
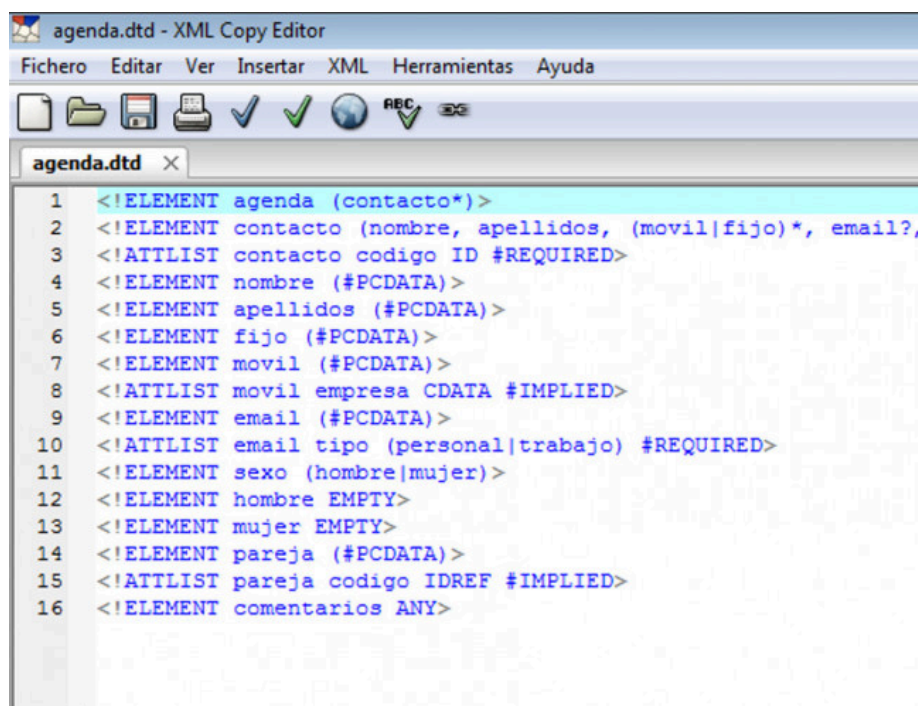


# Qué son los esquemas DTD

 [Rafa Morales](#)  5 Mayo 2014  20min  0

 TECNOLOGÍA



## 1. Qué es un esquema DTD

Un esquema DTD (Document Type Definition) es un documento que define la estructura válida de un documento XML: los elementos, atributos, entidades, notaciones, etc, que pueden aparecer, el orden y el número de veces que pueden aparecer, cuáles pueden ser hijos de cuáles, etc. El procesador XML utiliza el esquema DTD para verificar si un documento es válido, es decir, si el documento cumple las reglas del esquema DTD asociado.

Se trata de un mecanismo de validación de documentos que existía antes de la aparición de XML. Se usaba para validar documentos SGML (Standard Generalized Markup Language), que es el precursor de todos los lenguajes de marcas. Cuando apareció XML, se integró en su especificación como modelo de validación gramatical. Aunque ya ha quedado algo obsoleto, siendo actualmente reemplazado por los esquemas XSD (XML Schema Document), escritos también en XML y muchos más potentes que el esquema DTD.

## 2. Herramientas para el manejo de esquemas DTD

Para diseñar esquemas DTD podemos utilizar el **software** libre [XML Copy Editor](#) o cualquier otro software que comentamos [aquí](#).

### 3. Referencia a un esquema DTD en un documento XML

El esquema DTD que debe utilizar el procesador XML para validar el documento XML se indica mediante la etiqueta DOCTYPE. El esquema DTD puede estar incluido en el propio documento, ser un documento externo o combinarse ambas.

