

XML



INTRODUCCION

XML (eXtensible Markup Language) es un estándar desarrollado por el W3C (World Wide Web Consortium) en 1996 para el intercambio de información estructurada entre diferentes sistemas.

No tiene una versión "actual" como tal, ya que es una especificación que ha evolucionado a lo largo del tiempo mediante la introducción de nuevas características y mejoras.

Sin embargo, las versiones más comunes de XML incluyen:

1. XML 1.0: Esta es la versión original de XML, publicada por el W3C en 1998. Es la especificación fundamental que define la sintaxis y la semántica básica de XML.
2. XML 1.1: Publicada por el W3C en 2004, XML 1.1 es una revisión de XML 1.0 que introduce algunas mejoras y cambios en la sintaxis. Sin embargo, su adopción es limitada y no se utiliza ampliamente en comparación con XML 1.0

XML 1.1 Second Edition. Fue publicada por el W3C en 2006 y es una revisión de XML 1.1, que a su vez es una extensión de XML 1.0. Esta edición proporciona algunas correcciones de errores y clarificaciones sobre la especificación original de XML 1.1

La diferencia principal entre las dos versiones radica en los conjuntos de caracteres UNICODE que emplea. La versión XML 1.1 no depende de una versión específica de UNICODE sino siempre de la última.

La quinta edición de XML 1.0 fue publicada en noviembre de 2008. Esta edición incluye correcciones y clarificaciones, pero no introduce cambios significativos en la especificación principal de XML 1.0

¿QUÉ ES UN XML?

- Es un metalenguaje de marcas, lo que significa que no dispone de un conjunto fijo de etiquetas.
- Permite definir a los desarrolladores los elementos que necesiten y con la estructura que mejor les convenga
- Define una sintaxis general para maquetar datos con etiquetas sencillas y comprensibles al ojo humano.
- Provee un formato estándar para documentos informáticos.
- Es flexible y puede adaptarse al campo de aplicación que se desee.
- Tiene formato de texto plano editable por cualquier editor de texto.
- Los documentos xml están formados por marcas-etiquetas y por información que se representa como texto, no hay tipo de datos numéricos, fechas, etc..
- Ejemplos de marcas: <> para representar etiquetas. &código; para representar una entidad.
- Un documento XML no hace nada, es la base para estructurar, almacenar y transportar la información que otro software procesa.

LO QUE NO ES UN XML

- No es un lenguaje de programación, por lo que no hay compiladores que generen ejecutables a partir de un documento XML
- No es un protocolo de comunicación, de manera que no enviará datos por nosotros a través de internet
- No es un sistema gestor de bases de datos
- No es propietario, es decir, no pertenece a ninguna compañía

XML Y HTML

Comparten similitudes en su estructura de marcado y ambos lenguajes de marcado son utilizados para representar información de manera estructurada, pero están diseñados para propósitos diferentes y tienen diferencias fundamentales:

Propósito:

HTML: Se utiliza principalmente para crear páginas web y describir su estructura y contenido.

XML: Es un lenguaje de marcado genérico que se utiliza para almacenar y transportar datos de manera estructurada.

Sintaxis:

HTML: Está diseñado para presentar información en forma de páginas web. Tiene una estructura más rígida y está orientado hacia la presentación visual.

XML: Está diseñado para describir datos de manera estructurada. Tiene una sintaxis más flexible y permite definir etiquetas y atributos personalizados para adaptarse a diferentes necesidades de datos.

Semántica:

HTML: Tiene una semántica predefinida para representar elementos como encabezados, párrafos, listas, enlaces, etc., que son interpretados por los navegadores web para presentar contenido visualmente.

XML: No tiene una semántica predefinida y puede ser utilizado para representar cualquier tipo de datos de manera estructurada, como información de configuración, datos de intercambio entre aplicaciones, documentos científicos, entre otros.

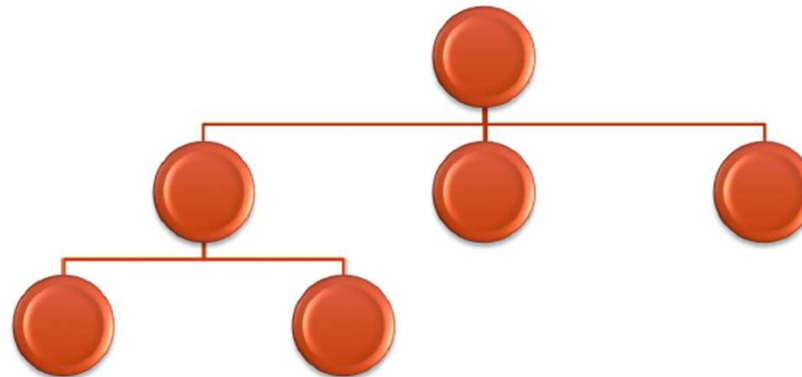
Validación:

HTML: Los navegadores web suelen ser tolerantes a errores y pueden renderizar el contenido incluso si hay errores de sintaxis en el HTML.

