

XML

## ¿Qué problema tiene HTML?

- Define mas la presentación que su contenido
- No es fácilmente procesables por las maquinas
- Su estructura es caótica
- Su interpretación es ambigua dependiendo S.W utilizado
- Solo vale para paginas WEB

# XML ¿QUÉ ES?

XML es un subconjunto de SGML(Estándar Generalised Mark-up Language),simplificado y adaptado a Internet



XML **no** es ,como su nombre puede sugerir, un lenguaje de marcado.

XML es un meta-lenguaje que nos permite definir lenguajes de marcado adecuados a usos determinados

## **Que no es XML**

- No es una versión mejorada de HTML
- HTML es una aplicación de SGML por lo tanto de XML
- No es un lenguaje para hacer paginas WEB
- Y sobre todo no es difícil

## ¿Por qué XML?

- Es un estándar internacionalmente conocido
- No pertenece a ninguna compañía
- Permite una utilización efectiva en Internet para sus diferentes terminales.

## Definición

Especificación para diseñar lenguajes de marcado, que permite definir etiquetas personalizadas para descripción y organización de datos.

## ¿Para que sirve XML?

Representar información estructurada en la web (todos documentos), de modo que esta información pueda ser almacenada, transmitida, procesada, visualizada e impresa, por muy diversos tipos de aplicaciones y dispositivos.

## Ventajas de XML

- Fácilmente procesable
- Separa radicalmente el contenido y el formato de presentación
- Diseñado para cualquier lenguaje y alfabeto. (encoding)

## EDI

NM1\*WT\*1\*Smith\*John\*C.~N3\*610 E. Bel Aire Dr.\*Suite  
300~N4\*Burbank\*CA\*91503

## AML

<Witness for Defendant>

<Person>

<Last name>Smith</Last name>

<First name>John</First name>

<Middle name>C.</Middle name>

<address1>610 E. Bel Aire Dr.</address1>

<address2>Suite 300</address2>

<city>Burbank</city>

<state>CA</state>

<zip>91503< /zip>

<Person>

</Witness for Defendant>

## Documento XML

- ✓ Conjunto de datos con sus respectivas etiquetas de marcado XML.
- ✓ Se almacena como texto en archivo con extensión .xml.
- ✓ Un documento XML puede incluir cualquier flujo de datos basado en texto: un artículo de una revista, un resumen de cotizaciones de bolsa, un conjunto de registros de una base de datos, etc..

## Estructura de un documento XML

Un documento XML está formado por **datos de caracteres** y **marcado**, el marcado lo forman las etiquetas:

<u>Prologo</u>	{	<code>&lt;?xml version="1.0" encoding="ISO-8859-1" standalone="no"?&gt;</code>
		<code>&lt;! DOCTYPE persona SYSTEM "persona.dtd" &gt;</code>
<u>Cuerpo</u>	{	<code>&lt;persona&gt;</code>
		<code>    &lt;nombre&gt;Luis&lt;/nombre&gt;</code>
		<code>    &lt;apellidos&gt;Pérez&lt;/apellidos&gt;</code>
		<code>&lt;/persona&gt;</code>

## Estructura de cada línea



## Sintaxis de XML

Representa las normas a seguir para la construcción de documentos XML. Estas reglas son dictadas por el organismo W3C (<http://www.w3.org/XML>). Entre ellas destacan:

- El XML es Case - Sensitive.
- Todo elemento tiene que tener su correspondiente etiqueta de inicio y de cierre, o una sola etiqueta vacía.
- Todo documento, debe haber un elemento (llamado raíz de documento) que contenga a los demás
- Todos los elementos deberán estar correctamente anidados.
- Todos los valores de los atributos deberán ir entre comillas.
- Normas de buena construcción
  - La primera letra de los nombre se escribirá en mayúscula
  - Los nombres compuestos se escribirán juntos o separados por guion bajo
  - Los elementos han de comenzar por un carácter o “\_” no numérico

## Normas de buena construcción II

Existen 2 tipos de construcciones

- Orientado a la presentación (No recomendable)

```
<Poema> <Negrita>El reino perdido</Negrita>  
<Cursiva>Las huestes de don Rodrigo desmayan y <Negrita> huían  
</Negrita> ..... <Cursiva> .....  
.....</Poema>
```

- No orientado a la presentación (recomendable)

```
<Poema> <Título>El reino perdido</Título>  
<Cuerpo>Las huestes de don rodrigo desmayan y huían.....  
<Cuerpo> ..... <Comentario> buen poema de.. <Comentario>  
.....</Poema>
```



# Componentes de un documento XML

En un documento XML existen los siguientes componentes:

**Elementos:** Pieza lógica del marcado, se representa con una cadena de texto(dato) encerrada entre etiquetas. Pueden existir elementos vacíos (<br/>). Los elementos pueden contener atributos.

**Instrucciones:** Ordenes especiales para ser utilizadas por la aplicación que procesa  
<?xml-stylesheet type="text/css" href="estilo.css">

**Las instrucciones XML.** Comienzan por <? Y terminan por ?>.

**Comentarios:** Información que no forma parte del documento. Comienzan por <!-- y terminan por -->. No se pueden escribir comentarios dentro de las etiquetas

**Declaraciones de tipo:** Especifican información acerca del documento:  
<!DOCTYPE persona SYSTEM "persona.dtd">

**Secciones CDATA:** Se trata de un conjunto de caracteres que no deben ser interpretados por el procesador:  
<![CDATA[ Aquí se puede meter cualquier carácter, como <, &, >, ... Sin que sean interpretados como marcación]]>

## EJEMPLO

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE peliculas SYSTEM "Peliculas.dtd">

<peliculas>
  <pelicula tipo="comedia" sistema="PG-13" ejemplares="5" año="1987">
    <titulo>Raising Arizona</titulo>
    <guionista>Ethan Coen</guionista>
    <guionista>Joel Coen</guionista>
    <productor>Ethan Coen</productor>
    <director>Joel Coen</director>
    <actor>Nicolas Cage</actor>
    <actor>Holly Hunter</actor>
    <actor>John Goodman</actor>
    <comentarios>Una pelicula clasica de Comedia.</comentarios>
  </pelicula>

  <pelicula tipo="ciencia-ficcion" sistema="PG-13" ejemplares="4" año="1989">
    <titulo>The Abyss</titulo>
    <guionista>James Cameron</guionista>
    <productor>Gale Anne Hurd</productor>
    <director>James Cameron</director>
    <actor>Ed Harris</actor>
    <actor>Mary Elizabeth Mastrantonio</actor>
    <comentarios>Una buena pelicula.</comentarios>
  </pelicula>
</peliculas>
```

## Elementos

Los documentos XML están formados por texto plano (sin formato) y contienen marcas (etiquetas) definidas por el desarrollador.