El esquema DTD puede incluirse **en el propio documento XML**, con la siguiente sintaxis:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE agenda [
    ...declaraciones...
]>
<agenda>
    ...
```

El esquema DTD puede estar **en un documento externo** con la extensión *.dtd*, con la sintaxis es la siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE agenda SYSTEM "ruta_fichero">
<agenda>
    ...
```

En este caso, el documento externo "ruta\_fichero" contendrá solamente las declaraciones (que se explicarán a continuación):

```
<!ELEMENT ejemplo (a, b)>
<!ELEMENT a EMPTY>
<!ELEMENT b EMPTY>
...
```

Se puede **combinar** un esquema DTD externo con un esquema DTD interno, utilizando la siguiente sintaxis:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE agenda SYSTEM "ruta_fichero" [
    ...declaraciones...
]>
<agenda>
    ...
```

En todos estos casos:

- "agenda", en el ejemplo, es el elemento raíz del documento XML.
- "ruta\_fichero" es el camino (absoluto o relativo) hasta el fichero que contiene el esquema DTD.

### 4. Declaraciones

Los esquemas DTD describen la estructura de los documentos XML mediante declaraciones. Hay cuatro **tipos de declaraciones**:

- Declaraciones de elementos: Indican los elementos permitidos en un documento y su contenido.
- Declaraciones de entidades: Indican valores constantes en todo el documento XML.
- Declaraciones de notaciones: Indican entidades externas que no se van a analizar.
- Declaraciones de atributos: Indican los atributos permitidos en cada elemento y el tipo o valores permitidos de cada elemento.

## 4.1. Declaración de elementos

Las declaraciones de los elementos siguen la siguiente sintaxis:

```
<!ELEMENT nombreElemento contenido >
```

```
<!ELEMENT nombreElemento (descendientes) >
```

Donde "nombreElemento" es el nombre del elemento, y "contenido" es una expresión que describe el tipo de contenido del elemento, que puede utilizar los términos EMPTY, (#PCDATA) o ANY. Si el elemento contiene otros elementos, se escriben los "descendientes" entre paréntesis, junto con las expresiones de orden y cardinalidad.

**EMPTY:** significa que el elemento es vacío, es decir, que no puede tener contenido. Los elementos vacíos pueden escribirse con etiquetas de apertura y cierre sin nada entre ellos, ni siquiera espacios, o con una etiqueta vacía. Por ejemplo:

```
<!DOCTYPE ejemplo [  
  <!ELEMENT ejemplo EMPTY>  

```

En el documento XML podría aparecer escrito de la siguiente manera:

```
<ejemplo></ejemplo>
```

```
<ejemplo />
```

Los siguientes casos darían un error de validación:

```
<ejemplo>Esto es un ejemplo</ejemplo> <!-- ERROR: contiene texto -->
```

```
<ejemplo><a></a></ejemplo> <!-- ERROR: contiene un elemento <a> -->
```

**(#PCDATA):** significa que el elemento puede contener texto (Parsed Character Data), pero no otros elementos (debe utilizarse en la definición los paréntesis de manera obligatoria). Por ejemplo:

```
<!DOCTYPE ejemplo [  
  <!ELEMENT ejemplo (#PCDATA)>  

```

En el documento XML podría aparecer escrito de la siguiente manera:

```
<ejemplo />
```

```
<ejemplo>Esto es un ejemplo</ejemplo>
```

Los siguientes casos darían un error de validación:

```
<ejemplo><a></a></ejemplo> <!-- ERROR: contiene un elemento <a> -->
```

**ANY:** significa que el elemento puede contener cualquier cosa (texto y otros elementos). Por ejemplo:

```
<!DOCTYPE ejemplo [  
  <!ELEMENT ejemplo ANY>  
  <!ELEMENT a ANY>  

```

En el documento XML podría aparecer escrito de la siguiente manera:

```
<ejemplo />
```

```
<ejemplo>Esto es un ejemplo</ejemplo>
```

```
<ejemplo><a>Esto es un ejemplo</a></ejemplo>
```

, (**coma**): significa que el elemento contiene los elementos en el orden indicado. Por ejemplo:

