Validación de documentos XML: **DTD (Documento de Definición de Tipos)**.

● Contenidos:

La DTD o documento de definición de tipos indica el formato que debe tener el documento XML (estructura y sintaxis).
Es decir, es una descripción de la estructura para un cierto tipo de 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.

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.

Nota: recordar cabecera documento XML:

```
<?xml version=”1.0” encoding=”UTF-8”?>
```

Si deseamos que el navegador valide el documento:

```
<?xml version="1.0" encoding ="UTF-8" standalone="yes"?>
```

1. Asociación de un DTD a documento XML.

La asociación puede realizarse de forma:

- Interna o Externa al documento XML.
- Pública (PUBLIC) o Privada (SYSTEM) al equipo que validará el documento XML.

Para asociar un documento XML a su validación siempre se incluirá en el documento XML:

<!DOCTYPE raíz … >

Donde "raíz": elemento raíz del documento XML.

▪ Declaración del DTD: definirá la forma de asociación del DTD al documento XML.

a) En función de la Ubicación:
   a. Interno: las reglas aparecen en el propio DTD.
   b. Externo: las reglas aparecen en un archivo .dtd independiente.
   c. Mixto: mezcla de las anteriores. Las reglas internas tienen prioridad sobre las externas.

   Se suele utilizar un documento externo (.dtd) por 2 razones:
   - Ahorro de espacio: si almacenamos las declaraciones dentro del documento, éste ocupará más 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.

b) En función del Carácter:
   a. Uso Privado (System): el DTD se almacena de forma local (propio equipo) y, por tanto, solo va a ser utilizado por nosotros mismos.
   b. 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).

▪ Formas de asociación: (no es posible una asociación pública interna)

- En el propio documento (Privado e Interno).
- En documento Privado Externo.
- En documento Público Externo.
- Mixta.

Los documentos DTD externos tendrán extensión .dtd

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.

a) **En el propio documento** (Interna y, lógicamente privada, ya que va en el documento XML):

<!DOCTYPE raíz [....códigoDTD...]>

Las reglas aparecen en el propio documento XML.
Reglas válidas únicamente para ese documento XML (por ello es la forma menos habitual). Su única (pero muy discutible) ventaja es que la validación está dentro del propio documento, por lo que siempre viajan juntas la validación y el contenido del mismo.

· [....códigoDTD...]: dentro de los símbolos [..] se especifican las instrucciones DTD.

Ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE persona [
        <!ELEMENT persona (nombre)>
        <!ELEMENT nombre (#PCDATA)>
]>
<persona>
        <nombre>Ana</nombre>
</persona>
```

b) **En un documento Externo Privado**:

<!DOCTYPE raíz SYSTEM "ruta_URL_al_DTD">

Las reglas aparecen en un archivo independiente, en el propio equipo.
La validación se crea en un documento DTD externo e independiente, aunque local. De modo que cuando un documento queremos que cumpla las reglas de ese, se debe indicar la ruta (sea relativa (en el mismo directorio en el que se encuentra el XML a validar) o absoluta, incluyendo el nombre del documento DTD) al mismo.
Sirve para validar diversos documentos XML, pero siempre en el ámbito de ese equipo validador.

En el documento DTD externo se introducen las reglas sin indicar cabecera, simplemente la estructura y sintaxis.

Ejemplo: Este fichero XML utilizará la DTD establecida en el fichero personas.dtd que se encuentra en el mismo directorio del equipo.

```
<?xml version=”1.0” encoding=”UTF-8”?>
<!DOCTYPE persona SYSTEM “personas.dtd”>
<persona>
    <nombre>Ana</nombre>
</persona>
```

El documento DTD externo tendría esta forma:

```
<!ELEMENT persona (nombre)>
    <!ELEMENT nombre (#PCDATA)>
```

También se puede añadir código DTD para ese documento concreto, añadiendo instrucciones de validación.

Ejemplo: Este fichero XML utilizará la DTD establecida en el fichero personas.dtd que se encuentra en el mismo directorio del equipo e incluirá reglas adicionales de validación impuestas en el propio XML.