```
<nombre>Elsa</nombre>
```

### Sintaxis:

```
<etiqueta>valor</etiqueta>
```

Un elemento puede **no** contener ningún valor.

```
<etiqueta></etiqueta>
```

```
<etiqueta/>
```

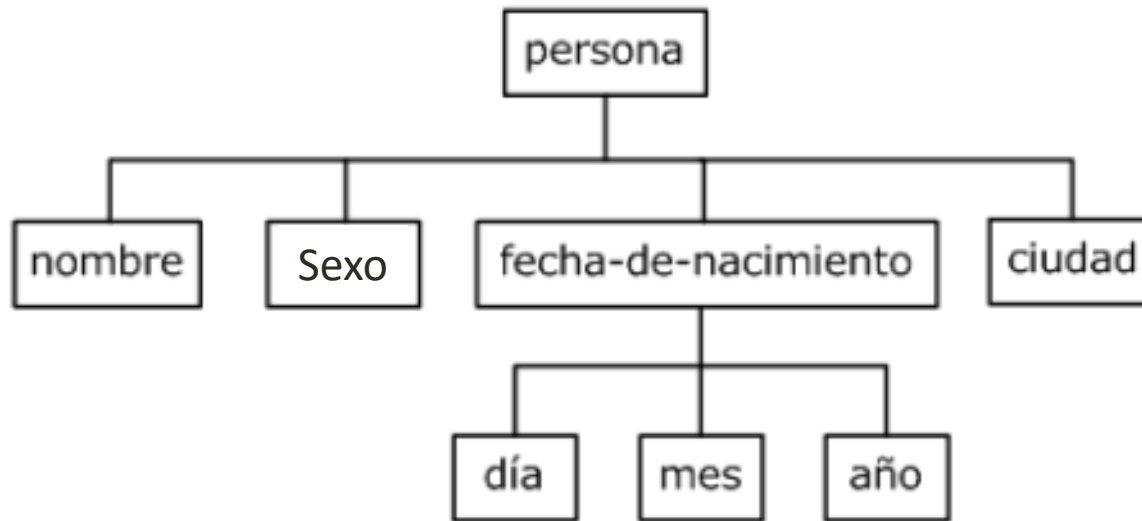
### • EJEMPLO

```
<nombre></nombre>
```

```
<nombre/>
```

## Elemento raíz de un documento XML

Todo documento XML tiene que tener **un único elemento raíz** (padre) del que desciendan todos los demás.



Los elementos son los que dan estructura semántica al documento.

## Relaciones padre-hijo entre elementos

Un elemento (padre) puede contener a otro u otros elementos (hijos).

<persona>

<nombre/>

<Sexo/>

<fecha-de-nacimiento>

<día/>

<mes/>

<año/>

</fecha-de-nacimiento>

<ciudad/>

</persona>

<persona>

<nombre>Elsa</nombre>

<sexo>Mujer</sexo>

<fecha-de-nacimiento>

<día>18</día>

<mes>6</mes>

<año>1996</año>

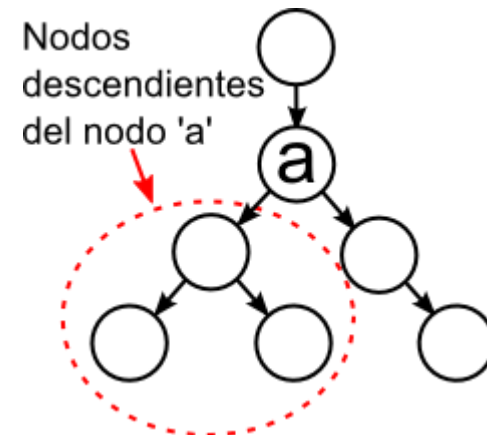
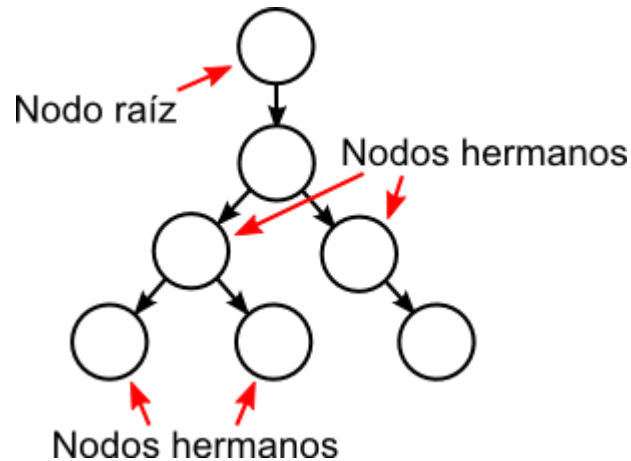
</fecha-de-nacimiento>

<ciudad>Pamplona</ciudad>

</persona>

## Relaciones padre-hijo entre elementos

Los documentos XML se estructuran de forma **jerárquica** en la que los elementos se relacionan entre sí mediante relaciones de padres, hijos, hermanos, ascendentes, descendentes, etc. es lo que se conoce como el **árbol del documento XML**.



## Elementos con contenido mixto

Un elemento puede contener contenido mixto, es decir, texto y otros elementos

```
<persona>  
    <nombre>Elsa</nombre> vive en <ciudad>Pamplona</ciudad>.  
</persona>
```

El elemento “persona” contiene los elementos “nombre” y “ciudad”, además de los textos " **vive en** "y "."

## Normas de sintaxis básicas

- Todos los nombres de los elementos son **case sensitive**.
- Pueden contener letras minúsculas, letras mayúsculas, números, *puntos“.”*, *guiones medios“-”* y *guiones bajos“\_”*.
- Pueden contener el carácter *dos puntos“:”*. No obstante, su uso se reserva para cuando se definan espacios de nombres.
- El primer carácter tiene que ser una letra o un *guion bajo“\_”*.
- **Detrás** del nombre de una etiqueta se permite escribir **un** espacio en blanco o un salto de línea.