```
<!DOCTYPE ejemplo [  
  <!ELEMENT ejemplo (a, b)>  
  <!ELEMENT a EMPTY>  
  <!ELEMENT b EMPTY>  

```

En el documento XML podría aparecer escrito de la siguiente manera:

```
<ejemplo><a /><b /></ejemplo>
```

Los siguientes casos darían un error de validación:

```
<ejemplo><a /><b /><c /></ejemplo> <!-- ERROR: contiene un elemento <c /> -->
```

```
<ejemplo><a /></ejemplo> <!-- ERROR: falta el elemento <b /> -->
```

```
<ejemplo><b /><a /></ejemplo> <!--ERROR: el orden no es correcto -->
```

| (**tubería**): significa que el elemento contiene uno de los dos elementos. Por ejemplo:

```
<!DOCTYPE ejemplo [  
  <!ELEMENT ejemplo (a | b)>  
  <!ELEMENT a EMPTY>  
  <!ELEMENT b EMPTY>  

```

En el documento XML podría aparecer escrito de la siguiente manera:

```
<ejemplo><a /></ejemplo>
```

```
<ejemplo><b /></ejemplo>
```

Los siguientes casos darían un error de validación:

```
<ejemplo><a /><b /></ejemplo> <!-- ERROR: están los dos elementos -->
```

```
<ejemplo></ejemplo> <!-- ERROR: no hay ningún elemento -->
```

? (**interrogante**): significa que el elemento puede aparecer o no, pero sólo una vez. Por ejemplo:

```
<!DOCTYPE ejemplo [  
  <!ELEMENT ejemplo (a, b?)>  
  <!ELEMENT a EMPTY>  
  <!ELEMENT b EMPTY>  

```

En el documento XML podría aparecer escrito de la siguiente manera:

```
<ejemplo><a /></ejemplo>
```

```
<ejemplo><a /><b /></ejemplo>
```

Los siguientes casos darían un error de validación:

```
<ejemplo><b /></ejemplo> <!-- ERROR: falta el elemento <a /> -->
```

```
<ejemplo><b /><b /></ejemplo> <!-- ERROR: el elemento <b /> aparece dos veces -->
```

**\*** (**asterísco**): significa que el elemento puede no aparecer, o aparecer una o más veces. Por ejemplo:

```
<!DOCTYPE ejemplo [  
  <!ELEMENT ejemplo (a*, b)>  
  <!ELEMENT a EMPTY>  
  <!ELEMENT b EMPTY>  

```

En el documento XML podría aparecer escrito de la siguiente manera:

```
<ejemplo><b /></ejemplo>
```

```
<ejemplo><a /><a /><b /></ejemplo>
```

Los siguientes casos darían un error de validación:

```
<ejemplo><b /><a /></ejemplo> <!-- ERROR: el elemento <a /> aparece después de <b /> -->
```

**+** (**suma**): significa que el elemento tiene que aparecer una o más veces de manera obligatoria. Por ejemplo:

```
<!DOCTYPE ejemplo [  
  <!ELEMENT ejemplo (a+, b)>  
  <!ELEMENT a EMPTY>  
  <!ELEMENT b EMPTY>  

```

En el documento XML podría aparecer escrito de la siguiente manera:

```
<ejemplo><a /><b /></ejemplo>
```

```
<ejemplo><a /><a /><b /></ejemplo>
```

Los siguientes casos darían un error de validación:

```
<ejemplo><b /></ejemplo> <!-- ERROR: falta el elemento <a /> -->
```

**()** (**paréntesis**): permite agrupar expresiones. Por ejemplo:

```
<!DOCTYPE ejemplo [  
  <!ELEMENT ejemplo (a, (a|b))>  
  <!ELEMENT a EMPTY>  
  <!ELEMENT b EMPTY>  

```

En el documento XML podría aparecer escrito de la siguiente manera:

```
<ejemplo><a /><a /></ejemplo>
```

```
<ejemplo><a /><b /></ejemplo>
```

Los siguientes casos darían un error de validación:

```
<ejemplo><a /></ejemplo> <!-- ERROR: falta el elemento <a /> o <b /> -->
```

Otro ejemplo:

```
<!DOCTYPE ejemplo [  
  <!ELEMENT ejemplo ((a, b)|(b, a))>  
  <!ELEMENT a EMPTY>  
  <!ELEMENT b EMPTY>  

```

En el documento XML podría aparecer escrito de la siguiente manera:

```
<ejemplo><a /><b /></ejemplo>
```

```
<ejemplo><b /><a /></ejemplo>
```

Los siguientes casos darían un error de validación:

```
<ejemplo><a /><a /></ejemplo> <!-- ERROR: sólo admite <a /><b /> o <b /><a /> -->
```

## 4.2. Declaración de entidades

Una **entidad** consiste en un nombre y su valor (son similares a las constantes en los lenguajes de programación). Con algunas excepciones, el procesador XML sustituye las referencias a entidades por sus valores antes de procesar el documento. Una vez definida la entidad, se puede utilizar en el documento escribiendo una referencia a la entidad, que empieza con el carácter "&", sigue con el nombre de la entidad y termina con punto y coma ";". (es decir, &nombreEntidad;).

Las entidades pueden ser internas o externas, y tanto unas como otras pueden ser generales o paramétricas.

Las declaraciones de **entidades internas (generales)** siguen la siguiente sintaxis:

```
<!ENTITY nombreEntidad "valorEntidad">
```

En las declaraciones de **entidades externas (generales)** se distinguen dos casos, según sea un fichero de texto o no.

1. La entidad hace referencia a un *fichero de texto* y en ese caso la entidad se sustituye por el contenido del archivo.

La entidad puede ser una entidad de sistema, con la siguiente sintaxis:

```
<!ENTITY nombreEntidad SYSTEM "uri">
```

O puede ser una entidad pública, con la siguiente sintaxis:

```
<!ENTITY nombreEntidad PUBLIC "fpi" "uri">
```

2. La entidad hace referencia a un *fichero que no es de texto* (por ejemplo, una imagen) y en ese caso la entidad no se sustituye por el contenido del archivo.

La entidad puede ser una entidad de sistema, con la siguiente sintaxis:

```
<!ENTITY nombreEntidad SYSTEM "uri" NDATA tipo>
```

O puede ser una entidad pública, con la siguiente sintaxis:

```
<!ENTITY nombreEntidad PUBLIC "fpi" "uri" NDATA tipo>
```

En todos estos casos:

- "nombreEntidad" es el nombre de la entidad.

- "valorEntidad" es el valor de la entidad.
- "uri" es el camino (absoluto o relativo) hasta un archivo.
- "tipo" es el tipo de archivo (gif, jpg, etc).
- "fpi" es un indentificador público formal (Formal Public Identifier).

Las declaraciones de **entidades paramétricas** siguen la mismas sintaxis que las generales, pero llevan el caracter "%" antes del nombre de la entidad. La diferencia entre entidades generales y paramétricas es que las entidades paramétricas se sustituyen por su valor en todo el documento (incluso en la propia declaración de tipo de documento), mientras que las generales no se sustituyen en la declaración de tipo de documento. Por ejemplo:

```
<!ENTITY % nombreEntidad "valorEntidad">
```

```
<!ENTITY % nombreEntidad SYSTEM "uri">
```

```
<!ENTITY % nombreEntidad SYSTEM "uri" NDATA tipo>
```

### 4.3. Declaración de notaciones

Las notaciones se usan en XML para definir las entidades externas que no va a analizar el procesador XML (aunque sí lo hará la aplicación que trate un documento). Para hacer referencia estas entidades no se utiliza la notación habitual (&nombreEntidad;), sino que se utiliza el nombre de la entidad directamente.

### 4.4. Declaración de atributos

Una declaración de **atributos** sigue la siguiente sintaxis:

```
<!ATTLIST nombreElemento nombreAtributo tipoAtributo valorInicialAtributo >
```

Donde:

- "nombreElemento": Es el nombre del elemento para el que se define un atributo.
- "nombreAtributo": Es el nombre del atributo.
- "tipoAtributo": Es el tipo de dato que almacenará el atributo.
- "valorInicialAtributo": Es el valor predeterminado del atributo o la opcionalidad u obligatoriedad del mismo.

Para **definir varios atributos de un mismo elemento**, se puede utilizar una o varias declaraciones de atributos. Los siguientes ejemplos son **equivalentes**:

```
<!ATTLIST nombreElemento nombreAtributo1 tipoAtributo1 valorInicialAtributo1>
<!ATTLIST nombreElemento nombreAtributo2 tipoAtributo2 valorInicialAtributo2>
```

```
<!ATTLIST nombreElemento nombreAtributo1 tipoAtributo1 valorInicialAtributo1
                             nombreAtributo2 tipoAtributo2 valorInicialAtributo2 >
```

Los **tipos de atributos** son los siguientes:

**CDATA:** El atributo contiene caracteres (sin restricciones). Por ejemplo:

```
<!DOCTYPE ejemplo [
  <!ELEMENT ejemplo EMPTY>
  <!ATTLIST ejemplo color CDATA #REQUIRED>
]>
```

En el documento XML podría aparecer escrito de la siguiente manera:

```
<ejemplo color=""></ejemplo>
```

```
<ejemplo color="amarillo"></ejemplo>
```

```
<ejemplo color="azul marino #000080"></ejemplo>
```

Los siguientes casos darían un error de validación:

```
<ejemplo></ejemplo> <!-- ERROR: falta el atributo "color", obligatorio debido al #REQUIRED -->
```

**NMTOKEN:** El atributo sólo contiene letras, dígitos y los caracteres punto ".", guión "-", subrayado "\_" y dos puntos ":". Por ejemplo:

```
<!DOCTYPE ejemplo [  
  <!ELEMENT ejemplo EMPTY>  
  <!ATTLIST ejemplo color NMTOKEN #REQUIRED>  

```

En el documento XML podría aparecer escrito de la siguiente manera:

```
<ejemplo color=""></ejemplo>
```

```
<ejemplo color="azul-marino"></ejemplo>
```

```
<ejemplo color="1"></ejemplo>
```

Los siguientes casos darían un error de validación:

```
<ejemplo color="azul marino"></ejemplo> <!-- ERROR: hay un espacio en blanco -->
```

```
<ejemplo color="#F0F0F0"></ejemplo> <!-- ERROR: contiene el carácter # -->
```

**NMTOKENS:** El atributo sólo contiene letras, dígitos y los caracteres punto ".", guión "-", subrayado "\_", dos puntos ":" (como el tipo NMTOKEN), pero también espacios en blanco. Por ejemplo:

```
<!DOCTYPE ejemplo [  
  <!ELEMENT ejemplo EMPTY>  
  <!ATTLIST ejemplo color NMTOKENS #REQUIRED>  

```

En el documento XML podría aparecer escrito de la siguiente manera:

```
<ejemplo color=""></ejemplo>
```

```
<ejemplo color="1"></ejemplo>
```

```
<ejemplo color="azul marino"></ejemplo>
```

Los siguientes casos darían un error de validación:

```
<ejemplo color="2*2"></ejemplo> <!-- ERROR: hay un asterisco -->
```

**enumeración:** El atributo sólo puede contener uno de los términos de una lista. La lista se escribe entre paréntesis, con los términos separados por una barra vertical "|". Por ejemplo:

```
<!DOCTYPE ejemplo [  
  <!ELEMENT ejemplo EMPTY>  
  <!ATTLIST ejemplo color (azul | blanco | rojo) #REQUIRED>  

```



En el documento XML podría aparecer escrito de la siguiente manera:

```
<ejemplo color=""></ejemplo>
```

```
<ejemplo color="azul"></ejemplo>
```

Los siguientes casos darían un error de validación:

```
<ejemplo color="verde"></ejemplo> <!-- ERROR: "verde" no está en la lista de valores -->
```

**ID:** El valor del atributo (no el nombre) debe ser único y no se puede repetir en otros elementos o atributos. Por ejemplo:

```
<!DOCTYPE ejemplo [  
  <!ELEMENT ejemplo (libro*)>  
  <!ELEMENT libro (#PCDATA) >  
  <!ATTLIST libro codigo ID #REQUIRED>  

```

En el documento XML podría aparecer escrito de la siguiente manera:

```
<ejemplo>  
  <libro codigo="L1">Poema de Gilgamesh</libro>  
  <libro codigo="L2">Los preceptos de Ptah-Hotep</libro>  
</ejemplo>
```

Los siguientes casos darían un error de validación:

```
<ejemplo>  
  <libro codigo="1">Poema de Gilgamesh</libro> <!-- ERROR: el valor de un atributo de tipo ID no puede  
empezar con un número -->  
  <libro codigo="L2">Los preceptos de Ptah-Hotep</libro>  
</ejemplo>
```

```
<ejemplo>  
  <libro codigo="L1">Poema de Gilgamesh</libro>  
  <libro codigo="L1">Los preceptos de Ptah-Hotep</libro> <!-- ERROR: el valor "L1" ya se ha utilizado --  
>  
</ejemplo>
```

**IDREF:** El valor del atributo debe coincidir con el valor del atributo ID de otro elemento. Por ejemplo:

```
<!DOCTYPE ejemplo [  
  <!ELEMENT ejemplo ((libro|prestamo)*)>  
  <!ELEMENT libro (#PCDATA) >  
  <!ATTLIST libro codigo ID #REQUIRED>  
  <!ELEMENT prestamo (#PCDATA) >  
  <!ATTLIST prestamo libro IDREF #REQUIRED>  

```

En el documento XML podría aparecer escrito de la siguiente manera:

```
<ejemplo>  
  <libro codigo="L1">Poema de Gilgamesh</libro>  
  <prestamo libro="L1">Numa Nigerio</prestamo>  
</ejemplo>
```

Los siguientes casos darían un error de validación:

```
<ejemplo>
  <libro codigo="L1">Poema de Gilgamesh</libro>
  <prestamo libro="L2">Numa Nigerio</prestamo> <!-- ERROR: el valor "L2" no es ID de ningún elemento -->
</ejemplo>
```

**IDREFS:** El valor del atributo es una serie de valores separados por espacios que coinciden con el valor del atributo ID de otros elementos.

```
<!DOCTYPE ejemplo [
  <!ELEMENT ejemplo ((libro|prestamo)*)>
  <!ELEMENT libro (#PCDATA) >
  <!ATTLIST libro codigo ID #REQUIRED>
  <!ELEMENT prestamo (#PCDATA) >
  <!ATTLIST prestamo libro IDREFS #REQUIRED>
]>
```

En el documento XML podría aparecer escrito de la siguiente manera:

```
<ejemplo>
  <libro codigo="L1">Poema de Gilgamesh</libro>
  <libro codigo="L2">Los preceptos de Ptah-Hotep</libro>
  <prestamo libro="L1 L2">Numa Nigerio</prestamo>
</ejemplo>
```

```
<ejemplo>
  <libro codigo="L1">Poema de Gilgamesh</libro>
  <libro codigo="L2">Los preceptos de Ptah-Hotep</libro>
  <prestamo libro="L1">Numa Nigerio</prestamo>
</ejemplo>
```

Los siguientes casos darían un error de validación:

```
<ejemplo>
  <libro codigo="L1">Poema de Gilgamesh</libro>
  <libro codigo="L2">Los preceptos de Ptah-Hotep</libro>
  <prestamo libro="L3">Numa Nigerio</prestamo> <!-- ERROR: el valor "L3" no es ID de ningún elemento -->
</ejemplo>
```

**ENTITY:** El valor del atributo es alguna entidad definida en la DTD.

**ENTITIES:** El valor del atributo es alguna de las entidades de una lista de entidades definida en la DTD.

**NOTATION:** el valor del atributo es alguna notación definida en la DTD.

Los **valores iniciales** de los atributos son los siguientes:

**#REQUIRED:** El atributo es obligatorio, aunque no se especifica ningún valor predeterminado. Por ejemplo:

```
<!DOCTYPE ejemplo [
  <!ELEMENT ejemplo EMPTY>
  <!ATTLIST ejemplo color CDATA #REQUIRED>
]>
```

En el documento XML podría aparecer escrito de la siguiente manera:

```
<ejemplo color=""></ejemplo>
```

```
<ejemplo color="amarillo"></ejemplo>
```

```
<ejemplo color="azul marino #000080"></ejemplo>
```

Los siguientes casos darían un error de validación:

```
<ejemplo></ejemplo> <!-- ERROR: falta el atributo "color" -->
```

**#IMPLIED:** El atributo no es obligatorio y no se especifica ningún valor predeterminado. Por ejemplo:

```
<!DOCTYPE ejemplo [  
  <!ELEMENT ejemplo EMPTY>  
  <!ATTLIST ejemplo color CDATA #IMPLIED>  

```

En el documento XML podría aparecer escrito de la siguiente manera:

```
<ejemplo></ejemplo>
```

```
<ejemplo color=""></ejemplo>
```

```
<ejemplo color="amarillo"></ejemplo>
```

```
<ejemplo color="azul marino #000080"></ejemplo>
```

**#FIXED valor:** El atributo tiene un valor fijo. Por ejemplo:

```
<!DOCTYPE ejemplo [  
  <!ELEMENT ejemplo EMPTY>  
  <!ATTLIST ejemplo color CDATA #FIXED "verde">  

```

En el documento XML podría aparecer escrito de la siguiente manera:

```
<ejemplo></ejemplo>
```

```
<ejemplo color="verde"></ejemplo>
```

Los siguientes casos darían un error de validación:

```
<ejemplo color=""></ejemplo> <!-- ERROR: el atributo "color" no tiene el valor "verde" -->
```

```
<ejemplo color="amarillo"></ejemplo> <!-- ERROR: el atributo "color" no tiene el valor "verde" -->
```

**valor:** El atributo tiene un valor predeterminado. Por ejemplo:

```
<!DOCTYPE ejemplo [  
  <!ELEMENT ejemplo EMPTY>  
  <!ATTLIST ejemplo color CDATA "verde">  

```

En el documento XML podría aparecer escrito de la siguiente manera:

```
<ejemplo></ejemplo>
```

```
<ejemplo color=""></ejemplo>
```

```
<ejemplo color="amarillo"></ejemplo>
```

```
<ejemplo color="verde"></ejemplo>
```

## 5. Limitaciones de los esquemas DTD

Algunas limitaciones de los DTD son:

- Un DTD no es un documento XML, luego no se puede verificar si está bien formado.
- No se pueden fijar restricciones sobre los valores y atributos, como su tipo de datos, su tamaño, etc.
- No soporta espacios de nombres.
- Sólo se puede enumerar los valores de los atributos, no de los elementos.
- Sólo se puede dar un valor por defecto para atributos, no para elementos.
- Existe un control limitado sobre las cardinalidades de los elementos, es decir, concretar el número de veces que pueden aparecer.

## 6. Generación automática de esquemas DTD

La mayoría de herramientas especializadas en lenguajes XML poseen alguna utilidad para generar de manera automática un esquema DTD a partir de un documento XML bien formado.

Por ejemplo, en XML Copy Editor, podemos encontrar esta opción en el menú *XML -> Create Schema...*

## 7. Validación de documentos XML con esquemas DTD

La mayoría de herramientas especializadas en lenguajes XML poseen la funcionalidad de validar documentos XML mediante su esquema DTD asociado.

## 8. Ficheros de ejemplos

Al final de la página podéis encontrar ficheros de ejemplo validados. Por un lado los ficheros "agenda.xml" y "agenda.dtd" es un ejemplo en el que el esquema DTD se almacena en un fichero externo. Por otro lado, el fichero "agenda-con-dtd.xml" es el mismo ejemplo pero incluyendo el esquema DTD en la declaración DOCTYPE.

## 9. Bibliografía y recursos

- Sintés Marco, Bartolomé. "DTD: Definición de Tipo de Documento" del curso "XML: Lenguaje de Marcas Extensible Licencia". Licencia CC BY-CN-SA. [Consulta: Abril 2014]. <http://www.mclibre.org/>
- DTD Tutorial. W3Schools. <http://www.w3schools.com/DTD/>

### Archivos adjuntos

Adjunto	Tamaño
<a href="#">agenda.xml</a>	907 bytes
<a href="#">agenda.dtd</a>	581 bytes
<a href="#">agenda-con-dtd.xml</a>	1.47 KB



Qué son los esquemas DTD escrito por [Rafa Morales](#) está protegido por una licencia [Creative Commons Atribución-NoComercial-CompartirIgual 4.0 Internacional](#)



[Inicie sesión](#) o [regístrese](#) para enviar comentarios

## Artículos similares

- 🔖 [Ejercicios prácticos de XSLT](#)
- 🔖 [Ejercicios prácticos de XQuery](#)
- 🔖 [Ejercicios prácticos de DTD](#)
- 🔖 [Qué es el lenguaje de transformación XSLT](#)
- 🔖 [Ejercicios prácticos de XSLT \(Soluciones\)](#)

## Susíbete a TicArte

Tu correo electrónico

☐ Acepto recibir en mi correo electrónico los artículos publicados [Términos y Condiciones de Uso](#) \*

Suscribirme

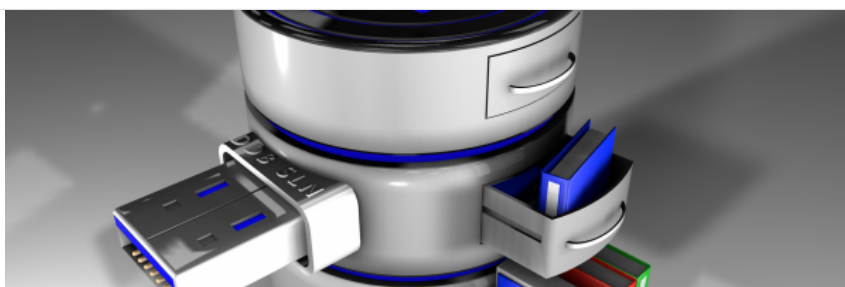
## Síguenos



## Ciclos formativos



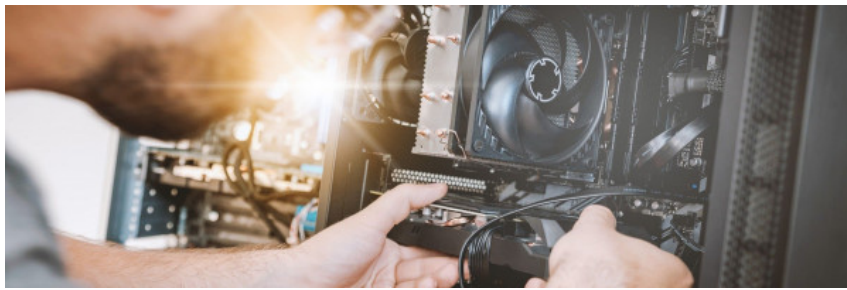
[Administración de Sistemas Operativos](#)



[Gestión de Bases de Datos](#)



[Lenguajes de Marcas y Sistemas de Gestión de Información](#)



### Montaje y Mantenimiento de Equipo



### Planificación y Administración de Redes



### Sistemas Operativos en Red

#### Invítanos a un café

Si te gusta nuestro trabajo, puedes colaborar invitándonos a un café.