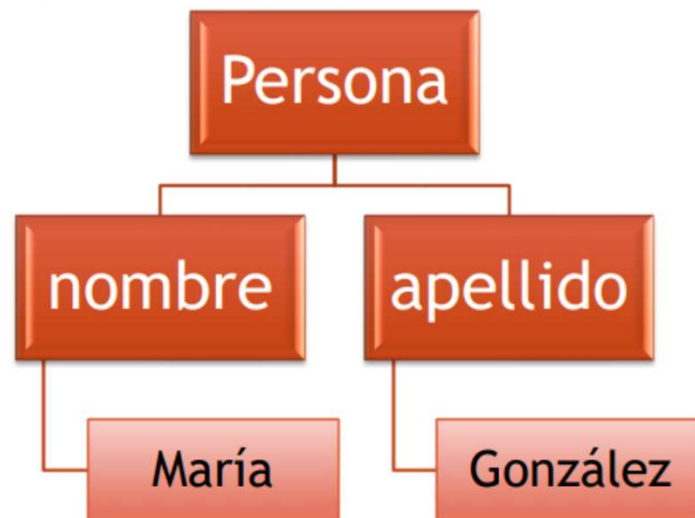
XML: Requiere una sintaxis estrictamente válida y puede ser validado mediante un DTD (Document Type Definition) o un esquema XML para garantizar su conformidad con la estructura definida

ESTRUCTURA DE UN DOCUMENTO XML

- Un documento XML tiene una estructura jerárquica
- A esta estructura jerárquica se la denomina árbol del documento XML
- A las partes del árbol que tienen hijos se las denomina nodos intermedios o ramas, mientras que a las que no tienen se conocen como nodos finales u hojas



```
<persona>
  <nombre>María</nombre>
  <apellido>González</apellido>
</persona>
```



No dispone de una visualización concreta en un navegador puesto que el documento no refleja una apariencia, sino unos datos.

Al abrir un archivo XML en un navegador web, editor de texto o herramienta de desarrollo lo que se mostrará es su estructura jerárquica, fácil de leer y entender.

Pero si lo que queremos es visualizar **los datos** de un documento XML con otro formato, hay que hacerlo:

- ✓ Mediante una hoja de estilo CSS que indique al navegador cómo convertir cada elemento del documento XML en un elemento visual

`<?xml stylesheet type="text/css" href="estilos.css"?>`

- ✓ Mediante el uso de una hoja de transformaciones XSLT. La instrucción equivalente para asociar a un documento XML una hoja de transformaciones es

`<?xml stylesheet type="text/xsl" href="estilos.xsl"?>`

- ✓ Mediante el uso de un lenguaje de programación como Java o Javascript

Estructura del documento

Prólogo: es una parte opcional pero recomendable, precede al elemento raíz y contiene información sobre la versión de XML utilizada y cualquier otra declaración necesaria para el procesamiento del documento.

Empieza con `<?` y terminan con `?>` Son instrucciones para el procesador XML y no forman parte del contenido del documento. Se utilizan para dar información a las aplicaciones que procesan el documento Ej: `<?xml version="1.0" encoding="UTF-8"?>`

Cuerpo: contiene toda la información del documento y se compone de:

- **Raíz:** por encima de cualquier elemento se ubica el nodo raíz. Se utiliza como punto de partida para recorrer el árbol XML y ubicar el resto de nodos
- **Elementos:** Es la unidad básica de un documento XML. Se identifican por una etiqueta de apertura y una de cierre Lo que se ubica entre ambas es el contenido de ese elemento que puede ser textual, otros elementos o vacío
- **Atributos:** son como los atributos de HTML. Son pares *nombre="valor"* que permiten especificar datos adicionales de un elemento. Se ubican en la etiqueta de apertura del elemento. Para asignar un valor a un atributo se utiliza el signo igual. Todos los atributos se tratarán como texto y aparecerán entre comillas

