto se hace separando los distintos elementos con una coma (,).

Ejemplo:

```
<!ELEMENT ordenador (procesador, memoria)>
<!ELEMENT procesador (#PCDATA)>
<!ELEMENT memoria (#PCDATA)>
```

Este fragmento de DTD declara un elemento `ordenador` que debe contener un elemento `procesador` y otro `memoria`, en ese orden. `procesador` y `memoria` pueden contener texto.

Ejemplo correcto:

```
<ordenador>
```

```
  <procesador>Ryzen 7 4600X</procesador>
  <memoria>16GB</memoria>
</ordenador>
```

Ejemplo incorrecto (los elementos no están en el orden indicado)

```
<ordenador>
  <memoria>16GB</memoria>
  <procesador>Ryzen 7 4600X</procesador>
</ordenador>
```

Ejemplo incorrecto (elemento no permitido)

```
<ordenador>
  <procesador>Ryzen 7 4600X</procesador>
  <memoria>16GB</memoria>
  <disco>256GB</disco>
</ordenador>
```

Ejemplo incorrecto (falta un elemento)

```
<ordenador>
  <memoria>16GB</memoria>
</ordenador>
```

A los elementos, además, se le puede añadir un indicador de repetición, con lo que además de los elementos y el orden, se puede especificar si un elemento se puede repetir o si puede aparecer de forma opcional.

Ejemplo:

```
<!ELEMENT ordenador (procesador, memoria, disco*)>
<!ELEMENT procesador (#PCDATA)>
<!ELEMENT memoria (#PCDATA)>
<!ELEMENT disco (#PCDATA)>
```

este fragmento declara un elemento `ordenador` que contiene los elementos `procesador`, `memoria` y, opcionalmente, 1 ó más elementos `disco`.

Ejemplo correcto (1 disco):

```
<ordenador>
  <procesador>Intel i7 10117</procesador>
  <memoria>16GB</memoria>
  <disco>256GB</disco>
</ordenador>
```

Otro ejemplo correcto (2 discos):

```
<ordenador>
```

```
  <procesador>Intel i7 10117</procesador>
  <memoria>16GB</memoria>
  <disco>256GB</disco>
  <disco>1024GB</disco>
</ordenador>
```

Ejemplo correcto (sin disco):

```
<ordenador>
  <procesador>Intel i7 10117</procesador>
  <memoria>16GB</memoria>
</ordenador>
```

4.1.5 Elementos alternativos

La especificación de elementos alternativos permite que un elemento pueda contener un elemento, y solo uno, de entre varios posibles. Se realiza utilizando el carácter barra vertical "|".

Ejemplo:

```
<!ELEMENT producto (euros | dolares | yenes)>
<!ELEMENT euros (#PCDATA)>
<!ELEMENT dolares (#PCDATA)>
<!ELEMENT yenes (#PCDATA)>
```

Este fragmento de DTD declara un elemento `producto` que puede contener o bien un elemento `euros`, o bien un elemento `dolares` o bien un elemento `yenes`. No puede estar vacío, contener más de un elemento ni contener texto o un elemento que no sea uno de los tres.

Ejemplo correcto (con euros):

```
<producto>
  <euros>210,26</euros>
</producto>
```

Ejemplo correcto (con dolares):

```
<producto>
  <dolares>125,76</dolares>
</producto>
```

Ejemplo correcto (con yenes):

```
<producto>
  <yenes>56789</yenes>
</producto>
```

Ejemplo incorrecto (vacío):

```
<producto/>
```

Ejemplo incorrecto (elemento no en la lista):

```
<producto>
  <libras>65,87</libras>
</producto
```

Ejemplo incorrecto (mas de un elemento):

```
<producto>
  <euros>210,26</euros>
  <dolares>125,76</dolares>
</producto
```

4.1.6 Combinaciones

Las especificaciones que hemos visto en los apartados anteriores se pueden combinar para especificar cualquier tipo de contenido. Se utilizan paréntesis para agrupar.