```
<?xml version=”1.0” encoding=”UTF-8”?>
<!DOCTYPE persona SYSTEM “personas.dtd” [
    <!ELEMENT nombre (#PCDATA)>
] >
<persona>
    <nombre>Ana</nombre>
</persona>
```

c) En un documento Externo Público:

```
<!DOCTYPE raíz PUBLIC “nombreDTD” “DTD_URL”>
```

Las reglas aparecen en un archivo independiente, en equipo o contenedor público.

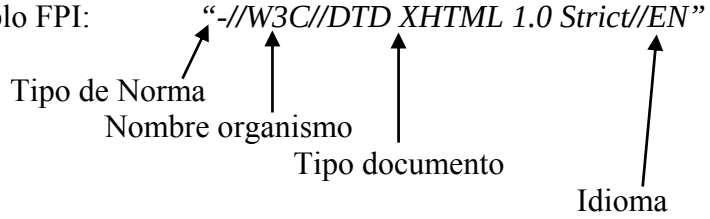
· “nombreDTD”: es el nombre público que se le da al DTD en cuestión (uso de FPI).
Si disponemos de un repositorio de DTDs públicos (como ocurre en entornos de trabajo como Oxygene por ejemplo) le cargaría sin ir a Internet.
· “DTD_URL”: si el “nombreDTD” no es reconocido se usa la dirección URL para descargarlo y utilizarlo.

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 FPI: *"-//W3C//DTD XHTML 1.0 Strict//EN"*



Ejemplo:

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

Este es el DOCTYPE para una página web escrita en XHTML 1.0 estricto, utilizada para validar miles de páginas web.

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"

2. Definiciones en un DTD.

En un código DTD (tanto externo como interno) se pueden definir 4 tipos de componentes:

- Los **elementos** que se pueden utilizar en un documento XML. En esta definición se indica además que pueden contener dichos elementos.
- Los **atributos** que pueden poseer los elementos. Además, incluso indicando sus posibles valores válidos.
- Las **entidades** que puede utilizar el documento XML.
- Las **notaciones**.
- **Secciones condicionales**.

A. **Elementos** o etiquetas:

En el DTD se pueden especificar los elementos que se utilizarán en el documento XML, su orden, tipo, etc.
El primer elemento será el nodo raíz, que deberá ser único, con una única ocurrencia (no se puede repetir).

<!ELEMENT nombre tipocontenido >

Donde:
· nombre: es el identificador que tendrá el elemento en el documento XML (hay que recordar que se distingue entre mayúsculas y minúsculas, y no se deben usar símbolos no permitidos).
· tipocontenido: indica el funcionamiento del elemento, relativo al contenido que puede tener.

○ Tipos de contenido en los elementos:

- **EMPTY**: el elemento no podrá tener contenido alguno (ni descendientes), es un elemento vacío. No obstante, los elementos vacíos pueden tener atributos si así se especifican en el DTD.

  Por ejemplo:
  <!ELEMENT línea EMPTY >
  (después se podrían vincular atributo/s al elemento línea)
  En el XML sería válido el uso de <línea></línea>
  o bien <línea />.

- **ANY**: permite cualquier contenido en el elemento, sin restricciones de ningún tipo (es decir, puede contener texto, otro tipo de datos y cualquier etiqueta), eliminando comprobaciones. Además, puede tener atributos.
  Puesto que un DTD se usa para restringir la escritura de un tipo de documentos XML, el uso de ANY no es recomendable.

  Por ejemplo:
  <!ELEMENT persona (nombre, apellidos)>
  <!ELEMENT nombre (#PCDATA)>
  <!ELEMENT apellidos ANY>

  En el documento XML sería válido:
  <persona>
  <nombre>Jorge</nombre>
  <apellidos>Sánchez Asenjo
  <nombre>Jorge</nombre>
  </apellidos>
  </persona>
  En este ejemplo el elemento "apellidos" puede contener cualquier cosa, incluido otro elemento.

- **(Contenido concreto)**: en los elementos se puede indicar que, dentro del mismo, tiene que aparecer un contenido concreto (pudiendo ser elementos, tipo de dato). Dicho contenido se indica entre paréntesis.
  Dicho contenido puede ser:

- **(#PCDATA)**: Datos. Cualquier conjunto de caracteres, ya sean textuales, numéricos u otro formato que no contenga marcas. El elemento podrá contener texto literal (tan largo como se desee).
  El texto de tipo #PCDATA tiene que cumplir las reglas de texto XML. Eso implica que no puede contener símbolos como prohibidos como < o >. En su lugar hay que utilizar entidades.

- **(lista_elementos)**: Descendientes. Indica que el elemento contendrá la lista de elementos indicada, la cual aparecerá en el mismo orden especificado en el DTD.
  Establece cuáles son los elementos descendientes del elemento que se está describiendo, y su orden.
  Se puede establecer:
  - Secuencia:
    - (elemento,elemento): indica el orden en que deben aparecer los elementos (,).
    - (elemento|elemento): indica que los elementos son excluyentes (|), es decir, aparecerá uno o el otro, no los dos a la vez.
  - Cardinalidad (?,*,+): indica el número de veces que puede aparecer un elemento o una secuencia de ellos.
    - ?: contenido opcional. Puede aparecer una sola vez o no aparecer. [0-1]
    - *: contenido opcional y repetible. Puede no aparecer, o aparecer varias veces. [0-N]
    - +: contenido obligatorio y repetible. Tiene que aparecer, e incluso aparecer varias veces. [1-N]

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

  Si un elemento tiene descendientes, después de enumerarlos hay que describirlos.

- **(#PCDATA, lista_elementos)**: Mixto. El contenido es una combinación de datos y elementos descendientes. (PCDATA deberá aparecer el primer elemento de la lista)

- **((lista_elementos)|(lista_elementos)**: Combinaciones de definiciones. Establece que puede aparecer una lista u otra lista.

Por ejemplo:
<!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)>

Por ejemplo, combinaciones de elementos:
<!ELEMENT coordenada ((longitud, latitud) | coordUniversal)>
<!ELEMENT longitud (#PCDATA)>
<!ELEMENT latitud (#PCDATA)>
<!ELEMENT coordUniversal (#PCDATA)>

En el documento XML sería válido:
<coordenada>
<longitud>234</longitud>
<latitud>-23</latitud>
</coordenada>

O también sería válido:
<coordenada>
<coordUniversal>1232332</coordUniversal>
</coordenada>

Por ejemplo, combinaciones de elementos y cardinalidad:
<!ELEMENT polígono ((coordX,coordY)+ | nombre)>
<!ELEMENT coordX (#PCDATA)>
<!ELEMENT coordY (#PCDATA)>
<!ELEMENT nombre (#PCDATA)>

En el documento XML sería válido: para un triángulo
<polígono>
<coordX>12</coordX>
<coordY>13</coordY>
<coordX>17</coordX>
<coordY>23</coordY>
<coordX>34</coordX>
<coordY>56</coordY>
</polígono>

B. **Atributos**:

Los atributos permiten añadir información a un elemento.
Un atributo:
- No puede constar de más atributos.
- Cada atributo sólo puede aparecer una vez en cada elemento.