Ejemplo:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<nota>
<para>Pedro</para>
<de>Laura</de>
<titulo>Recordatorio</titulo>
<contenido>A las 7:00 pm en la puerta del teatro</contenido>
</nota>
```

- La primera línea es la declaración XML. En esta línea se especifica la versión XML que se utiliza (1.0), y el tipo de codificación utilizada (en este caso ISO 8859 1, que corresponde al juego de caracteres Latin 1 / West European).
- La siguiente línea contiene la marca de apertura del elemento raíz del documento: “nota”
- Las cuatro líneas siguientes describen 4 elementos hijos del elemento raíz (“para”, “de”, “título” y “contenido”)
- La última línea contiene el cierre del elemento raíz, y el final del documento.

MODELO DE DATOS DE UN DOCUMENTO XML

Toda la información de un documento de XML se representa con texto, caracteres ya sean de las familias UNICODE o ISO IEC 10646

Texto: puede aparecer como contenido de un elemento o como valor de un atributo
No puede aparecer en ningún otro lugar.

Los *espacios* reciben un tratamiento especial

Tabulador:	\t			TAB
Nueva línea:	\n	
	LF
Retorno de carro:	\r		CR
Espacio:	\s	 	NO-BREAK SPACE

Dentro del contenido de un elemento se mantendrán como están y así serán tratados por el procesador

Como valor de un atributo, los espacios en blanco adyacentes se condensarán en uno solo y los espacios en blanco entre elementos serán ignorados

Ejemplo:

<code><A> Dato </code>	<code>≠</code>	<code><A>Dato</code>
<code></code>	<code>=</code>	<code></code>
<code><A>Más datos Y más</code>	<code>=</code>	<code><A>Más datosY más</code>

Comentarios: Son iguales que en HTML, empiezan con `<!--` y se cierran con `-->`

Dentro de ellos se puede escribir cualquier cosa excepto `--`

Pueden ubicarse en cualquier lugar del documento excepto dentro de una etiqueta de apertura o cierre

Entidades predefinidas: representan caracteres especiales de marcado pero son interpretados como texto por el procesador

Entidad	Carácter
&	&
<	<
>	>
'	'
"	"

Ejemplo:

`<bebida>Barnes & Noble</bebida>`

El contenido del elemento `<bebida>` será interpretado por el analizador como Barnes & Noble

Secciones CDATA: son conjuntos de caracteres que el procesador no debe analizar

Permite agilizar el análisis del documento y deja libertad al autor para introducir libremente en ellas caracteres como < y &

No pueden aparecer antes del elemento raíz ni después de su cierre.

No pueden contener el propio signo delimitador de final de sección CDATA, es decir,]]>

Ejemplo:

```
<código>
```

```
<![CDATA[<html><body><h3>Título de la página</h3></body></]]>
```

```
</código>
```

Definición de Tipo de Documento (DTD): Permite definir reglas que fuercen ciertas restricciones sobre la estructura de un documento xml. Un documento XML bien formado que tiene asociado un documento de declaración de tipos y cumple con las restricciones allí declaradas se dice que es válido.

Debe aparecer en la segunda línea del documento, entre la instrucción de procesamiento inicial y el elemento raíz

ETIQUETAS Y ATRIBUTOS

En XML, los nombres de elementos y atributos deben seguir algunas reglas para ser considerados válidos:

- ❖ Comienzan con una letra o guion bajo (_): Los nombres pueden comenzar con cualquier letra del alfabeto (mayúscula o minúscula) o con un guion bajo _
- ❖ Pueden contener letras, dígitos y otros símbolos especiales: Después del primer carácter, pueden contener letras, dígitos, guiones bajos, comas, puntos, guion medio y dos puntos (-, ., : _)
- ❖ No pueden contener espacios en blanco.
- ❖ No pueden comenzar con ciertas secuencias reservadas: No pueden comenzar con ciertas secuencias reservadas como xml (o cualquier combinación de letras en mayúsculas o minúsculas que sean equivalentes a xml), o cualquier secuencia que comience con :, .-, o _:.
- ❖ Es sensible a mayúsculas y minúsculas: lo que significa que Nombre y nombre se consideran diferentes

ESPACIOS DE NOMBRES

Los espacios de nombres en XML (XML namespaces) se utilizan para evitar conflictos de nombres entre elementos y atributos en un documento XML.

Permiten asignar un identificador único a cada elemento y atributo, incluso si tienen el mismo nombre pero pertenecen a diferentes vocabularios o dominios.

Los espacios de nombres se definen utilizando una sintaxis especial en XML. El formato típico de un espacio de nombres es:

```
<elemento xmlns:prefijo="URI del espacio de nombres">  
...  
</elemento>
```

Donde ***xmlns*** es el atributo utilizado para el espacio de nombre, ***prefijo*** es un identificador único que se utiliza para hacer referencia al espacio de nombres dentro del documento, y ***URI del espacio de nombres*** es una cadena que identifica de manera única el espacio de nombres.

URI (*Uniform Resource Identifier*, Identificador Uniforme de Recurso)

En el ejemplo:

xmlns es un atributo que se ha utilizado en la etiqueta de inicio del elemento **<ejemplo>** y, en este caso, se han definido dos espacios de nombres que hacen referencia a los siguientes

- <http://www.abrirllave.com/ejemplo1>
- <http://www.abrirllave.com/ejemplo2>

Los prefijos definidos son **e1** y **e2**, respectivamente.

Véase que, se han añadido dichos prefijos a las etiquetas que aparecen en el documento:
<e1:carta>, **<e2:carta>**, **<e1:palo>**, etc

Los URI especificados en un documento XML no tienen por qué contener nada, su función es ser únicos. No obstante, en un URI se puede mostrar información si se considera oportuno.

Pueden definirse en el elemento raíz –como en el ejemplo– o, directamente, en los elementos que los vayan a utilizar y es posible definir todos los espacios de nombres que se necesiten