```
<ciudad >Pamplona</ciudad>
```

- No puede haber un salto de línea o un espacio en blanco **antes** del nombre de una etiqueta.

```
<ciudad>Pamplona</ ciudad>
```



## EJEMPLOS

### Elementos escritos incorrectamente

<Ciudad>Pamplona</ciudad>

<día>18</dia>

<mes>6<mes/>

<ciudad>Pamplona</finciudad>

<\_rojo>

<2colores>Rojo y Naranja</2colores>

< Aficiones >Cine, Bailar, Nadar</ Aficiones >

<persona><nombre>Elsa</persona></nombre>

<color favorito>azul</color favorito>

# EJEMPLOS

## Elementos escritos correctamente

<Ciudad>Pamplona</Ciudad>

<día>18</día>

<mes>6</mes>

<ciudad>Pamplona</ciudad>

<\_rojo/>

<colores2>Rojo y Naranja</colores2>

<Aficiones >Cine, Bailar, Nadar</Aficiones >

<persona><nombre>Elsa</nombre></persona>

<color.favorito>azul</color.favorito>

<color-favorito>azul</color-favorito>

<color\_favorito>azul</color\_favorito>

## Atributos

- Un atributo proporciona información extra del elemento que lo contiene.
- Los atributos son pares nombre-valor que se sitúan en la etiqueta de apertura del elemento

```
<producto codigo="G45">  
    <nombre color="negro" precio="12.56">Gorro de lana</nombre>  
</producto>
```

```
<producto>  
    <codigo>G45</codigo>  
    <nombre color="negro" precio="12.56">Gorro de lana</nombre>  
</producto>
```

- Los valores de los atributos van entre comillas dobles (") o simples (').
- Los nombres de los atributos deben cumplir las mismas normas de sintaxis que los nombres de los elementos.
- Además, todos los atributos de un elemento tienen que ser únicos. Por ejemplo, es incorrecto escribir:

```
<datos x="3" x="4" y="5"/>
```

- Sí es correcto escribir:

```
<datos x="3" X="4" y="5"/>
```

# Atributos vs Elementos

En muchas ocasiones la información se podrá almacenar como elemento o como atributo.

**Se recomienda** utilizar elementos siempre que sea posible teniendo en cuenta las siguientes propiedades:

## ➤ Elementos

- Se emplean para representar jerarquías.
- Se pueden extender con otros elementos.
- El orden en que aparecen es representativo.
- Pueden tener atributos.
- Puede haber múltiples ocurrencias de un elemento

## ➤ Atributos

- Son modificadores de la información.
- Van asociados a los elementos.
- No se pueden extender con otros elementos o atributos.
- El orden en que aparecen dentro del elemento no es representativo.
- No puede haber múltiples ocurrencias de un atributo dentro del mismo elemento.

# Elementos: Contenido

Los elementos pueden contener texto, otro elemento o estar vacíos.

- ✓ **Otro elemento:** los elementos se anidan unos dentro de otros formando una estructura **jerárquica**. Como ocurre en HTML las etiquetas deberán cerrarse en **orden inverso** al que se abrieron.
- ✓ **Texto:** representa información de un elemento y puede contener cualquier carácter aunque para algunos hay que utilizar su entidad

Referencias a entidades en XML		
Carácter	Entidad	Referencia a entidad
< (menor que)	<b>lt</b> ( <i>less than</i> )	<b>&amp;lt;</b>
> (mayor que)	<b>gt</b> ( <i>greater than</i> )	<b>&amp;gt;</b>
" (comilla doble)	<b>quot</b> ( <i>quotation mark</i> )	<b>&amp;quot;</b>
' (comilla simple)	<b>apos</b> ( <i>apostrophe</i> )	<b>&amp;apos;</b>
& (ampersand)	<b>amp</b> ( <i>ampersand</i> )	<b>&amp;amp;</b>

El Euro (€):

- valor decimal →&#8364;
- valor hexadecimal &#x20AC;

# Elementos: Contenido

Referencias a entidades en XML		
Carácter	Entidad	Referencia a entidad
< ( <i>menor que</i> )	<b>lt</b> ( <i>less than</i> )	<b>&amp;lt;</b>
> ( <i>mayor que</i> )	<b>gt</b> ( <i>greater than</i> )	<b>&amp;gt;</b>
" ( <i>comilla doble</i> )	<b>quot</b> ( <i>quotation mark</i> )	<b>&amp;quot;</b>
' ( <i>comilla simple</i> )	<b>apos</b> ( <i>apostrophe</i> )	<b>&amp;apos;</b>
& ( <i>ampersand</i> )	<b>amp</b> ( <i>ampersand</i> )	<b>&amp;amp;</b>

[elementos.xml](#)

```
<?xml version="1.0" encoding="UTF-8"?> <entidades>
<menor_que>&lt;</menor_que>
<mayor_que>&gt;</mayor_que>
<comilla_doble>&quot;</comilla_doble>
<comilla_simple>&apos;</comilla_simple>
<ampersand>&amp;</ampersand> </entidades>
```

# Declaración XML

La declaración XML no es una instrucción de procesamiento (o proceso).

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

En un documento XML no es obligatorio que aparezca la declaración XML.

Si se incluye, tiene que aparecer en la primera línea del documento, y el carácter “<” debe ser el primero de dicha línea.

- version:** indica la versión de XML. Aunque existe la versión 1.1, la más extendida sigue siendo la 1.0.
- encoding:** establece la codificación de caracteres, siendo utf-8 la más extendida.
- standalone:** indica si el documento está listo para procesarse independientemente o requiere de archivos externos, por defecto es "no".