```
<!ATTLIST elemento
        nombre_atributo tipo_atributo carácter valorPorDefecto
        nombre_atributo tipo_atributo carácter valorPorDefecto
        …
>
```

Donde:
· elemento: es el nombre del elemento que podrá utilizar el atributo.
· nombre_atributo: es el identificador del atributo que estamos declarando (y que debe de cumplir las reglas de identificadores de XML).
· tipo_atributo: es el tipo de valores que podemos asignar al atributo.
· carácter: indica las características de los valores que puede tomar el atributo: si es obligatorio, si hay valor por defecto, etc.
· valorPorDefecto: permite especificar un valor que el atributo tomará si no especifica otro explícitamente en el documento XML.

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

- Tipo de Atributo:
  - **CDATA**: Texto. Caracteres que no contienen etiquetas.
    A diferencia de #PCDATA, su contenido no es procesado, lo que significa que puede contener cualquier valor (incluidos símbolos prohibidos en los #PCDATA como <, >, &, etc.)

    <!ATTLIST precio moneda CDATA #REQUIRED>

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

    <!ATTLIST semáforo color (rojo | naranja | verde)>

  - **ID**: Sirve para especificar que el atributo contendrá un identificador de elemento, identificador que permite identificar, diferenciar, 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.
    Los atributos ID solo pueden indicar #IMPLIED o #REQUIRED en el apartado del valor por defecto.

  - **IDREF**: representa el valor de un atributo ID de otro elemento. (Contienen como valor el ID de otro elemento. Es una referencia a otro elemento).
    Para ser válido debe coincidir con el valor del atributo ID de otro elemento en el documento XML.
    Mediante el uso de ID e IDREF se pueden relacionar elementos.

- **IDREFS**: múltiples IDs de otros elementos separados por espacios. (Similar a IDREF, pero permite indicar varias referencias, que deben existir en el documento XML).

  <!ELEMENT persona (#PCDATA) >
  <!ATTLIST persona id ID #REQUIRED
  padres IDREFS #IMPLIED >
  En el documento XML:
  <persona id="p1">Pedro</persona>
  <persona id="p2">Marisa</persona>
  <persona id="p3" padres="p1 p2">Carmen</persona>

- **NMTOKEN**: el valor del atributo será un texto que cumple unas reglas estrictas: nombre XML válido, es decir cualquier nombre sin espacios en su interior ni símbolos reservados.
  (sólo puede contener letras, dígitos, punto [.], guión [-], subrayado [_] y dos puntos [:])

  <!ATTLIST rio pais NMTOKEN #REQUIRED >
  Sería válido:
  <rio pais="EEUU"/>
  No sería válido:
  <rio pais="Estados Unidos"/>

- **NMTOKENS**: el valor del atributo puede contener varios valores de tipo NMTOKEN separados por espacios, es decir, una lista de nombres XML válidos separados entre sí por espacios.

- **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).

o Carácter:

- **"valor"**: establece un valor por defecto para el atributo. Se debe introducir entre comillas.
  Se podría no utilizar el atributo en un elemento y entonces dicho atributo tomaría dicho valor.

  <!ATTLIST persona nacionalidad CDATA "Española">

En el documento XML:
<persona nacionalidad=”Francesa”>Vivi</persona>
<persona>Juan</persona>

A la primera persona se la ha indicado la nacionalidad. A la segunda no se le ha indicado nacionalidad, por lo que adoptará el valor por defecto, española.

- **#IMPLIED**: el atributo es opcional y no se le asigna ningún valor por defecto. Implica que dicho atributo puede quedarse sin valor, por lo que no tiene sentido asignar valor por defecto (si no se indica valor quedaría nulo).