```
<?xml version="1.0" encoding="UTF-8"?>
<e1:ejemplo xmlns:e1="http://www.abrirllave.com/ejemplo1"
  xmlns:e2="http://www.abrirllave.com/ejemplo2">

  <e1:carta>
    <e1:palo>Corazones</e1:palo>
    <e1:numero>7</e1:numero>
  </e1:carta>

  <e2:carta>
    <e2:carnes>
      <e2:filete_de_tenera precio="12.95"/>
      <e2:solomillo_a_la_pimienta precio="13.60"/>
    </e2:carnes>
    <e2:pescados>
      <e2:lenguado_al_horno precio="16.20"/>
      <e2:merluza_en_salsa_verde precio="15.85"/>
    </e2:pescados>
  </e2:carta>

</e1:ejemplo>
```

Espacio de nombres por defecto

Un espacio de nombres por defecto es aquel en el que no se define un prefijo

`xmlns="URI"`

Su ámbito de aplicación es el del elemento en el que se ha declarado y sus descendientes, pero no sus atributos.

```
<?xml version="1.0" encoding="UTF-8"?>
<ejemplo xmlns="http://www.abrirllave.com/ejemplo1">

  <carta>
    <palo>Corazones</palo>
    <numero>7</numero>
  </carta>

</ejemplo>
```

Cómo indicar que un elemento no pertenece a ningún espacio de nombres

Para indicar que determinados elementos –o todos– no pertenecen a ningún espacio de nombres, se escribe el atributo `xmlns` vacío, es decir:

`xmlns=""`

```
<?xml version="1.0" encoding="UTF-8"?>
<ejemplo xmlns="http://www.abrirllave.com/ejemplo1">

  <carta>
    <palo>Corazones</palo>
    <numero>7</numero>
  </carta>

  <carta xmlns="http://www.abrirllave.com/ejemplo2">
    <carnes>
      <filete_de_tenera precio="12.95"/>
      <solomillo_a_la_pimienta precio="13.60"/>
    </carnes>
    <pescados xmlns="">
      <lenguado_al_horno precio="16.20"/>
      <merluza_en_salsa_verde precio="15.85"/>
    </pescados>
  </carta>

</ejemplo>
```

PARSER XML

Un analizador XML es un procesador que lee un documento XML y determina la estructura y propiedades de los datos que contiene.

Si el analizador comprueba las reglas de buena formación y además valida el documento contra un DTD o esquema, se trata de un **analizador-validador**

Existen validadores XML en línea, como XML Validation

<http://www.xmlvalidation.com>

XML BIEN FORMADO

Se dice que un documento XML está bien formado si cumple las reglas establecidas por el W3C en las especificaciones para XML. Algunas de estas reglas son:

- El documento puede empezar por una instrucción de procesamiento xml que indica la versión del xml y, opcionalmente, el encoding y si requiere o no archivos externos standalone. Es recomendable, pero no obligatorio.
- Debe existir un único elemento raíz, que cuelga del nodo raíz(/). Este elemento tendrá como descendientes a todos los demás elementos
- Los elementos que no sean vacíos deben tener una etiqueta de apertura y otra de cierre
- Los elementos vacíos deben cerrarse con `</>` `<elemento/>` o `<elemento></elemento>`
- Los elementos deben aparecer correctamente anidados en cuanto a su apertura y su cierre, no solaparse
- Los nombres de etiquetas y atributos son sensibles a mayúsculas/minúsculas
- Los valores de los atributos deben aparecer entre comillas simples o dobles, pero del mismo tipo
- No puede haber dos atributos con el mismo nombre asociados a un mismo elemento
- No se pueden introducir ni instrucciones de procesamiento ni comentarios en ningún lugar del interior de las etiquetas de apertura o cierre de los elementos
- No puede haber nada antes de la instrucción de procesamiento xml
- No pueden aparecer los signos `<` ni `&` en el contenido textual de elementos ni atributos

DOCUMENTOS XML VÁLIDOS

Se dice que un documento XML es válido si existen unas reglas de validación asociadas a él, de manera que el documento deba cumplir con dichas reglas. Estas reglas especifican la estructura gramatical y sintaxis que debe tener el documento XML.

Es decir, además de estar bien formado cumple con las especificaciones y reglas establecidas para ese tipo de documento.





DTD

¿QUÉ ES UN DTD?

Un DTD (Document Type Definition) es una descripción de la estructura de un documento XML.

En un DTD se especifica qué elementos tienen que aparecer, en qué orden, cuáles son optativos, cuales obligatorios, qué atributos tienen los elementos, etc

El procesador XML utiliza la DTD para verificar si un documento cumple las reglas del DTD, en definitiva, certifican la calidad de los documentos XML. Esto conlleva resultados más fiables y mayor sencillez a la hora de programar las aplicaciones que procesan los documentos, ya que se trabaja a partir de una base de seguridad y no será necesario contemplar ciertas situaciones de error.