```
<?xml version="1.0" encoding="UTF-8"?>
<persona profesion="cantante">
  <nombre>Elsa</nombre>
  <mujer/>
  <fecha_de_nacimiento>
    <dia>18</dia>
    <mes>6</mes>
    <año>1996</año>
  </fecha_de_nacimiento>
  <ciudad>Pamplona</ciudad>
</persona>
```

## Ejercicios

Escribir un documento XML que almacene la siguiente información:

CIUDADES		
Nombre	País	Continente
Nueva Delhi	India	Ásia
Lisboa	Portugal	Europa
El Cairo	Egipto	África
Madrid	España	Europa
Toronto	Canada	América

[Países 1](#)

[Países 2](#)



## Ejercicios

Escribir un documento XML que almacene la siguiente información:

HECHOS HISTÓRICOS			
Descripción de cada hecho	Fecha		
	Día	Mes	Año
IBM da a conocer el PC	12	8	1981
Se funda Google	4	9	1998
Se funda Facebook.	4	2	2004

Hecho Histórico

# Ejercicios

## Corregir errores de sintaxis

```
<?xml version="1.0" encoding="UTF-8"?>
<figuras>
  <figura tipo="plana">
    <nombre>cuadrado</nombre>
    <lados>4</lados>
  </figura>
  <figura tipo="plana">
    <nombre>triángulo</nombre>
    <lados>3</lados>
  </figura>
  <figura tipo="tridimensional">
    <nombre>cubo</nombre>
    <aristas>12</aristas>
    <caras>6</caras>
  </figura>
</figuras>
```

## Ejercicios

### Corregir errores de sintaxis

```
<?xml version="1.0" encoding="UTF-8"?>
<triangulos>
  <triangulo base="7" altura="5"/>
  <triangulo base="2" altura="6"/>
  <triangulo base="3" altura="3"/>
</triangulos>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<numeros>
  <numero1 letra="u" letra2="n" letra3="o">1</numero1>
  <numero2 letra="d" letra2="o" letra3="s">22</numero2>
  <numero6 letra="s" letra2="e" letra3="i" letra4="s">666666</numero6>
</numeros>
```

# Instrucciones de Procesamiento

Las instrucciones de procesamiento no forman parte del contenido del documento pero proporcionan información a las aplicaciones que procesan el XML.

- ✓ **Declaración:** es obligatoria. Se utiliza para indicar la versión de XML y otras características.
- ✓ **Visualización:** XML no dispone de una visualización concreta en el navegador puesto que refleja datos y no presentación, pero podemos establecer la forma de representar el documento mediante una hoja de estilos (CSS) o una hoja de transformaciones (XSLT).

Las instrucciones de proceso se escriben empezando con la pareja de caracteres "<?" y finalizando con "?>".

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


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

# Instrucciones de Procesamiento

## Ejemplo

### animales.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="estilo-animales.css"?>
<animales>
  <animal>
    <nombre>perro</nombre>
    <patas>4</patas>
  </animal>
  <animal>
    <nombre>pato</nombre>
    <patas>2</patas>
  </animal>
  <animal>
    <nombre>ballena</nombre>
    <patas>0</patas>
  </animal>
</animales>
```

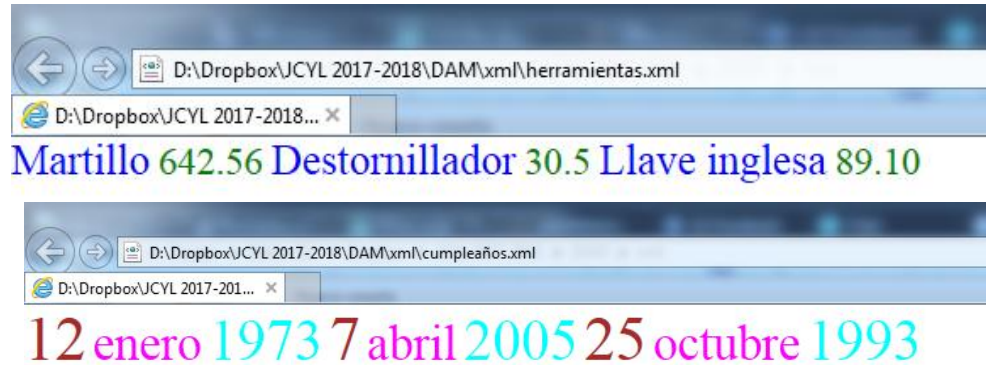


### **estilo-animales.css**

```
nombre{color:blue;font-size:40px}
patas{color:red;font-size:22px}
```

# Ejercicios

Escribir un documento XML y su hoja de estilo para obtener:



Dado el archivo "***estilo-colores.css***", cuyo contenido es:

```
color{color:grey;font-size:20px}
```

## Representar

MEZCLAS DE COLORES	
<i>Mezcla</i>	<i>Resultado</i>
rojo y amarillo	naranja
negro y blanco	gris
amarillo, azul y blanco	verde claro

## Elementos de Contenido CDATA

Las secciones **CDATA** (*Character DATA*) sirven para para escribir texto que no se desea que sea analizado.

Se escribe comenzando con la cadena de caracteres "<![CDATA[" y terminando con los caracteres "]]>".

Dentro de una sección CDATA no se puede escribir la cadena "]]>". En consecuencia, **no se pueden anidar secciones CDATA**.

<![CDATA[*contenido*]]>

```
<?xml version="1.0" encoding="UTF-8"?>
<ejemplo_CDATA>
<![CDATA[
#include <stdio.h>
int main() {
printf( "\n Hola mundo " );
}
]]>
</ejemplo_CDATA>
```

## Espacios de nombres

Utilizamos los espacios de nombres cuando queremos juntar dos o mas documentos XML y dichos documentos comparten ciertos elementos.

### Naipes.xml

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

### Restaurante.xml

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



## Espacios de nombres

Al incluir ambos elementos **<carta>** en un documento XML, se origina un conflicto de nombres.

Para resolverlo, se pueden utilizar **espacios de nombres** (*XML Namespaces*)

Indicaremos al documento XML que vamos a usar el espacio de nombres mediante una línea **xmlns**

Para definir un espacio de nombres se utiliza la siguiente sintaxis:

**xmlns:prefijo="URI"**

Para quitarnos de problemas en vez de una URI usaremos una URL

Usaremos una URL ya que son únicas en Internet

## Espacios de nombres

Restaurante.xml

```
<?xml version="1.0" encoding="UTF-8"?>
  <e1:ejemplo xmlns:e1="http://www.miweb.com/naipes"
    xmlns:e2="http://www.miweb.com/restaurante">
    <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>
```

## Espacios de nombres

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

`http://www.miweb.com/naipes`

`http://www.miweb.com/restaurante`

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

Los prefijos definidos son **e1** y **e2**, respectivamente, añadido a dichos prefijos a las etiquetas que aparecen en el documento:

**`<e1:carta>`, `<e2:carta>`, `<e1:palo>`, etc.**

hacen que el elemento pertenezca al espacio de nombres del prefijo al que hacen referencia