Ejemplos:

```
<!ELEMENT complejo (a, (b | c))>
```

Declara un elemento `complejo` que contiene un elemento `a`, seguido de un elemento `b` o un elemento `c`.

```
<!ELEMENT complejo2 ((a,b) | (c,d))>
```

Declara un elemento `complejo2` que contiene, o bien un elemento `a` seguido de un elemento `b`, o bien un elemento `c` seguido de un elemento `d`.

```
<!ELEMENT complejo3 (c | (a,b+))>
```

Declara un elemento `complejo3` que contiene, o bien un elemento `c`, o bien un elemento `a`, seguido de uno o más elementos `b`.

4.1.7 Contenido mixto

Para especificar contenido mixto (texto y elementos), se hace de la siguiente manera:

```
<!ELEMENT parrafo (#PCDATA | negrita | cursiva)*>
<!ELEMENT negrita (#PCDATA)>
<!ELEMENT cursiva (#PCDATA)>
```

Declara un elemento `parrafo` que puede contener texto, elementos `negrita` y elementos `cursiva`, en cualquier cantidad y orden. Los elementos `negrita` y `cursiva` sólo pueden contener texto.

Nótese que para declarar contenido mixto se debe hacer como en el ejemplo. `#PCDATA` debe aparecer primero junto con una lista de los elementos hijos permitidos, separados por `|`. La repetición se debe indicar con `*` obligatoriamente. Dicho de otra forma, lo único que puede variar es el nombre del elemento que se declara y el nombre y número de los elementos hijos. El resto debe escribirse tal y como se ve.

Ejemplo correcto:

```
<parrafo>
En un lugar de la <negrita>Mancha</negrita> de cuyo nombre no
<cursiva>quiero acordarme</cursiva>
</parrafo>
```

4.1.8 Elementos vacíos

Para declarar un elemento que no puede tener contenido se utiliza la palabra clave EMPTY.

Ejemplo:

```
<!ELEMENT br EMPTY>
```

Esto declara un elemento llamado br que no tiene contenido. Si se incluye cualquier contenido es un error.

Ejemplo:

```
<br/>
```

Otro ejemplo:

```
<br></br>
```

Ejemplo incorrecto:

```
<br>Hola</br>
```

Otro ejemplo incorrecto:

```
<br><item>Hola</item></br>
```

4.1.9 Contenido libre

Para declarar un elemento que puede contener cualquier contenido se utiliza la palabra clave ANY.

```
<!ELEMENT misc ANY>
```

Declara un elemento misc que puede contener cualquier combinación de elementos y texto.

IMPORTANTE: Los elementos que se utilicen dentro de ANY deben haber sido declarados en el DTD. Si no se han declarado es un error.

4.2 Declaración de atributos

Para declarar los atributos de un elemento se utiliza la declaración `ATTLIST`, que es de la forma:

```
<!ATTLIST elemento atributos>
```

donde `ATTLIST` es una palabra clave, `elemento` es el elemento al que pertenecen los atributos a declarar y `atributos` es la lista de atributos del elemento (se pueden declarar más de uno en una sola declaración `ATTLIST`.

IMPORTANTE: Es obligatorio declarar los atributos de un elemento DESPUES de haber declarado el elemento en el DTD y sólo se puede utilizar una declaración de atributos por elemento.

Los atributos se declaran con tres indicaciones:

- nombre. Nombre del atributo
- tipo. Tipo del atributo.
- valor. Especificaciones sobre el valor del atributo.

El nombre del atributo debe cumplir las reglas de nombrado XML y ser único dentro de los atributos del elemento que se está declarando. Dicho en otras palabras, dentro de un elemento no puede haber atributos con el mismo nombre pero si en distintos elementos.

El tipo puede ser uno de: CDATA, NMTOKEN, NMTOKENS, ENTITY, ENTITIES, ID, IDREF, IDREFS, NOTATION o una enumeración. De todos ellos sólo vamos a describir ID, IDREF, CDATA y enumeración que son los más utilizados. El resto se utilizan en circunstancias muy específicas que se van a dar poco.