```
<!ELEMENT Casas_Rurales (Casa)*>  
<!ELEMENT Casa (Dirección, Descripción, Estado, Tamaño)>  
<!ELEMENT Dirección (#PCDATA) >  
<!ELEMENT Descripción (#PCDATA) >  
<!ELEMENT Estado (#PCDATA) >  
<!ELEMENT Tamaño (#PCDATA) >
```


ESTRUCTURA DE UN DTD

Declaración DTD

<!DOCTYPE>

Es la instrucción dónde se indica qué DTD validará el XML. Aparece al comienzo del documento XML. El primer dato que aparece es el nombre de elemento raíz del documento XML (el nombre del DOCTYPE debe coincidir con el elemento raíz)

Según la declaración donde se indica la ubicación de este elemento se distingue entre, interno, externo o mixto y si es privado o público.

- ❖ **Interno**, las declaraciones de marcado van dentro del propio XML.
- ❖ **Externo**, las declaraciones de marcado van en un fichero externo. Implica *standalone="no"* en la definición del xml `<?xmlversión="1.0"encoding="iso-8859-1"standalone="no"?>`. El dtd se puede aplicar a varios xml.
(el atributo standalone puede tomar dos valores: **yes** En caso de el documento XML no utilice DTD externa. **no** Cuando el documento obligatoriamente hace uso de DTD externa)
- ❖ **Mixto**: Se complementan y si hay contradicciones prevalecen las del interno.

Sintaxis	Tipo de DTD
<!DOCTYPE elemento_raíz [reglas]>	Interno (privado)
<!DOCTYPE elemento_raíz SYSTEM URL>	Externo y privado
<!DOCTYPE elemento_raíz SYSTEM URL [reglas]>	Mixto y privado
<!DOCTYPE elemento_raíz PUBLIC FPI URL>	Externo y público
<!DOCTYPE elemento_raíz PUBLIC FPI URL [reglas]>	Mixto y público

La combinación interno y público no tiene sentido, ya que el hecho de ser público fuerza a que el DTD esté en un documento independiente.

La ruta puede ser absoluta y entonces se indica su URL: <!DOCTYPE raíz SYSTEM "http://www.empresa.com/docs.dtd">

Pero puede ser relativa: <!DOCTYPE raíz SYSTEM " docs.dtd">

El FPI es un identificador público formal (Formal Public Identifier)

EJEMPLOS

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE Casas_Rurales [
  <!ELEMENT Casas_Rurales (Casa)*>
    <!ELEMENT Casa (Dirección, Descripción, Estado, Tamaño)>
    <!ELEMENT Dirección (#PCDATA) >
    <!ELEMENT Descripción (#PCDATA) >
    <!ELEMENT Estado (#PCDATA) >
    <!ELEMENT Tamaño (#PCDATA) >
]>
<Casas_Rurales>
  <Casa>
```

Interno

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<!DOCTYPE Casas_Rurales SYSTEM
"http://www.casasrurales.com/casasrurales.dtd">
<Casas_Rurales>
  ...
</Casas_Rurales>
```

```
<!ELEMENT Casas_Rurales (Casa)*>
<!ELEMENT Casa (Dirección, Descripción, Estado, Tamaño)>
<!ELEMENT Dirección (#PCDATA) >
<!ELEMENT Descripción (#PCDATA) >
<!ELEMENT Estado (#PCDATA) >
<!ELEMENT Tamaño (#PCDATA) >
```

Externo

```
<?xml version="1.0" standalone="no" ?>
<!DOCTYPE document SYSTEM "subjects.dtd" [
  <!ATTLIST assessment assessment_type (exam | assignment | prac)>
]>
```

SUBJETC.DTD