## Documentos bien formados y válidos

Se dice que un documento XML está bien formado (*well-formed document*) cuando no tiene errores de sintaxis. Esto incluye los siguientes aspectos:

- Los nombres de los elementos y sus atributos deben estar escritos correctamente.
- Los valores de los atributos deben estar escritos entre comillas dobles o simples.
- Los atributos de un elemento deben separarse con espacios en blanco.
- Se tienen que utilizar referencias a entidades donde sea necesario.
- Tiene que existir un único elemento raíz.
- Todo elemento debe tener un elemento padre, excepto el elemento raíz.
- Todos los elementos deben tener una etiqueta de apertura y otra de cierre.
- Las etiquetas deben estar correctamente anidadas.
- Las instrucciones de proceso deben estar escritas de forma correcta.
- La declaración XML debe estar en la primera línea escrita correctamente.
- Las secciones CDATA y los comentarios deben estar correctamente escritos

Por otro lado, se dice que **un documento XML es válido (*valid*)** cuando, además **de no tener errores de sintaxis, no incumple ninguna de las normas establecidas en su estructura**. Dicha estructura se puede definir utilizando distintos métodos, tales como:

**DTD** (*Document Type Definition*, Definición de Tipo de Documento).

**XML Schema**.

**RELAX NG** (*REgular LAnguage for XML Next Generation*).

Se pueden validar ficheros XML de varias formas:

- Online:
  - <https://www.xmlvalidation.com/>
  - [https://www.w3schools.com/xml/xml\\_validator.asp](https://www.w3schools.com/xml/xml_validator.asp)
- Mediante Programas

# DTD

**DTD** es una manera de describir de forma concisa el lenguaje XML.

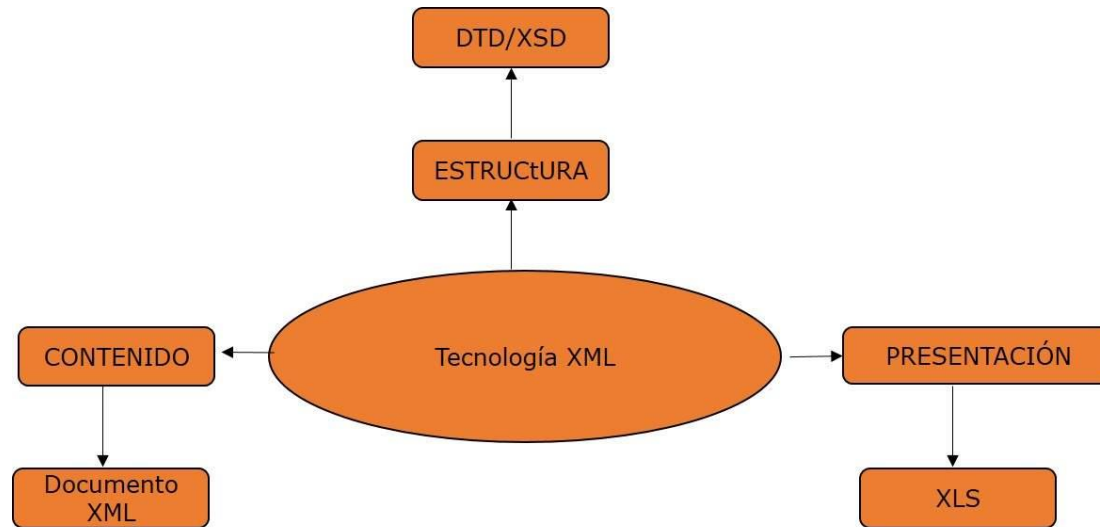
Los **DTDs** evalúan la validez de la estructura y del vocabulario de un documento XML en contraste con las reglas gramaticales del lenguaje XML apropiado.

Recordemos que un documento **XML** se puede definir como:

**Bien formado:** Si el documento XML se ajusta a las reglas generales de XML como la de que las etiquetas se deben encajar adecuadamente, se debe equilibrar la apertura y el cierre de etiquetas, y las etiquetas vacías deben terminar en `'/>'`.

**Válido:** Un documento XML es válido no solamente cuando está *bien formado*, sino también cuando conforma un DTD disponible que especifica qué etiquetas usa, qué atributos pueden contener esas etiquetas, y qué etiquetas se pueden formar dentro de otras, entre otras propiedades.

# DTD



## Tipos

El DTD se puede clasificar en base a sus declaraciones en el documento XML document, en:

- **DTD Interno**     DTD declarado en el archivo XML
- **DTD Externo**     DTD declarado en un archivo independiente alXML

# DTD

## Características

Puntos importantes que un **DTD** describe:

- los elementos que pueden aparecer en un documento XML y su orden.
- elementos obligatorios y opcionales.
- atributos del elemento sean opcionales o obligatorios.
- si los atributos pueden tener valores predeterminados.

## Ventajas del uso de DTD

**Documentación** - Podemos definir nuestro propio formato para los archivos XML.

**Validación** - Aporta una modalidad para evaluar la validez de archivos XML examinando si cada elemento del **XML** cumple con la especificación dada.



# DTD

## Desventajas del uso de DTD

Puntos importantes que un DTD describe:

- No es compatible con el espacio de nombres.
- Solamente es compatible con el *tipo de dato de texto en cadena*.
- No está orientado al objeto. Por consiguiente, el concepto de herencia no se puede aplicar en los DTDs.
- Posibilidades limitadas de expresar la cardinalidad para los elementos

## DTD Interno

Para referenciar un XML como DTD interno, el atributo ***standalone*** en la declaración del archivo se debe marcar con un **Si**. Esto significa que la declaración funciona al margen de la fuente externa.

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

### Sintaxis

La definición de un **DTD interno** es como sigue:

```
<!DOCTYPE root-element [element-declarations]>
```

donde *root-element* es el nombre del elemento raíz y *element-declarations* es donde se declaran los elementos.

# DTD Interno

Declaración

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

DTD

```
<!DOCTYPE address [ <!ELEMENT address (name,company,phone)>
    <!ELEMENT name (#PCDATA)>
    <!ELEMENT company (#PCDATA)>
    <!ELEMENT phone (#PCDATA)>
```

> Fin de la declaración

```
<address>
    <name>IES San Andres</name>
    <company>Instituto</company>
    <phone> (+34)987846315 </phone>
</address>
```

# DTD Interno

## Normas

La declaración del tipo de documento debe aparecer al inicio del documento (*solamente precedido por un encabezado XML*) - No está permitido en ningún otro lugar del documento.