El tipo ID indica que el atributo es un identificador. Su contenido debe seguir las mismas reglas que un nombre de etiqueta o atributo (sólo letras, números, guión o subrayado, sin comenzar por número), sólo puede haber un elemento de tipo ID por documento y cada valor debe ser único dentro de un mismo documento, es decir, no puede haber dos ids repetidos en un mismo documento XML.

El tipo IDREF indica que el atributo es una referencia a un ID. El contenido debe seguir las mismas reglas que para el ID con la excepción de que los IDREFs se pueden repetir, puede haber más de uno en un mismo elemento y que el valor debe coincidir con el dado a algún ID del documento.

El tipo CDATA indica que el contenido del atributo puede ser cualquier cadena de caracteres, sin limitaciones excepto las que pone el lenguaje XML.

El tipo enumeracion permite especificar una lista de posibles valores. Cualquier valor que se asigne al atributo deberá ser obligatoriamente uno de los especificados. Se indica poniendo una lista con los valores separadas por la barra vertical | y encerrada entre paréntesis. Los valores sólo podrán contener los caracteres válidos para nombres de XML (letras, números, subrayado (_) y guión (-). No podrán contener espacios o carácteres no permitidos en nombres XML.

Las especificaciones de valor indican restricciones sobre el valor del atributo. Puede ser una de:

- Un valor entre comillas (simples o dobles). El atributo es opcional. Puede aparecer en el elemento o no. Si no aparece en el documento el procesador supone que SI ha aparecido y que tiene el valor indicado.
- #REQUIRED. El atributo es obligatorio y toma el valor que se de en el documento. Los atributos de tipo ID deben ser obligatorios.
- #IMPLIED. El atributo es opcional y si no aparece el procesador no le da un valor por defecto.
- #FIXED "valor". El atributo es obligatorio y DEBE llevar el valor indicado en la declaración.

Ejemplos:

```
<!ELEMENT body ANY>
<!ATTLIST body
        background-color CDATA "#FFFFFF"
        href CDATA #REQUIRED
        version CDATA #IMPLIED
        padre CDATA #FIXED "html"
        alineacion (izquierda | centro | derecha) "derecha"
>
```

Declara un elemento llamado `body` que contiene los siguientes atributos:

- `background-color`, de tipo texto, opcional. Si no se da en el documento su valor es `#FFFFFF`.
- `href`, de tipo texto, obligatorio. Si no se da se produce un error.
- `version`, de tipo texto, opcional. Si no se da no se le da ningún valor.
- `padre`, de tipo texto, obligatorio y hay que darle el valor `html`.
- `alineacion`. Puede valer o bien `izquierda`, o bien `centro` o bien `derecha`. Cualquier otro valor da error. Es opcional y si no se proporciona su valor es `derecha`.

4.3 Declaración de entidades

XML declara por defecto 5 entidades: `lt`, `gt`, `amp`, `quot` y `apos`. Las entidades se referencian en un documento escribiendo `&nombre_entidad;`

En un DTD se pueden definir entidades que más tarde se pueden referenciar en el documento. Para ello se utiliza la declaración `ENTITY`. Ésta tiene la forma:

```
<!ENTITY nombre contenido>
```

donde `nombre` es el nombre de la entidad a declarar. Un nombre de entidad debe ser único dentro de un DTD, esto es, no se pueden declarar dos entidades con el mismo nombre. `contenido` es el contenido de la entidad. Cuando se referencie la entidad en el documento XML utilizando `&entidad;`, esta referencia será sustituida por el contenido que aquí se indique.

Hay 4 tipos de entidades:

4.3.1 Entidades generales

En estas entidades se proporciona directamente el texto a sustituir, en la forma:

```
<!ENTITY nombre "texto>
```

Por ejemplo, si declaramos en el DTD:

```
<!ENTITY onu "Organización de las Naciones Unidas">
```

y tenemos un documento tal que:

```
<parrafo>La &onu; es la organización que vela por el cumplimiento de la paz mundial y la concordia entre las naciones. La &onu; fue fundada en 1945. España es miembro de la &onu; desde 1955.</parrafo>
```

se cambiaría por:

```
<parrafo>La Organización de las Naciones Unidas es la organización que vela por el cumplimiento de la paz mundial y la concordia entre las naciones. La Organización de las Naciones Unidas fue fundada en 1945. España es miembro de la Organización de las Naciones Unidas desde 1955.</parrafo>
```

El contenido de la entidad debe ser bien formado. Las entidades generales no pueden referenciarse desde dentro del DTD.