```
<!ELEMENT document
(title*,subjectID,subjectname,classes,assessment)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT subjectID (#PCDATA)>
<!ELEMENT subjectname (#PCDATA)>
<!ELEMENT classes (#PCDATA)>
<!ELEMENT assessment (#PCDATA)>
<!ATTLIST assessment assessment_type (exam | assignment)
```

Mixto

- **Componentes del DTD**

- Elemento
- Atributo
- Entidad
- Notación

<!ELEMENT> elementos

Indica la existencia de un elemento en el documento XML

Sintaxis:

```
<!ELEMENT nombre-del-elemento tipo-de-contenido>
```

El `nombre-del-elemento` es el identificador que tendrá el elemento en el documento XML (hay que recordar que distingue entre mayúsculas y minúsculas).

El `tipo-de-contenido` indica el funcionamiento del elemento, relativo al contenido que puede tener.

Los tipos de contenido en los elementos pueden ser:

- ✓ EMPTY
- ✓ ANY
- ✓ elemento concreto
- ✓ secuencias
- ✓ elecciones
- ✓ combinaciones

A tener en cuenta:

- No existe ninguna separación entre "<!" y "ELEMENT", lo contrario implica un error de formación.
- Los tipos de elementos se declaran de uno en uno, es decir, las Declaraciones de Tipos de Elementos son individuales.
- No es posible tampoco un tipo de elemento dos veces, la recomendación de XML considera este hecho como un error de buena formación.

✓ *EMPTY*

Significa que el elemento no podrá tener contenido alguno, es un elemento vacío

`<!ELEMENT nombre_elemento EMPTY>`

Un elemento vacío no puede tener contenido entre la etiqueta inicial y la etiqueta de cierre.

En el xml solo pueden utilizarse así:

`<nombre-elemento></nombre-elemento>` o `<nombre-elemento/>`

✓ *ANY*

Permite cualquier contenido en el elemento, sin restricciones de ningún tipo. Es decir, puede contener texto, otro tipo de datos y cualquier etiqueta. Además, puede tener atributos.

```
<!ELEMENT nombre_elemento ANY>
```

Puesto que un DTD se usa para restringir la escritura de un tipo de documentos XML, el uso de ANY debe de ser muy cauteloso ya que no restringe nada.

✓ *ELEMENTO CONCRETO*

En los elementos se puede indicar claramente un contenido concreto para el mismo. Dicho contenido se indica entre paréntesis. Este contenido puede ser:

- Un elemento uno y sólo uno, que deberá ser declarado después. Sintaxis:

```
<!ELEMENT nombre_elemento (elemento_hijo)>
```

- La indicación #PCDATA significa que el elemento podrá contener texto literal (tan largo como se desee). Sintaxis:

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

PCDATA (Parsed Character Data). Indica que entre la etiqueta de apertura y cierre de ese elemento, se almacenarán caracteres como texto y serán analizados por el parser.

Los elementos declarados como de tipo PCDATA no pueden contener los caracteres siguientes:

- ✓ El carácter [**<**], porque se interpretará como el comienzo de un nuevo elemento y el tipo PCDATA no puede contener etiquetas, ni elementos.
- ✓ El carácter [**&**], porque se interpretará siempre como el comienzo de una entidad.
- ✓ La cadena [**]]>**], porque marca el final de una sección CDATA.

Si se desean incluir estos caracteres, para que no se interpreten erróneamente, se hace como en el xml. A parte de estas restricciones obligatorias no existe forma de limitar el valor de un elemento PCDATA a un determinado grupo de caracteres.

✓ *SECUENCIAS*

Para especificar que el elemento se compone de una secuencia de elementos, se añaden estos como una lista de los elementos que puede contener separados por comas

```
<!ELEMENT nombre_elemento (elemento1,...,elementoN)>
```

De esta forma indicamos que debe haber todos y cada uno de los elementos que aparecen y en ese orden.

✓ *ELECCIONES O ALTERNATIVAS*

Para especificar una alternativa de elementos, en lugar de comas se utiliza como separador la barra vertical [|]. En este caso se especifica que únicamente un elemento de la lista puede formar parte del contenido en cada realización de dicho elemento.

```
<!ELEMENT nombre_elemento (elemento1 | ... | elementoN)>
```

✓ *COMBINACION DE TIPOS*

Se pueden utilizar paréntesis para agrupar secuencias y alternativas de modelos.