De forma similar a la declaración de DOCTYPE, las declaraciones de elemento deben empezar con un signo de exclamación (!), esto indica al analizador que es un DTD asociado al XML

El nombre en la declaración del tipo de documento debe coincidir con el tipo de elemento del elemento raíz.

La declaración finaliza con ]>

# DTD Interno

## Cuerpo del DTD

Después de la declaración DOCTYPE viene el cuerpo del DTD, donde se declaran los elementos, atributos, entidades, y anotaciones en orden según la estructura deseada:

```
<!ELEMENT address (name,company,phone)>  
<!ELEMENT name (#PCDATA)>  
<!ELEMENT company (#PCDATA)>  
<!ELEMENT phone_no (#PCDATA)>
```

**<!ELEMENT name (#PCDATA)>** define el *name* del elemento para que se escriba "**#PCDATA**".

En el ejemplo #PCDATA significa datos de texto interpretables por la máquina.

## DTD Externo

Para referenciar un XML como DTD externo, el atributo ***standalone*** en la declaración del archivo se debe marcar con un **No**.

Los elementos se declaran fuera del archivo XML. Se puede acceder a ellos especificando los atributos del sistema que pueden ser o un archivo legal ***.dtd*** o una dirección ***URL válida***

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?  
<!DOCTYPE root-element SYSTEM "file-name">
```

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?  
<!DOCTYPE elemento-raíz PUBLIC "identificador-público" "URI">
```

## DTD Externo

**EJEMPLO** Si en un archivo llamado *"marcadores.dtd"*

```
<!ELEMENT marcadores (pagina)*>
<!ELEMENT pagina (nombre, descripcion, url)>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT descripcion (#PCDATA)>
<!ELEMENT url (#PCDATA)>
```

*marcadores-con-dtd-externa.xml*

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE marcadores SYSTEM "marcadores.dtd">
```

```
<marcadores>
  <pagina>
    <?xml version="1.0" standalone="no"?>
    <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
    <html>
      <head>
        <title>Título</title>
      </head>
      <body>
        <p>Párrafo</p>
      </body>
    </html>
```

## DTD - Elementos

Un elemento **DTD** se declara con una declaración de **ELEMENT**.

Cuando un archivo XML es valorado por una DTD, el analizador inicialmente examina el elemento raíz y luego los elementos secundarios

```
<!ELEMENT elementname (content)>
```

- **elementname** es el nombre del elemento (también llamado *Identificador genérico*) que se está definiendo.
- **content** define qué contenido, si hubiera alguno, puede ir en el elemento.



# DTD - Elementos

## Contenido de un elemento.

Un elemento puede tener cuatro posibles contenidos

- Contenido vacío

Este tipo de declaración de elemento no contiene ningún contenido. Estas se declaran con la palabra clave **EMPTY**. `<!ELEMENT elementname EMPTY >`

- Contenido de elemento

El contenido sería los elementos permitidos entre paréntesis. También podemos incluir más de un elemento.

```
<!ELEMENT elementname (child1, child2...)>
```

- Contenido mixto

Esta es la combinación de (#PCDATA) y de sus elementos secundarios

```
<!ELEMENT elementname (#PCDATA|child1|child2)*>
```

- Cualquier contenido

Se usar la palabra clave ANY cuando aun se tienen que decidir los contenidos que se permitirán en el elemento. `<!ELEMENT elementname ANY>`

## DTD - Elementos

Operador	Sintaxis	Descripción
+	<b>&lt;!ELEMENTO nombre del elemento (elemento secundario1+)&gt;</b>	Indica que el elemento secundario se puede dar una o más veces dentro del elemento principal.
*	<b>&lt;!ELEMENTO nombre del elemento (elemento secundario1*)&gt;</b>	Indica que el elemento secundario se puede dar cero o más veces dentro del elemento principal.
?	<b>&lt;!ELEMENTO nombre del elemento (elemento secundario1?)&gt;</b>	Indica que el elemento secundario se puede dar cero veces o una vez dentro del elemento principal.
,	<b>&lt;!ELEMENTO nombre del elemento (elemento secundario1, elemento secundario2)&gt;</b>	Da una secuencia de los elementos secundarios separados por comas, los cuales deben ser incluidos en el nombre del elemento.
	<b>&lt;!ELEMENTO nombre del elemento (elemento secundario1   elemento secundario2)&gt;</b>	Permite hacer elecciones en el elemento secundario.

## DTD - Elementos

¿Cuáles son los fallos?

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE perifericos [
  <!ELEMENT perifericos (#PCDATA)>
]>
<perifericos>
  <periferico>impresora</periferico>
  <periferico>monitor</periferico>
  <periferico>teclado</periferico>
</perifericos>
```

<perifericos>**impresora monitor teclado**</perifericos>

Perifericos no puede tener hijos

## DTD - Elementos

¿Cuáles son los fallos?

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

```
<!DOCTYPE perro [
```

```
  <!ELEMENT perro (edad, nombre)>
```

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

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

```
<perro>
```

```
  <nombre>Pancho</nombre>
```

```
  <edad>8</edad>
```

```
</perro>
```

```
<perro>
```

```
  <edad>8</edad>
```

```
  <nombre>Pancho</nombre>
```

```
</perro>
```

Edad ha de ir antes de nombre

## DTD - Elementos

¿Cuáles son los fallos?

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

```
<!DOCTYPE caballo [
```

```
  <!ELEMENT caballo (nombre, raza, edad)>
```

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

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

```
  <!ELEMENT edad EMPTY> ]>
```

```
<caballo>
```

```
  <nombre>Silvestre</nombre>
```

```
  <raza>Morgan</raza>
```

```
</caballo>
```

```
<caballo>
```

```
  <nombre>Silvestre</nombre>
```

```
  <raza>Morgan</raza>
```

```
  <edad/>
```

```
</caballo>
```

Edad ha de aparecer y ha de ser vacío

## DTD - Elementos

¿Cuáles son los fallos?