  <!ATTLIST persona nacionalidad CDATA #IMPLIED>
  No es obligatorio especificar nacionalidad, puede quedar sin valor.

- **#REQUIRED**: el atributo es obligatorio, pero no se le asigna un valor por defecto. Se indica que siempre hay que dar valor al atributo. El uso de #REQUIRED no permite indicar un valor por defecto, al no poder dejarse sin especificar el atributo.

  <!ATTLIST persona nacionalidad CDATA #REQUIRED>
  El elemento persona deberá especificar obligatoriamente la nacionalidad.

- **#FIXED "valor"**: el atributo es obligatorio y su valor no puede ser otro más que el establecido. Mediante fixed se define un valor fijo para un atributo, que no se puede modificar.
  En la práctica este tipo de atributos no se usa demasiado.

  <!ATTLIST persona nacionalidad CDATA #FIXED “Española”>
  Todas las personas tendrán nacionalidad española, no pudiendo modificarse.

C. 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.
(Se usan como abreviaturas que aparecerán en el documento XML. La razón de su uso es facilitar la escritura de nombres repetitivos (nombres de la empresa, direcciones muy utilizadas, etc.).)

Existen 3 **tipos de entidades**:
- Entidades generales (internas o externas)
- Entidades parámetro (internas o externas)
- Entidades no procesadas (unparsed)

a) Entidades Generales Internas

<!ENTITY nombre_entidad "valor_entidad">

Para usar en un documento XML la entidad declarada:
&nombre_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.

Por ejemplo:

<!ENTITY rsa "República Sudafricana">
En el documento XML:
<pais>
<nombre>&rsa;</nombre>
</pais>

b) Entidades Generales Externas

<!ENTITY nombre_entidad privacidad “url_archivo”>

Para usar en un documento XML la entidad declarada:
&nombre_entidad;

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

Por ejemplo:

<!ELEMENT escritores (#PCDATA)>
<!ENTITY autores SYSTEM "autores.txt">
En el documento XML:
<escritores>&autores;</escritores>
Es la palabra SYSTEM la que indica que la entidad no es un texto, sino que es el contenido de un archivo.
El uso de la entidad &autores; en un documento XML provocará que en dicho documento se añada el contenido del archivo autores.txt (en la posición exacta en la que esté colocada la referencia a la entidad).

c) Entidades Parámetro

Sólo pueden utilizarse dentro del propio DTD (no en el documento XML) y se utilizan para agrupar elementos del DTD que se repiten con cierta frecuencia (a fin de ahorrar trabajo al crear el propio DTD).

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

• Entidades parámetro internas:

<!ENTITY % nombre_entidad "valor_entidad">

• Entidades parámetro externas:

<!ENTITY % nombre_entidad privacidad “url”>

Por ejemplo:

<!ENTITY %dimensiones
alto CDATA #IMPLIED
ancho CDATA #IMPLIED">
<!ATTLIST objeto %dimensiones>
Esta declaración del atributo será equivalente a:
<!ATTLIST objeto
alto CDATA #IMPLIED
ancho CDATA #IMPLIED>

Por ejemplo, para construir un DTD basado en otro DTD:

<!ENTITY % directorio SYSTEM “directorio.dtd” >
%directorio;
<!ELEMENT empresa (razónSocial, dirección) >
<!ELEMENT razónSocial (#PCDATA) >
De esta forma se construye un DTD con el contenido ya especificado en otro DTD. En el ejemplo las empresas constan de elementos razónSocial y de directorio. El elemento directorio no se define, sino que su descripción está especificada en directorio.dtd.

d) 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.

Por ejemplo:

<!NOTATION JPG SYSTEM "image/jpeg">
<!ENTITY mediterraneo SYSTEM "mediterraneo.jpg" NDATA JPG>
<!ELEMENT mar…>
<!ATTLIST mar imagen ENTITY #IMPLIED>