```
<!ELEMENT nombre_elemento (elemento1 | (elemento2,elemento3))>
```

El tipo #PCDATA también puede formar parte de esta combinación, en cuyo caso debe aparecer el primero.

✓ *CARDINALIDAD O FRECUENCIA*

Se puede definir el número de veces que puede aparecer un determinado contenido en un elemento.

Se realiza mediante estos símbolos:

Símbolo	Significado
?	El elemento o secuencia de elementos puede aparecer 0 o 1 vez (contenido opcional)
*	El elemento o secuencia de elementos puede aparecer de 0 a N veces (opcional y repetible)
+	El elemento o secuencia de elementos puede aparecer de 1 a N veces (obligatorio y repetible)

Ejemplo : <!ELEMENT película (título, dirección+, argumento?, actor*)>

El elemento película consta de un título, uno o más elementos de dirección, puede o no tener argumento, y de varios a ningún actor (además se tendría que respetar ese orden).

<!ATTLIST atributos

Los atributos permiten añadir información a los elementos. En general se utiliza un solo ATTLIST para declarar todos los atributos de un elemento, pero podría utilizarse un ATTLIST para cada atributo.

Sintaxis:

```
<!ATTLIST elemento
```

```
    nombre tipo tratamiento_por_defecto
```

```
    nombre tipo tratamiento_por_defecto
```

```
    ...
```

```
>
```

- **elemento**: es el nombre del elemento al que corresponden los atributos
- **nombre**: es el nombre del atributo
- **tipo**: CDATA, (valor | valor | ...), ID, IDREF, IDREFS, NMTOKEN, NMTOKENS
- **tratamiento_por_defecto**: #REQUIRED, #IMPLIED, #FIXED valor_por_defecto, valor_por_defecto

`<!ATTLIST nombre_elemento nombre_atributo tipo presencia/valorPorDefecto>`

Los atributos son sencillos de declarar y de utilizar, pero hay que tener en cuenta que:

- Entre "<!" y "ATTLIST" no hay ninguna separación.
- El orden de las declaraciones es como se especifica. Cualquier alteración en este sentido será causa de errores.
- Los valores de los atributos por defecto deben delimitarse por comillas dobles o simples.
- Son case-sensitive. Los nombres de los atributos empleados en la declaración deben ser los mismos que luego reciban los valores en los elementos.
- El número de atributos que se declaran para un elemento puede ser tan grande como se desee. XML no impone ninguna restricción en este sentido.
- El orden en que se especifican los atributos para un elemento en concreto es irrelevante y no viene impuesto por el orden definido en la declaración ni por ningún otro factor.
- Un atributo no puede constar de más atributos y cada atributo sólo puede aparecer una vez en cada elemento.

```
<!ATTLIST elemento
  nombre tipo tratamiento_por_defecto
  nombre tipo tratamiento_por_defecto
  ...
>
```

- **elemento:** es el nombre del elemento al que corresponden los atributos
- **nombre:** es el nombre del atributo
- **tipo:** CDATA, (valor | valor | ...), ID, IDREF, IDREFS, NMTOKEN, NMTOKENS
- **tratamiento_por_defecto:** #REQUIRED, #IMPLIED, #FIXED valor_por_defecto, valor_por_defecto

✓ **TRATAMIENTO POR DEFECTO**

Valor por defecto concreto: Si al final de la declaración de un atributo aparece un valor concreto, se entiende que ese será el valor por defecto. Es decir que se podría no utilizar el atributo en un elemento y entonces dicho atributo tomaría ese valor.

Valores fijos: Se puede utilizar el término **#FIXED** antes de indicar el valor por defecto de un atributo. En ese caso en ningún documento XML que utilice este DTD se podrá modificar dicho atributo. El atributo es obligatorio y se le asigna un valor por defecto que, además, es el único valor que puede tener el atributo

Valores requeridos: En este caso se usa la palabra **#REQUIRED** indicando con ello que obligatoriamente hay que asignársele algún valor en el documento XML.

Valores opcionales: La palabra **#IMPLIED** especificada en el atributo indicaría que dicho atributo puede quedarse sin valor y no posee valor por defecto, puede quedarse sin especificar en el XML

✓ TIPOS DE ATRIBUTOS

CDATA caracteres que no contienen etiquetas, es el tipo más general.

ID identificador único. Un elemento puede tener solo un atributo de tipo ID

IDREF el atributo contendrá el nombre de un ID de otro elemento. Para que sea válido, debe existir otro elemento en el documento XML que tenga un atributo de tipo ID y cuyo valor sea el mismo que el del atributo de tipo IDREF del primer elemento.

IDREFS representa múltiples IDs de otros elementos separados por espacios. Es igual que el anterior sólo que permite indicar varias referencias (que deben existir en el documento XML) a otros ID, separadas por espacios.

NMTOKEN cualquier nombre sin espacios en blanco en su interior. Los espacios en blanco anteriores o posteriores se ignoran. Análogos a los de tipo CDATA, pero con la diferencia de que sólo puede contener letras.

NMTOKENS una lista de nombres, sin espacios en blanco en su interior, separados por espacios