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

```
<!DOCTYPE partes [
```

```
<!ELEMENT partes (secciones)>
```

```
<!ELEMENT secciones ANY>
```

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

```
]>
```

```
<secciones>
```

```
<letras>abc</letras>
```

```
MNT
```

```
<letras>xyz</letras>
```

```
</secciones>
```

```
<partes>
```

```
<secciones>
```

```
<letras>abc</letras>
```

```
MNT
```

```
<letras>xyz</letras>
```

```
</secciones>
```

```
</partes>
```

Falta el elemento raíz

## DTD - Atributos

La **declaración de atributos** es muy similar a la declaración de elementos en muchos sentidos, excepto en uno; en vez de declarar contenido permitido para elementos, se declara una lista de atributos permitidos para cada elemento. Estas listas se denominan declaración '**ATTLIST**'

```
<!ATTLIST element-name attribute-name attribute-type attribute-value>
```

- **element-name** especifica el nombre del elemento al que se aplica el atributo.
- **attribute-name** especifica el nombre del atributo que se incluye con el nombre del elemento.
- **attribute-type** define el tipo de atributo. Hablaremos de este apartado en las siguientes secciones.
- **attribute-value** toma un valor fijo que el atributo debe definir. Hablaremos más sobre este asunto en las siguientes secciones.

# DTD - Atributos

## Normas de los atributos

- **Todos** los atributos que se usan en un documento **XML** deben **declararse** en el documento **DTD** usando una declaración de lista de tributos (Attribute-List Declaration).
- Los atributos puede que solo aparezcan al inicio o en una etiqueta vacía.
- La lista de atributos claves debe especificarse en mayúsculas
- No se permitirán nombres de atributos repetidos en la lista de atributos para un elemento determinado.

La tipología de atributos se puede dividir en tres categorías:

- Tipo de cadena
- Tipología de caso (en inglés token)
- Tipos enumerados

[https://www.w3schools.com/xml/xml\\_dtd\\_attributes.asp](https://www.w3schools.com/xml/xml_dtd_attributes.asp)



# **DTD - Atributos**

## **Atributos**

[http://codexexempla.org/articulos/2008/leer\\_dtd\\_2.php](http://codexexempla.org/articulos/2008/leer_dtd_2.php)

## **XML DTD y Atributos - Ejercicios**

<http://www.mclibre.org/consultar/xml/ejercicios/dtd.html>

# DTD

## Ejercicio

**<?xml version="1.0" encoding="ISO-8859-1" standalone="yes" ?>**

...

**<nota>**

**<para>Pedro</para>**  Puede no aparecer

**<de>Laura</de>**

**<titulo>Recordatorio</titulo>**

**<contenido>A las 7:00 en la puerta del teatro</contenido>**

**</nota>**

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

**<!DOCTYPE nota [**

**<!ELEMENT nota (para\*, de, titulo, contenido)>**

**<!ELEMENT para (#PCDATA)>**

**<!ELEMENT de (#PCDATA)>**

**<!ELEMENT titulo (#PCDATA)>**

**<!ELEMENT contenido (#PCDATA)>**

**]>**

**<nota>**

**<para>Pedro</para>**

**<de>Laura</de>**

**<titulo>Recordatorio</titulo>**

**<contenido>A las 7:00 en la puerta del teatro</contenido>**

**</nota>**

## DTD Externo

```
<?xml version="1.0" encoding="ISO-8859-1"? Standalone="no">
<!DOCTYPE nota SYSTEM "nota.dtd">
<nota>
    <para>Pedro</para>
    <de>Laura</de>
    <titulo>Recordatorio</titulo>
    <contenido>A las 7:00 en la puerta del teatro</contenido>
</nota>
```

### nota.dtd

```
<!ELEMENT nota (para*, de, titulo, contenido)>
<!ELEMENT para (#PCDATA)>
<!ELEMENT de (#PCDATA)>
<!ELEMENT titulo (#PCDATA)>
<!ELEMENT contenido (#PCDATA)>
```

Ejercicio. Escribir un DTD que valide este XML

```
<lista_de_notas>
  <nota>
    <para>Pedro</para>
    <de>Laura</de>
    <titulo>Recordatorio</titulo>
    <contenido>A las 7:00 pm en la puerta del teatro </contenido>
  </nota>
  <nota dia="23/01/2010" hora="13:15">
    <para>Miguel</para>
    <de>Juan</de>
    <titulo>Informes</titulo>
    <contenido>Tienes los informes en el casillero</contenido>
  </nota>
</lista_de_notas>
```

```
<!ELEMENT lista_de_notas (nota+)>
<!ELEMENT nota (para, de, titulo, contenido)>
<!ELEMENT para (#PCDATA)>
<!ELEMENT de (#PCDATA)>
<!ELEMENT titulo (#PCDATA)>
<!ELEMENT contenido (#PCDATA)>
<!ATTLIST nota dia CDATA #IMPLIED>
<!ATTLIST nota hora CDATA #IMPLIED>
```

Ejercicio. Escribir un DTD que valide este XML

```
<lista_de_notas>
  <nota>
    <para>Pedro</para>
    <de>Laura</de>
    <titulo>Recordatorio</titulo>
    <contenido>A las 7:00 pm en la puerta del teatro </contenido>
  </nota>
  <nota dia="23/01/2010" hora="13:15">
    <para>Miguel</para>
    <de>Juan</de>
    <titulo>Informes</titulo>
    <contenido>Tienes los informes en el casillero</contenido>
  </nota>
</lista_de_notas>
```

```
<!ELEMENT lista_de_notas (nota+)>
<!ELEMENT nota (para, de, titulo, contenido)>
<!ELEMENT para (#PCDATA)>
<!ELEMENT de (#PCDATA)>
<!ELEMENT titulo (#PCDATA)>
<!ELEMENT contenido (#PCDATA)>
<!ATTLIST nota dia CDATA #IMPLIED>
<!ATTLIST nota hora CDATA #IMPLIED>
```

Ejercicio. Escribir un DTD que valide este XML

Ediciones Aranda Informe regional del ventas Descripción: informe de ventas para las regiones Norte, Centro y Sur Fecha del informe: 30/12/2009		
Region	Trimestre	Libros Vendidos
Norte	1	24000
	2	38600
	3	NO_INFO
	4	NO_INFO
Centro	1	NO_INFO
	2	16080
	3	25000
	4	29000
Sur	1	27000
	2	31400
	3	40100
	4	30000

**Se deben tener en cuenta las siguientes consideraciones:**

- Es obligatorio que el informe lleve una fecha
- Se debe poder diferenciar la parte de la cabecera del informe de la parte con los datos
- Siempre deben aparecer las tres regiones en el informe, y ninguna más
- Para cada zona deben aparecer siempre los cuatro trimestres, aunque falte la información sobre los libros vendidos
- Si no se incluye el número de libros vendidos en los datos, en el informe aparecerá la cadena NO\_INFO
- El número de trimestre sólo puede tomar los valores 1, 2, 3 o 4