4.3.2 Entidades externas

En estas entidades el contenido está en un archivo externo al documento donde se está declarando el DTD. Utilizan la forma:

```
<!ENTITY nombre SYSTEM "url">
```

donde `ENTITY` y `SYSTEM` son palabras clave, `nombre` es el nombre de la entidad que se está declarando y `url` es la URL al archivo cuyo contenido se desea asociar al nombre.

Cuando se referencie la entidad en el documento XML (no se puede hacer en el DTD), se sustituirá por el contenido que tuviera el archivo.

Por ejemplo

```
<!ENTITY cabecera SYSTEM "cabecera.xml">
```

declara la entidad `cabecera` y le asocia el contenido del archivo "cabecera.xml".

Cuando se use `cabecera` en el documento, su lugar será sustituido por el contenido del archivo.

Al igual que con las generales, el contenido del archivo debe ser XML bien formado.

4.3.3 Entidades parámetricas

Las entidades paramétricas nos permiten utilizar entidades *dentro* del DTD. Se declaran utilizando la forma:

```
<!ENTITY % nombre "valor">
```

donde `ENTITY` es palabra clave, `nombre` es el nombre de la entidad a definir, que debe ser única dentro del DTD y `valor` es el valor que va a sustituir a las referencias a la entidad.

Las entidades paramétricas se utilizan para evitar repeticiones en las declaraciones de los DTDs, sobre todo para vocabularios grandes.

Por ejemplo, en HTML, todos los elementos pueden llevar atributos id y class. Podríamos hacerlo de la siguiente manera (suponemos que los elementos ya se han declarado):

```
<!ATTLIST html id ID #IMPLIED class ID #IMPLIED>
<!ATTLIST head id ID #IMPLIED class ID #IMPLIED>
<!ATTLIST body id ID #IMPLIED class ID #IMPLIED>
<!ATTLIST div id ID #IMPLIED class ID #IMPLIED>
......
```

Como se puede ver, como **todos** los elementos tienen dichos atributos, hay que incluirlos en **todas** las declaraciones. Peor aún, si se incluye un nuevo atributo, por ejemplo style, habría que modificar **todas** las declaraciones ya existentes:

```
<!ATTLIST html id ID #IMPLIED class ID #IMPLIED style CDATA #IMPLIED>
<!ATTLIST head id ID #IMPLIED class ID #IMPLIED style CDATA #IMPLIED>
<!ATTLIST body id ID #IMPLIED class ID #IMPLIED style CDATA #IMPLIED>
<!ATTLIST div id ID #IMPLIED class ID #IMPLIED style CDATA #IMPLIED>
......
```

Si usamos entidades paramétricas, la tarea se automatiza grandemente. En lugar de lo que hicimos en primer lugar, podríamos haber hecho:

```
<!ENTITY % common-attrs " id ID #IMPLIED class ID #IMPLIED">
<!ATTLIST html %common-attrs;>
<!ATTLIST head %common-attrs;>
<!ATTLIST body %common-attrs;>
<!ATTLIST div %common-attrs;>
....
```

Si luego hubiera que incluir el atributo style, se podría hacer simplemente con un sólo cambio:

```
<!ENTITY % common-attrs " id ID #IMPLIED class ID #IMPLIED style CDATA #IMPLIED">
<!ATTLIST html %common-attrs;>
<!ATTLIST head %common-attrs;>
<!ATTLIST body %common-attrs;>
<!ATTLIST div %common-attrs;>
```

. . . .

5 Resumen

En este documento hemos intentado dar una visión rápida pero a la vez rigurosa y con ejemplos del lenguaje DTD para definición de vocabularios en XML.

Hemos aprendido como declarar los contenidos de los elementos, atributos y entidades que componen un vocabulario.