En el documento XML:
<mar imagen="mediterraneo"> … </mar>

e) Entidades Definidas en XML:
En XML están definidas las siguientes entidades, de forma implícita, por lo que no hay que declararlas en ningún DTD (todos los analizadores de XML estándar conocen estas entidades).

Entidad	Significado
&lt;	Símbolo de menor (<)
&gt;	Símbolo de mayor (>)
&amp;	Ampersand (&)
&apos;	Comilla simple (‘)
&quot;	Comilla doble (“)

Por ejemplo:

<autor>Leopoldo Alas 'Clarín'</autor>

El texto PCDATA del autor es Leopoldo Alas ‘Clarín’ (así se visualizará en el navegador).

f) Entidades para referencias a Caracteres Especiales:
Para indicar un carácter especial que no está contenido en nuestro teclado, conociendo su código en el juego de caracteres que utiliza el documento, se puede especificar con la sintaxis:

• Código del carácter en decimal: &#código;

• Código del carácter en hexadecimal: &#xnúmero;

D. Notaciones:

En la DTD de un documento XML, las notaciones se pueden utilizar para especificar el formato de entidades externas (datos no XML, como por ejemplo un archivo que contenga una imagen). Dichas entidades externas no las analizará un procesador XML, sino que serán tratadas por el programa que procese el documento.
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">

Las notaciones podrán ser usadas en Entidades o en Atributos:

La declaración de la entidad que la utiliza: (entidades no procesadas)

<!ENTITY entidad privacidad “url” NDATA notacion>

La declaración del atributo que la utiliza:
<!ATTLIST nombre_elemento nombre_atributo NOTATION valor_defecto>

Por ejemplo: Notación asociada a Entidad

En la DTD del siguiente documento XML, se indica que los elementos "fruta" que se escriban, tienen que incluir obligatoriamente el atributo foto, cuyo valor será una entidad y, para indicar el formato de dicha entidad, se usa la notación gif:

<!ELEMENT fruta EMPTY>
<!ATTLIST fruta foto ENTITY #REQUIRED>
<!ENTITY manzana SYSTEM "manzana.gif" NDATA gif>
<!ENTITY naranja SYSTEM "naranja.gif" NDATA gif>
<!NOTATION gif SYSTEM "image/gif">
Los valores de las entidades manzana o naranja se cargarán desde el propio equipo, haciendo referencia a los archivos "manzana.gif" y "naranja.gif".
La notación gif es una declaración del tipo MIME image/gif.

Por ejemplo: Notación asociada a un Atributo
En la DTD del siguiente documento XML, se indica que los elementos "documento" que se escriban, tienen que incluir obligatoriamente el atributo version, cuyo valor será una notación (h4 o h5):
<!ELEMENT documento (#PCDATA)>
<!ATTLIST documento version NOTATION (h4|h5) #REQUIRED>
<!NOTATION h5 PUBLIC "HTML 5">
<!NOTATION h4 PUBLIC "HTML 4.01">
El documento XML podría ser:
<documento version="h4"><!—Código del documento 1. --></documento>
<documento version="h5"><!—Código del documento 1. --></documento>

Por ejemplo: Notación asociada a un Atributo
<!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>
El documento XML podría ser:
<mar imagen="mediterraneo" formato_imagen="JPG">
<nombre>Mediterráneo</nombre>
</mar>

E. Secciones Condicionales:

En DTD externas se pueden definir las secciones IGNORE e INCLUDE, para ignorar o incluir declaraciones.
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. (El uso de las secciones condicionales suele estar ligado a entidades paramétricas).

Reglas ignoradas:	<![*IGNORE* [declaraciones]]>
por ejemplo:	<![%Nombre_Igno [declaraciones]]>
Reglas incluidas:	<![*INCLUDE* [declaraciones]]>
por ejemplo:	<![%Nombre_Incl [declaraciones]]>

Ejemplo:

En un archivo denominado "persona.dtd" se ha descrito:

```
<![ %datos_basicos; [
  <!ELEMENT persona (nombre, edad)>
]]>

<![ %datos_ampliados; [
  <!ELEMENT persona (nombre, apellidos, edad, ciudad)>
]]>

<!ELEMENT nombre (#PCDATA)>
<!ELEMENT apellidos (#PCDATA)>
<!ELEMENT edad (#PCDATA)>
<!ELEMENT ciudad (#PCDATA)>
```

El documento XML podría ser:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE persona SYSTEM "persona.dtd" [
  <!ENTITY % datos_basicos "INCLUDE">
  <!ENTITY % datos_ampliados "IGNORE">
]>

<persona>
  <nombre>Elsa</nombre>
  <edad>23</edad>
</persona>
```

Y este otro documento XML también sería válido:

```
<!DOCTYPE persona SYSTEM "persona.dtd" [
  <!ENTITY % datos_basicos "IGNORE">
  <!ENTITY % datos_ampliados "INCLUDE">
]>

<persona>
  <nombre>Ana</nombre>
  <apellidos>Sanz Tin</apellidos>
  <edad>19</edad>
  <ciudad>Pamplona</ciudad>
</persona>
```

Entre los dos ficheros XML detallados y asociados a la DTD externa se ha indicado –por medio de IGNORE e INCLUDE– si el elemento "persona" tiene que contener 2 ó 4 hijos, es decir, ("nombre" y "edad") o ("nombre", "apellidos", "edad" y "ciudad").

Lo que debe ser congruente es la decisión entre los que se incluye o se ignora y los datos reflejados en el XML.

F. Comentarios:

```
<!-- comentarios -->
```

En una DTD asociada a un documento XML, se pueden escribir comentarios entre los caracteres "<!--" y "-->".

3. 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.

- 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.

4. Limitaciones de los DTD

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

5. Verificación de un documento XML: Bien Formado y Validación

▪ Podemos usar el validador del W3C, que se encuentra en la página:
http://validator.w3.org/#validate_by_uri.