## Ejercicio. Escribir un DTD que valide este XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ediciones_aranda [
  <!ELEMENT ediciones_aranda (cabecera, tabla)>
  <!ELEMENT cabecera (titulo, contenido, descripcion, fecha)>
  <!ELEMENT titulo (#PCDATA)>
  <!ELEMENT contenido (#PCDATA)>
  <!ELEMENT descripcion (#PCDATA)>
  <!ELEMENT fecha (#PCDATA)>
  <!ELEMENT tabla (region+, trimestre, libros_vendidos)>
  <!ELEMENT region (norte | centro | sur)>
  <!ELEMENT norte (#PCDATA|trim|lib)*>
  <!ELEMENT centro (#PCDATA|trim|lib)*>
  <!ELEMENT sur (#PCDATA|trim|lib)*>
  <!ELEMENT trim (#PCDATA)>
  <!ELEMENT lib (#PCDATA)>
  <!ELEMENT trimestre (#PCDATA)>
  <!ELEMENT libros_vendidos (#PCDATA)>
]>
```

## Ejercicio. Escribir un DTD que valide este XML

```
<ediciones_aranda>
  <cabecera>
    <titulo>Ediciones Aranda</titulo>
    <contenido>Informe regional del ventas</contenido>
    <descripcion>Descripcion: informe de ventas para las regiones Norte, Centro y
Sur</descripcion>
    <fecha>Fecha: 30/12/2009</fecha>
  </cabecera>
  <tabla>
    <region>
      <norte>Norte
        <trim>1</trim>
        <lib>24000</lib>
        <trim>2</trim>
        <lib>38600</lib>
        <trim>3</trim>
        <lib></lib>
        <trim>4</trim>
        <lib></lib>
      </norte>
    </region>
    <region>
      <centro>Centro
        <trim>1</trim>
        <lib></lib>
        <trim>2</trim>
        <lib>16080</lib>
        <trim>3</trim>
        <lib>25000</lib>
        <trim>4</trim>
        <lib>29000</lib>
      </centro>
    </region>
    <region>
      <sur>Sur
        <trim>1</trim>
        <lib>27000</lib>
        <trim>2</trim>
        <lib>31400</lib>
        <trim>3</trim>
        <lib>40100</lib>
        <trim>4</trim>
        <lib>30000</lib>
      </sur>
    </region>
    <trimestre>Trimestre</trimestre>
    <libros_vendidos>
      Libros vendidos
    </libros_vendidos>
  </tabla>
</ediciones_aranda>
```



```
<factura n_fac="f999">
  <datos_empresa>
    <nombre>Equipos Digitales S.L.</nombre>
    <dir>Av. Valladolid</dir>
    <poblacion cod_postal="28043" Provincia="Madrid">Madrid</poblacion>
    <cif>Q-9876543</cif>
    <telefono/>
  </datos_empresa>
  <datos_cliente n_cli="c879">
    <nombre>Darío, Bueno Gutiérrez</nombre>
    <dir_env>Av. Oporto nº7 4ºd</dir_env>
    <poblacion cod_postal="28043">Madrid</poblacion>
    <provincia>Madrid</provincia>
  </datos_cliente>
  <datos_factura n_ped="p731" iva="16" f_pago="efectivo" moneda="euro">
    <fecha>12-01-2005</fecha>
    <linea>
      <ref>MII93000F/8</ref>
      <desc>MICRO PENTIUM IV 3000MHZ FB800</desc>
      <cant>1</cant>
      <precio>230</precio>
      <importe>266,80</importe>
    </linea>
    <base>970,00</base>
    <cuota_iva>155,20</cuota_iva>
    <total>1125,20</total>
  </datos_factura>
</factura>
```

# DTD

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!-- ***** -->
<!-- DEFINICIÓN DEL TIPO DE DOCUMENTO -->
<!DOCTYPE factura [
  <!-- Definición de elementos -->
  <!ELEMENT factura (datos_empresa, datos_cliente, datos_factura)>
  <!ELEMENT datos_empresa (nombre,dir,poblacion,cif,telefono?,fax?)>
  <!ELEMENT datos_cliente (nombre, dir_env, poblacion, provincia)>
  <!ELEMENT datos_factura (fecha, linea*, base, cuota_iva, total)>
  <!ELEMENT linea (ref, desc, cant, precio, importe)>
  <!ELEMENT ref (#PCDATA)>
  <!ELEMENT desc (#PCDATA)>
  <!ELEMENT cant (#PCDATA)>
  <!ELEMENT precio (#PCDATA)>
  <!ELEMENT importe (#PCDATA)>
  <!ELEMENT nombre (#PCDATA)>
  <!ELEMENT dir (#PCDATA)>
  <!ELEMENT poblacion (#PCDATA)>
  <!ELEMENT provincia (#PCDATA)>
  <!ELEMENT cif (#PCDATA)>
  <!ELEMENT telefono EMPTY>
  <!ELEMENT fax EMPTY>
  <!ELEMENT dir_env (#PCDATA)>
  <!ELEMENT fecha (#PCDATA)>
  <!ELEMENT base (#PCDATA)>
  <!ELEMENT cuota_iva (#PCDATA)>
  <!ELEMENT total (#PCDATA)>

  <!-- Definición de atributos -->
  <!ATTLIST factura n_fac ID #REQUIRED>
  <!ATTLIST telefono num_tel CDATA #FIXED "917776688">
  <!ATTLIST fax num_fax CDATA #FIXED "917776699">
  <!ATTLIST datos_cliente n_cli ID #REQUIRED>
  <!ATTLIST datos_factura n_ped ID #REQUIRED>
  <!ATTLIST datos_factura iva NMTOKEN #REQUIRED>
  <!ATTLIST datos_factura f_pago (efectivo|tarjeta|plazos) #REQUIRED>
  <!ATTLIST datos_factura moneda CDATA #FIXED "euro">
  <!ATTLIST poblacion cod_postal CDATA "">
  <!ATTLIST poblacion provincia CDATA "">
]>
```

# **DTD**

Ejercicios DTD:

<http://www.abrirllave.com/dtd/ejercicios-resueltos.php>