```
<!ATTLIST elemento
    nombre tipo tratamiento_por_defecto
    nombre tipo tratamiento_por_defecto
    ...
>
```

- **elemento**: es el nombre del elemento al que corresponden los atributos
- **nombre**: es el nombre del atributo
- **tipo**: CDATA, (valor | valor | ...), ID, IDREF, IDREFS, NMTOKEN, NMTOKENS
- **tratamiento_por_defecto**: #REQUIRED, #IMPLIED, #FIXED valor_por_defecto, valor_por_defecto

<!ENTITY> entidad

Las entidades son elementos XML que permiten indicar abreviaturas de texto (o referencias a elementos externos abreviadas) o utilizar caracteres que de otra forma serían inválidos en el documento.

También permiten definir constantes de los documentos.

Hay diferentes tipos de entidades y, en función del tipo, la sintaxis varía.

Los tipos de entidades son:

- Entidades generales (internas o externas)

- Entidades parámetro (internas o externas)

- Entidades no procesadas (unparsed)

✓ **ENTIDADES GENERALES INTERNAS**

Se utilizan dentro del documento XML

Sintaxis:

```
<!ENTITY nombre_entidad definición_entidad>
```

Ejemplo:

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

En el XML se usa anteponiendo al nombre de la entidad el carácter ampersand y a continuación un carácter punto y coma. El programa analizador del documento realizará la sustitución

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

✓ **ENTIDADES GENERALES EXTERNAS**

Ubicadas en otros archivos

Sintaxis:

```
<!ENTITY nombre_entidad tipo_uso url_archivo>
```

Siendo el tipo de uso privado (SYSTEM) o público (PUBLIC)

Ejemplo:

Se dispone de un archivo de texto, autores txt, que contiene el siguiente texto plano “Juan Manuel y José Ramón” Se crea un documento XML que hace referencia a ese archivo de texto, en forma de entidad externa. Al visualizar el documento, la referencia a la entidad general externa se sustituirá por el texto contenido en el archivo

```
<!DOCTYPE escritores[  
    <!ELEMENT escritores (#PCDATA)>  
    <!ENTITY autores SYSTEM “autores txt”>  
>  
<escritores>&autores;</escritores>
```


✓ ENTIDADES PARAMETROS INTERNAS

Se usarán en el propio DTD (no se pueden usar en el XML) y funcionan cuando la definición de las reglas del DTD se realiza en un archivo externo.

Sintaxis:

```
<!ENTITY % nombre_entidad definición_entidad>
```

Ejemplo:

Se declara una entidad parámetro dimensiones y se referencia dentro del propio DTD. Los siguientes códigos son equivalentes

```
<!ENTITY % dimensiones "alto CDATA #IMPLIED  
    ancho CDATA #IMPLIED profundo CDATA #IMPLIED">  
<!ELEMENT objeto (nombre)>  
<!ATTLIST objeto  
    codigo ID #REQUIRED  
    %dimensiones;>
```

```
<!ELEMENT objeto (nombre)>  
<!ATTLIST objeto  
    codigo ID #REQUIRED  
    alto CDATA #IMPLIED  
    ancho CDATA #IMPLIED  
    profundo CDATA #IMPLIED>
```

✓ ENTIDADES PARAMETROS EXTERNAS

Permiten utilizar archivos externos, es decir crear una DTD compuesta por otras DTD.

Sintaxis:

```
<!ENTITY % nombre_entidad SYSTEM "URL">
```

✓ ENTIDADES NO PROCESADAS

Referencian a datos que no deben ser procesados por el analizador XML, sino por la aplicación que lo use.

Sintaxis:

```
<!ENTITY nombre_entidad tipo_uso fpi valor_entidad NDATA tipo>
```

Crear la Entidad No Procesada es como crear una entidad general externa añadiéndole al final NDATA y el nombre de una notación

Ejemplo:

- Se declara una notación de nombre JPG (describas a continuación)
- Se declara una entidad no procesada de nombre mediterraneo, asociada al archivo de imagen mediterraneo.jpg
- Se declara un elemento <mar> que cuenta con un atributo imagen que es del tipo ENTITY recién declarado
- En el XML, el valor del atributo imagen del elemento <mar> es la entidad no procesada declarada en el DTD

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

```
<mares>
  <mar imagen="mediterraneo">
    <nombre>Mediterráneo</nombre>
  </mar>
  ...
</mares>
```

<!NOTATION> notación

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).

Se usa para especificar un formato de datos que no sea XML. Se utiliza con frecuencia para describir tipos como image/gif o image/jpg

Cuando declaramos una notación, indica que el contenido del elemento debe ser procesado mediante las herramientas que indican en la propia la notación.

Es un elemento avanzado en el diseño de DTD.

```
<!NOTATION nombre_notation SYSTEM "identificador_externo">
```

Sintaxis general del atributo que la usa:

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