Las pestañas de la parte de arriba del validador nos permiten comprobar la validación tanto de un archivo que está en la web (Validate by URI) como de un archivo local de nuestro ordenador (Validate by File Upload), o escribir directamente el código que queremos validar (Validate by Direct Input). Una vez escrita la ruta (dos primeros casos) o el archivo (tercer caso) pulsamos en Check y nos dirá si el archivo es válido o no.

▪ La mayoría de los navegadores no tienen en cuenta si el documento se ajusta a su DTD sino simplemente si el documento está bien formado. Es decir, sólo tienen en cuenta que el documento se ajuste a las normas básicas de creación de archivos XML.

Sin embargo, podemos poner el atributo standalone="yes" en la primera declaración del archivo XML para comprobar si el XML se ajusta a la DTD:

```
<?xml version="1.0" encoding ="UTF-8" standalone="yes"?>
```

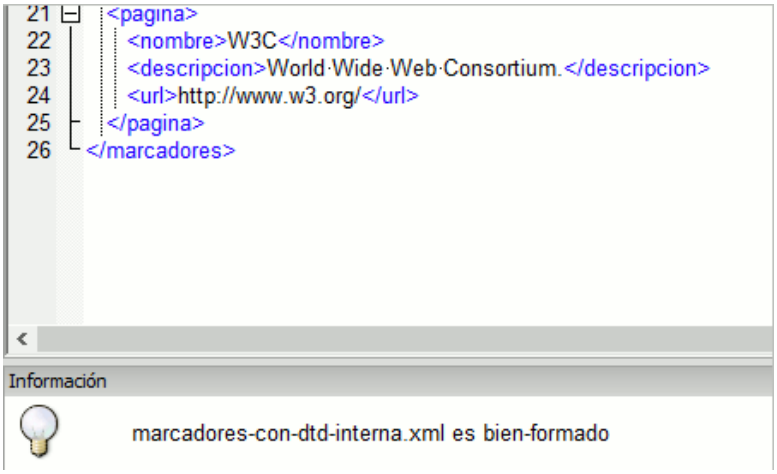
▪ Verificación a través de la aplicación XML Copy Editor:
Para validar un documento XML asociado a una DTD podemos utilizar la aplicación XML Copy Editor, software libre.

http://xml-copy-editor.sourceforge.net/

Una vez instalada la aplicación y cargado el fichero XML:

a) Comprobar si el documento XML está bien formado:
Menú XML: Comprobar Bien-Formado

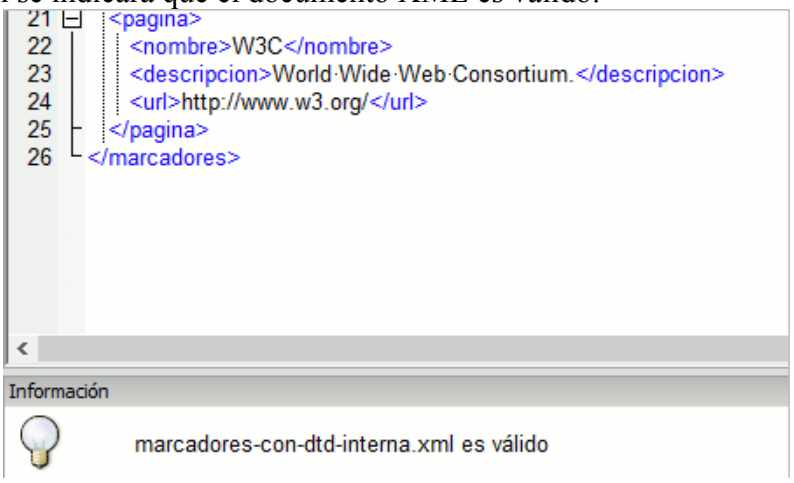
Si XML Copy Editor no detecta ningún error, en la esquina inferior izquierda de la aplicación se visualizará un mensaje indicando que el documento XML está bien formado:



Si el documento no está bien formado, se verá un aviso de error.

b) Comprobar si el documento XML es válido:
Menú XML: Validar: DTD/XML Schema

Si XML Copy Editor valida el documento, en la esquina inferior izquierda de la aplicación se indicará que el documento XML es válido:



Si el documento no es válido, se verá un aviso de error.